

Tero Kokkonen

Anomaly-Based Online Intrusion
Detection System as a Sensor for
Cyber Security Situational
Awareness System



JYVÄSKYLÄ STUDIES IN COMPUTING 251

Tero Kokkonen

Anomaly-Based Online Intrusion
Detection System as a Sensor for
Cyber Security Situational
Awareness System

Esitetään Jyväskylän yliopiston informaatioteknologian tiedekunnan suostumuksella
julkisesti tarkastettavaksi yliopiston Agora-rakennuksen Lea Pulkkisen salissa
joulukuun 15. päivänä 2016 kello 12.

Academic dissertation to be publicly discussed, by permission of
the Faculty of Information Technology of the University of Jyväskylä,
in building Agora, Lea Pulkkinen hall, on December 15, 2016 at 12 o'clock noon.



UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2016

Anomaly-Based Online Intrusion
Detection System as a Sensor for
Cyber Security Situational
Awareness System

JYVÄSKYLÄ STUDIES IN COMPUTING 251

Tero Kokkonen

Anomaly-Based Online Intrusion
Detection System as a Sensor for
Cyber Security Situational
Awareness System



UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2016

Editors

Timo Männikkö

Department of Mathematical Information Technology, University of Jyväskylä

Pekka Olsbo, Ville Korhonen

Publishing Unit, University Library of Jyväskylä

URN:ISBN:978-951-39-6832-8

ISBN 978-951-39-6832-8 (PDF)

ISBN 978-951-39-6831-1 (nid.)

ISSN 1456-5390

Copyright © 2016, by University of Jyväskylä

Jyväskylä University Printing House, Jyväskylä 2016

ABSTRACT

Kokkonen, Tero

Anomaly-Based Online Intrusion Detection System as a Sensor for Cyber Security
Situational Awareness System

Jyväskylä: University of Jyväskylä, 2016, 82 p.(+included articles)

(Jyväskylä Studies in Computing

ISSN 1456-5390; 1456-5390; 251)

ISBN 978-951-39-6831-1 (nid.)

ISBN 978-951-39-6832-8 (PDF)

Finnish summary

Diss.

Almost all the organisations and even individuals rely on complex structures of data networks and networked computer systems. That complex data ensemble, the cyber domain, provides great opportunities, but at the same time it offers many possible attack vectors that can be abused for cyber vandalism, cyber crime, cyber espionage or cyber terrorism. Those threats produce requirements for cyber security situational awareness and intrusion detection capability. This dissertation concentrates on research and development of anomaly-based network intrusion detection system as a sensor for a situational awareness system. In this dissertation, several models of intrusion detection systems are developed using clustering-based data-mining algorithms for creating a model of normal user behaviour and finding similarities and dissimilarities compared to that model. That information can be used as a sensor feed in a situational awareness system in cyber security. A model of cyber security situational awareness system with multi-sensor fusion capability is presented in this thesis. Also a model for exchanging the information of cyber security situational awareness is generated. The constructed intrusion detection system schemes are tested with different scenarios even in online mode with real user data.

Keywords: Anomaly Detection, Clustering, Cyber Security, Early Warning, Information Sharing, Intrusion Detection System, Network Security, Situational Awareness

Author Tero Kokkonen
Department of Mathematical Information Technology
University of Jyväskylä
Finland

Supervisors Professor Timo Hämäläinen
Department of Mathematical Information Technology
University of Jyväskylä
Finland

Ph.D., Adjunct Professor Jarmo Siltanen
Director
Institute of Information Technology
JAMK University of Applied Sciences
Finland

Ph.D. Mikhail Zolotukhin
Department of Mathematical Information Technology
University of Jyväskylä
Finland

Reviewers Professor Janos Sztrik
Department of Informatics Systems and Networks
University of Debrecen
Hungary

Ph.D., Adjunct Professor Kari Luostarinen
Development Manager
Regional Council of Central Finland
Finland

Opponent Professor Jarno Linnéll
Department of Communications and Networking
Aalto University
Finland

ACKNOWLEDGEMENTS

I would like to express my sincerest gratitude to my supervisors Professor Timo Hämäläinen, Ph.D. Jarmo Siltanen and Ph.D Mikhail Zolotukhin for the guidance with this dissertation, and working as authors with the included papers.

I would like to thank the Department of Mathematical Information Technology and Faculty of Information Technology of University of Jyväskylä about the opportunity to write this dissertation, and I would like to thank the Institute of Information Technology of JAMK University of Applied Sciences for offering me opportunity to do the cyber security related research activities as part of my duties in the JYVSECTEC research, development and training center.

I would also like to thank Jari Hautamäki, Antti Niemelä, Marko Silokunnas and Mikko Neijonen who has worked as authors with the included papers, and all the other colleagues and collaborators who have worked with me and helped me with this research.

My warm and special thank goes to my wife Heidi, my children Ida, Eetu and Leevi, and of course to my Mother Tarja and Father Raimo.

Jyväskylä 16.11.2016
Tero Kokkonen

GLOSSARY

AE	Auto-Encoder
APT	Advanced Persistent Threat
CCOP	Cyber Common Operational Picture
CIA	Confidentiality, Integrity, Availability
CLT	Central Limit Theorem
COP	Common Operational Picture
CyboX™	Structured Cyber Observable eXpression
C2	Command and Control
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DDoS	Distributed Denial of Service
DoS	Denial of Service
EM	Expectation Maximisation -algorithm
EWMA	Exponentially Weighted Moving Average
FADS	Flow Anomaly Detection System
FIRST	Forum of Incident Response and Security Teams
FPR	False Positive Rate
HIDS	Host Intrusion Detection System
HMI	Human Machine Interface
HTTP	Hypertext Transfer Protocol
IDS	Intrusion Detection System
IOC	Indicator Of Compromise
ISP	Internet Service Provider
JDL	Joint Directors of Laboratories
JYVSECTEC	Jyväskylä Security Technology
KNN	K-Nearest Neighbors
NIDS	Network Intrusion Detection System
OSI	Open Systems Interconnection
PCA	Principal Component Analysis
PCAP	Packet CAPture
RGCE	Realistic Global Cyber Environment
ROC	Receiver Operating Characteristic
R2L	Remote to Local
SA	Situational Awareness
SAE	Stacked Auto-Encoder
SDN	Software-Defined Networking
SOM	Self-Organizing Map
SSL	Secure Sockets Layer
STIX™	Threat Information eXpression
SVDD	Support Vector Data Description
TAXII™	Trusted Automated eXchange of Indicator Information
TLP	Traffic Light Protocol
TLS	Transport Layer Security
TPR	True Positive rate
TTP	Tactics, Techniques, and Procedures
U2R	User to Roar

LIST OF FIGURES

FIGURE 1	Relationship of included articles.....	17
FIGURE 2	HMI of Internet data generation application (PII).....	24
FIGURE 3	ROC curves of anomalous web resources by SVDD (PI).....	31
FIGURE 4	ROC curves of anomalous query attributes by k-means (PI).....	31
FIGURE 5	ROC curves of anomalous user agents by DBSCAN (PI).....	32
FIGURE 6	Normalized entropy values with 60-second time window and 10-second time steps (PI).....	32
FIGURE 7	Detection Accuracy with different n_C values (PIV).....	36
FIGURE 8	ROC curves of compared algorithms (PIV).....	37
FIGURE 9	Chi-square values of extracted features of time intervals (PIII).....	41
FIGURE 10	ROC curves of compared detection algorithms (PIII).....	41
FIGURE 11	ROC curves of Slowloris detection (PVI).....	45
FIGURE 12	ROC curves of Slowpost detection (PVI).....	45
FIGURE 13	ROC curves of intermediate DDoS attack detection (PVI).....	46
FIGURE 14	ROC curves of DDoS attack detection (PVIII).....	50
FIGURE 15	Setup of live scenario.....	51
FIGURE 16	Confusion matrix.....	52
FIGURE 17	Average costs of conversation clustering (PIX).....	56
FIGURE 18	Average costs of user session clustering (PIX).....	56
FIGURE 19	ROC curves of Sslsqueeze detection for different clustering parameters (PIX).....	57
FIGURE 20	ROC curves of Slowloris detection for different clustering pa- rameters (PIX).....	57
FIGURE 21	ROC curves of advanced DDoS attack detection for different time window sizes and clustering parameters (PIX).....	58
FIGURE 22	Modified OODA-loop, quoted from Brehmer (2005).....	60
FIGURE 23	OODA-loop in cyber defence.....	60
FIGURE 24	Threat sharing models of TAXII™, quoted from Connolly et al. (2014).....	61
FIGURE 25	Architecture of STIX™, quoted from Barnum (2014).....	62
FIGURE 26	Example of STIX™ use cases, quoted from Barnum (2014).....	62
FIGURE 27	Cyber security information sharing community (PV).....	63
FIGURE 28	Cyber security information sharing topology with a minimum risk level implementation (PV).....	64
FIGURE 29	High level architecture (PVII).....	67
FIGURE 30	Use case example of cyber security SA system (PVII).....	67

LIST OF TABLES

TABLE 1	Day four, scans and bruteforce attacks (PI).	33
TABLE 2	Detection Accuracy of the time interval analysis applying the entropy-based method (PI).....	33
TABLE 3	Performance metrics of compared algorithms (PIV).....	37
TABLE 4	Performance metrics of compared detection algorithms (PIII)....	41
TABLE 5	Performance metrics of Slowloris detection (PVI).....	45
TABLE 6	Performance metrics of Slowpost detection (PVI).....	45
TABLE 7	Performance metrics of intermediate DDoS attack detection (PVI).....	46
TABLE 8	Detection Accuracy of DDoS attacks (PVIII).	50
TABLE 9	Confusion matrix of a live scenario.	52
TABLE 10	Performance metrics of live scenario.....	52
TABLE 11	Detection Accuracy of DDoS attacks (PIX).....	58

CONTENTS

ABSTRACT

ACKNOWLEDGEMENTS

GLOSSARY

LIST OF FIGURES

LIST OF TABLES

CONTENTS

LIST OF INCLUDED ARTICLES

1	INTRODUCTION	13
1.1	Research motivation	13
1.2	Research questions	14
1.3	Research approach.....	15
1.4	Structure of the work	16
1.5	Research contribution	17
2	IMPLEMENTING THE INTRUSION DETECTION SYSTEM.....	20
2.1	Intrusion detection system.....	20
2.2	Realistic network traffic generation for testing the IDSs (PII).....	22
2.2.1	Requirements for traffic generation (PII)	22
2.2.2	Implemented solution (PII)	23
3	ANOMALY-DETECTION-BASED IDS WITH CAPTURED NETWORK DATA.....	25
3.1	Analysing HTTP requests (PI).....	25
3.1.1	Feature extraction (PI)	25
3.1.1.1	Statistics based on requests (PI)	26
3.1.1.2	Statistics based on time intervals (PI)	27
3.1.2	Analysis (PI)	27
3.1.2.1	Analysis of requested web resource (PI)	27
3.1.2.2	Analysis of query attribute values (PI)	28
3.1.2.3	Analysis of user agent values (PI)	29
3.1.2.4	Analysis of time intervals (PI)	29
3.1.3	Testing the model (PI)	30
3.2	Analysing network flows (PIV)	33
3.2.1	Feature extraction (PIV).....	33
3.2.2	Analysis (PIV)	34
3.2.2.1	Clustering flows (PIV)	34
3.2.2.2	Finding anomalous time intervals (PIV)	35
3.2.2.3	Finding intrusive flows (PIV)	36
3.2.3	Testing the model (PIV)	36
3.3	Analysing encrypted traffic, first model (PIII).....	37
3.3.1	Feature extraction (PIII).....	38
3.3.2	Analysis (PIII)	39

3.3.2.1	Finding anomalous time intervals (PIII)	39
3.3.2.2	Finding intrusive flows (PIII)	39
3.3.3	Testing the model (PIII)	40
3.4	The implemented general model for analysing encrypted traffic (PVI)	41
3.4.1	Feature extraction (PVI).....	42
3.4.2	Analysis (PVI)	43
3.4.3	Testing the model (PVI).....	44
4	ANOMALY-DETECTION-BASED IDS WITH ONLINE NETWORK DATA.....	47
4.1	Online training (PVIII)	47
4.1.1	Anomaly detection (PVIII).....	49
4.1.2	Testing the model (PVIII).....	49
4.2	Online analysis.....	50
4.2.1	Testing the model	51
4.3	Online analysis in high-speed networks (PIX)	53
4.3.1	Clustering (PIX).....	53
4.3.2	Anomaly detection (PIX)	55
4.3.3	Testing the model (PIX)	55
5	SITUATIONAL AWARENESS IN CYBER SECURITY	59
5.1	Situational awareness for decision making (PV; PVII)	59
5.2	Sharing the cyber security situational awareness (PV)	61
5.3	Cyber security situational awareness system (PVII)	64
5.3.1	Multi-sensor data fusion (PVII)	64
5.3.2	Required Interfaces (PVII)	65
5.3.3	High level architecture (PVII).....	66
6	CONCLUSION	68
	YHTEENVETO (FINNISH SUMMARY)	69
	REFERENCES.....	70
	INCLUDED ARTICLES	

LIST OF INCLUDED ARTICLES

- PI Mikhail Zolotukhin, Timo Hämäläinen, Tero Kokkonen, Jarmo Siltanen. Analysis of HTTP Requests for Anomaly Detection of Web Attacks. *The 12th IEEE International Conference on Dependable, Autonomic and Secure Computing, DASC 2014, Dalian, China, August 24-27, 2014, Proceedings, 2014.*
- PII Tero Kokkonen, Timo Hämäläinen, Marko Silokunnas, Jarmo Siltanen, Mikhail Zolotukhin, Mikko Neijonen. Analysis of Approaches to Internet Traffic Generation for Cyber Security Research and Exercise. *Lecture Notes in Computer Science, vol. 9247, pp. 254-267, 2015.*
- PIII Mikhail Zolotukhin, Timo Hämäläinen, Tero Kokkonen, Antti Niemelä, Jarmo Siltanen. Data Mining Approach for Detection of DDoS Attacks Utilizing SSL/TLS Protocol. *Lecture Notes in Computer Science, vol. 9247, pp. 274-285, 2015.*
- PIV Mikhail Zolotukhin, Timo Hämäläinen, Tero Kokkonen, Jarmo Siltanen. Online detection of anomalous network flows with soft clustering. *7th International Conference on New Technologies, Mobility and Security, NTMS 2015, Paris, France, July 27-29, 2015, Proceedings, 2015.*
- PV Tero Kokkonen, Jari Hautamäki, Jarmo Siltanen, Timo Hämäläinen. Model for sharing the information of cyber security situation awareness between organizations. *23rd International Conference on Telecommunications, ICT 2016, Thessaloniki, Greece, May 16-18, 2016, Proceedings, 2016.*
- PVI Mikhail Zolotukhin, Timo Hämäläinen, Tero Kokkonen, Jarmo Siltanen. Increasing web service availability by detecting application-layer DDoS attacks in encrypted traffic. *23rd International Conference on Telecommunications, ICT 2016, Thessaloniki, Greece, May 16-18, 2016, Proceedings, 2016.*
- PVII Tero Kokkonen. Architecture for the Cyber Security Situational Awareness System. *Lecture Notes in Computer Science, vol. 9870, pp. 294-302, 2016.*
- PVIII Mikhail Zolotukhin, Tero Kokkonen, Timo Hämäläinen, Jarmo Siltanen. Weighted Fuzzy Clustering for Online Detection of Application DDoS Attacks in Encrypted Network Traffic. *Lecture Notes in Computer Science, vol. 9870, pp. 326-338, 2016.*
- PIX Mikhail Zolotukhin, Tero Kokkonen, Timo Hämäläinen, Jarmo Siltanen. On Application-Layer DDoS Attack Detection in High-Speed Encrypted Networks. *International Journal of Digital Content Technology and its Applications (JDCTA), Accepted Oct/2016.*

1 INTRODUCTION

Research motivation, and the research question and approach are explained in this chapter. The overall structure of the dissertation is covered, as well as the included articles and contribution of the author in the included articles.

1.1 Research motivation

Almost all the software systems are connected to data networks. According to the Finland's Cyber Security Strategy (2013), that versatile environment for data processing is called the cyber domain. Societies and organisations rely on that cyber domain which offers great possibilities but also poses serious threats.

Cyber threats pose risks for all the levels of modern societies from individuals to big corporations or even countries. United Kingdom National Crime Agency (2016) determines cyber crime as a serious and growing threat for businesses. Situational awareness is required for achieving understanding of the situation of one's own assets and for making decisions related to a business or mission. One extremely important input for Situational Awareness (SA) in cyber security is sensor information originating from the Intrusion Detection System (IDS).

IDS consist of a network located or host located sensors which are analysing data and reporting detected intrusions (Gollmann, 2011). Intrusion detection of network traffic can be divided into two basic solutions: anomaly detection and misuse detection, also known as anomaly based detection and signature based detection. With signature based misuse detection, intrusions are detected by comparing samples with known predefined signatures or attack patterns. Misuse detection is accurate and effective for detecting known attacks, but it cannot detect state-of-the-art attacks with novel, unknown signatures or attack patterns. Also mistakes in the signature definition will prejudice the intrusion detection. With anomaly detection, the normal behaviour profile is established. The differences from that norm with the threshold are indicated as anomalies and detected as in-

trusions. The advantage of anomaly detection approach is its capability to detect novel zero-day attacks with unknown attack patterns (Jabez and Muthukumar, 2015; Gollmann, 2011; Kaushik and Deshmukh, 2011; Ghorbani et al., 2010; Kumar et al., 2010; Hernández-Pereira et al., 2009; Kruegel et al., 2005; Asaka et al., 2003; Kruegel et al., 2003; Mukkamala and Sung, 2003).

Online processing of network data for intrusion detection is required as a sensor information for situational awareness in cyber domain. A systematic literature review by Franke and Brynielsson (2014) substantiates the fact that there are several high level national strategies that indicate the importance of situational awareness in cyber security. For example, requirement for real-time situational awareness is expressed in Finland's Cyber Security Strategy (2013), and early warning information is described in Cybersecurity Strategy of the European Union (2013). US International Strategy For Cyberspace (2011) details shared situational awareness between government organisations and companies, and also states co-operation between countries in global situational awareness and incident response capability. The cyber domain is extremely complicated and includes the complex structures of networks and dynamic interactions of networked computer systems added with growing amount of data which is potentially encrypted. Anomaly-detection-based online analysis of network traffic gives effective sensor information feed for cyber security situational awareness. It might even give an early warning indication of an occurring zero-day attack. For achieving such capability, the research questions are formulated in the following chapter.

1.2 Research questions

The research objective of this dissertation is to research and develop anomaly-detection-based online intrusion detection system that can be used as a sensor for situational awareness system in cyber security. The dissertation includes several case studies of situational awareness and anomaly-detection-based intrusion detection solutions. Those case studies combine to answer the following main research question of the study:

- How to develop an anomaly-detection-based online IDS that can be used as a sensor for a cyber security SA system? The main research question can be divided into the following sub-questions:
 - How to generate realistic user traffic for research and development of anomaly-detection-based intrusion detection system?
 - How to detect intrusions from HTTP requests by using anomaly detection?
 - How to detect intrusions from encrypted network traffic by using anomaly detection?
 - How to detect intrusions from encrypted online traffic with near-real-

- time capability?
- What are the architecture requirements for a cyber security situational awareness system?
 - How to exchange information between different organisations on situational awareness in cyber security?

1.3 Research approach

Constructive research approach is used in this dissertation to answer the research question of the study. When using the constructive research approach, innovative constructions or artefacts, e.g. processes, practices or tools, are created as solutions for domain-specific real world problems. (Kasanen et al., 1993; Crnkovic, 2010; Piirainen and Gonzalez, 2013; Lehtiranta et al., 2015)

Both, theoretical and practical component should be considered for constructive research. The problem and the solution should be tied with the theoretical comprehension. The four elements that should be included for problem-solving constructs of constructive research are practical relevance, practical functioning, theory connection and theoretical contribution. (Kasanen et al., 1993; Lehtiranta et al., 2015)

The problem of using the online network anomaly-detection-based intrusion detection system as a sensor for cyber security situational awareness system with real encrypted network data is studied. The proposed solution is the cyber security situational awareness system utilising an anomaly-detection-based network intrusion detection system as the online sensor. Differently constructed theoretical methods are always tested in practice by using generated network traffic in a realistic test environment or with real user-traffic and real environment. The model for sharing the cyber security situational awareness information is also tested using realistic scenario, and architecture for cyber security situational awareness system is evaluated by an example use case. The above-mentioned four elements of constructive problem-solving are applied in every included article of the thesis. All the included articles describe practical relevance in their introduction part; practical functioning is considered when testing the constructions with realistic simulations or with real environment; and theory connection and theoretical contribution are described as a theoretical part of the included articles and also when analysing the test results. The overall structure of the dissertation includes both practical and theoretical considerations of the research question.

1.4 Structure of the work

The dissertation is structured as follows. The theoretical background of situational awareness in cyber security as well as theoretical background of anomaly-detection-based intrusion detection and realistic Internet network data generation for testing the intrusion detection systems is introduced. Results achieved on the included research articles and theoretical background of those articles are represented and summarised. Also unpublished results of online analysis are represented. Discussion about research directions for future work is included and conclusion summarises the dissertation.

Theoretical background for intrusions and intrusion detection systems is explained in the chapter 2. Because realistic data is extremely important for testing the intrusion detection systems, implementation of realistic Internet data generation application is also summarised. When constructing and testing the model for intrusion detection system and situational awareness system, there is requirement for realistic test environment with realistic network data generation, because when testing IDSs there should be real attacks jeopardizing the target system. That causes the fact that real production environments should not be used for research and testing of the intrusion detection system, thus there shall be realistic environments that can be jeopardized.

Information about anomaly-detection-based intrusion detection approach is presented in chapter 3. It covers implementation of anomaly-detection-based intrusion detection system using several anomaly detection and data mining algorithms with generated network traffic and real attacks. Analysis of HTTP requests and analysis of network flows with generated traffic are explained and summarised. Especially analysis of encrypted network traffic is also covered.

Near-real-time anomaly detection, where both the training phase and detection phase are done online with real network data, is covered in chapter 4. Analysis of encrypted network data is also included with the capability of implemented near-real-time intrusion detection system.

Chapter 5 presents how to utilise the sensor information explained in the previous chapters for the situational awareness system, and the importance of situational awareness in decision making and the special features with complexity of situational awareness in the cyber domain compared to the kinetic world. One specific and severe requirement, exchange of the situational awareness information is also covered. There is also presented discussion about proposed architecture for cyber security situational awareness system.

Finally results of the dissertation and future work as guidelines for future research activities are concluded in chapter 6.

1.5 Research contribution

The contribution of the author to the included articles consists of the design, implementation and practical demonstrations of the studies with co-authors, and construction of the architecture for the situational awareness system in cyber security. Figure 1 illustrates the total scheme of the study with the included articles. The articles form an ensemble for dissertation. Article PII describes an Internet data generation application that is utilized for simulation-based test scenarios in many of the other articles. Articles PI, PIII, PIV, PVI, PVIII and PIX focus on anomaly-detection-based network IDS with different algorithms implemented and then proceed towards near-real-time online analysis. Articles PV and PVII describe special features of the cyber security situational awareness system.

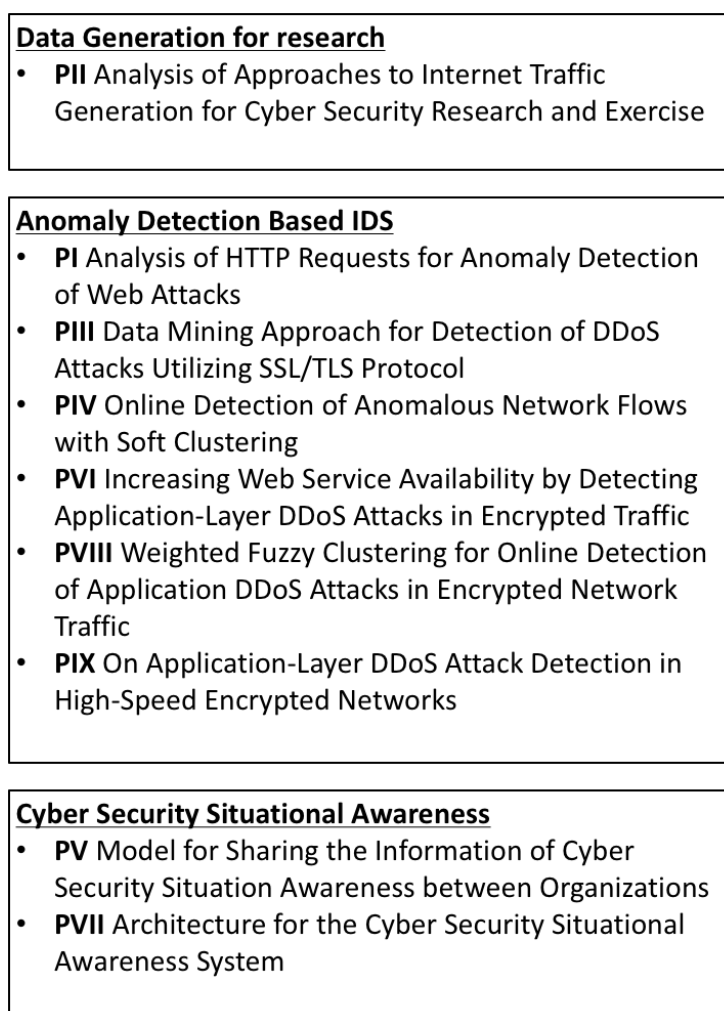


FIGURE 1 Relationship of included articles.

Article PI presents network intrusion detection utilizing the analysis of HTTP requests. A theoretical model is presented. In the experimental part, there is a demonstration where the construction is tested with the real attacks. First there is a training set of HTTP requests with legitimate traffic. Several different anomaly

detection algorithms have been used for solving the behaviour model of normal legitimate users, which is used for labelling deviations from that normal mode as attacks. This early face of implementation uses HTTP log data for analysis. The author's contribution in the article consist of organising the experimental part of the research and taking part in the writing of the theoretical background.

Article PII describes the analysis of Internet data generation for cyber security research and exercise purposes. The paper discusses different approaches for Internet data generation. The requirements for data generation are determined and also existing solutions are analysed. Because no suitable existing solution were found, an Internet data generation software is implemented and the performance of implemented solution is studied. The author's contribution in the article consist of the planning and implementation of the study, taking part of writing the theoretical parts and organising the evaluation of the solution.

Article PIII proposes an anomaly-detection-based algorithm for detecting the Denial of Service (DoS) or Distributed Denial of Service (DDoS) attacks utilizing encrypted Secure Sockets Layer / Transport Layer Security (SSL/TLS) protocols. When using such protocols, the data is encrypted in the application-layer, and it is unthinkable to analyse activity based on the payload of the packets. Our model focuses on statistics extracted from the packet headers and by using anomaly-detection algorithms: normal user behaviour is analysed and deviations from that norm are detected as attacks. Model is tested with simulation with the realistic data generated. The author's contribution in the article concerns organising the experimental part of the research and taking part to the data analysis.

Article PIV describes the anomaly-detection-based analysis of network flows with soft clustering. Line speeds of data networks have increased, and at the same time the amount of network traffic in those lines has increased; that requires heavy computational power for packet payload analysis. For this reason, study concentrates on flow-based network traffic analysis. First the traffic is considered as a time series, and for each time interval in that time series, per-flow information is extracted. Feature vectors of the time intervals are divided into clusters and the distributions of the resulting clusters are analysed for finding anomalous time intervals with anomalous network traffic. After that, traffic in the anomalous time intervals is analysed and the flows characterized as attacks are detected. The constructed model is tested in the simulation environment. The author's contribution in the article consist of organising the experimental part of the research and taking part in the writing of the theoretical background.

Article PV proposes a model for creating topologies for exchanging information of cyber security situational awareness between different organisations. The article has a discussion about situational awareness in cyber security, and the requirement for cyber security information sharing is rationalized. The proposed model for information sharing is based on the risk level estimation of shared sensitive information and the organisation attached. Dijkstra's shortest path algorithm with those risk level estimations is utilized for the proposed model for creating the information sharing topologies, and model is tested with an example scenario. The author's contribution in the article consist of writing the theoret-

ical background, taking part of the implementation of the proposed model and demonstration of the model.

Article PVI proposes a model for detecting application-layer DoS/DDoS attacks in encrypted traffic. The proposed scheme is based on the analysis of conversations between the server and the client, and the model is suitable for encrypted traffic without decryption. Conversations are divided into clusters by feature vectors, and conversations deviating from normal user behaviour are detected as attacks. For more complex attacks where the attacker is capable for mimicking normal user behaviour stacked auto-encoder is used. The constructed model is tested with the simulation environment with realistic user traffic and attack data. The author's contribution in the article concerns organising the experimental part, implementing the attack scenario of the research and taking part in the writing of the theoretical background.

Article PVII presents the architecture for the cyber security situational awareness system with an example use case of threat mitigation process. The paper describes multi-sensor data fusion in a cyber domain, the required interfaces for situational awareness system in cyber security and the Human Machine Interface (HMI) with suggestions for solving the visualisation problem. Finally, the proposed architecture and next steps for the study are dealt with. The author implemented the whole study by himself.

Article PVIII proposes an anomaly detection construct based on weighted fuzzy clustering for application-layer DoS/DDoS attacks with encrypted traffic. There are two different versions for normal user behaviour: the offline training algorithm and the online training algorithm. The concept is also tested with a simulation environment. The author's contribution in the article concerns organising the experimental part, implementing the attack scenario of the research and taking part in the writing of the theoretical background.

Journal PIX widens the analysis of the near-real-time implementation of DoS/DDoS attack detection in high-speed encrypted networks: theoretical background and implementation are described along with algorithm evaluation including a description of the test environment, data set and evaluated performance metrics. The author's contribution concerns taking part in the writing of the theoretical parts of the journal.

In addition, the author's contribution in the online analysis described in section 4.2 consist of organising the live tests, analysing the test data and taking part in testing and optimising the Python implementation of the application.

2 IMPLEMENTING THE INTRUSION DETECTION SYSTEM

The concept of intrusion detection system and data generation for developing and testing the intrusion detection system is discussed in this chapter.

2.1 Intrusion detection system

Already in the 1980s, a model of a real-time intrusion-detection expert system had been introduced by Denning for identifying security violations (Paliwal and Gupta, 2012; Botha and von Solms, 2004; Denning, 1987, 1986). The information security covers three standard security goals, the triple of confidentiality-integrity-availability, also known as the CIA triad (Amoli, 2015; Juvonen, 2014; Gollmann, 2011; National Institute of Standards and Technology NIST, 2007). Any set of activities attempting to compromise one or more element of that triad can be considered as an intrusion, and intrusion detection tries to recognize those violations (Juvonen, 2014; Hernández-Pereira et al., 2009). According to National Institute of Standards and Technology NIST (2007) intrusion detection in a computer system or network is the process that itemizes events by the characteristics of possible incident, and incident is defined as violation against standard security practices, acceptable use policies, or computer security policies, covering also impending threat of violation. A violation to compromise a computer system or network may be referred to an intrusion (Amoli, 2015).

One often-used classification divides intrusions or attacks into four main categories, as in DARPA 1998 and KDD Cup 1999 intrusion detection datasets: Probe (explore information about target network, system or service), Denial of Service (Deny the usage of a system or service), Remote to Local (R2L, achieve unauthorized access to the target system), User to Root (U2R, achieve unauthorized access to Root/Super User privileges by using local user access) (Jabez and Muthukumar, 2015; Gifty Jeya et al., 2012; Paliwal and Gupta, 2012; Kumar et al., 2010; Lippmann et al., 2000; Hernández-Pereira et al., 2009).

Saber et al. (2010) states that multiple different studies for the classification of the intrusions or attacks can be found. Very often an attack is a chain of events or chain of individual attacks starting for example with a reconnaissance or probing phase to get information in order to achieve access to a system or a computer and use that access as a route to another system or computer for gaining the mission objectives. That chain of events, or possible different chains of events, can be illustrated with Attack Trees (Gollmann, 2011). As an example, one of the top criminal threats in Europe is ransomware (Europol, 2015). Ransomware takes control of the system or files and demands ransom to be paid for releasing the control back to the user, and in some cases the ransomware will encrypt the system or files. Ransomware can be distributed e.g. in a payload of an exploit kit, by attachment of e-mail or by botnet that have access to the vulnerable system (F-Secure, s. a.).

Advanced Persistent Threat (APT) attacks are targeted, very sophisticated and stealthy network attacks with a selection of attack vectors. The attacker with ample resources and skills strives to establish a long-lasting and undetected persistent threat with the Command and Control (C2) communication channel in a system to achieve the objectives of the attack mission. A special feature of APT is the ability to stay undetected for long periods of time, concealed within the normal legitimate network traffic. The mission objectives may be for example to get access to sensitive data or disturb processes. Because APT attacks require ample resources and skills, their targets are so called high-value targets, for example governmental organisations. (Li et al., 2016; Kao, 2015; Zhao et al., 2015; Tankard, 2011)

For example, network of Ministry for Foreign Affairs of Finland has been compromised through an APT attack (Computer Emergency Readiness Team of Finland CERT-FI, 2013).

As stated earlier in a section 1.1, intrusion detection can be divided into two basic solutions: anomaly detection and misuse detection. Intrusion detection systems can also be divided according to the location: Network Intrusion Detection System (NIDS) and Host Intrusion Detection System (HIDS). Network intrusion detection systems are located in essential points of a certain network or segment of a network, where they monitor and analyse traffic for the benefit of intrusion detection. Host intrusion detection systems are located in a specific system or host, where they analyse the characteristics of e.g. incoming and outgoing packets, audit trails or software calls in terms of intrusion detection (Mokarian et al., 2013; Kaushik and Deshmukh, 2011; Gollmann, 2011; National Institute of Standards and Technology NIST, 2007; Tian et al., 2005; Zhou et al., 2003). There is one widely recognised problem with anomaly-based IDSs, namely that of generating a large amount of false positives (Mokarian et al., 2013; Gollmann, 2011; Kumar et al., 2011; Spathoulas and Katsikas, 2010; Tian et al., 2005). For example, a sudden burst of high volume service calls might be a legitimate attempt to use that service or an indication of an ongoing attack, and there is a high probability of alert in that case.

This dissertation focuses on the anomaly-detection-based network intrusion

detection system, which offers the near-real-time online sensor feed for situational awareness system in cyber security.

2.2 Realistic network traffic generation for testing the IDSs (PII)

In the article PII, presented and summarised here, the Internet data generation application is implemented and evaluated for testing the IDSs. Anomaly-based network intrusion detection system has a potential for detecting novel attacks. However, research and development of IDS requires a lot of testing with realistic network traffic data. The availability of such data sets is weak. Some public data sets are available, but often these are anonymized, there is lack of realistic statistical characteristics or there are no modern attack patterns available (Shiravi et al., 2012). Saber et al. (2012) has introduced research about scenarios of attacks when testing IDS, and Luo and Marin (2005) states that the main challenge for simulation environment in IDS testing is to have realistic Internet background traffic generation, because there is a requirement to generate attack-free legitimate Internet traffic and intentionally mix it with illegal attack traffic.

JAMK University of Applied Sciences has developed an Internet-like Cyber Range called Realistic Global Cyber Environment (RGCE). RGCE was developed to mimic the structures of real Internet. For example, RGCE Cyber Range contains realistic Internet Service Providers (ISPs) and numerous different web services as in the real Internet. RGCE is an isolated environment offering the possibility to use public IP addresses with real GeoIP locations and generate scenarios where attackers are scattered globally around various locations of the world. Isolation also allows the use of malicious software, attacks or vulnerabilities, without jeopardizing real networks or systems. (JAMK University of Applied Sciences, Institute of Information Technology, JYVSECTEC, s. a.)

Because RGCE is isolated from production networks and more widely isolated from the Internet there is a problem of how to conduct a scenario with mixed legitimate user traffic and attack traffic. For generating the traffic, there are two different approaches: trace-based replaying of traffic and analytic model-based generation of traffic (Botta et al., 2012; Hong and Wu, 2006). Realistic analytic model-based generation is hard because of the random variables of the traffic characteristics (Luo and Marin, 2005). Floyd and Paxson (2001) and Luo and Marin (2005) prefer source-level simulation where traffic is simulated on application level instead of packet level.

2.2.1 Requirements for traffic generation (PII)

As described in PII, the main objective is to generate Internet traffic data for RGCE Cyber Range by fulfilling the following requirements:

- system shall be capable for centralized control,

- system shall be capable for generating legitimate traffic according to a generated protocol,
- system shall be capable for generating verifiable traffic on OSI layers 3-6,
- system shall be capable for generating realistic traffic of various attacks,
- system shall be capable for generating traffic indistinguishable for a machine and human being,
- system shall be capable for generating traffic from globally scattered locations and IP addresses,
- system shall be capable for model-based traffic generation, not trace-based replaying,
- system shall be capable for generating traffic applicable with existing servers and services, e.g. generation of requests to a specified HTTP server shall have variable HTTP headers and variable intervals,
- system shall be capable for autonomous traffic generation for long time periods and with an automatic error recovery capability.

To achieve the capability that fulfils the above-mentioned requirements, several existing solutions and different approaches were analysed. The overall result was to implement our own Internet data generation application that simulates clients. The chosen approach requires substantial exertion for create an application that can generate traffic from multiple clients with multiple protocols while allowing generation of a specific kind of traffic, for example broken TCP traffic, from certain clients like in the real Internet.

2.2.2 Implemented solution (PII)

A hierarchical tree-like network structure, bot-net, is implemented. Traffic-generating bots can be globally scattered within RGCE Cyber Range while generating traffic by using various services and protocols, each capable of acting according to a different profile. For example HTTP-protocol profiles are employed by following traffic generations: users browsing in the Internet, Slowloris DoS attack, Slowpost DoS attack, repeatedly HTTP basic authentication and volumetric DoS with continuous HTTP requests.

The implementation of the Internet traffic generation application is done using the Go programming language for GNU/Linux platform with the HMI as shown in Figure 2. The scalability of the implemented solution was evaluated in three different networks: Localhost, Local Area Network (LAN) and Internet. The Internet scenario was utilized in real Internet between a server in Netherlands and clients in Finland. The LAN scenario was carried out within the JAMK University of Applied Sciences.

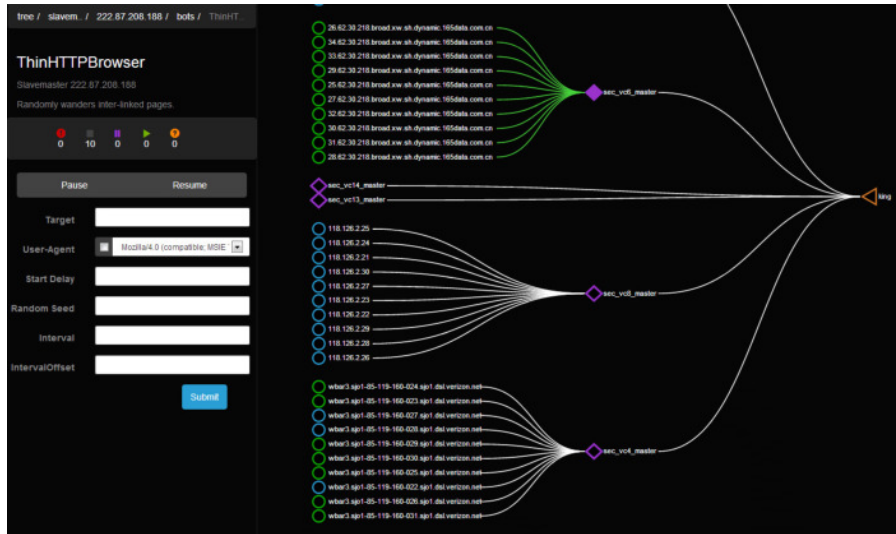


FIGURE 2 HMI of Internet data generation application (PII).

A constructed system was used as a traffic generator for several articles included to this dissertation. All except one of the anomaly-detection-based intrusion detection system demonstrations in this dissertation that were conducted using simulated network data utilized the implemented Internet data generation application. It is also worth of mentioning that the application developed is used as a main component of Internet data generation in the National Cyber Exercise (The Finnish Defence Forces, 2016).

3 ANOMALY-DETECTION-BASED IDS WITH CAPTURED NETWORK DATA

This chapter presents research and development of anomaly-detection-based intrusion detection system. First the analysis of HTTP logs is described for the detection of network intrusions. Mukkamala and Sung (2003) regards overhead as one of the main problems of the IDSs. In this study, the HTTP log analysis is done afterwards from the HTTP logs with a delay. This is considered here as the first step towards online analysis with lower overhead. As the next step, the captured network traffic is analysed for identifying intrusions. Analysis of anomalous network flows and analysis of encrypted network traffic, the latter being extremely important feature of modern network traffic and modern network-based intrusions, are considered. Both of those analysis were done afterwards using captured incoming and outgoing network data.

3.1 Analysing HTTP requests (PI)

Web servers offer wide variety of most common services in the Internet. Clients request and send information to web servers with the help of HTTP requests. If an attacker is capable of producing or manipulating these string-based queries or parameters of these queries as applicable requests, there is a possibility to compromise the server to get access for interacting with its resources. (Kumar and Pateriya, 2013; Nguyen-Tuong et al., 2005). In article PI, presented and summarised here, the model for analysis of HTTP requests for anomaly detection is implemented and evaluated using realistic test environment.

3.1.1 Feature extraction (PI)

HTTP logs can include lot of information about client and client interaction with the web server. According to The Apache Software Foundation (s. a.), a combined format of Apache server log can be used as follows:

```
LogFormat "%h %l %u %t \"%r\" %>s %b
\"%{Referer}i\" \"%{User-agent}i\" combined
```

This produces the following example log (The Apache Software Foundation, s. a.),

```
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700]
```

```
"GET /apache_pb.gif HTTP/1.0" 200 2326
"http://www.example.com/start.html"
"Mozilla/4.08 [en] (Win98; I ;Nav) "
```

indicating the requesting client's IP address, the client's identity (RFC 1413), user id as determined by HTTP authentication, time stamp of the request received, the client's request line, the server status code to the client, returned object's size, referrer site that the client reports being referred from, and User-Agent header as the identifying information by the browser.

When detecting attacks from HTTP logs, two types of statistics are analysed: statistics based on requests and statistic based on time intervals.

3.1.1.1 Statistics based on requests (PI)

A n -gram model is used for the following parameters of lines in the HTTP log: web resource requested, query attributes of the request and user agent. N -gram is a sequence's sub-sequence of n overlapping items. N -gram models are used for analysing texts and languages for example for speech recognition (Hirsimaki et al., 2009), handwriting recognition (Schall et al., 2016) and understanding of natural language (Suen, 1979). Non-alphanumeric symbol combinations are a subject of interest when analysing injections of code. Thus in the model used, letters and numbers are considered equal for focusing on the non-alphanumeric symbols and allowing shrinking the dimensionality without losing essential information. Feature vectors from each of the three textual parameters are assembled using unique n -grams and the number of all possible n -grams included in the training set determines the size of each feature vector.

- Model 1: the i -th feature vector's j -th feature x_{ij} equals the i -th line's j -th n -gram's occurrence frequency f_{ij}^n :

$$x_{ij} = f_{ij}^n. \quad (1)$$

- Model 2: the i -th feature vector's j -th feature x_{ij} equals the ratio between i -th line's j -th n -gram's occurrence frequency f_{ij}^n and the j -th n -gram in the whole training set occurrence frequency F_j^n :

$$x_{ij} = \frac{f_{ij}^n}{F_j^n}. \quad (2)$$

- Model 3: the i -th feature vector's j -th feature x_{ij} equals the ratio between i -th line's j -th n -gram's occurrence frequency f_{ij}^n and the product of symbol frequencies contained in current n -gram for the whole training set, where

j -th n-gram's k -th symbol's occurrence frequency is F_{jk}^1 :

$$x_{ij} = \frac{f_{ij}^n}{\prod_{k=1}^n F_{jk}^1}. \quad (3)$$

3.1.1.2 Statistics based on time intervals (PI)

Time period of HTTP logs under analysis is divided into overlapping time intervals, and from each of those time intervals the following parameters are extracted: IP address of the client, web resource requested with query attributes, server status code and returned object's size. Also the number of HTTP requests in time interval is considered.

For measuring the uncertainty of the parameters' distribution, the entropy is calculated (Ozcelik and Brooks, 2016). For each time interval, a sample entropy of every extracted parameter is produced and the entropy value matrix is used as an extracted feature matrix. The j -th parameter of the i -th time interval has N_{ij} unique values occurring with frequencies $p_{ij}^1, \dots, p_{ij}^{N_{ij}}$. Sample entropy E_{ij} for the j -th parameter of the i -th time interval:

$$E_{ij} = - \sum_{k=1}^{N_{ij}} p_{ij}^k \log_2 p_{ij}^k. \quad (4)$$

If all values of the j -th parameter are the same, sample entropy E_{ij} is zero and if $p_{ij}^1 = p_{ij}^2 = \dots = p_{ij}^{N_{ij}}$, sample entropy E_{ij} has a maximal value $\log_2 N_{ij}$.

3.1.2 Analysis (PI)

The feature vectors of normal users' behaviour model are constructed using HTTP logs of legitimate user traffic generated during the training period.

3.1.2.1 Analysis of requested web resource (PI)

Principal Component Analysis (PCA) is used for reducing the dimensionality of web resources feature vector. PCA is a widely employed transformation technique used for example for feature extraction, data compression and characterisation in pattern recognition, image processing and signal processing (Karhunen and Joutsensalo, 1995). PCA converts data to a new coordinate system. Here, in this model, the transformation is done using the principal components corresponding to the covariance matrix's non-zero eigenvalues. The transformed vectors of the training set are analysed using Support Vector Data Description (SVDD). SVDD constructs a hypersphere around the data belonging to one category (Tax and Duin, 2004).

Requested web resources of the training set corresponds to the q number of transformed feature vectors x_1, x_2, \dots, x_q . There exists SVDD hypersphere (c, R)

with radius $R > 0$ and centre $c = \sum_{i=1}^q \alpha_i x_i$, where parameter $\alpha = (\alpha_1, \dots, \alpha_q)$ can be defined using the optimisation:

$$\begin{aligned} & \max_{\alpha} \sum_{i=1}^q \alpha_i (\phi(x_i)^T \phi(x_i)) - \sum_{i=1}^q \sum_{j=1}^q \alpha_i \alpha_j \phi(x_i)^T \phi(x_j), \\ & \text{subject to } \begin{cases} \sum_{i=1}^q \alpha_i = 1, \\ 0 \leq \alpha_i \leq C, \forall i \in \{1, \dots, q\}, \end{cases} \end{aligned} \quad (5)$$

where C is the parameter controlling the balance of the hypersphere volume and classification errors and function $\phi(x)$ maps x to a higher-dimensional space. Here, in this analysis, parameter C is defined to be one and $\phi(x) = x$. For any vector x_k from a dataset where $\alpha_k < C$, radius R of SVDD hypersphere (c, R) can be defined as:

$$\begin{aligned} R^2 = & \phi(x_k)^T \phi(x_k) - 2 \sum_{i:\alpha_i < C} \alpha_i \phi(x_i)^T \phi(x_k) + \\ & \sum_{i:\alpha_i < C} \sum_{j:\alpha_j < C} \alpha_i \alpha_j \phi(x_i)^T \phi(x_j). \end{aligned} \quad (6)$$

After the hypersphere is optimised for the training set, the HTTP request's feature vector x under analysis is labelled anomalous if:

$$\begin{aligned} R^2 - & (\phi(x)^T \phi(x) - 2 \sum_{i=1}^q \alpha_i (\phi(x)^T \phi(x_i)) + \\ & \sum_{i=1}^q \sum_{j=1}^q \alpha_i \alpha_j \phi(x_i)^T \phi(x_j)) < 0. \end{aligned} \quad (7)$$

3.1.2.2 Analysis of query attribute values (PI)

Iterative refinement technique is used for the k-means clustering algorithm for anomaly detection of query attribute values. This technique was proposed by Lloyd for pulse-code modulation already in 1957 and published in 1982 (Lloyd, 1982). K-means allocates instances to k clusters, where k is a predefined number. At the beginning, a set of k instances are e.g. randomly set as centres of the clusters. After that, every instance of data set is assigned to the nearest cluster and new cluster centres are calculated. The process is iterated until there are no more changes. (Syarif et al., 2012; Munz et al., 2007)

K-means is used to find the clusters of the query attributes of the training set. Let there be vector x corresponding to the query attribute of the HTTP request. Let $d(x_i, x_j)$ be the distance function for two feature vectors x_i and x_j and C_i is the i -th cluster. Centre of C_i corresponds to the mean value m_i of the vectors in C_i . Radius r_i is defined as follows:

$$r_i = \frac{|C_i| + 1}{|C_i|} \cdot \max_{x \in C_i} d(m_i, x). \quad (8)$$

If $d(x, m_i) > r_i, \forall i \in \{1, 2, \dots, k\}$, vector x is labelled anomalous because it does not belong to any of the training set clusters (m_i, r_i) , and the whole request is considered as an attack if any attribute of an HTTP request is labelled anomalous.

3.1.2.3 Analysis of user agent values (PI)

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is used for anomaly detection of user agent values. For creating clusters, DBSCAN retrieves all points which are density-reachable from arbitrary starting point (Ester et al. (1996)). For DBSCAN, parameters ε (size of ε -neighbourhood) and N_{min} (the minimum number of cluster points) are defined before the algorithm starts with randomly selected feature vector x . To check, whether x is in a cluster or considered as noise, is done by comparing the number of feature vectors $N_\varepsilon(x)$ in the ε -neighbourhood of x and N_{min} as follows:

$$\begin{cases} \text{If } N_\varepsilon(x) < N_{min}, \text{ then } x \text{ is considered as noise,} \\ \text{If } N_\varepsilon(x) \geq N_{min}, \text{ then } x \text{ is in cluster.} \end{cases} \quad (9)$$

Vectors considered as noise in this comparison might later be in the ε -environment of another vector and a part of a cluster. If vector x is a part of a cluster, also its ε -neighbourhood is part of that cluster. Density-reachability of all y of that ε -neighbourhood is checked: y is density-reached from x and y is added to the cluster if there exists a chain of points x_1, x_2, \dots, x_m , where $x_1 = x$ and $x_m = y$, such that $\forall i \in \{1, 2, \dots, m-1\}$ the Euclidean distance $d^E(x_i, x_{i+1})$ between x_i and x_{i+1} is smaller than predefined size of ε -neighbourhood $d^E(x_i, x_{i+1}) \leq \varepsilon$ and $N_\varepsilon(x_i) \geq N_{min}$. After that, the new vector is analysed and finally all the remaining points without a cluster are considered as anomalies. Basic equation for calculation of Euclidean distance is presented for example in the studies by Bouhmala (2016) and Munz et al. (2007).

In this study, the size of the ε -neighbourhood is determined as the maximal pair distance between feature vectors $\varepsilon = \max_{x_i, x_j \in X} d^E(x_i, x_j)$ of the training set X . By considering that ε , the minimum number of cluster points N_{min} is determined as a minimal value of neighbours of each training set's feature vector $N_{min} = \min_{x \in X} N_\varepsilon(x)$.

The normal user behaviour model, according to the training set, is created using all the feature vectors extracted from user agent values of the training set. When analysing the user agent of a new HTTP request, DBSCAN is applied to the new feature vector of HTTP request under analysis and all the feature vectors from the training set. If the new feature vector is without cluster it is labelled as anomaly.

3.1.2.4 Analysis of time intervals (PI)

Training set's entropy vectors are standardized by z-score normalisation, also known as zero mean normalisation, where the data normalisation is done based on the mean and standard deviation of the data (Saranya and Manikandan, 2013).

Mean value, μ_j , of j -th entropy values and standard deviation, σ_j , of j -th entropy values are applied for the normalisation of entropy values E_{ij} for the j -th parameter of the i -th time interval:

$$E_{ij}^z = \frac{E_{ij} - \mu_j}{\sigma_j}. \quad (10)$$

With a new time interval, an entropy vector is extracted and normalized using the training set's mean value and standard deviation. Anomaly detection is done by comparing the euclidean norm of the vectors. Normalized vector e^z of HTTP request in the new time interval is classified as an anomaly if

$$\|e^z\|_2 > \max_{i \in T} \|E_i^z\|_2, \quad (11)$$

where T is the set of training phase time intervals.

3.1.3 Testing the model (PI)

The implemented model was tested using RGCE Cyber Range and Internet traffic generation application described in section 2.2. In the RGCE Cyber Range, there was a web server with known vulnerabilities installed, and during a five-day time period there was legitimate user traffic mixed with traffic of intrusions. During the first day, only legitimate traffic was generated for training purposes. The next four days included legitimate user traffic mixed with illegal attack traffic, e.g. DoS/DDoS attacks, bruteforce attacks, scanning attacks and several targeted attacks using known vulnerabilities in the server software. The HTTP logs of the web server were gathered and analysed using the implemented model.

For the evaluation of the performance metrics of the model, the following characteristics of anomaly detection are defined (Mokarian et al., 2013; Fawcett, 2006):

- *True Positives, TP* - the number of anomaly instances correctly classified as anomaly,
- *True Negatives, TN* - the number of normal instances correctly classified as normal,
- *False Positives, FP* - the number of normal instances incorrectly classified as anomaly,
- *False Negatives, FN* - the number of anomaly instances incorrectly classified as normal,
- *True Positive Rate (also called Recall), TPR* = $\frac{TP}{TP + FN}$ - the ratio between TP and the total number of anomaly instances,
- *False Positive Rate, FPR* = $\frac{FP}{TN + FP}$ - the ratio between FP and the total number of normal instances,
- *Detection Accuracy* = $\frac{TN + TP}{TN + FP + TP + FN}$ - the ratio between correctly classified instances and the total number of instances,

- Receiver Operating Characteristic curve, ROC curve - curve that is plotting the TPR against the FPR.

For the anomaly detection of the requested web resource, SVDD is performed as described in section 3.1.2.1. Feature vectors are extracted applying models introduced in equation 1, equation 2 and equation 3, with n-gram modes 1-gram, 2-gram, and 3-gram (Figure 3). With carefully chosen best parameters, the introduced model detects intrusion performance metrics of TPR (97.96%), FPR (0.52%) and Detection Accuracy (99.20%).

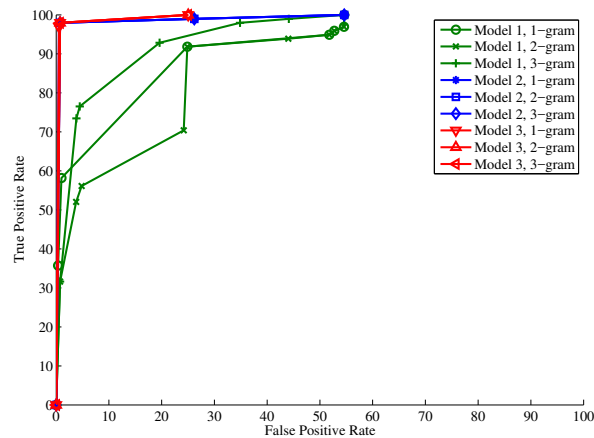


FIGURE 3 ROC curves of anomalous web resources by SVDD (PI).

For the anomaly detection of the query attribute values, the k-means is performed as described in section 3.1.2.2. Feature vectors are extracted by applying the models introduced in equation 1, equation 2 and equation 3, with n-gram modes 1-gram, 2-gram, and 3-gram (Figure 4). The introduced model is capable for detecting all intrusions with 1-gram and models described with equation 2 and equation 3.

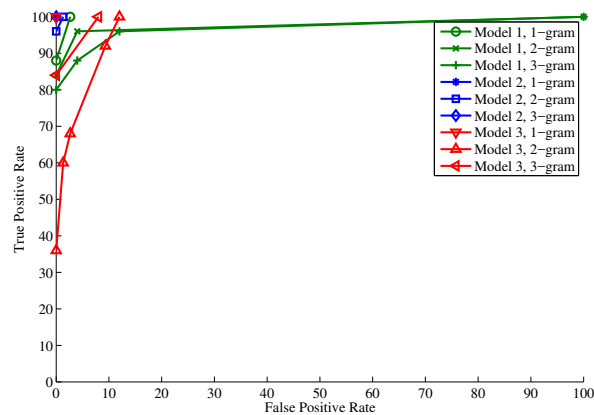


FIGURE 4 ROC curves of anomalous query attributes by k-means (PI).

For the anomaly detection of the user agent values, DBSCAN is performed as described in section 3.1.2.3. Feature vectors are extracted by applying the mod-

els introduced in equation 1, equation 2 and equation 3, with n-gram modes 1-gram, 2-gram, and 3-gram (Figure 5). Using the introduced IDS model, anomalous user agent values are detected with the Detection Accuracy of 97.5% when using 1-gram with equation 2.

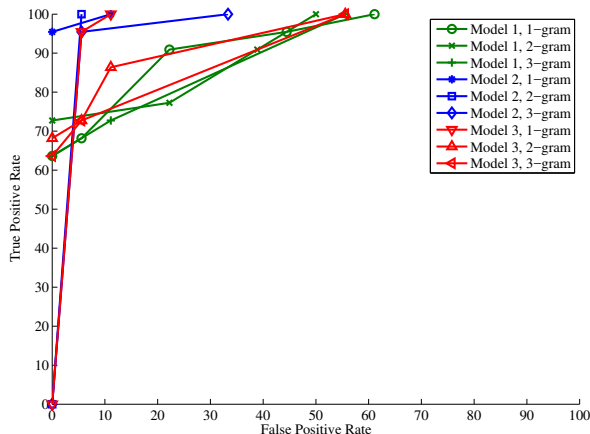


FIGURE 5 ROC curves of anomalous user agents by DBSCAN (PI).

Also time series consideration is applied as described in section 3.1.2.4. Figure 6 shows the normalized entropy values during the first and the fourth day. During the fourth day, there are intrusions mixed with legitimate traffic (Table 1). If the size of the time window is big enough, all intrusions are detected (Table 2).

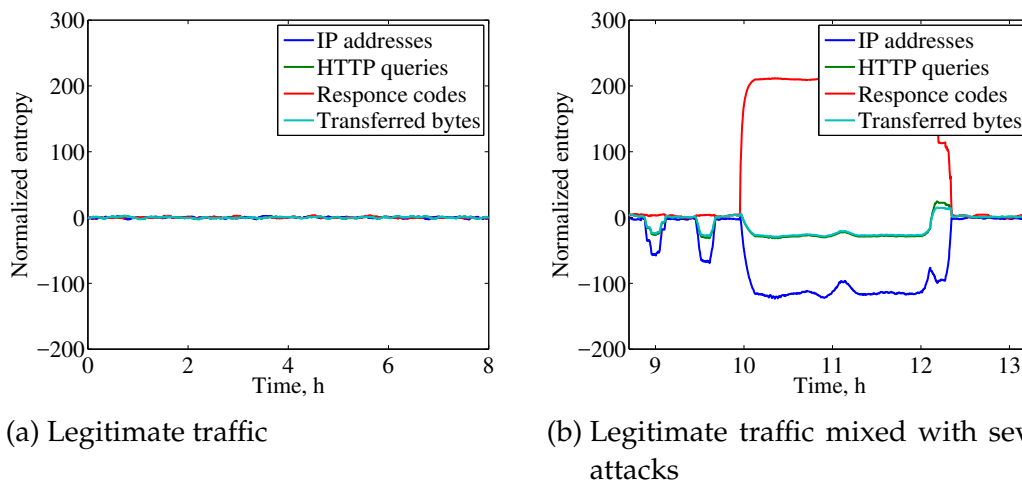


FIGURE 6 Normalized entropy values with 60-second time window and 10-second time steps (PI).

The introduced model for the analysis of HTTP requests' log data, demonstrates that there are potential for research and development of online IDS. It was also shown that RGCE Cyber Range and Internet data generation application are suitable for IDS research and development. The next step for the implementation of IDS is focusing on network flows instead of server log level or network packet level considerations.

TABLE 1 Day four, scans and bruteforce attacks (PI).

Attack	Time
Password bruteforce by Hydra	08:57 - 09:11
Password bruteforce by Hydra	09:28 - 09:41
Database takeover by Sqlmap	09:58 - 12:21
Active scan by Owasp	12:07 - 12:21
Forced browse by Owasp (dirbuster)	13.06 - 13:14

TABLE 2 Detection Accuracy of the time interval analysis applying the entropy-based method (PI).

Time window	Time step	Attacks detected	Detection Accuracy in the time step
10 minutes	1 minute	5 of 5	98.35 %
10 minutes	10 seconds	5 of 5	98.76 %
10 minutes	1 second	5 of 5	98.29 %
1 minute	10 seconds	5 of 5	99.24 %
1 minute	1 second	5 of 5	98.70 %

3.2 Analysing network flows (PIV)

In article PIV, presented and summarised here, the model for flow-based analysis of internet traffic for anomaly detection is implemented and evaluated using realistic test environment. There are lot of research interests for flow-based anomaly detection. Elrawy et al. (2016) introduces a flow-based IDS for mobile traffic, whereas AlEroud and Karabatis (2016) proposes an approach for flow-based intrusion detection. Dong et al. (2016) introduces flow-based detection of DDoS against Software-Defined Networking (SDN) controllers. Saboor and Aslam (2015) analyses and evaluates flow-based DDoS detection systems and Sheikhan and Jadidi (2014) presents flow-based anomaly detection in high-speed networks.

3.2.1 Feature extraction (PIV)

In the proposed method, network traffic is recognized as a time series and analysed period of time $[T_s, T_e]$ is allocated to overlapping time intervals with length of ΔT with points $T_s + \frac{t}{w}\Delta T$, where $t = \{w, w + 1, \dots, w\frac{T_e - T_s}{\Delta T} - 1\}$. The intrusion detection is applied based on the analysis of network flows. Flow, as defined in RFC3917 (The Internet Society Network Working Group, 2004), is in a specified time interval, a group of IP packets with some common properties passing a monitoring point. Here those common properties are assumed to be source and destination IP addresses and ports, and time interval is the above-defined time interval with length ΔT . To simplify and compile network data to be anal-

used, packets with the same source and destination IP addresses and ports are measured as a flow. At each time interval with length ΔT , following per-flow information is extracted:

1. Source IP address ($I_i^{src,t}$)
2. Source port ($P_i^{src,t}$)
3. Destination IP address ($I_i^{dst,t}$)
4. Destination port ($P_i^{dst,t}$)
5. Number of packets (N_i^t)
6. Maximal packet size ($B_i^{max,t}$)
7. Minimal packet size ($B_i^{min,t}$)
8. Average packet size ($B_i^{avg,t}$)
9. Flow duration (D_i^t)

Total amount of flows n^t in the t -th time interval is also required for analysis. Extracted features from the i -th flow at the t -th time interval are compiled as vectors x_i^t , y_i^t , and f_i^t , where

$$\begin{aligned} x_i^t &= (I_i^{src,t}, P_i^{src,t}, I_i^{dst,t}, P_i^{dst,t}), \\ y_i^t &= (N_i^t, B_i^{max,t}, B_i^{min,t}, B_i^{avg,t}, D_i^t), \\ f_i^t &= (x_i^t, y_i^t). \end{aligned} \quad (12)$$

3.2.2 Analysis (PIV)

As earlier, the introduced model requires a training set of data with only legitimate traffic included for the construction of normal user behaviour. Network flows included in all time intervals are clustered with soft clustering approach and cluster distributions are analysed for finding anomalous time intervals. Traffic included in those anomalous time intervals is analysed for finding intrusive flows.

3.2.2.1 Clustering flows (PIV)

When using soft clustering, each data sample is associated with a set of membership level that indicates the probability of a data sample belonging to a cluster. The clustering technique here is based on the Expectation Maximisation - algorithm (EM) (Syarif et al., 2012; Lu and Tong, 2009; Bilmes, 1998). Distribution of feature vectors $Y^t = \{y_i^t\}_{i=1}^{n^t}$ are according to the probability density function $p(y_i^t|\Theta^t)$ of distribution parameters $\Theta^t = (\alpha_1^t, \dots, \alpha_M^t, \mu_1^t, \dots, \mu_M^t, \Sigma_1^t, \dots, \Sigma_M^t)$:

$$p(y_i^t|\Theta^t) = \sum_{k=1}^M \alpha_k^t p_k^t(y_i^t|\mu_k^t, \Sigma_k^t), \quad (13)$$

where Bayesian Information Criterion (Fraley and Raftery, 1998) can be used for number of Gaussians M , $\alpha_1^t, \dots, \alpha_M^t$ are parameters where $\sum_{k=1}^M \alpha_k^t = 1$, $p_k^t(y_i^t|\mu_k^t, \Sigma_k^t)$

is the density function of the k -th Gaussian using covariance matrix Σ_k^t and mean value μ_k^t .

Iterative expectation and maximisation steps are used for finding appropriate values of distribution parameters Θ^t and Expectation Maximisation -algorithm defines the probability $p(k, y_i^t, \Theta^t)$ for data sample y_i^t belonging to the k -th distribution according to parameters Θ^t . Probability P_{ij}^t that flow $f_i^t = (x_i^t, y_i^t)$ and flow $f_j^t = (x_j^t, y_j^t)$ are in the same cluster, where $d_H(x_i^t, x_j^t)$ is the Hamming distance of x_i^t and x_j^t , is as follows:

$$P_{ij}^t = (1 - d_H(x_i^t, x_j^t)) \sum_{k=1}^M p(k, y_i^t, \Theta^t) \times p(k, y_j^t, \Theta^t). \quad (14)$$

Hamming distance is introduced by Hamming (1950), and it defines the number of elements where two inputs of the distance function are different.

Hierarchical clustering method namely a single-linkage clustering algorithm (Rafsanjani et al., 2012; Laskov et al., 2005) is used for achieving n_C number of clusters for all the time intervals. When algorithm starts, each vector forms a cluster. Until the defined n_C number of clusters is achieved, the clusters C_l and C_k with the shortest distance $d(C_l, C_k)$ are combined iteratively, where distance $d(C_l, C_k)$ is:

$$d(C_l, C_k) = \min_{f_i^t \in C_l, f_j^t \in C_k} (P_{ij}^t). \quad (15)$$

3.2.2.2 Finding anomalous time intervals (PIV)

Flows of every time interval t in the training set Ω are clustered, and for each cluster, the number of feature vectors is defined. The histogram vectors $h^t = (h_1^t, h_2^t, \dots, h_{n_C}^t)$ consisting, in descending order, numbers of feature vectors in clusters are used for constructing matrix H with rows $H_t = (h_1^t, h_2^t, \dots, h_{n_C}^t)$.

The Mahalanobis distance measures the number of standard deviations that certain point is away from mean of the cluster (Li et al., 2016; Xie et al., 2014). The Mahalanobis distance $d_M(H_t, H)$ between the t -th row H_t and matrix H is defined for each row H_t :

$$d_M(H_t, H) = \sqrt{(H_t - \mu^H)^T C^H (H_t - \mu^H)}, \quad (16)$$

where $\mu^H = (\mu_1^H, \mu_1^H, \dots, \mu_{n_C}^H)$ is the mean of H and C^H is the covariance matrix of H .

In the training set, there is $|\Omega|$ amount of time intervals. For the anomaly detection of the new time interval, the feature vectors are extracted and clustered for achieving histogram vector h . Mahalanobis distance $d_M(h, H)$ between histogram vector h and training matrix H is defined. Histogram vector h is classified as an anomaly if

$$d_M(h, H) > \frac{|\Omega| + 1}{|\Omega|} \max_{t \in \Omega} d_M(H_t, H). \quad (17)$$

3.2.2.3 Finding intrusive flows (PIV)

Clusters $\{c_1, \dots, c_{n_C}\}$ found from anomalous time interval and clusters from the training phase are compared for finding intrusive flows. The mean values of y_i^t , $i \in C_k^t$ are defined as the centre of cluster C_k^t . Distance $d(C_j^t, C_k^t)$ is the distance between the cluster centre of cluster C_j^t and the cluster centre of cluster C_k^t . Number of flows in cluster c_j is $|c_j|$, in which case the flows of cluster c_j are labelled intrusive if any of following is true:

$$\begin{aligned} \min_{t \in \Omega, 1 \leq k \leq n_C} d(c_j, C_k^t) &> \frac{|\Omega| + 1}{|\Omega|} \max_{\tau, t \in \Omega, 1 \leq l, k \leq n_C} d(C_l^\tau, C_k^t), \\ |c_j| &> \frac{|\Omega| + 1}{|\Omega|} \max_{t \in \Omega} |C_{k^{t*}}^t|, \quad k^{t*} = \arg \min_{1 \leq k \leq n_C} d(c_j, C_k^t). \end{aligned} \quad (18)$$

3.2.3 Testing the model (PIV)

The implemented model was tested using the RGCE Cyber Range and Internet traffic generation application described in section 2.2. In the RGCE Cyber Range, there was a web server with known vulnerabilities installed, and during a four-day time period there was legitimate user traffic mixed with anomalous attack traffic. During the first day, there was only legitimate traffic, and the rest of the period had legitimate user traffic mixed with illegal attack traffic including brute-force attempts, illegal scanning, and targeted attacks. For the numerical parameters of the model, one-minute time interval ΔT was selected, and parameter w was set to 60. TPR, FPR and Detection Accuracy were defined for estimating the performance of the proposed model.

The proposed method was tested with different values of parameter n_C (Figure 7). All the values exhibited good detection accuracy: if the value of n_C is between 10 and 15, the Detection Accuracy is better than 99% with the used data set.

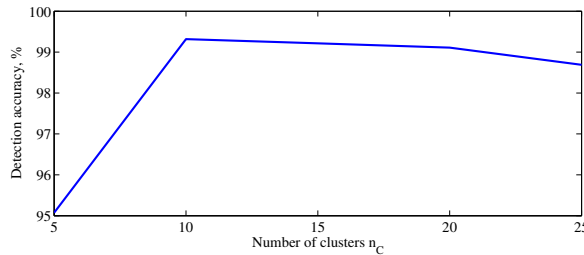


FIGURE 7 Detection Accuracy with different n_C values (PIV).

First phase of the proposed method, with n_C value of 10, was also compared against other well-known flow-based IDS techniques such as Flow Anomaly Detection System (FADS) (Chapple et al., 2006), Exponentially Weighted Moving Average (EWMA) (Cisar et al., 2010) and Chi-square Detection Mechanism (Muralidharan et al., 2010) (Figure 8 and Table 3).

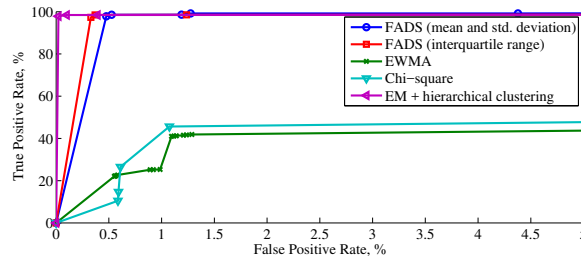


FIGURE 8 ROC curves of compared algorithms (PIV).

TABLE 3 Performance metrics of compared algorithms (PIV).

Method	TPR	FPR	Detection Accuracy
FADS (mean and standard deviation)	97.84 %	0.48 %	99.10 %
FADS (interquartile range)	97.31 %	0.33 %	99.08 %
EWMA	22.17 %	0.55 %	79.94 %
Chi-square	14.65 %	0.59 %	76.99 %
EM + hierarchical clustering	98.39 %	0.39 %	99.32 %

The second part of the proposed method for detecting intrusive flows was tested, and also compared against other well-known clustering techniques, achieving the best Detection Accuracy (99.88%). Other clustering techniques were: single-linkage clustering (see section 3.2.2.1), k-means (see section 3.1.2.2), K-Nearest Neighbors (KNN) (Farooqi and Munir, 2008) and Self Organizing Map (SOM) (Kohonen, 1990).

As a result, the performance of the proposed method was good, but the model could not detect password brute force attempts when the number of the attempts included in the time interval was low. The proposed method performed with a higher rate of detection accuracy than compared well-known anomaly detection methods. The proposed model is one step forward for the implementation of online intrusion detection capability with encrypted traffic.

3.3 Analysing encrypted traffic, first model (PIII)

Various systems are nowadays using encrypted traffic, so intrusion detection cannot be done by analysing packet payloads. According to Stevanovic and Vlajic (2014), HTTP-based DDoS attacks can be divided into three main categories: trivial (stand-alone HTTP requests from each bot), intermediate (semi-random HTTP requests from each bot that appear to origin from a legitimate web browser) and advanced (HTTP requests from each bot that appear to origin from a legitimate web browser and mimic sequences of human interaction). In article PIII, presented and summarised here, the model for analysis of encrypted network traffic

for anomaly detection is implemented and evaluated using realistic test environment. The presented algorithm can be divided into two steps: anomaly detection for finding time intervals during attacks, and detection of attack flows during those time intervals.

3.3.1 Feature extraction (PIII)

As in section 3.2.1, network traffic is recognized as a time series, and analysed period of time $[T_s, T_e]$ is allocated into overlapping time intervals with length of ΔT with points $T_s + \frac{t}{w}\Delta T$, where $t = \{w, w + 1, \dots, w\frac{T_e - T_s}{\Delta T} - 1\}$. The intrusion detection is applied based on the analysis of network flows with common properties as source IP address and port and destination IP address and port. The length of time interval ΔT should be big enough for containing suitable information for anomaly detection and parameter w should be adequate for detecting attacks early enough.

For finding intrusive time intervals, following features are calculated for each time interval, where the entropy can be calculated according to equation 4:

1. Sample entropy of source IP address,
2. Sample entropy of source port,
3. Sample entropy of destination IP address,
4. Sample entropy of destination port,
5. Total number of flows,
6. Average duration of flow,
7. Average number of packets in one flow from source to destination,
8. Average number of packets in one flow from destination to source,
9. Average size of packets from source to destination,
10. Average size of packets from destination to source,
11. Average size of TCP window for packets from source to destination,
12. Average size of TCP window for packets from destination to source.

Element x_i^t of an extracted feature vector is the t -th time interval's i -th feature value.

For finding intrusive flows, the following information is extracted separately from source to destination and from destination to source for each flow:

1. Size of packets (average, minimal and maximal),
2. Size of TCP window (average, minimal and maximal),
3. Time from the previous packet (average, minimal and maximal),
4. Time to live (average, minimal and maximal),
5. Percentage of packets with TCP flag SYN,
6. Percentage of packets with TCP flag ACK,
7. Percentage of packets with TCP flag PSH,
8. Percentage of packets with TCP flag RST,
9. Percentage of packets with TCP flag FIN.

Elements of the extracted feature vector y_i of length $n = 34$ consist of feature values of the i -th flow normalized for range $[0, 1]$ applying the max-min normalisation.

3.3.2 Analysis (PIII)

Similarly as in the section 3.2, the model for detecting anomalous time intervals is applied and secondly the intrusive flows are detected.

3.3.2.1 Finding anomalous time intervals (PIII)

Chi-square values are used for calculating the model of legitimate normal user behaviour. Chi-square test is used to determine the difference between the distribution's expected values and observed values (Vardasbi et al., 2011; Muraleedharan et al., 2010; Ye et al., 2003). For the n_x number of features, the t -th time interval's chi-square χ_t^2 can be defined as:

$$\chi_t^2 = \sum_{i=1}^{n_x} \frac{(x_i^t - \mu_i)^2}{\mu_i}, \quad (19)$$

where x_i^t is the t -th time interval's i -th feature value and μ_i is the i -th feature's mean value in $[T_s, T_e]$. Distance $d(\chi_{t_1}^2, \chi_{t_2}^2)$ between two patterns can be calculated using probability $p(x \text{ is normal})$, which determines the probability value for x being normal (Corona and Giacinto, 2010)

$$d(\chi_{t_1}^2, \chi_{t_2}^2) = p(\chi_{t_1}^2 \text{ is normal}) - p(\chi_{t_2}^2 \text{ is normal}). \quad (20)$$

Similarly to section 3.2.2.1, a single-linkage clustering algorithm (Rafsanjani et al., 2012; Laskov et al., 2005) is used. A single-linkage clustering algorithm is applied for dividing chi-square values into two clusters, cluster of normal values C_n and cluster of outlier values C_o , using distance d , where the minimum distance between two chi-square values of clusters defines the distance between clusters. Now all the outliers can be removed, and only normal values are used for anomaly detection. According to the Central Limit Theorem (CLT), the χ^2 statistic of a sequences of independent signals approximately follow a normal distribution (Dehay et al., 2013), and, according to the classical 68-95-99.7, rule, approximately 99.7% of all values from normal distribution are within three standard deviations σ from mean value μ (Gallego et al., 2013). At the current time interval, network traffic is considered anomalous if

$$\chi^2 > \bar{\mu}_{\chi^2} + \alpha \bar{\sigma}_{\chi^2}, \quad (21)$$

where $\bar{\mu}_{\chi^2}$ is the mean of chi-square values and $\bar{\sigma}_{\chi^2}$ is the standard deviation of chi-square values from cluster of normal values C_n , and parameter $\alpha \geq 3$.

3.3.2.2 Finding intrusive flows (PIII)

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is used as in section 3.1.2.3. Once the feature vectors of the flows of the legitimate training

set are extracted, normalised and clustered. The maximal pairwise Euclidean distance, m_i , is defined for the i -th cluster C_i :

$$m_i = \max_{z_j, z_k \in C_i} d(z_j, z_k). \quad (22)$$

In a case of each cluster-less point, m_i is assigned as minimal $m_{i, \min}$ of all clusters' elements.

In a current time interval, for the intrusion detection of a new flow, all the earlier-mentioned features of the flow are extracted to the feature vector z , and the cluster or cluster-less point, with least distant vector from z is discovered:

$$i^* = \arg \min_{z_j \in C_i} d(z, z_j). \quad (23)$$

If the distance between vector z under analysis and the found least distant vector is bigger than the maximal pairwise distance, m_{i^*} , of that cluster, then the current flow is labelled as an intrusion:

$$\min_{z_j \in C_{i^*}} (d(z, z_j)) > m_{i^*}. \quad (24)$$

3.3.3 Testing the model (PIII)

Tests were conducted in the RGCE Cyber Range with Internet data generation software as described in section 2.2. As part of the RGCE, there is a web server with a static main page through HTTPS (SSL/TLS) implemented. Mixed legitimate traffic and DoS/DDoS traffic was generated to that web server. All the traffic was captured as PCAP-files (Packet CAPture) containing mainly HTTPS -traffic, a minimal amount of HTTP -traffic are being made of client-server handshakes before the creation of encrypted channel. The analysed PCAP-file contained 429202 traffic flows with a total length of 80 minutes. The training set with only legitimate traffic was 8 minutes long. ΔT was assigned to five seconds and parameter w to 5 as a result of iterative testing for achieving reasonable detection capability.

For plotting the behaviour of chi-square values based on seconds, features were extracted from the data set and chi-square values were calculated (Figure 9). For finding the time intervals with anomalous traffic, parameter α of equation 21 was assigned to 5.

For evaluation of the performance of introduced method TPR, FPR and Detection Accuracy were defined and ROC curves plotted. Performance of DBSCAN was compared against other well-known clustering techniques, which are already introduced in this thesis, with different parameters: K-means, K-Nearest Neighbours (KNN), Support Vector Data Description (SVDD) and Self-Organizing Map (SOM) (Figure 10 and Table 4).

All intrusive conversations were detected with all of those algorithms, but DBSCAN had the best results with other performance metrics (Table 4). According to the results, it can be said that the proposed model had good performance metrics. The next steps of the development consist of improvement of the algorithm for online capability and testing it online with more complex data sets.

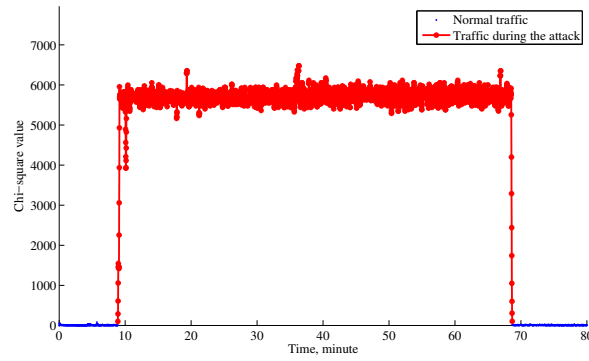


FIGURE 9 Chi-square values of extracted features of time intervals (PIII).

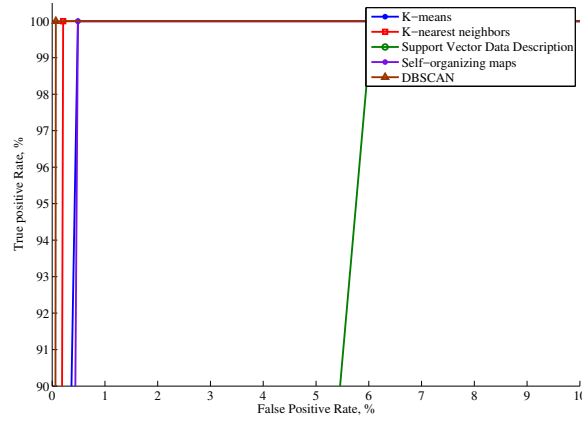


FIGURE 10 ROC curves of compared detection algorithms (PIII).

TABLE 4 Performance metrics of compared detection algorithms (PIII).

Algorithm	TPR	FPR	Detection Accuracy
K-means	100 %	0.4878 %	99.9951 %
KNN	100 %	0.2091 %	99.9979 %
SVDD	100 %	6.0627 %	99.9390 %
SOM	100 %	0.4878 %	99.9951 %
DBSCAN	100 %	0.0697 %	99.9993 %

3.4 The implemented general model for analysing encrypted traffic (PVI)

In article PVI, presented and summarised here, the model for analysis of encrypted network traffic for the detection of application-layer DoS/DDoS attacks is implemented and evaluated using realistic test environment. The payload of the network packet information is unreachable because of traffic encryption, thus

the introduced model implements the anomaly-detection-based approach, focusing to the statistics of the headers of the packet. In this study, the basic assumption is that there is a web server providing several services with HTTP and HTTPS protocols, and the implemented IDS is analysing incoming and outgoing traffic of that web server. The intrusion detection consists of the following phases: forming the normal user behaviour model, finding the conversations which are segregating from that normal user behaviour model for detecting trivial DoS/DDoS attacks, and analysing the distribution of feature vectors in the clusters for detecting more sophisticated DoS/DDoS attacks.

3.4.1 Feature extraction (PVI)

All inbound and outbound traffic during time period $[T_s, T_e]$ is captured and divided into equal non-overlapping time intervals ΔT . The intrusion detection is applied based on the analysis of network flows with common source IP address and port and destination IP address and port. Every flow found on time interval $[T_s + i\Delta T, T_s + (i + 1)\Delta T]$, and packets transferred in earlier time intervals belonging to that flow are found and grouped $[T_s + (i - 1)\Delta T, T_s + i\Delta T]$, $[T_s + (i - 2)\Delta T, T_s + (i - 1)\Delta T]$, etc. In addition, flow with equal source IP address and port as another flow's destination IP address and port and vice versa, are grouped as conversations. For each conversation at each time interval, the following information is extracted and normalized using max-min normalisation:

1. duration of that conversation
2. during 1 second, number of packets sent from client to server
3. during 1 second, number of packets sent from server to client
4. during 1 second, number of bytes from client to server
5. during 1 second, number of bytes from server to client
6. from client to server, maximal, minimal and average size of packet
7. from server to client, maximal, minimal and average size of packet
8. from client to server, maximal, minimal and average size of TCP window
9. from server to client, maximal, minimal and average size of TCP window
10. from client to server, maximal, minimal and average time to live (TTL)
11. from server to client, maximal, minimal and average time to live (TTL)
12. from client to server, percentage of packets with different TCP flags: FIN, SYN, RST, PSH, ACK and URG
13. from server to client, percentage of packets with different TCP flags: FIN, SYN, RST, PSH, ACK and URG
14. from client to server, percentage of encrypted packets with different properties: handshake, alert, etc
15. from server to client, percentage of encrypted packets with different properties: handshake, alert, etc

3.4.2 Analysis (PVI)

Features extracted during the training phase are assumed to be legitimate and can be used for the determination of normal user behaviour by using clustering for finding similarities from the patterns of vector data. For detecting trivial DoS/DDoS attacks, the feature vector of a new conversation under analysis is created. If the vector differs from the clusters created from the legitimate traffic in training phase, the conversation is identified as intrusive. The most popular categories for clustering algorithms are hierarchical clustering algorithm (Rafsanjani et al., 2012), centroid-based clustering algorithms (Uppada, 2014) and density-based clustering algorithms (Loh and Park, 2014), which can be used for analysis whether the new vector belongs to the cluster extracted in the training phase or not. In the numerical tests of the introduced model, different clustering algorithms are used.

If the attacker is capable for interactions in accordance with to normal user behaviour, there is a need for more efficient analysis; thus conversations related to the attack might be included into the normal behaviour model. Because of that, the feature vectors' distribution across clusters are noticed. Xu et al. (2014), Stevanovic et al. (2013), Chwalinski et al. (2013), Ye et al. (2012), Stevanovic et al. (2011) and Ranjan et al. (2009) have been analysing HTTP sessions for intrusion identification. When analysing encrypted traffic, session ID is not available, thus in this introduced method conversations from a certain client to a certain server at the same time interval are grouped and those groups are approximated as user sessions (the same source IP address, and the same destination IP address and destination port).

As in section 3.2.2.2 histogram vector is constructed from the training set. Here, in this method, for user sessions, the percentage of feature vectors in each n_C number of clusters is defined for histogram vector $h^{it} = (h_1^{it}, h_2^{it}, \dots, h_{n_C}^{it})$. At the t -th time interval and j -th cluster, h_j^{it} is calculated as the ratio between the number of feature vectors of the training set's i -th conversation group and the total number of the feature vectors of the i -th conversation group. Histogram vector $(h_1^{it}, h_2^{it}, \dots, h_{n_C}^{it})$ is the t -th row of matrix $H^{it} = (H_1^{it}, \dots, H_{n_C}^{it})$.

Anomaly detection of histogram vectors is done by using Stacked Auto-Encoders (SAE), which can be applied for dimensionality reduction (Sakurada and Yairi, 2014; Chen et al., 2014). There is a an assumption, when using dimensionality reduction for anomaly detection, that data can be projected into lower dimensional space because the data variables correlate with each other, and, when projected to a lower dimensional space, anomaly and normal data perform disparate (Sakurada and Yairi, 2014). In the proposed model, the t -th row H^{it} of histogram matrix H^i is reconstructed to \hat{H}^{it} and reconstruction error $E^{it} = \sqrt{\sum_{j=1}^{n_C} (H_j^{it} - \hat{H}_j^{it})^2}$ is defined. For feature vectors of the i -th conversation group of the training set, the mean value μ_i^E and the standard deviation σ_i^E of the reconstruction errors are used for the threshold $T^E = \mu_i^E + \omega\sigma_i^E$, where ω is the tuning parameter of the threshold value.

An Auto-Encoder (AE) consist of an input layer, hidden layer, reconstruction layer, and activation function f . First the input is mapped using the encoder to the hidden layer. Then the units of hidden layer are mapped using the decoder to the reconstruction layer as the output. The size of the input layer equals that of the reconstruction layer. The error between input vector h and its reconstruction \hat{h} is minimized during the training of the algorithm. If the encoder-decoder reconstructs the original input perfectly, it means that hidden layer of AE conserves enough information of the input: that is, during the reconstruction, the decoder uses only the information of hidden layer encoded from the input. Information loss is minimized by stacking the encoders with the help of method called Stacked Auto-Encoder (SAE). SAE is trained as follows: first AE is trained as described previously and the following AE layers are trained by the output of the earlier layer. Since all the layers are trained, fine-tuning for improving results can be applied by using back-propagation to tune the parameters of all layers at the same time. (Chen et al., 2014)

During the current time interval, histogram vector h can be reconstructed for a client with user session of type i . By using SAE, the reconstruction error E^{it} can be calculated. If reconstruction error E^{it} is bigger than threshold T^E , vector h is labelled anomalous and the user session of the current client can be labelled as intrusion.

3.4.3 Testing the model (PVI)

RGCE Cyber Range is utilised for the test simulation of proposed method. There was a fictitious service provider hosting and defending the web shop service implemented in RGCE Cyber Range. Communication to the web shop service is provided through encrypted HTTPS protocol. There is a total of 55 web shop users scattered with different global GeoIP locations within an roughly 2-hour-long scenario. All web shop users generate legitimate traffic, but some of them are also attackers who scan the target and use Slowloris, Slowpost and advanced DDoS attacks. The first 12 minutes of the dataset are only legitimate traffic used as the training set. The time step size is defined as 1 second.

To evaluate the performance of the proposed method for detecting trivial attacks as Slowloris and Slowpost, a single-linkage clustering algorithm, k-means, fuzzy c-means (Duan et al., 2016), self-organizing map and DBSCAN are used for clustering. TPR, FPR and Detection Accuracy are calculated and ROC curves plotted (Figure 11, Figure 12, Table 5 and Table 6).

Trivial Slowpost and Slowloris attacks can be detected with extremely good performance metrics, and still the number of false alarms remains very low.

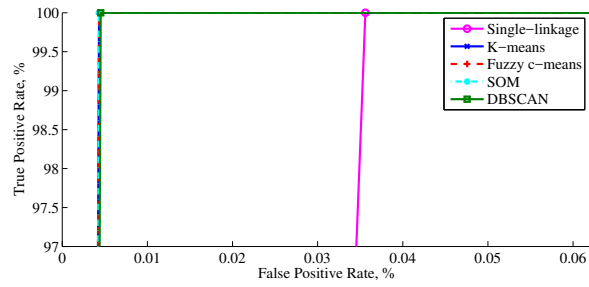


FIGURE 11 ROC curves of Slowloris detection (PVI).

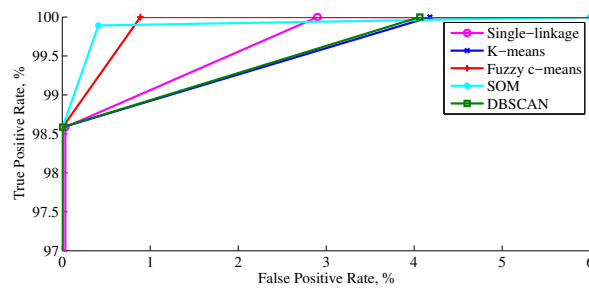


FIGURE 12 ROC curves of Slowpost detection (PVI).

TABLE 5 Performance metrics of Slowloris detection (PVI).

Method	TPR	FPR	Detection Accuracy
Single-linkage	100 %	0.0356 %	99.9644 %
K-means	100 %	0.0043 %	99.9957 %
Fuzzy c-means	100 %	0.0043 %	99.9957 %
SOM	100 %	0.0043 %	99.9957 %
DBSCAN	100 %	0.0045 %	99.9955 %

TABLE 6 Performance metrics of Slowpost detection (PVI).

Method	TPR	FPR	Detection Accuracy
Single-linkage	98.587 %	0.0356 %	99.9619 %
K-means	98.587 %	0.0043 %	99.9931 %
Fuzzy c-means	98.587 %	0.0043 %	99.9931 %
SOM	98.587 %	0.0043 %	99.9931 %
DBSCAN	98.587 %	0.0045 %	99.9929 %

To detect the more advanced attacks where attacker can mimic normal user behaviour, a SAE based method is tested. K-means, fuzzy c-means and SOM are applied for clustering the conversations, because those gave the best performance metrics with Slowloris and Slowpost attacks above. Between 1 and 5 of the hidden layers for SAE are selected and a dimension of each hidden layer is smaller than on the earlier layer (Figure 13 and Table 7).

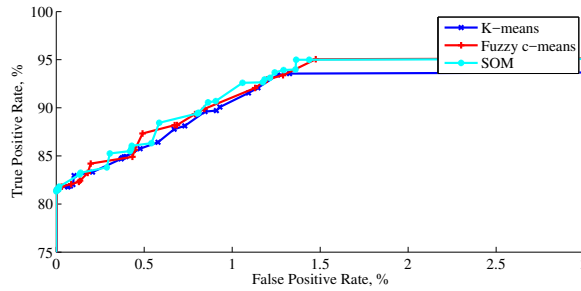


FIGURE 13 ROC curves of intermediate DDoS attack detection (PVI).

TABLE 7 Performance metrics of intermediate DDoS attack detection (PVI).

Method	TPR	FPR	Detection Accuracy
K-means	93.5569 %	1.3261 %	96.1058 %
Fuzzy c-means	95.051 %	1.475 %	96.7815 %
SOM	94.9906 %	1.3633 %	96.8068 %

Also these attacks can be detected with rather good performance values. The overall method was good and has potential for near-real-time implementation with online data, thus such method is considered as a general model for future IDS constructions in this dissertation.

4 ANOMALY-DETECTION-BASED IDS WITH ONLINE NETWORK DATA

This chapter introduces the construction of an online anomaly-detection-based IDS. First the training phase is implemented as online version and after that the general model, described in section 3.4, is implemented as online version and tested in a military exercise in co-operation with the Finnish Defence Forces. Finally, the performance of the IDS model is improved for high-speed networks.

4.1 Online training (PVIII)

In article PVIII, presented and summarised here, the model for online anomaly detection is implemented and evaluated using realistic test environment. The online approach is based on the general IDS model as described in section 3.4. Feature extraction, normalisation and the basic idea of anomaly detection model is similar in both. However, the training algorithm is constructed to use fuzzy clustering which is tailored for online traffic instead of captured traffic. When using captured network data, the whole training data set can be saved to memory for analysis, which is not possible when analysing online training data. With online data, a model of normal user behaviour is reconstructed when new network traffic is available.

From the training set, standardized feature vectors of conversations $X = \{x_1, \dots, x_n\}$ are defined. During the standardisation, vectors with minimal and maximal feature value are found as x_{min} and x_{max} . The cluster centroids $V^x = \{v_1^x, \dots, v_c^x\}$ and partition matrix $U^x = \{u_{ij}^x\}$ are calculated by using fuzzy c-means clustering. Reconstruction criterion can be used for determining how similar a feature vector is to the discovered normal user behaviour patterns or their combination (Izakian and Pedrycz, 2013). The reconstruction \bar{x}_j and reconstruction error, E , can be calculated for vector x_j , where that reconstruction error can be considered as an anomaly score. The average value μ^x and the standard deviation σ^x of the reconstruction errors of the training set's feature vectors are calculated.

Similar approximation for the user session, $S = \{s_1, \dots, s_N\}$, as introduced in the general model (section 3.4), is used here. With user sessions, earlier introduced general model considered percentage of feature vectors in each cluster for vector distribution, here with this improved model, that consideration is changed. By using fuzzy clustering, the membership matrix $A = \{a_{ij}\}$, with $i \in \{1, \dots, N\}$ and $j \in \{1, \dots, n\}$, and new feature matrix Y are defined, where the average probability that the j -th cluster of conversations includes a conversation from the i -th user session corresponds to the element y_{ij} of matrix Y :

$$a_{ij} = \begin{cases} 1, & \text{if } x_j \in s_i, \\ 0, & \text{if } x_j \notin s_i \end{cases} \quad (25)$$

$$Y = A(U^x)^T. \quad (26)$$

Similarly, cluster centroids $V^y = \{v_1^y, \dots, v_c^y\}$ and partition matrix $U^y = \{u_{ij}^y\}$, can be defined for user session's feature vectors in matrix Y with the help of the fuzzy c-means clustering. Reconstruction errors of new feature vectors, with average value μ^y and standard deviation σ^y of reconstruction errors, are calculated. As a result of the offline training period, the model of normal user behaviour is:

$$M_{offline} = \{x_{min}, x_{max}, V^x, \mu^x, \sigma^x, V^y, \mu^y, \sigma^y\}. \quad (27)$$

This offline method requires improvements for online use, which is done by changing the model so that network traffic is captured in short time intervals, where feature vectors of conversations are extracted. For the calculation of x_{min} , x_{max} , V^x , μ^x , σ^x , V^y , μ^y and σ^y , the previous offline version of training can be used to the vectors of the first time interval. Matrix H is then calculated:

$$H = V^y(e^T w^x)^{-1}, \quad (28)$$

where partition matrices U^x and U^y are used for calculation of weights of cluster centroids $w^x = (w_1^x, \dots, w_c^x)$ and $w^y = (w_1^y, \dots, w_c^y)$. Vector e has length c , with each element having value of 1. After the first time interval, when there exists a v^x conversations and v^y user sessions, the model of normal user behaviour M_{online} is defined:

$$M_{online} = \{x_{min}, x_{max}, V^x, w^x, v^x, \mu^x, \sigma^x, H, V^y, w^y, v^y, \mu^y, \sigma^y\}. \quad (29)$$

For the traffic captured during the second time interval, features from conversations are extracted: $X = \{x_1, \dots, x_n\}$. Vectors x_{min} and x_{max} analysed during the previous time interval are marked by x_{min}^{old} and x_{max}^{old} . New x_{min} and x_{max} are found by comparing new vector values and previous x_{min}^{old} and x_{max}^{old} for finding new vectors x_{min} and x_{max} . New cluster centroids V^x are found with a help of new and old values x_{min} , x_{max} , x_{min}^{old} and x_{max}^{old} :

$$v_{ij}^x = \frac{(x_{max,j}^{old} - x_{min,j}^{old})v_{ij}^x + x_{min,j}^{old} - x_{min,j}}{x_{max,j} - x_{min,j}}. \quad (30)$$

Weighted fuzzy c-means is used for vectors in X , using centroids V^x and weights w^x ; also new reconstruction error for vectors in X is calculated with mean value $\bar{\mu}^x$ and standard deviation $\bar{\sigma}^x$ of reconstruction errors. Now the total number of conversations $v^x = v^{x,old} + n$.

For the user sessions, the new cluster centroids $V^y = Hw^x$ and the new feature matrix $Y = \{y_1, \dots, y_N\}$ are defined. Definition of the new feature matrix is done according to equation 26. Fuzzy c-means is used for Y with recalculated centroids V^y and weights w^y for the resulting new cluster centroids with updated weights. For vectors in matrix Y , reconstruction errors are calculated with the mean value of reconstruction errors μ^y and standard deviation of reconstruction errors σ^y . Total number of user sessions is updated $v^y = v^{y,old} + N$, and new matrix H is defined.

Until all traffic of the training set is analysed, the user behaviour model is reconstructed upon new vectors.

4.1.1 Anomaly detection (PVIII)

Anomaly detection based on conversations and user sessions is carried out. Conversation based analysis can be applied for detection of trivial DoS/DDoS attacks by defining reconstruction error e^x for new conversation's feature vector x by using centroids V^x . If $e^x > \mu^x + \alpha^x \sigma^x$, the conversation is labelled as anomaly.

Similarly, user session based analysis can be applied for detection of more sophisticated attacks by defining reconstruction error e^y for the user session's feature vector y by using centroids V^y . If $e^y > \mu^y + \alpha^y \sigma^y$, the user session is labelled as anomaly.

In both cases, parameters α^x and α^y shall be tuned during testing and optimising the IDS.

4.1.2 Testing the model (PVIII)

The presented IDS model is implemented with Python programming language. RGCE Cyber Range was utilized for testing the model. A web shop server was installed to RGCE where several global users (both bots and humans) use the web shop through encrypted HTTPS-protocol. Three DoS/DDoS attack models were used by multiple attackers against that web shop. The model was tested with the offline and online training modes with legitimate user traffic. For the performance evaluation of the implemented model, TPR, FPR and Detection Accuracy were calculated, using both online and offline modes of training, and ROC curves were plotted (Figure 14 and Table 8).

Good performance metrics can be achieved, with Slowloris and Sslsqueeze. With more advanced DDoS attacks, the second part of the analysis, based on user sessions, performs quite well. Table 8 shows the detection accuracy with optimal α^x and α^y values for achieving maximal detection accuracy. The results proofed that the proposed weighted fuzzy clustering could be applied for the construction of normal user behaviour. Also online training, in particular, allows rebuilding

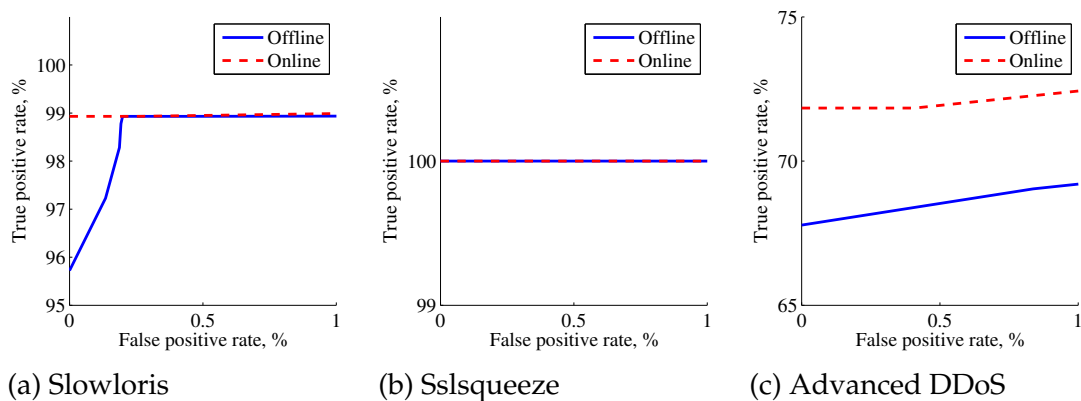


FIGURE 14 ROC curves of DDoS attack detection (PVIII).

TABLE 8 Detection Accuracy of DDoS attacks (PVIII).

Training algorithm	Slowloris	Sslsqueeze	Advanced DDoS
Offline	99.113 %	100 %	68.619 %
Online	99.223 %	100 %	71.837 %

of behavioural with a new set of network traffic.

4.2 Online analysis

This online approach is based on the general online IDS model as described in section 3.4. The programming language for the implemented application is Python (Version 2.7) and the hardware platform is normal office PC with the Ubuntu Linux (Version 14.04) operating system. The implementation is called a near-real-time system, which in this context is defined as following: application is buffering analysed traffic and indicates anomalies within a time interval that equals to 1 second for timely detection of attacks. That 1-second buffering is chosen because it is quick enough for incident response and risk mitigation activities. It must be mentioned, that even for achieving that 1-second interval, with large volumes of online traffic data, considerable amount of optimisation was required.

Feature extraction and anomaly detection is based on the earlier introduced general model (see section 3.4). In the training period, the network data is buffered and clustered at the end of the training period. Clustering and analysis of the feature vectors of conversations is implemented using k-means algorithm (see section 3.1.2.2). The number of clusters is implemented to be equal to the number of unique clients, however the maximum value is defined as a threshold parameter. Instead of using SAE, as in general model, the histogram vectors are clustered using k-means clustering. When detecting anomalous user sessions, the histogram vector is defined. If this vector does not belong to any of the conversation distri-

bution clusters extracted during the training, then it is classified as an anomaly and the user session as an attack.

4.2.1 Testing the model

Tests were conducted in a live scenario as part of the military exercise in a cooperation with the Finnish Defence Forces. During the tests, the workstation network with an unknown number of workstations was implemented and all network traffic was mirrored to the implemented IDS (Figure 15). The implementation was parametrised so that the number of clusters was equal to the number of unique clients. During the iterative implementation and tuning of the application, the maximum number of clusters was defined as equal to 20.

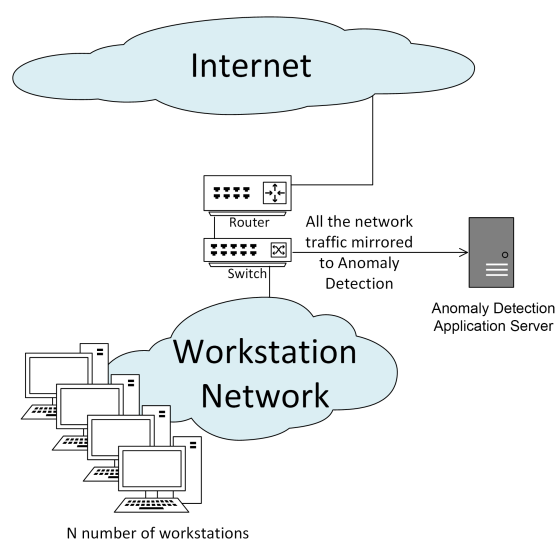


FIGURE 15 Setup of live scenario.

Legitimate traffic during the training phase consist of traffic of typical end-point applications such as web browsers. The length of the training phase was 10 minutes with 105 unique and active source IP-addresses. In the detection phase there were also illegitimate traffic of the ongoing attack campaigns mixed with legitimate network traffic. Illegitimate intrusive traffic included command and control traffic using application-layer protocols such as HTTPS, and, as in the training phase, legitimate traffic consisted typical endpoint applications such as web browsers. It shall be mentioned, that trivial volumetric DoS/DDoS was not generated during the scenario. Length of detection phase was approximately 2 hours with total number of 279 unique and active source IP addresses. Because of the classified nature of the military exercise, any other information concerning the network, computers or traffic was not released: for example, the amount of network traffic was not released. Generally, it can be stated that those characteristics were extremely realistic and it was a privilege to have that opportunity for testing implemented IDS in such realistic and unique environment.

As a result, the analysis identified 14 source IP addresses as anomalous and all the rest as legitimate. Afterwards, the number of intrusive source IP addresses

released was equal to 2. The performance metrics of the conducted test scenario are presented in Table 9 and Table 10. In addition to the characteristics of anomaly detection presented in section 3.1.3, also Precision and F-measure are defined (Mokarian et al., 2013; Fawcett, 2006):

- $Precision = \frac{TP}{TP + FP}$ - the ratio between TP and all instances classified as anomaly,
- $F-measure = \frac{2 \times Precision \times Recall}{Precision + Recall} = \frac{2 \times TP}{2 \times TP + FP + FN}$ - combines Precision and Recall.

ROC curves couldn't be plotted because only the information of intrusive IPs were released afterwards. A confusion matrix (Figure 16) that forms the basis for common performance metrics was constructed (Fawcett, 2006).

		True class	
		TP	FP
Hypothesized class	TP	TP	FP
	FN	FN	TN

FIGURE 16 Confusion matrix.

TABLE 9 Confusion matrix of a live scenario.

TP=2	FP=12
FN=0	TN=265

TABLE 10 Performance metrics of live scenario.

True Positive Rate, TPR	1
False Positive Rate, FPR	0,043321
Detection Accuracy	0,956989
Precision	0,142857
F-measure	0,25

The implemented application analysed, quite successfully, intrusions in a complicated live scenario. All the performance metrics values were good; however, there were quite a few false positives. One practical result is that a 10-minute training period is long enough for achieving good performance metrics with the implemented method. That training result is highly dependable of the variety of legitimate traffic included in the training data set. Identified future research will be on anomaly-detection-based IDS for a real-time high-capacity encrypted networks.

4.3 Online analysis in high-speed networks (PIX)

In article PIX, presented and summarised here, the model for online anomaly detection in high-speed networks is implemented and evaluated. Network traffic flows are analysed, and flows are combined together as conversations. Feature extraction and normalisation are carried out the way it is done in the generated general model introduced in section 3.4. In the training phase, consisting only of legitimate traffic, a clustering algorithm is applied for dividing feature vectors of conversations into clusters to create a model of normal user behaviour. Conversations from certain time interval with equal source and destination IP addresses and destination port are grouped as an approximation of user sessions and conversations of the user session are clustered. After that, the labels of those clusters are obtained as sequence of label numbers. Using the n-gram model that sequence is transformed to a sequence of n-labels. Frequency vector is implemented by counting the number of n-labels in the user session. Similarly, a clustering algorithm is used for clustering those new feature vectors as normal user behaviour.

4.3.1 Clustering (PIX)

In high-speed networks there exists large amount of network data, and it requires lot of memory resources to analyse such amount of data. Analysis shall be done sequentially. The implemented model uses following scheme for clustering the conversations. Let there be τ sequential time windows $t \in \{1, \dots, \tau\}$. There exists n_c^t number of conversations during the t -th time window and non-standardised raw feature vectors $X^t = \{x_1^t, \dots, x_{n_c^t}^t\}$ are extracted from training set during that t -th time window. Those raw vectors X^t are standardised applying max-min standardisation with values $x_{min,j}^t$ and $x_{max,j}^t$, for achieving standardised feature vectors $Z^t = \{z_1^t, \dots, z_{n_c^t}^t\}$. Arthur and Vassilvitskii (2007) introduced the kmeans++ algorithm, which can be applied for defining clusters p_i^t and k number of centroids m_i^t from standardised feature vectors Z^t . Standardised feature vectors are used for this clustering, but statistics are calculated using non-standardised raw feature vectors. For the clusters p_i^t , raw feature vectors $x(z)$ included in the clusters are used for calculating the new centroids μ_i^t and the sum of squared features $\zeta(p_i^t)$. In addition, the number of feature vectors in cluster p_i^t , $w(p_i^t)$, is defined.

For all the τ sequential time windows $t \in \{1, \dots, \tau\}$, the $\tau \times k$ clusters, considered as partitions, are determined. It should be noticed that, when implementing the application according to presented scheme, value τ is parametrised in compliance with the usable memory resource of the computing environment. Those $\tau \times k$ partitions shall be compiled to k new clusters. Minimal $\bar{x}_{min,j}^\tau$ and maximal $\bar{x}_{max,j}^\tau$ feature values of all the τ sequential time windows are found and used for standardising $\tau \times k$ centroids μ_i^t to m_i^t , where $t \in \{1, \dots, \tau\}$ and $i \in \{1, \dots, k\}$. New k centroids are obtained similarly as previously but consider-

ing previously saved values of $w(p_i^t)$. For new k clusters, \bar{p}_i^τ , its centroid $\bar{\mu}_i^\tau$, the number of feature vectors $w(\bar{p}_i^\tau)$ and the sum of squared features in all of these vectors $\zeta(\bar{p}_i^\tau)$ are defined. Values $x_{min,j}^t$ and $x_{max,j}^t$ for $t \in \{1, \dots, \tau\}$ are assigned to values $\bar{x}_{min,j}^\tau$ and $\bar{x}_{max,j}^\tau$.

After the new k clusters, \bar{p}_i^τ , are compiled from $\tau \times k$ partitions p_i^t , implementation is capable to delete information of previous partitions, and continues defining partitions for the next $\tau - 1$ sequential time windows. New k clusters are compiled using clusters \bar{p}_i^τ and partitions from next $\tau - 1$ sequential time windows. Finally, at the end of the training phase, k conversation clusters p_1, \dots, p_k are achieved. As detailed in PIX, from the implementation perspective, reducing the computational power required for the above procedure, clustering can be replaced with the use of the streaming k-means introduced by Braverman et al. (2011).

The implemented model uses following scheme for clustering the user sessions. At the time window t , after all the conversations are divided into the k clusters, an n -gram vector of size k^n is defined for all the sessions. Let n_s^t be the number of user sessions at time window t , and feature matrix $Y^t = \{y_1^t, \dots, y_{n_s^t}^t\}$. The same method for clustering is applied for achieving K number of session clusters. Also for each cluster P_i^t , centroid M_i^t and number of feature vectors $w(P_i^t)$ are defined. With user sessions, sum of squared features is used, but here it is replaced by $k^n \times k^n$ size matrix $S(P_i^t)$, where $j, l \in \{1, \dots, k^n\}$ and the elements of the matrix are $S_{jl}(P_i^t) = \sum_{y \in P_i^t} y_j y_l$.

When considering τ sequential time windows $t \in \{1, \dots, \tau\}$, updates of the connection clusters induce modifications in the n -gram vectors. Let there be function $f(j, p_i^t, \bar{p}_i^\tau)$ where $j \in \{1, \dots, k^n\}$. That function $f(j, p_i^t, \bar{p}_i^\tau)$ returns the index of the n -gram which is achieved from the j -th gram by replacing label l with label i , if label l is in the j -th n -gram and a partition p_i^t is associated with the new cluster \bar{p}_i^τ . It is assumed that q number of partitions $p_{i_1}^t, \dots, p_{i_q}^t$ of the time window τ are included in the cluster \bar{p}_i^τ from conversations. The i -th session centroid's j -th component, M_{ij}^t , and both $S_{jl}(P_i^t)$ and $S_{lj}(P_i^t)$ elements of matrix $S(P_i^t)$, where $l \in \{1, \dots, k^n\}$, are modified using function f , if label i is in the j -th gram. However M_i^t and both elements $S_{jl}(P_i^t)$ and $S_{lj}(P_i^t)$ are assigned to zero if label $i_a \in \{1, \dots, i_q\}$ is in the j -th gram and label i is not in that j -th gram.

New K clusters are defined similarly to those with conversations: after the updating of $\tau \times K$ session partitions P_i^t , where $t \in \{1, \dots, \tau\}$ and $i \in \{1, \dots, K\}$, these partitions are compressed to K number of clusters, \bar{P}_i^τ , and for each resulting cluster \bar{P}_i^τ , centroid $m(\bar{P}_i^\tau)$, the number of the feature vectors $w(\bar{P}_i^\tau)$ and matrix $S(\bar{P}_i^\tau)$ are defined, where $S_{jl}(\bar{P}_i^\tau) = \sum_{m(x) \in \bar{P}_i^\tau} S_{jl}(x)$. Finally, at the end of the training phase, K session clusters P_1, \dots, P_k are achieved.

As a summary, after the training phase with legitimate network traffic, the normal user behaviour model consists of:

- For the final k conversation clusters p_1, \dots, p_k :
 - minimal and maximal feature values $x_{min,j}$ and $x_{max,j}$,

- centroids $\mu_i = m(p_i)$,
- amount of associated feature vectors $w_i = w(p_i)$,
- sums of squared feature values $\zeta_i = \zeta(p_i)$.
- For the final K user session clusters P_1, \dots, P_K :
 - centroids $M_i = m(P_i)$,
 - amount of associated vectors $W_i = w(P_i)$,
 - matrices $S_i = S(P_i)$.

4.3.2 Anomaly detection (PIX)

For anomaly detection of the conversations, new centroids m_{ij} and the sums of the squared feature values s_{ij} of the j -th feature, where $i \in \{1, \dots, k\}$ are defined according to values $x_{min,j}$ and $x_{max,j}$:

$$m_{ij} = \frac{\mu_{ij} - x_{min,j}}{x_{max,j} - x_{min,j}}, \quad (31)$$

$$s_{ij} = \frac{\zeta_{ij} + w_i(x_{min,j}^2 - 2x_{min,j}\mu_{ij})}{(x_{max,j} - x_{min,j})^2}.$$

Radius r_i and diameter ψ_i are defined for clusters p_i by using values of s_i , w_i and m_i . Similarly radius R_i and diameter Ψ_i are defined for clusters P_i by using values of M_i , W_i and S_i .

When detecting attacks from conversations, new feature vector x is extracted and standardised. The conversation is identified as an attack if:

$$d(x, m_{i(x)}) > r_{i(x)} + \alpha\psi_{i(x)}, \quad (32)$$

where $d(x, m_{i(x)})$ is the distance between vector x and the nearest centroid $m_{i(x)}$, and $\alpha > 0$ is a tuning parameter which is tuned during the testing and optimising period of the algorithm's implementation.

Correspondingly, when detecting attacks from user sessions, new feature vector y is extracted from the user session. The user session is identified as an attack if:

$$d(y, M_{i(y)}) > R_{i(y)} + \beta\Psi_{i(y)}, \quad (33)$$

where $d(y, M_{i(y)})$ is the distance between vector y and the nearest centroid $M_{i(y)}$, and $\beta > 0$ is a tuning parameter which is tuned during the testing and optimising period of the algorithm's implementation.

4.3.3 Testing the model (PIX)

The IDS implementation according to clustering and attack detection algorithms described in sections 4.3.1 and 4.3.2 is implemented using Python programming language. A virtual test environment with a web-bank server is established. Legitimate traffic that mimics human user behaviour is generated and mixed with

three types of DoS/DDoS attack traffic. As with earlier models, those three selected attack types are Sslsqueeze, Slowloris and advanced DDoS.

First the clustering is evaluated with a legitimate user traffic, with a duration of 10 minutes and generated by 45 user bots using web-bank services with 15-45 second delay between sequential sessions. The chosen time window is 5 seconds, and partitions to new cluster centroids are defined every 10 time windows. For evaluating the clustering of conversations and clustering of user sessions using offline mode and online mode, the average cost $C = \frac{1}{|X|} \sum_{x \in X} \min_{y \in m} d(x, y)$ is calculated, where X is the set of vectors and m is cluster centres (Figure 17 and Figure 18).

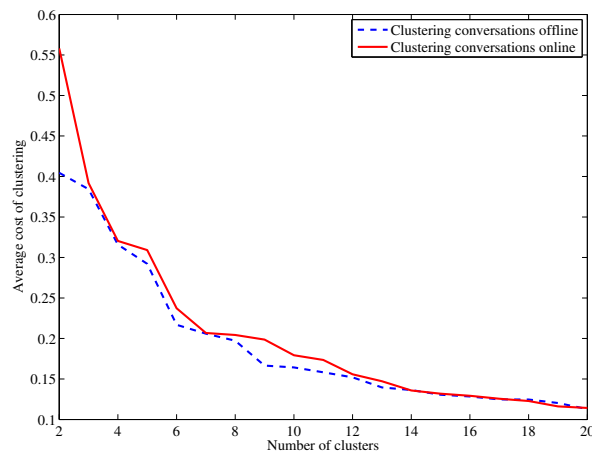


FIGURE 17 Average costs of conversation clustering (PIX).

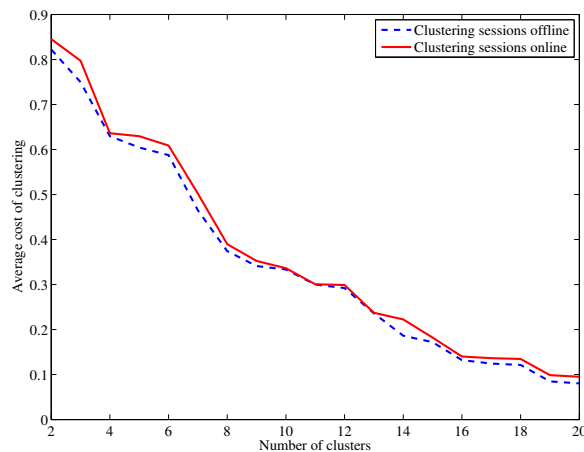


FIGURE 18 Average costs of user session clustering (PIX).

For evaluating the anomaly detection performance metrics of the introduced algorithm FPR, TPR and Detection Accuracy are calculated, and ROC curves plotted. With Sslsqueeze and Slowloris attacks, results of the online mode are compared with the results of the offline mode (Figure 19 and Figure 20). The offline

mode is implemented by using k-means combined with k-means++. The 1 second time window is selected and different values of parameter α and number of clusters k are used when plotting TPR against FPR for ROC curves.

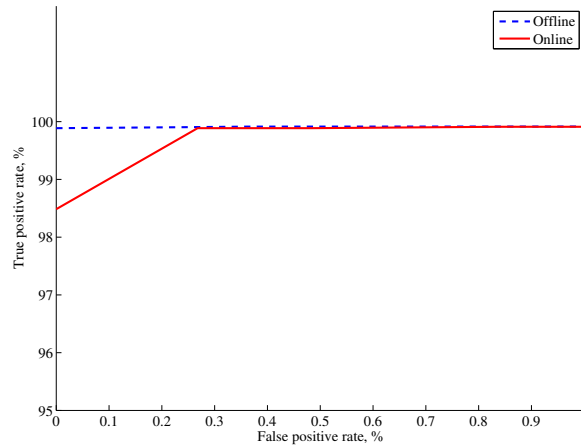


FIGURE 19 ROC curves of Sslsqueeze detection for different clustering parameters (PIX).

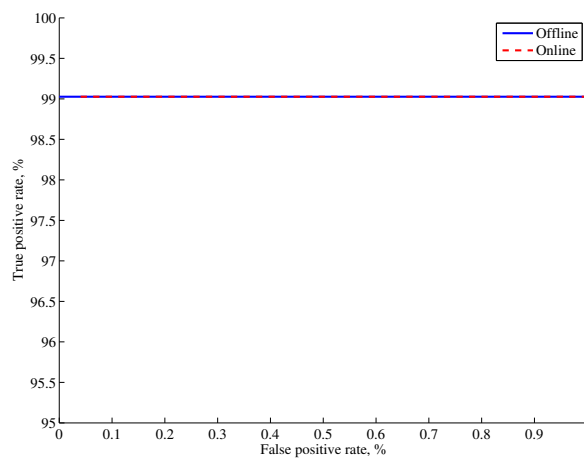


FIGURE 20 ROC curves of Slowloris detection for different clustering parameters (PIX).

For evaluating performance metrics with advanced DDoS attacks, 2-gram model is used with different sizes of time window (Figure 21). In calculation of Detection Accuracy, clustering parameters are selected with tuned optimal values (Table 11).

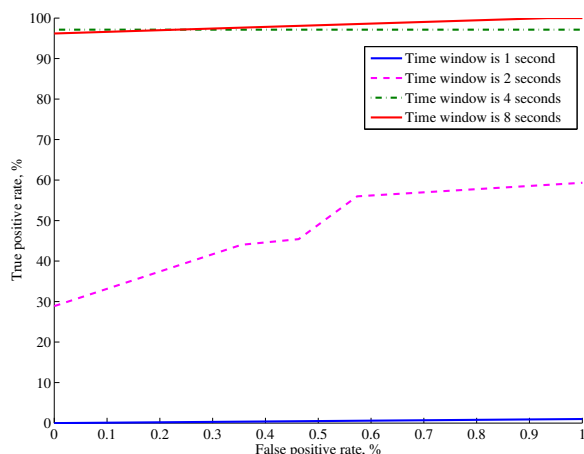


FIGURE 21 ROC curves of advanced DDoS attack detection for different time window sizes and clustering parameters (PIX).

TABLE 11 Detection Accuracy of DDoS attacks (PIX).

Attack	Time window size			
	1 second	2 seconds	4 seconds	8 seconds
Sslsqueeze	99.89%			
Slowloris	99.77%			
Intermediate DDoS	63.29%	85.21%	98.52%	99.34%

As it can be seen, performance metrics are good with Sslsqueeze and Slowloris attacks, and also with more advanced DoS attacks when the size of time window is increased. With Sslsqueeze attacks, the algorithm is capable for detecting TPR (98.5%) without false alarms, and TPR (99.9%) with FPR (0.8%). When considering Slowloris attacks, the algorithm is capable for detecting anomalous conversations without false alarms. With more advanced DDoS attacks, analysis of user session's n-gram vectors achieved TPR (97%) with the 4-second time window and with 8-second time window all the intrusive sessions are detected, but with the 1-second time window, it is impossible to detect intrusions from user sessions.

The introduced intrusion detection capability depends highly on the selected clustering parameters and requires tuning and optimisation of those parameters during the implementation. With correctly chosen parameters, model does not require lot of resources from the platform and is capable for detecting intrusions with quite a good performance metrics. As a future work, the model will be tested with bigger data sets generated by real human-made end-user traffic.

5 SITUATIONAL AWARENESS IN CYBER SECURITY

Because almost all the business areas are using networked systems or services, cyber threats, cyber attacks, or more commonly intrusions, might affect to the continuity of business. Organisations sharing or exchanging information related to those intrusions would use it as an early warning information for immediate intrusion mitigation and threat response activities. Information related to cyber threat is often sensitive and might be classified, so when that information is shared with other organisations, there is a risk of being compromised.

5.1 Situational awareness for decision making (PV; PVII)

Situation awareness and situational awareness are mixed in a literature and in this dissertation the term situational awareness is used. When considering dynamic environments and decision making, SA has an important role. One of the most used definition of SA is written by Endsley (1995): "*Situation awareness is the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future*". He also states that even highly trained decision makers will make mistake decisions if they use improper SA for decision making.

Cyber SA can be defined as awareness of own cyber domain for assistance of decision making for defensive or reactive actions, one special feature of relative new concept of Cyber SA is that it focuses on awareness of data networks for attack or intrusion identification of cyber defence (Tadda, 2008). SA is a state of knowledge depending on the person's competence in a specific task and on events related to that task (Tadda and Salerno, 2010; Kuusisto et al., 2005; Endsley, 1995). Common Operational Picture (COP) is a production of information available related to the task, as a source for construct the SA (Kuusisto et al., 2005). In military, even if cyber domain is widely appreciated as an operational domain (and in kinetic environments it is common to use COP for decision making) it is still stated that there is no versatile Cyber Common Operational Picture

(CCOP) available (Conti et al., 2013). Cyber domain and physical domain have fundamental differences concerning sensors, and consideration of the time and location information (Conti et al., 2013; Barford et al., 2010; Tadda and Salerno, 2010). For example, a cyber attack from the other side of the world is just as sudden as a cyber attack originated nearby, which is not true with the kinetic world. Sensor information from IDS is quite different from sensor information of kinetic environment e.g. obtained from surveillance radar.

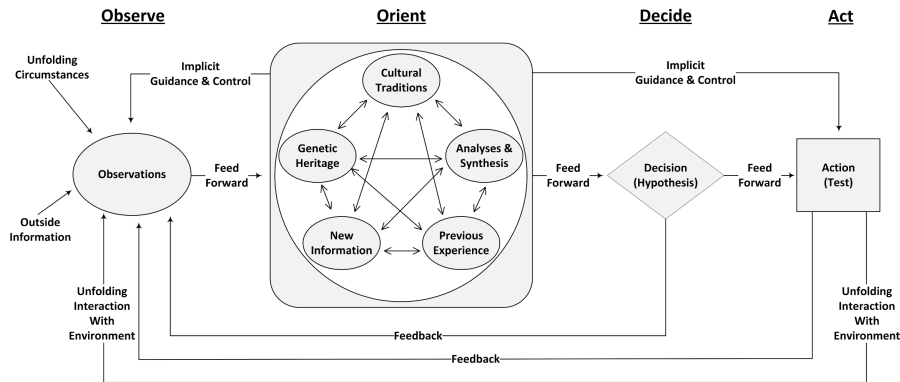


FIGURE 22 Modified OODA-loop, quoted from Brehmer (2005).

An often used model for decision making in Command and Control (C2) is the OODA-loop by Boyd, which in its basic form is Observation-Orienta-tion-Decision-Action phases cycle originally implemented for decision making in air operations (Fusano et al., 2011; Brehmer, 2005). When thinking about the OODA-loop and especially of the two first phases of the loop, it can be clearly seen that SA depends on person’s competence, and the total decision making loop illustrates that proper cyber security SA is required for making decisions related to cyber domain (Figure 22). The OODA-model in cyber defence works in phases: during the observation phase, sensor information concerning the infrastructure and assets is gathered; during the orientation phase, that information is analysed to find out what is happening; during the decision phase, the countermeasures, incident response, mitigation or recovery activities are chosen; and during the action phase, those chosen activities are employed (Figure 23). After that the new loop starts with the observation.

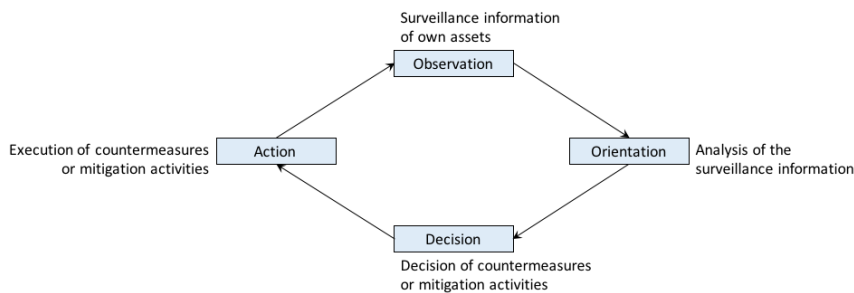


FIGURE 23 OODA-loop in cyber defence.

5.2 Sharing the cyber security situational awareness (PV)

In article PV, presented and summarised here, the model for creating the information sharing communities for the cyber security situational awareness information is implemented and evaluated. Development of proper cyber SA is important for decision making, and sharing that information is the key element of cyber security (Davies and Patel, 2016). In the United States, two laws about sharing the information on cyber SA were recently signed (The United States Congress, 2015b,a). The Cybersecurity Information Sharing Act requires the parties to develop procedures for sharing threat information of cyber security between different stakeholders, whereas the Cyber Intelligence Sharing and Protection Act obliges the parties to provide sharing of situational information of cyber threats in real-time between nominated stakeholders.

Among the globally developed standards for sharing the information of cyber security required in cyber SA, the most popular technical standards are Structured Cyber Observable eXpression (CybOXTM), Threat Information eXpression (STIXTM), and Trusted Automated eXchange of Indicator Information (TAXIITM). CybOXTM is a language for standardized structured information of cyber observables, STIXTM is a language for standardized structured communication of cyber threat information for improving interoperability and cyber security situational awareness, and TAXIITM is a framework for exchanging cyber threat information that determines the set of messages, protocols, and services (The MITRE Corporation, s. a.; Barnum, 2014; Connolly et al., 2014). For example, The U.S Department of Homeland Security has been using a system called Automated Indicator Sharing for providing the bidirectional sharing of the cyber security threat indicator information utilizing TAXIITM capability and STIXTM profile (The United States Computer Emergency Readiness Team, US-CERT, s. a.).

As seen in Figure 24, TAXIITM supports following information sharing models: hub-and-spoke, peer-to-peer and source-subscriber (Connolly et al., 2014). The STIXTM architecture consists of eight constructs (Figure 25), which are utilized to the XML schema: Observable, Indicator, Incident, TTP (tactics, techniques, and procedures), ExploitTarget, CourseOfAction, Campaign and ThreatActor (Barnum, 2014). Figure 26 demonstrates STIXTM use cases where also cyber security information sharing between organisations is implemented (Barnum, 2014).

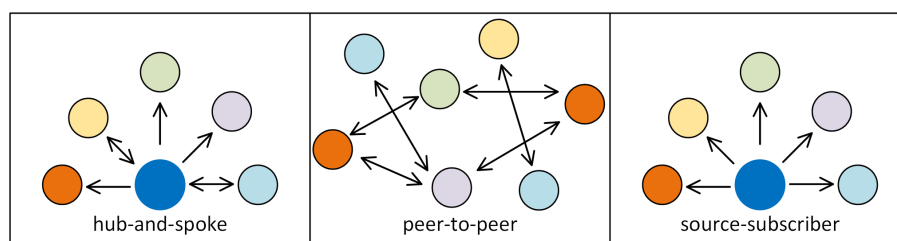


FIGURE 24 Threat sharing models of TAXIITM, quoted from Connolly et al. (2014).

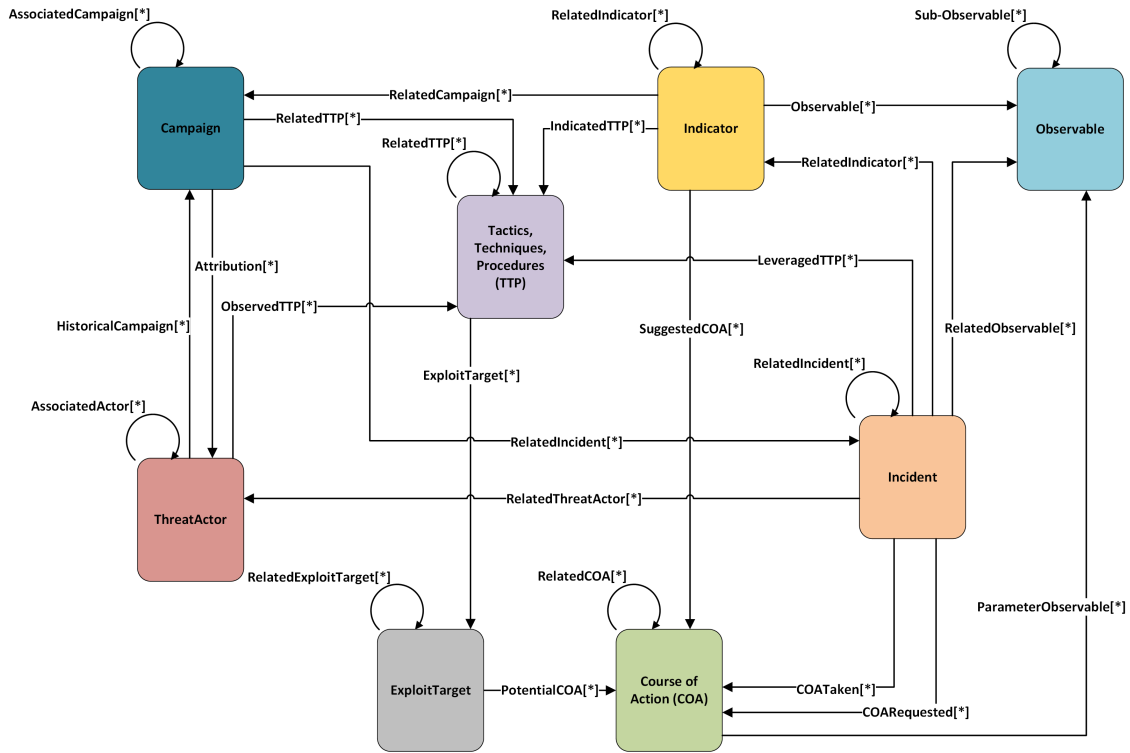


FIGURE 25 Architecture of STIX™, quoted from Barnum (2014).

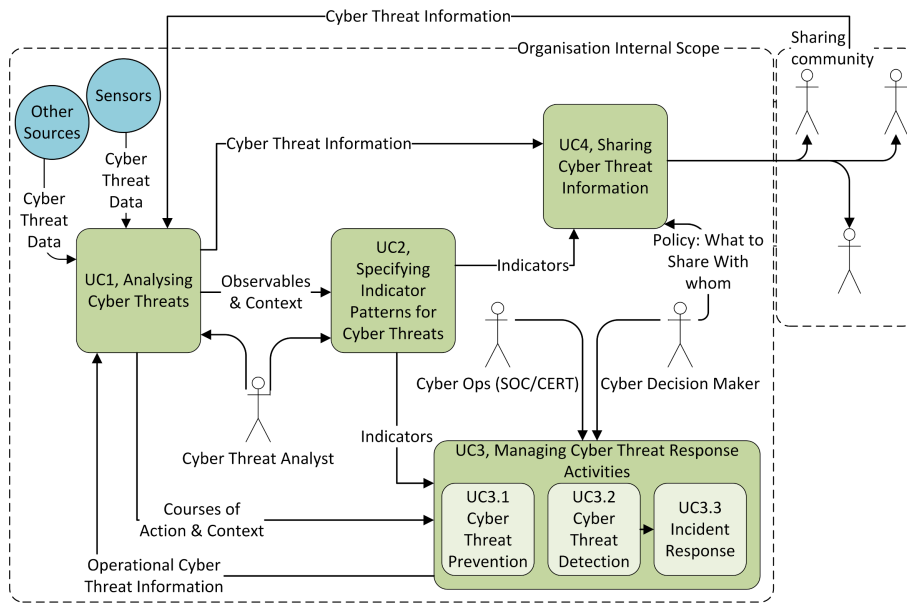


FIGURE 26 Example of STIX™ use cases, quoted from Barnum (2014).

The basic assumption when sharing cyber security information is that information might be classified. There is recognised risk of being compromised when sharing classified information; thus information sharing requires trusted relationships (Davies and Patel, 2016). The model developed is used for constructing the topology of the information sharing community. The model is based on the assumption that there is a predefined risk level for sharing the information between

organisations. The TAXII™ peer-to-peer information sharing model is used with STIX™ architecture; risk level values are required to have the same scale and organisations are sharing information only to trusted partners. Constructing such topology is similar to find the shortest paths between nodes. For constructing information sharing community with minimum risk levels, the implemented model utilizes Dijkstra's shortest path algorithm (Dijkstra, 1959), presented as pseudo code by Li et al. (2009, 2010).

Let there be a real life scenario with three different national CERTs as the highest national authority, the national and international Internet Service Providers (ISPs) as the next level and various national and international enterprises, as seen in Figure 27. Every peer-to-peer TAXII™ link has risk level value of $[1, 20]$, where the risk values are defined as $1 = \text{min-risk}$ and $20 = \text{max-risk}$.

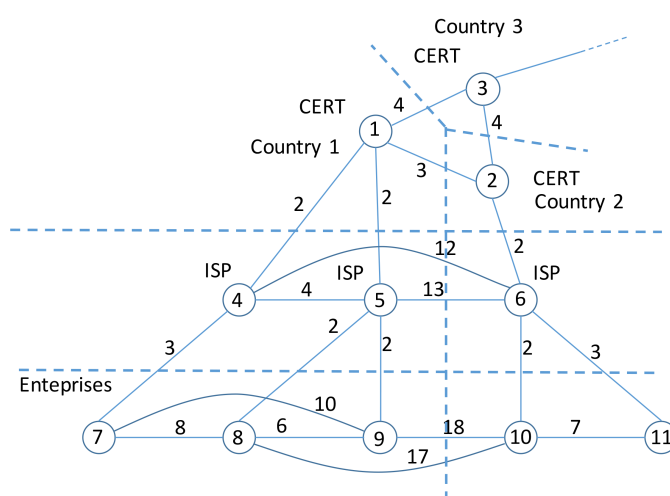


FIGURE 27 Cyber security information sharing community (PV).

Information sharing topology with a minimum risk level implementation can be achieved by applying Dijkstra's shortest path algorithm, as shown in Figure 28. Even if there are no direct connections between all the organisations, the data flow still goes to every organisation in that community. It is assumed that information might be anonymized, according to the company policies for example the origin of the information shall not be shared to the whole community.

The implemented model can be used for creating a minimum risk topology in order for share classified cyber security information between organisations in an information sharing community. It should be realised that even if there is a topology of minimum risk, filters should be implemented in outbound interfaces to filter out the information that is not allowed to be shared. As a future undertaking, the model will be implemented with one-way source-subscriber links and risk levels instead of peer-to-peer links and risk levels. The model shall be tested with shortest path algorithms other than Dijkstra's algorithm and numerical parameters will be added to describe the weight of the path, e.g. latency of the link, reliability of the information or accuracy of the information.

The Forum of Incident Response and Security Teams (FIRST) has released

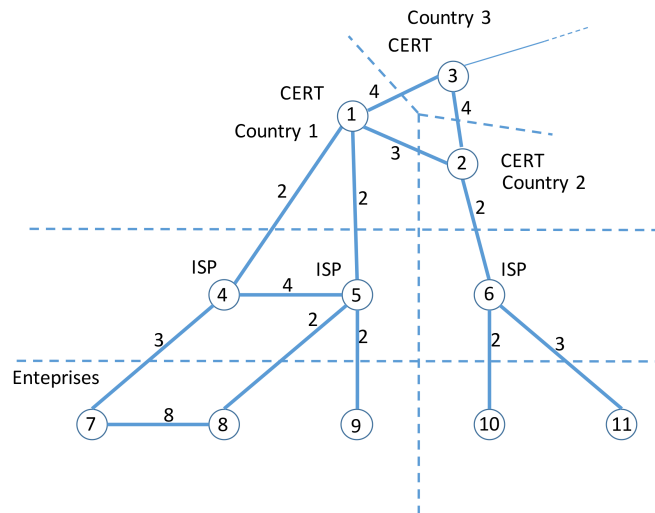


FIGURE 28 Cyber security information sharing topology with a minimum risk level implementation (PV).

version 1.0 of the consolidated Traffic Light Protocol (TLP), which facilitates a four-colour category for information sharing (red, amber, green, white): for example the meaning of TLP:RED is *"Not for disclosure, restricted to participants only"* and the meaning of TLP:WHITE is *"Disclosure is not limited"* (Forum of Incident Response and Security Teams FIRST, 2016; Forum of Incident Response and Security Teams, FIRST, 2016). The TLP categories can be applied as a part of information sharing rules and topology construction for filtering data between organisations.

5.3 Cyber security situational awareness system (PVII)

In article PVII, presented and summarised here, the model for cyber security SA system is created and evaluated. When developing a system for cyber security SA, it should be kept in mind that in modern networks there is a wide and increasing repertoire of systems and devices generating information to be analysed and visualised as part of the cyber security SA. The increasing amount of data requires high computational resources (Yu et al., 2013). Data fusion or multi-sensor data fusion refers to a process that uses overlapping information from multiple sources and merges it to refine a fused projection of environment to get a better understanding of what is happening (Azimirad and Haddadnia, 2015; Khaleghi et al., 2013; Giacobe, 2010; Liu et al., 2007a,b; Bass, 2000; Steinberg et al., 1999; Hall and Llinas, 1997).

5.3.1 Multi-sensor data fusion (PVII)

Research and development of data fusion is originating from military applications such as surveillance systems, battle management systems and target or

threat recognition systems. Besides military applications, there is a wide range of non-military applications such as applications for medical diagnosis (Bass, 2000; Hall and Llinas, 1997). One of the commonly employed model of data fusion is the JDL model implemented by the Joint Directors of Laboratories. The JDL fusion model is divided in different levels 0-6 (originally there were levels 0-4), each level enabling merging and refining information. At each level, there is possibilities to draw conclusions for achieving a better SA. The JDL levels in the cyber security context are (Swart et al., 2015; Azimirad and Haddadnia, 2015; Castanedo, 2013; Blasch et al., 2013; Khaleghi et al., 2013; Giacobe, 2010; Bass, 2000; Steinberg et al., 1999):

0. Data Assessment -level: Sensor information feed to the cyber security SA system.
1. Object Assessment -level: Identification of entities in the cyber domain, thus physical connections of network, devices, services or data flows.
2. Situation Assessment -level: Evaluation of the state of the systems in the cyber domain, e.g. information of software versions or patches, and information of known vulnerabilities, risks or threats.
3. Impact Assessment -level: Information about an ongoing attack or a threat combined with information about possible damage, incident response activities ongoing or already done.
4. Process Refinement/Resource Management -level: Cyber security sensor management, including configuration of individual sensors with settings and definition of the sensors' reliability score and selection of sensors in use.
5. User Refinement/Knowledge Management -level: Access to control each layer of fusion by using Human Machine Interface (HMI), and visualisation of information to the user.
6. Mission Management -level: Set of policies and mission objectives that should be known in decision making.

In a kinetic environment, the sensor feed has a spatio-temporal nature: thus data correlation and data association can be done based on time and physical location. This differs from cyber security sensor data, where correlation can be done using IP-addresses or other characteristics of network behaviour (Bass, 2000). Zhang et al. (2011), Zhao et al. (2009) and Tian et al. (2005) are using the Dempster-Shafer theory for data fusion of IDS sensor data while Fessi et al. (2011) apply clustering method for data fusion of IDS and Beheshti and Wasniowski (2007) merge IDS data with equal time stamps.

5.3.2 Required Interfaces (PVII)

The SA system for supporting decision making in cyber domain requires different input and output interfaces. The main categories of interfaces are as follows:

- Sensor information interfaces. The system implements interfaces for input of cyber security sensor information.

- Interfaces for status information. The system implements interfaces for inputting the status information of all the known cyber entities. Information of systems, devices and sensors with their status and configuration information, but also the spare parts of physical devices are relevant information for a cyber security SA system. Also information about the status of saved data and the status of information flows should be reported. Some of that information can be automatically generated using data interfaces and some should be user generated by using HMI.
- Interfaces for analysis information. The system implements interfaces for information based on analysis. That kind of information includes analysed impact assessment information, Indicator Of Compromise (IOC) information and early warning information from open source intelligence using e.g. social media or CERT-bulletins. Also required policies and objectives should be input to the system.
- Interfaces for information exchange. The system implements interfaces for cyber security information exchange with trusted companions, as described in section 5.2.
- HMI. The system implements HMI for effective visualisation of the current status of the cyber domain under control and for input of information that cannot be entered automatically. HMI is also used for controlling the data fusion process. HMI should implement different visualisations for different levels of users: e.g. technical user who requires detailed technical information, whereas a decision maker needs totally different visualisation. As in the study of Laaperi and Vankka (2015), HMI also implements filters for data allowed for different users.

5.3.3 High level architecture (PVII)

Cyber security SA system includes the data fusion engine, information interfaces and the HMI providing an effective visualisation layer. The functionalities described above should be as automatic as possible without human interaction; however, there should be an operator for controlling the sensors and data fusion algorithms and inputting information to the system. The high level architecture of the cyber security SA system can be seen in Figure 29, and there is an example use case of such system in Figure 30. It should be noticed that there should also be normalisation layer for feature extraction and normalisation of input information before inputting it to the fusion engine.

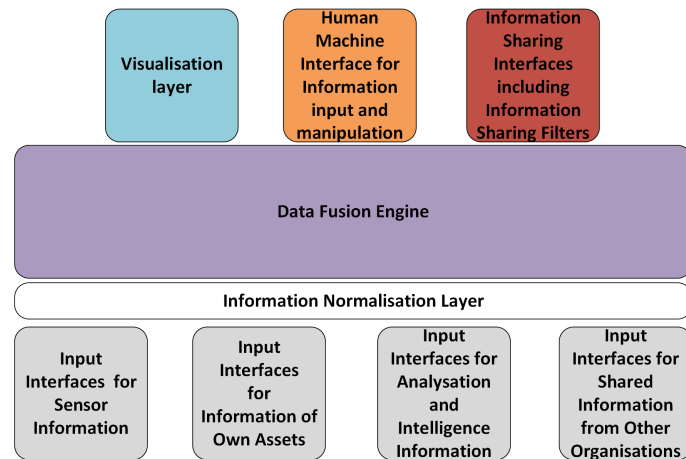


FIGURE 29 High level architecture (PVII).

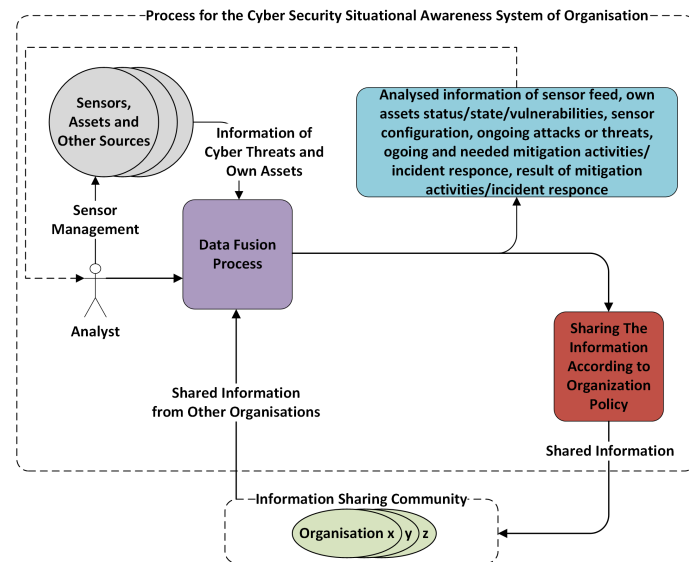


FIGURE 30 Use case example of cyber security SA system (PVII).

For the development of such cyber security SA system, the system requirements in detailed level, can be produced by using the proposed high-level architecture, which utilises both the multi-sensor data fusion process and the information exchange process; both are required for achieving cyber resilience and continuity for a business or mission using decisions based on SA.

6 CONCLUSION

In modern societies almost all organisations and individuals depend on networks and networked systems and services. The amount and complexity of networks, systems and users are constantly increasing, which enables the use of multitudinous attack vectors. The dilemma of digitalisation poses the requirement for comprehensive situational awareness in cyber security as a backbone for decision making. SA in cyber security helps us to understand the current situation of cyber domain and make predictions about what will happen in a near future.

This dissertation approaches the above-described challenge from the viewpoint of intrusion detection system as a sensor for cyber security situational awareness system. Different approaches for constructing an intrusion detection system are described in order to achieve an online intrusion detection system for modern encrypted high-speed networks. Also a model for a situational awareness system is constructed as is a model for creating topologies for changing the cyber security SA information between organisations to enhance situational awareness.

When creating constructions like those above, the theoretical background of the chosen method is described. The mathematical model of the current construction is described, and finally the construction created is tested in a realistic (or real) environment. Also, the results are discussed and analysed. The described models and approach can be used as a basic models of an intrusion detection system which can be utilised for a system of situational awareness. The produced model for information sharing can be utilised by different commercial or governmental organisations inside the country or between different countries as early warning information of a possible threat. Future research can be carried out to achieve better performance metrics and visualisation about intrusions in complex networked systems - actually a basic implementation of cyber security situational awareness system.

YHTEENVETO (FINNISH SUMMARY)

Tietoverkoissa tapahtuvien poikkeamien havainnointiin perustuva tunkeutumisen havainnointijärjestelmä sensorina kyberturvallisuuden tilannekuvajärjestelmälle

Nykyaikana lähes kaikki organisaatiot ja ihmiset ovat riippuvaisia tietoverkoista ja tietoverkkopohjaisista järjestelmistä. Tietoverkkojen ja verkotettujen järjestelmien, sekä verkossa liikkuvan tiedon määrä kasvaa jatkuvasti ja muodostaa entistä monimutkaisempia riippuvuussuhteita, tarjoten enenevässä määrin mahdollisuuksia tietoverkkorikollisuudelle. Tämä väitöskirja käsittelee tunkeutumisten havainnointijärjestelmän kehittämistä poikkeamien havainnointiin perustuen.

Poikkeamien havainnointiin perustuvassa tunkeutumisen tunnistamisessa hyödynnetään klustereihin perustuvia tiedonlouhinta-algoritmeja, joiden avulla järjestelmän opetusvaiheessa verkkoliikenteestä määritellään malli normaalille käyttäytymiselle ja tunnistamisvaiheessa pyritään löytämään poikkeamia tästä määritellystä normaalimallista. Löydetyt poikkeamat havainnoidaan tunkeutumisiksi. Tällä tavalla kyetään havainnoimaan myös ennalta tuntemattomia hyökkäyksiä salatusta verkkoliikenteestä. Tätä tunkeutumisen havainnointitietoa voidaan käyttää yhtenä sensoritietona kyberturvallisuuden tilannekuvajärjestelmälle, jolla mahdollistetaan tilannetietoisuus päätöksenteon tukena. Tätä kybertilannetietoisuutta ja sen mukaista päätöksentekokykyä tarvitaan liiketoiminnan jatkuvuuden ylläpidossa kyberpoikkeamatilanteissa.

Aluksi väitöskirjassa esitellään tunkeutumisen havainnointijärjestelmän perusteet ja tunkeutumisen havainnointijärjestelmien kehittämisessä ja testauksessa tarvittavan verkkoliikennetiedon tuottamisjärjestelmä. Tämän jälkeen esitellään vaiheittain eri lähestymistapoja tunkeutumisen havainnointijärjestelmän kehittämiseksi. Aluksi malleissa käytetään generoitua ja tallennettua verkkoliikennetietoa, ja myöhemmin mallin kehityksen edetessä, havainnointi tapahtuu lähes reaaliaikaisesti salattua verkkoliikennetietoa käyttäen. Kehitettyjen mallien teoreettinen viitekehys ja suoritettavat testitulokset esitellään tapauskohtaisesti. Osana väitöskirjaa esitellään testiskenaario, joka on suoritettu yhteistyössä Puolustusvoimien kanssa, osana harjoitustoimintaa. Väitöskirja esittelee mallin kyberturvallisuuden tilannekuvajärjestelmälle, joka käyttää edellä mainittua tunkeutumisen tunnistamisjärjestelmän sensoritietoa tilannekuvan muodostamiseen sensorifuusioon perustuen. Tämän lisäksi esitellään malli kyberturvallisuuspoikkeamatiedon jakamiselle eri organisaatioiden välillä. Myös jatkotutkimuskohteet esitellään aihealueittain. Tärkeimpänä jatkotutkimuskohteena mainittakoon algoritmien tehokkuuden kehittäminen suurilla verkkoliikenne- ja käyttäjämäärillä, sekä tunkeutumisten visualisointi monimutkaisten järjestelmäkokonaisuuksien yhteydessä. Kehitettyä mallia voidaan hyödyntää ja käyttää perustana kyberturvallisuuden tilannekuvajärjestelmäkehitykselle.

REFERENCES

- AlEroud, A. F. & Karabatis, G. 2016. Queryable semantics to detect cyber-attacks: A flow-based detection approach. *IEEE Transactions on Systems, Man, and Cybernetics: Systems PP* (99), 1–17. doi:10.1109/TSMC.2016.2600405.
- Amoli, P. V. 2015. Unsupervised Network Intrusion Detection Systems for Zero-Day Fast-Spreading Network Attacks and Botnets. phdthesis.
- The Apache Software Foundation s. a. Apache HTTP Server Version 2.4, Log Files. <https://httpd.apache.org/docs/2.4/logs.html>. (Accessed: 12 September 2016).
- Arthur, D. & Vassilvitskii, S. 2007. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics. SODA '07, 1027–1035.
- Asaka, M., Onabuta, T., Inoue, T., Okazawa, S. & Goto, S. 2003. Remote attack detection method in ida: Mlsi-based intrusion detection with discriminant analysis. *Electronics and Communications in Japan (Part I: Communications)* 86 (4), 50–62. doi:10.1002/ecja.10053.
- Azimirad, E. & Haddadnia, J. 2015. The comprehensive review on jdl model in data fusion networks: Techniques and methods. (*IJCSIS*) *International Journal of Computer Science and Information Security* 13 (1), 53–60.
- Barford, P., Dacier, M., Dietterich, T. G., Fredrikson, M., Giffin, J., Jajodia, S., Jha, S., Li, J., Liu, P., Ning, P., Ou, X., Song, D., Strater, L., Swarup, V., Tadda, G., Wang, C. & Yen, J. 2010. Cyber sa: Situational awareness for cyber defense. *Cyber Situational Awareness, Advances in Information Security* 46, 3–13. doi:10.1007/978-1-4419-0140-8_1.
- Barnum, S. 2014. Structured Threat Information eXpression (STIX™), white paper, Version 1.1, Revision 1. <http://stixproject.github.io/getting-started/whitepaper/>. (Accessed: 29 September 2016).
- Bass, T. 2000. Intrusion detection systems and multisensor data fusion. *Commun. ACM* 43 (4), 99–105. doi:10.1145/332051.332079.
- Beheshti, M. & Wasniowski, R. A. 2007. Data fusion support for intrusion detection and prevention. In *Fourth International Conference on Information Technology, 2007, ITNG '07*, 966–966. doi:10.1109/ITNG.2007.62.
- Bilmes, J. 1998. A Gentle Tutorial of the EM algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. *International Computer Science Institute ICSI, TR-97-021*.

- Blasch, E., Steinberg, A., Das, S., Llinas, J., Chong, C., Kessler, O., Waltz, E. & White, F. 2013. Revisiting the jdl model for information exploitation. In 2013 16th International Conference on Information Fusion (FUSION), 129–136.
- Botha, M. & von Solms, R. 2004. Utilizing neural networks for effective intrusion detection. In J. H. P. Eloff, L. Labuschagne, M. M. Eloff & H. S. Venter (Eds.) Proceedings of the ISSA 2004 Enabling Tomorrow Conference, 30 June - 1 July 2004, Gallagher Estate, Midrand, South Africa. ISSA, Pretoria, South Africa.
- Botta, A., Dainotti, A. & Pescapé, A. 2012. A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks* 56 (15), 3531–3547. doi:10.1016/j.comnet.2012.02.019.
- Bouhmala, N. 2016. How good is the euclidean distance metric for the clustering problem. In 2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI), 312–315. doi:10.1109/IIAI-AAI.2016.26.
- Braverman, V., Meyerson, A., Ostrovsky, R., Roytman, A., Shindler, M. & Tagiku, B. 2011. Streaming k-means on well-clusterable data. In Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics. SODA '11, 26–40.
- Brehmer, B. 2005. The dynamic ooda loop: Amalgamating boyd's ooda loop and the cybernetic approach to command and control. In 10th International Command and Control Research and Technology Symposium, The Future of C2.
- Castanedo, F. 2013. A review of data fusion techniques. *The Scientific World Journal* 2013, 1–19. doi:10.1155/2013/704504.
- Chapple, M. J., Wright, T. E. & Winding, R. M. 2006. Flow anomaly detection in firewalled networks. In *Securecomm and Workshops, 2006*, 1–6. doi:10.1109/SECCOMW.2006.359576.
- Chen, Y., Lin, Z., Zhao, X., Wang, G. & Gu, Y. 2014. Deep learning-based classification of hyperspectral data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 7 (6), 2094–2107. doi:10.1109/JSTARS.2014.2329330.
- Chwalinski, P., Belavkin, R. & Cheng, X. 2013. Detection of application layer DDoS attacks with clustering and Bayes factors. In 2013 IEEE International Conference on Systems, Man, and Cybernetics, 156–161. doi:10.1109/SMC.2013.34.
- Cisar, P., Bosnjak, S. & Cisar, S. M. 2010. EWMA based threshold algorithm for intrusion detection. *Computing and Informatics* 29, 1089–1101.
- Computer Emergency Readiness Team of Finland CERT-FI 2013. Tietoturvakatsaus (Security Review) 4/2013. https://www.viestintavirasto.fi/attachments/tietoturva/Katsaus_4_13.pdf. (Accessed: 08 September 2016).

- Connolly, J., Davidson, M. & Schmid, C. 2014. The Trusted Automated eXchange of Indicator Information (TAXII™), white paper. <http://taxiiproject.github.io/getting-started/whitepaper/>. (Accessed: 29 September 2016).
- Conti, G., Nelson, J. & Raymond, D. 2013. Towards a cyber common operating picture. In 2013 5th International Conference on Cyber Conflict (CyCon), 1–17.
- Corona, I. & Giacinto, G. 2010. Detection of server-side web attacks. In JMLR: Workshop and Conference Proceedings 11, Workshop on Applications of Pattern Analysis, 160–166.
- Crnkovic, G. D. 2010. Constructive research and info-computational knowledge generation. In L. Magnani, W. Carnielli & C. Pizzi (Eds.) *Model-Based Reasoning in Science and Technology: Abduction, Logic, and Computational Discovery*. Springer Berlin Heidelberg, 359–380. doi:10.1007/978-3-642-15223-8_20.
- Davies, M. & Patel, M. 2016. Are we managing the risk of sharing cyber situational awareness? a uk public sector case study. In 2016 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (CyberSA), 1–2. doi:10.1109/CyberSA.2016.7503292.
- Dehay, D., Leskow, J. & Napolitano, A. 2013. Central limit theorem in the functional approach. *IEEE Transactions on Signal Processing* 61 (16), 4025–4037. doi:10.1109/TSP.2013.2266324.
- Denning, D. E. 1986. An intrusion-detection model. In *IEEE Symposium on Security and Privacy* (1986), 118–131. doi:10.1109/SP.1986.10010.
- Denning, D. E. 1987. An intrusion-detection model. *IEEE Transactions on Software Engineering* SE-13 (2), 222–232. doi:10.1109/TSE.1987.232894.
- Dijkstra, E. W. 1959. A note on two problems in connexion with graphs. *Numer. Math.* 1 (1), 269–271. doi:10.1007/BF01386390.
- Dong, P., Du, X., Zhang, H. & Xu, T. 2016. A detection method for a novel DDoS attack against sdn controllers by vast new low-traffic flows. In 2016 IEEE International Conference on Communications (ICC), 1–6. doi:10.1109/ICC.2016.7510992.
- Duan, L., Yu, F. & Zhan, L. 2016. An improved fuzzy c-means clustering algorithm. In 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), 1199–1204. doi:10.1109/FSKD.2016.7603349.
- Elrawy, M. F., Awad, A. I. & Hamed, H. F. A. 2016. Flow-based features for a robust intrusion detection system targeting mobile traffic. In 2016 23rd International Conference on Telecommunications (ICT), 1–6. doi:10.1109/ICT.2016.7500483.

- Endsley, M. 1995. Toward a theory of situation awareness in dynamic systems. *Human Factors* 37 (1), 32–64. doi:10.1518/001872095779049543.
- Ester, M., Kriegel, H.-P., Sander, J. & Xu, X. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. *Kdd 96*, 226–231.
- European Commission 2013. *Cybersecurity Strategy of the European Union: An Open, Safe and Secure Cyberspace*.
- Europol 2015. *The Internet Organised Crime Threat Assessment (IOCTA)*.
- Farooqi, A. H. & Munir, A. 2008. Intrusion detection system for ip multimedia subsystem using k-nearest neighbor classifier. In *IEEE International Multitopic Conference INMIC 2008*, 423–428. doi:10.1109/INMIC.2008.4777775.
- Fawcett, T. 2006. An introduction to roc analysis. *Pattern Recogn. Lett.* 27 (8), 861–874. doi:10.1016/j.patrec.2005.10.010.
- Fessi, B. A., Abdallah, S. B., Djemaiel, Y. & Boudriga, N. 2011. A clustering data fusion method for intrusion detection system. In *2011 IEEE 11th International Conference on Computer and Information Technology (CIT)*, 539–545. doi:10.1109/CIT.2011.92.
- The Finnish Defence Forces 2016. Finnish authorities developing interagency interoperability in national cyber exercise, Press Release 9 May 2016. http://puolustusvoimat.fi/en/articles/-/asset_publisher/kansallisessa-kyberharjoituksessa-kehitetaan-viranomaisten-operatiivista-yhteistoimintaa. (Accessed: 05 September 2016).
- Floyd, S. & Paxson, V. 2001. Difficulties in simulating the internet. *IEEE/ACM Transactions on Networking* 9 (4), 392–403. doi:10.1109/90.944338.
- Forum of Incident Response and Security Teams FIRST 2016. FIRST announces Traffic Light Protocol (TLP) version 1.0, Press Release 31 August 2016. <https://www.first.org/newsroom/releases/20160831>. (Accessed: 10 October 2016).
- Forum of Incident Response and Security Teams, FIRST 2016. TRAFFIC LIGHT PROTOCOL (TLP), FIRST Standards Definitions and Usage Guidance - Version 1.0. https://www.first.org/_assets/resources/tlp-v1.pdf. (Accessed: 10 October 2016).
- Fraley, C. & Raftery, A. E. 1998. How many clusters? Which clustering method? Answers via model-based cluster analysis. *The Computer Journal* 41 (8), 578–588.
- Franke, U. & Brynielsson, J. 2014. Cyber situational awareness - a systematic review of the literature. *Computers & Security* 46, 18–31. doi:http://dx.doi.org/10.1016/j.cose.2014.06.008.

- Fusano, A., Sato, H. & Namatame, A. 2011. Study of multi-agent based combat simulation for grouped ooda loop. In Proceedings of 2011 SICE Annual Conference (SICE), 131–136.
- F-Secure s. a. Ransomware threat description. <https://www.f-secure.com/v-descs/ransomware.shtml>. (Accessed: 10 September 2016).
- Gallego, G., Cuevas, C., Mohedano, R. & Garc a, N. 2013. On the mahalanobis distance classification criterion for multidimensional normal distributions. *IEEE Transactions on Signal Processing* 61 (17), 4387–4396. doi:10.1109/TSP.2013.2269047.
- Ghorbani, A. A., Lu, W. & Tavallaee, M. 2010. Chapter 2, detection approaches. In *Network Intrusion Detection and Prevention: Concepts and Techniques*. Springer US. *Advances in Information Security* 47. doi:10.1007/978-0-387-88771-5_2.
- Giacobe, N. A. 2010. Application of the jdl data fusion process model for cyber security. *Proc. SPIE* 7710. doi:10.1117/12.850275.
- Gifty Jeya, P., Ravichandran, M. & Ravichandran, C. S. 2012. Efficient classifier for r2l and u2r attacks. *International Journal of Computer Applications* 45 (21), 28–32.
- Gollmann, D. 2011. *Computer Security* (3rd edition edition). Wiley.
- Hall, D. L. & Llinas, D. L. 1997. An introduction to multisensor data fusion. *Proceedings of the IEEE* 85 (1), 6–23. doi:10.1109/5.554205.
- Hamming, R. W. 1950. Error detecting and error correcting codes. *The Bell System Technical Journal* 29 (2), 147–160. doi:10.1002/j.1538-7305.1950.tb00463.x.
- Hern andez-Pereira, E., Su arez-Romero, J., Fontenla-Romero, O. & Alonso-Betanzos, A. 2009. Conversion methods for symbolic features: A comparison applied to an intrusion detection problem. *Expert Systems with Applications* 36 (7), 10612–10617. doi:http://dx.doi.org/10.1016/j.eswa.2009.02.054.
- Hirsimaki, T., Pylkkonen, J. & Kurimo, M. 2009. Importance of high-order n-gram models in morph-based speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing* 17 (4), 724–732. doi:10.1109/TASL.2008.2012323.
- Hong, S.-S. & Wu, S. F. 2006. On interactive internet traffic replay. In A. Valdes & D. Zamboni (Eds.) *Lecture Notes in Computer Science, Recent Advances in Intrusion Detection: 8th International Symposium, RAID 2005*, Vol. 3858. Springer Berlin Heidelberg, 247–264. doi:10.1007/11663812_13.
- The Internet Society Network Working Group 2004. RFC3917, Requirements for IP Flow Information Export (IPFIX).

- Izakian, H. & Pedrycz, W. 2013. Anomaly detection in time series data using a fuzzy c-means clustering. In 2013 Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS), 1513–1518. doi:10.1109/IFSA-NAFIPS.2013.6608627.
- Jabez, J. & Muthukumar, B. 2015. Intrusion detection system (ids): Anomaly detection using outlier detection approach. *Procedia Computer Science* 48, 338–346. doi:10.1016/j.procs.2015.04.191.
- JAMK University of Applied Sciences, Institute of Information Technology, JYVSECTEC s. a. RGCE Cyber Range. <http://www.jyvsectec.fi/en/rgce/>. (Accessed: 05 September 2016).
- Juvonen, A. 2014. *Intrusion Detection Applications Using Knowledge Discovery and Data Mining*. phdthesis.
- Kao, D. Y. 2015. Performing an apt investigation: Using people-process-technology-strategy model in digital triage forensics. In 2015 IEEE 39th Annual Computer Software and Applications Conference (COMPSAC), Vol. 3, 47–52. doi:10.1109/COMPSAC.2015.10.
- Karhunen, J. & Joutsensalo, J. 1995. Generalizations of principal component analysis, optimization problems, and neural networks. *Neural Networks* 8 (4), 549–562. doi:[http://dx.doi.org/10.1016/0893-6080\(94\)00098-7](http://dx.doi.org/10.1016/0893-6080(94)00098-7).
- Kasanen, E., Lukka, K. & Siitonen, A. 1993. The constructive approach in management accounting research. *Journal of Management Accounting Research* 5, 243–264.
- Kaushik, S. S. & Deshmukh, P. R. 2011. Detection of attacks in an intrusion detection system. (*IJCSIT*) *International Journal of Computer Science and Information Technologies* 2, 982–986.
- Khaleghi, B., Khamis, A., Karray, F. O. & Razavi, S. N. 2013. Multisensor data fusion: A review of the state-of-the-art. *Inf. Fusion* 14 (1), 28–44. doi:10.1016/j.inffus.2011.08.001.
- Kohonen, T. 1990. The self-organizing map. *Proceedings of the IEEE* 78 (9), 1464–1480. doi:10.1109/5.58325.
- Kruegel, C., Mutz, D., Robertson, W. & Valeur, F. 2003. Bayesian event classification for intrusion detection. In *Proceedings of the 19th Annual Computer Security Applications Conference*. ACSAC '03.
- Kruegel, C., Vigna, G. & Robertson, W. 2005. A multi-model approach to the detection of web-based attacks. *Comput. Netw.* 48 (5), 717–738. doi:10.1016/j.comnet.2005.01.009.

- Kumar, G., Kumar, K. & Sachdeva, M. 2010. The use of artificial intelligence based techniques for intrusion detection: a review. *Artificial Intelligence Review* 34 (4), 369–387. doi:10.1007/s10462-010-9179-5.
- Kumar, M., Hanumanthappa, M. & Suresh Kumar, T. V. 2011. Intrusion detection system - false positive alert reduction technique. *ACEEE International Journal on Network Security* 2 (3), 37–40. doi:01.IJNS.02.03.104.
- Kumar, P. & Pateriya, R. K. 2013. Enhanced intrusion detection system for input validation attacks in web application. *IJCSI International Journal of Computer Science Issues* 10, 435–441.
- Kuusisto, R., Kuusisto, T. & Armistead, L. 2005. Common operational picture, situation awareness and information operations. In *Proceedings of 4th European Conference on Information Warfare and Security*, Glamorgan, UK. Academic Conferences Limited, Reading, UK, 175–185.
- Laaperi, L. & Vankka, J. 2015. Architecture for a system providing a common operating picture of critical infrastructure. In *2015 IEEE International Symposium on Technologies for Homeland Security (HST)*, 1–6. doi:10.1109/THS.2015.7446228.
- Laskov, P., Düssel, P., Schäfer, C. & Rieck, K. 2005. Learning Intrusion Detection: Supervised or Unsupervised? doi:10.1007/11553595_6.
- Lehtiranta, L., Junnonen, J.-M., Kärnä, S. & Pekuri, L. 2015. The constructive research approach: Problem solving for complex projects. In B. Pasian (Ed.) *Designs, Methods and Practices for Research of Project Management*. Gower Publishing Limited.
- Li, M., Huang, W., Wang, Y., Fan, W. & Li, J. 2016. The study of apt attack stage model. In *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, 1–5. doi:10.1109/ICIS.2016.7550947.
- Li, X. Y., Li, G. S. & Zhang, S. T. 2009. Routing space internet based on dijkstra's algorithm. In *2009 First Asian Himalayas International Conference on Internet, AH-ICI*, 1–4. doi:10.1109/AHICI.2009.5340354.
- Li, X. Y., Li, G. S. & Zhang, S. T. 2010. Routing space internet based on dijkstra's algorithm. In *2010 Second International Conference on Networks Security Wireless Communications and Trusted Computing (NSWCTC)*, 118–121. doi:10.1109/NSWCTC.2010.163.
- Li, Z., Filev, D. P., Kolmanovsky, I., Atkins, E. & Lu, J. 2016. A new clustering algorithm for processing gps-based road anomaly reports with a mahalanobis distance. *IEEE Transactions on Intelligent Transportation Systems* PP (99), 1–9. doi:10.1109/TITS.2016.2614350.

- Lippmann, R. P., Fried, D. J., Graf, I., Haines, J. W., Kendall, K. R., McClung, D., Weber, D., Webster, S. E., Wyschogrod, D., Cunningham, R. K. & Zissman, M. A. 2000. Evaluating intrusion detection systems: the 1998 darpa off-line intrusion detection evaluation. In Proceedings of 2000 DARPA Information Survivability Conference and Exposition DISCEX, Vol. 2, 12–26. doi:10.1109/DISCEX.2000.821506.
- Liu, X., Wang, H., Lai, J., Liang, Y. & Yang, C. 2007a. Multiclass support vector machines theory and its data fusion application in network security situation awareness. In 2007 International Conference on Wireless Communications, Networking and Mobile Computing, 6349–6352. doi:10.1109/WICOM.2007.1557.
- Liu, X., Wang, H., Liang, Y. & Lai, J. 2007b. Heterogeneous multi-sensor data fusion with multi-class support vector machines: Creating network security situation awareness. In 2007 International Conference on Machine Learning and Cybernetics, 2689–2694. doi:10.1109/ICMLC.2007.4370604.
- Lloyd, S. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory* 28 (2), 129–137. doi:10.1109/TIT.1982.1056489.
- Loh, W.-K. & Park, Y.-H. 2014. A survey on density-based clustering algorithms. In Y.-S. Jeong, Y.-H. Park, C.-H. R. Hsu & J. J. H. Park (Eds.) *Lecture Notes in Electrical Engineering* 280, Ubiquitous Information Technologies and Applications. Springer Berlin Heidelberg, 775–780. doi:10.1007/978-3-642-41671-2_98.
- Lu, W. & Tong, H. 2009. Detecting Network Anomalies Using CUSUM and EM Clustering. doi:10.1007/978-3-642-04843-2_32.
- Luo, S. & Marin, G. A. 2005. Realistic internet traffic simulation through mixture modeling and a case study. In Proceedings of the 37th Conference on Winter Simulation. Winter Simulation Conference. WSC '05, 2408–2416.
- The MITRE Corporation s. a. Cyber Observable eXpression (CybOX™) - homepage. <https://cyboxproject.github.io/>. (Accessed: 29 September 2016).
- Mokarian, A., Faraahi, A. & Delavar, A. G. 2013. False positives reduction techniques in intrusion detection systems-a review. *IJCSNS International Journal of Computer Science and Network Security* 13 (10), 128–134.
- Mukkamala, S. & Sung, A. H. 2003. A comparative study of techniques for intrusion detection. In Proceedings of 15th IEEE International Conference on Tools with Artificial Intelligence, 2003, 570–577. doi:10.1109/TAI.2003.1250243.
- Munz, G., Li, S. & Carle, G. 2007. Traffic anomaly detection using kmeans clustering. In In GI/ITG Workshop MMBnet.

- Muraleedharan, N., Parmar, A. & Kumar, M. 2010. A flow based anomaly detection system using chi-square technique. In 2010 IEEE 2nd International Advance Computing Conference (IACC), 285–289. doi:10.1109/IADCC.2010.5422996.
- National Institute of Standards and Technology NIST 2007. Guide to Intrusion Detection and Prevention Systems (IDPS), NIST Special Publication 800-94.
- Nguyen-Tuong, A., Guarnieri, S., Greene, D., Shirley, J. & Evans, D. 2005. Automatically hardening web applications using precise tainting. In Security and Privacy in the Age of Ubiquitous Computing. Springer. doi:10.1007/0-387-25660-1_20.
- Ozcelik, I. & Brooks, R. R. 2016. Cusum - entropy: an efficient method for ddos attack detection. In 2016 4th International Istanbul Smart Grid Congress and Fair (ICSG), 1–5. doi:10.1109/SGCF.2016.7492429.
- Paliwal, S. & Gupta, R. 2012. Denial-of-service, probing & remote to user (r2l) attack detection using genetic algorithm. *International Journal of Computer Applications* 60 (19), 57–62.
- Piirainen, K. A. & Gonzalez, R. A. 2013. Seeking constructive synergy: Design science and the constructive research approach. In J. vom Brocke, R. Hekkala, S. Ram & M. Rossi (Eds.) *Lecture Notes in Computer Science, 8th International Conference on Design Science at the Intersection of Physical and Virtual Design DESRIST, Vol. 7939*. Springer Berlin Heidelberg, 59–72. doi:10.1007/978-3-642-38827-9_5.
- Rafsanjani, M. K., Varzaneh, Z. A. & Chukanlo, N. E. 2012. A survey of hierarchical clustering algorithms. *The Journal of Mathematics and Computer Science* 5 (3), 229–240.
- Ranjan, S., Swaminathan, R., Uysal, M., Nucci, A. & Knightly, E. 2009. Ddos-shield: Ddos-resilient scheduling to counter application layer attacks. *IEEE/ACM Transactions on Networking* 17 (1), 26–39. doi:10.1109/TNET.2008.926503.
- Saber, M., Bouchentouf, T., Benazzi, A. & Azizi, M. 2010. Amelioration of attack classifications for evaluating and testing intrusion detection system. *Journal of Computer Science* 6, 716–722. doi:10.3844/jcssp.2010.716.722.
- Saber, M., Bouchentoufand, T. & Benazzi, A. 2012. Generation of attack scenarios for evaluating ids. *IACSIT International Journal of Engineering and Technology* 4 (3), 298–302. doi:10.7763/IJET.2012.V4.369.
- Saboor, A. & Aslam, B. 2015. Analyses of flow based techniques to detect distributed denial of service attacks. In 2015 12th International Bhurban Conference on Applied Sciences and Technology (IBCAST), 354–362. doi:10.1109/IBCAST.2015.7058529.

- Sakurada, M. & Yairi, T. 2014. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In 2014 2nd Workshop on Machine Learning for Sensory Data Analysis. ACM. MLSDA'14, 4–11. doi:10.1145/2689746.2689747.
- Saranya, C. & Manikandan, G. 2013. A study on normalization techniques for privacy preserving data mining. *International Journal of Engineering and Technology (IJET)* 5 (3), 2701–2704.
- Schall, M., Schambach, M. P. & Franz, M. O. 2016. Increasing robustness of handwriting recognition using character n-gram decoding on large lexica. In 2016 12th IAPR Workshop on Document Analysis Systems (DAS), 156–161. doi:10.1109/DAS.2016.43.
- Secretariat of the Security Committee 2013. Finland's Cyber security Strategy, Government Resolution 24.1.2013.
- Sheikhan, M. & Jadidi, Z. 2014. Flow-based anomaly detection in high-speed links using modified gsa-optimized neural network. *Neural Computing and Applications* 24 (3), 599–611. doi:10.1007/s00521-012-1263-0.
- Shiravi, A., Shiravi, H., Tavallaee, M. & Ghorbani, A. A. 2012. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Comput. Secur.* 31 (3), 357–374. doi:10.1016/j.cose.2011.12.012.
- Spathoulas, G. P. & Katsikas, S. K. 2010. Reducing false positives in intrusion detection systems. *Computers & Security* 29 (1), 35–44. doi:http://dx.doi.org/10.1016/j.cose.2009.07.008.
- Steinberg, A. N., Bowman, C. L. & White, F. E. 1999. Revisions to the jdl data fusion model. *Proc. SPIE* 3719, 430–441. doi:10.1117/12.341367.
- Stevanovic, D. & Vlajic, N. 2014. Next generation application-layer DDoS defenses: Applying the concepts of outlier detection in data streams with concept drift. In 2014 13th International Conference on Machine Learning and Applications (ICMLA), 456–462. doi:10.1109/ICMLA.2014.80.
- Stevanovic, D., Vlajic, N. & An, A. 2011. Unsupervised clustering of web sessions to detect malicious and non-malicious website users. *Procedia Computer Science* 5, 123 – 131. doi:http://dx.doi.org/10.1016/j.procs.2011.07.018.
- Stevanovic, D., Vlajic, N. & An, A. 2013. Detection of malicious and non-malicious website visitors using unsupervised neural network learning. *Applied Soft Computing* 13 (1), 698 – 708. doi:http://dx.doi.org/10.1016/j.asoc.2012.08.028.
- Suen, C. Y. 1979. n-gram statistics for natural language understanding and text processing. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-1* (2), 164–172. doi:10.1109/TPAMI.1979.4766902.

- Swart, I., Irwin, B. & Grobler, M. 2015. Multisensor national cyber security data fusion. In Proceedings of the 10th International Conference on Cyber Warfare and Security ICCWS-2015, 320–328.
- Syarif, I., Prugel-Bennett, A. & Wills, G. 2012. Unsupervised clustering approach for network anomaly detection. In R. Benlamri (Ed.) Communications in Computer and Information Science, 4th International Conference on Networked Digital Technologies, Vol. 293. Springer Berlin Heidelberg, 135–145. doi:10.1007/978-3-642-30507-8_13.
- Tadda, G. 2008. Measuring performance of cyber situation awareness systems. In 2008 11th International Conference on Information Fusion, 1–8.
- Tadda, G. P. & Salerno, J. S. 2010. Overview of cyber situation awareness. In S. Jajodia, P. Liu, V. Swarup & C. Wang (Eds.) Cyber Situational Awareness - Issues and Research, Vol. 46. Springer. Advances in Information Security, 15–35. doi:10.1007/978-1-4419-0140-8_2.
- Tankard, C. 2011. Advanced persistent threats and how to monitor and deter them. *Network Security* 2011 (8), 16–19. doi:http://dx.doi.org/10.1016/S1353-4858(11)70086-1.
- Tax, D. M. & Duin, R. P. 2004. Support vector data description. *Machine Learning* 54 (1), 45–66. doi:10.1023/B:MACH.0000008084.60811.49.
- Tian, J., Zhao, W. & Du, R. 2005. D-s evidence theory and its data fusion application in intrusion detection. In Lecture Notes in Computer Science: 2005 International conference on Computational Intelligence and Security CIS'05, Vol. 3802. Springer, 244–251. doi:10.1007/11596981_36.
- United Kingdom National Crime Agency 2016. Cyber Crime Assessment 2016, Version 1.2.
- The United States Computer Emergency Readiness Team, US-CERT s. a. Automated Indicator Sharing (AIS). <https://www.us-cert.gov/ais>. (Accessed: 27 September 2016).
- The United States Congress 2015a. H.R.234 - Cyber Intelligence Sharing and Protection Act. <https://www.congress.gov/bill/114th-congress/house-bill/234>. (Accessed: 27 September 2016).
- The United States Congress 2015b. S.754 - Cybersecurity Information Sharing Act of 2015. <https://www.congress.gov/bill/114th-congress/senate-bill/754>. (Accessed: 27 September 2016).
- Uppada, S. K. 2014. Centroid based clustering algorithms - a clarion study. *International Journal of Computer Science and Information Technologies (IJCSIT)* 5 (6), 7309–7313.

- Vardasbi, A., Salmasizadeh, M. & Mohajeri, J. 2011. Multiple-chi-square tests and their application on distinguishing attacks. In 2011 8th International ISC Conference on Information Security and Cryptology (ISCISC), 55–60. doi:10.1109/ISCISC.2011.6062336.
- The White House, signed by President Barack Obama 2011. International Strategy for cyberspace; Prosperity, Security, and Openness in a Networked World.
- Xie, Q., Zhang, Y., Jia, H. & Lu, Y. 2014. Research on mahalanobis distance algorithm optimization based on opencl. In 2014 IEEE International Conference on High Performance Computing and Communications (HPCC), 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security (CSS) and 2014 IEEE 11th Intl Conf on Embedded Software and Syst (ICCESS), 84–91. doi:10.1109/HPCC.2014.19.
- Xu, C., Zhao, G., Xie, G. & Yu, S. 2014. Detection on application layer DDoS using random walk model. In 2014 IEEE International Conference on Communications (ICC), 707–712. doi:10.1109/ICC.2014.6883402.
- Ye, C., Zheng, K. & She, C. 2012. Application layer ddos detection using clustering analysis. In 2012 2nd International Conference on Computer Science and Network Technology (ICCSNT), 1038–1041. doi:10.1109/ICCSNT.2012.6526103.
- Ye, N., Borrer, C. M. & Parmar, D. 2003. Scalable chi-square distance versus conventional statistical distance for process monitoring with uncorrelated data variables. *Quality and Reliability Engineering International* 19 (6), 505–515. doi:10.1002/qre.539.
- Yu, W., Xu, G., Chen, Z. & Moulema, P. 2013. A cloud computing based architecture for cyber security situation awareness. In 2013 4th International Workshop on Security and Privacy in Cloud Computing, IEEE Conference on Communications and Network Security (CNS), 488–492. doi:10.1109/CNS.2013.6682765.
- Zhang, Y., Huang, S., Guo, S. & Zhu, J. 2011. Multi-sensor data fusion for cyber security situation awareness. *Procedia Environmental Sciences* 10, 1029–1034. doi:http://dx.doi.org/10.1016/j.proenv.2011.09.165.
- Zhao, G., Xu, K., Xu, L. & Wu, B. 2015. Detecting apt malware infections based on malicious dns and traffic analysis. *IEEE Access* 3, 1132–1142. doi:10.1109/ACCESS.2015.2458581.
- Zhao, X., Jiang, H. & Jiao, L. 2009. A data fusion based intrusion detection model. In 2009 First International Workshop on Education Technology and Computer Science, ETCS '09, 1017–1021. doi:10.1109/ETCS.2009.232.
- Zhou, Y., Abad, C., Taylor, J., Yurcik, W., Sengul, C. & Rowe, K. 2003. Log correlation for intrusion detection: A proof of concept. 2003 Annual Computer Security Applications Conference, 255–. doi:http://doi.ieeecomputersociety.org/10.1109/CSAC.2003.1254330.

ORIGINAL PAPERS

PI

ANALYSIS OF HTTP REQUESTS FOR ANOMALY DETECTION OF WEB ATTACKS

by

Mikhail Zolotukhin, Timo Hämäläinen, Tero Kokkonen, Jarmo Siltanen 2014

The 12th IEEE International Conference on Dependable, Autonomic and Secure
Computing, DASC 2014, Dalian, China, August 24-27, 2014, Proceedings

Reproduced with kind permission of IEEE,
Copyright © 2014, IEEE.

Analysis of HTTP requests for anomaly detection of web attacks

Mikhail Zolotukhin, Timo Hämäläinen
Department of Mathematical Information Technology
University of Jyväskylä, Jyväskylä, Finland
email: {mikhail.m.zolotukhin, timo.t.hamalainen}@jyu.fi

Tero Kokkonen, Jarmo Siltanen
Institute of Information Technology
JAMK University of Applied Sciences, Jyväskylä, Finland
email: {tero.kokkonen, jarmo.siltanen}@jamk.fi

Abstract—Attacks against web servers and web-based applications remain a serious global network security threat. Attackers are able to compromise web services, collect confidential information from web data bases, interrupt or completely paralyze web servers. In this study, we consider the analysis of HTTP logs for the detection of network intrusions. First, a training set of HTTP requests which does not contain any attacks is analyzed. When all relevant information has been extracted from the logs, several clustering and anomaly detection algorithms are employed to describe the model of normal users behavior. This model is then used to detect network attacks as deviations from the norms in an online mode. The simulation results presented show that, compared to other data mining algorithms, the method results in a higher accuracy rate.

I. INTRODUCTION

In modern society, the use of computer technologies, both for work and personal use, is growing with time. Unfortunately, with the growing number and size of computer networks and systems their vulnerability is also increasing. One of the most popular attack targets are web-servers and web-based applications. Usually, the users of web-servers and web-based applications request and send information using queries, which in HTTP traffic are strings containing a set of parameters having some values. Attackers are able to manipulate these queries and create requests which can corrupt the server or collect confidential information [1], [2]. In addition, certain types of intrusions such as scanning and probing attacks are able to identify running services on a web server with the view to compromise it. Finally, password brute-force attacks aim to give an attacker administrators privileges for the web server, whereas Denial of Service (DoS) allows attackers to interrupt or completely paralyze the server. To ensure the security of web-servers and web-based applications Intrusion Detection Systems (IDSs) can be used. As a rule, IDS gathers data from the system under inspection, analyzes this data to detect suspicious activities and determines suitable responses to these activities [3].

There are two basic approaches for detecting intrusions from the network data: misuse detection and anomaly detection [4]. The misuse detection approach allows us to detect attacks by searching for predefined attack signatures. Since this approach is usually accurate, it is successfully used in commercial intrusion detection. However, misuse detection approach cannot detect attacks for which it has not been programmed. Therefore, it is prone to ignore all new types of attack if the system is not kept up to date with the latest

intrusions. The anomaly detection approach learns the normal behavior of users and detects intrusions by observing patterns that deviate from established norms. Thus, systems which use the anomaly detection approach are able to detect zero-day attacks. However, the number of false alerts will probably increase because not all anomalies are intrusions.

Nowadays, many studies concentrate on approaches that are based on to be able to detect network intrusions that probably have never been seen before. Study [5] considers a method of detecting web attacks which is based on dimensionality reduction by applying diffusion maps and on subsequent application of spectral clustering. Paper [6] introduces an adaptive anomaly detection approach for web attacks that utilizes hidden Markov models to adaptively identify normal HTTP requests. Application of growing hierarchical self-organizing maps for finding intrusive HTTP queries is investigated in [7]. In study [8], a novel anomaly-based HTTP-flooding detection approach based on a density-based cluster algorithm is proposed.

In this study, we consider the analysis of HTTP logs for the detection of network intrusions. Normal user behavior is analyzed with the help of a training set of HTTP requests which does not contain any attacks. After all relevant information is extracted from these logs, several data mining techniques are employed to describe the model of legitimate user behavior. These techniques are based on unsupervised machine learning and, therefore, allow one to build the model without a priori knowledge about the structure of the data in the training set. Furthermore, all parameters are calculated beforehand so that the system administrator is not supposed to adjust them during the training phase. The model is then used to detect network intrusions within recent HTTP requests in an online mode. Finally, the proposed framework is tested with the help of a web server installed in a realistic cyber environment that enables one to construct real attack vectors against this server.

The remainder of this paper is organized as follows. Section II describes the process of data acquisition and feature extraction from network logs. In Section III, a method that is based on a data mining approach for detecting intrusions is described. Results of the application of a proposed algorithm to logs of a real web server are presented in Section IV. Section V concludes this paper and outlines future work.

II. FEATURE EXTRACTION

In order to detect attacks against a web server, two types of statistics are extracted from HTTP logs. HTTP logs can include information about a user's IP address, time and timezone, the HTTP request including HTTP method used, requested resource with several attributes, server response code, amount of data transferred, referer and browser software used. Here is an example of a single line from an Apache server log file in which information is stored in a combined log format [9]:

```
127.0.0.1 - frank [10/Oct/2000:13:55:36
-0700] "GET /resource?parameter1=value1&
parameter2=value2 HTTP/1.0" 200 2326
"http://www.example.com/start.html"
"Mozilla/4.08 [en] (Win98; I ;Nav)"
```

A. Statistics per request

First, we concentrate on analyzing HTTP queries which can include the requested web resources with several attributes and the user's browser agents. A requested web resource and its attributes can be used by an attacker to perform such dangerous attacks as SQL injections, buffer overflow and directory traversal attacks. The attacker can also inject malicious code to the user agent field to construct various kinds of attacks based on HTTP response splitting or malicious redirecting [10]. For this reason, for each HTTP request the following parameters are extracted: the resource requested, values of the query's attributes and the user agent. The analysis of each of these parameters is considered separately.

To extract features from all of these parameters n-gram models are applied. An n-gram is a sub-sequence of n overlapping items (characters, letters, words, etc) from a given sequence. N-gram models are widely used in statistical natural language processing [11] and speech recognition [12]. For example, for the line: /wp-content/lapland2-529x320.jpg 2-grams are: 'w', 'wp', 'p-', '-c', 'co', 'on', 'nt', 'j', 'jp', 'pg'. Since, when making code injections, attackers use specific combinations of non-alphanumeric symbols, the usage of those symbols is of the most interest. To deal with them, all numbers and Latin letters are considered as the same character. In other words, for a 2-gram model there is no difference between 'w' and 'r' and between 'd2' and 'x3'. It allows us to reduce the dimensionality of the problem significantly without losing relevant information needed for intrusion detection. All unique n-grams are counted to calculate feature vectors. A size of each feature vector is equal to the number of all possible n-grams used in the training set.

In this study, three types of feature vectors are extracted from each of the mentioned textual parameters.

- 1) Model 1: the j -th feature of the i -th feature vector x_{ij} is equal to the frequency of occurrences of j -th n-gram in the i -th line f_{ij}^n :

$$x_{ij} = f_{ij}^n. \quad (1)$$

- 2) Model 2: the j -th feature of the i -th feature vector x_{ij} is calculated as the ratio of the frequency of occurrences of j -th n-gram in the i -th line f_{ij}^n and

the frequency of occurrences of the j -th n-gram in the whole training set F_j^n :

$$x_{ij} = \frac{f_{ij}^n}{F_j^n}. \quad (2)$$

- 3) Model 3: the j -th feature of the i -th feature vector x_{ij} is defined as the ratio of the frequency of occurrences of j -th n-gram in the i -th line f_{ij}^n and the product of frequencies of symbols contained in this n-gram calculated for the whole training set:

$$x_{ij} = \frac{f_{ij}^n}{\prod_{k=1}^n F_{jk}^1}, \quad (3)$$

where F_{jk}^1 is the frequency of occurrences of the k -th symbol of the j -th n-gram in the training set.

B. Statistics per time bin

As a rule, the detection of attacks such as scanning and bruteforcing requires consideration of network traffic as time series. For this reason, the analyzed time period is divided into overlapping time bins. First, for each HTTP request in each time bin we extract the following parameters: user's IP address, requested web resource with its attributes, the amount of transferred bytes and server's response code. In addition, the total number of HTTP requests sent into each time bin is taken into account.

Per-request statistics is then reduced into aggregated time-bin based statistics. For this purpose, a sample entropy of every extracted parameter is calculated for each time bin. Let us assume that in the i -th time bin the j -th parameter has N_{ij} unique values which appear with frequencies $p_{ij}^1, \dots, p_{ij}^{N_{ij}}$. In this case, sample entropy E_{ij} for the j -th parameter in the i -th time bin is defined as follows:

$$E_{ij} = - \sum_{k=1}^{N_{ij}} p_{ij}^k \log_2 p_{ij}^k. \quad (4)$$

Sample entropy allows one to capture the degree of dispersal or concentration of the parameter's distribution. E_{ij} is equal to zero when all values of the j -th parameter are the same, and it takes on its maximal value $\log_2 N_{ij}$ when $p_{ij}^1 = p_{ij}^2 = \dots = p_{ij}^{N_{ij}}$. The matrix of entropy values is used as a feature matrix for the detection of several web attacks.

III. METHOD

When all relevant features have been extracted from network logs, several data mining techniques are applied to feature matrices obtained to construct a model of normal users behavior. For different HTTP request parameters, different techniques are used, because they allow us to extract a model of normal users behavior more accurately (see Section IV). All these techniques employ an unsupervised machine learning approach which allows one to build the model without a priori knowledge about the structure of the data in the training set. All the feature vectors extracted from the training set correspond to legitimate HTTP requests. In addition, all algorithm parameters are calculated beforehand so that the system administrator is not supposed to adjust them during the training phase. Once the normal user behavior model is found, it can be further used to classify new HTTP connections.

A. Web resource

The construction of a normal users behavior model starts with the analysis of requested web resources. First, the dimensionality of feature vectors corresponding to web resources is reduced with the help of Principal Component Analysis (PCA). PCA is an unsupervised dimensionality reduction technique that maps the data to a new coordinate system where the axis directions contain maximal variance [19]. These axes are ordered in such a way that the greatest variance by any projection of the data lies on the first coordinate, which is known as the first principal component, the second greatest variance is on the second coordinate, and so on. This mapping is performed by analyzing the eigenvectors of the covariance matrix calculated for extracted feature vectors. In this study, the principal components which correspond to the non-zero eigenvalues of the covariance matrix are used for the transformation.

After that, Support Vector Data Description (SVDD) is applied to the transformed vectors of the training set. By constructing a hypersphere which contains all data in one category, an SVDD finds a closed boundary around the data belonging to that category [19]. This sphere is characterized by center c and radius $R > 0$.

Let us assume that there are q transformed feature vectors x_1, x_2, \dots, x_q which correspond to the requested web resources of the training set. The center c of SVDD hypersphere (c, R) for these vectors can be defined as $c = \sum_{i=1}^q \alpha_i x_i$. Here $\alpha = (\alpha_1, \dots, \alpha_q)$ is the solution of the following optimization problem:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^q \alpha_i (\phi(x_i)^T \phi(x_i)) - \sum_{i=1}^q \sum_{j=1}^q \alpha_i \alpha_j \phi(x_i)^T \phi(x_j), \\ \text{subject to} \quad & \begin{cases} \sum_{i=1}^q \alpha_i = 1, \\ 0 \leq \alpha_i \leq C, \forall i \in \{1, \dots, q\}, \end{cases} \end{aligned} \quad (5)$$

where function $\phi(x)$ maps x to a higher dimensional space and C is the penalty parameter which controls the trade-off between the hypersphere volume and classification errors. In this study, parameter C is equal to one and a linear kernel ϕ is used, i.e. $\phi(x) = x$. Radius R of sphere (c, R) is calculated as follows:

$$\begin{aligned} R = & \phi(x_k)^T \phi(x_k) - 2 \sum_{i:\alpha_i < C} \alpha_i \phi(x_i)^T \phi(x_k) + \\ & + \sum_{i:\alpha_i < C} \sum_{j:\alpha_j < C} \alpha_i \alpha_j \phi(x_i)^T \phi(x_j), \end{aligned} \quad (6)$$

where x_k is any vector from a dataset such as $\alpha_k < C$.

Once optimal hypersphere (c, R) is found, a vector x corresponding to the resource parameter of a new HTTP request is classified as an attack if

$$\begin{aligned} R^2 - (\phi(x)^T \phi(x)) - 2 \sum_{i=1}^q \alpha_i (\phi(x)^T \phi(x_i)) + \\ \sum_{i=1}^q \sum_{j=1}^q \alpha_i \alpha_j \phi(x_i)^T \phi(x_j) < 0. \end{aligned} \quad (7)$$

B. Query attributes

To detect anomalies within values of queries attributes, the well-known k-means technique is applied. K-means is a unsupervised partitioning technique which classifies a dataset of objects into k clusters. As a rule, the number of clusters k is given beforehand. This algorithm tries to minimize the sum of distances between each feature vector and the mean value of the cluster this vector belongs to.

The most common of these algorithms uses an iterative refinement technique [13]. First, k means m_1, \dots, m_k are initiated, e.g. by randomly choosing k feature vectors from the dataset. After that, each feature vector x is assigned to the cluster corresponding to the least distant mean. Thus, the i -th cluster C_i is formed as follows:

$$C_i = \{x_j : d(x_j, m_i)^2 \leq d(x_j, m_l)^2 \forall l, 1 \leq l \leq k\}, \quad (8)$$

where d is the distance function. For recently-built clusters, new means are calculated:

$$m_i^{new} = \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j. \quad (9)$$

These two steps, the assignment of feature vectors to clusters and the calculation of new means, are repeated until there are no longer changes in clusters during the assignment step

The clustering result strongly depends on the number of clusters. In this study, we select this number based on study [14]. Furthermore, the distance function for two feature vectors x_i and x_j of length l is defined as one minus the linear correlation between these vectors:

$$d(x_i, x_j) = \frac{\sum_{k=1}^l (x_{ik}(x_{jk} - x_j^\mu) - x_i^\mu(x_{jk} - x_j^\mu))}{\sqrt{\left(\sum_{k=1}^l (x_{ik} - x_i^\mu)^2\right) \left(\sum_{k=1}^l (x_{jk} - x_j^\mu)^2\right)}}, \quad (10)$$

where

$$x_i^\mu = \frac{\sum_{k=1}^l x_{ik}}{l}, \quad x_j^\mu = \frac{\sum_{k=1}^l x_{jk}}{l}. \quad (11)$$

After the number of clusters is selected, k-means is applied to feature vectors of the training set to find the optimal clusters of web resources. For each such cluster its center and radius are defined. The center of the i -th cluster corresponds to mean value m_i of vectors belonging to the cluster C_i . Radius r_i is calculated as

$$r_i = \frac{|C_i| + 1}{|C_i|} \cdot \max_{x \in C_i} d(m_i, x). \quad (12)$$

Vector x corresponding to a value of an attribute of a new HTTP request is classified as anomalous if it does not belong to any of the spheres (m_i, r_i) obtained in the training phase, i.e. $d(x, m_i) > r_i, \forall i \in \{1, 2, \dots, k\}$. If at least one attribute of a new HTTP request is anomalous then the whole request is classified as an attack.

C. User Agent

The detection of intrusions within user agent values is carried out with the help of Density-Based Spatial Clustering of Applications with Noise (DBSCAN). DBSCAN is a powerful density-based clustering algorithm, which is often

used for detecting outliers. It discovers clusters in the training dataset, starting from the estimated density distribution of feature vectors [15]. All cluster-less vectors are classified as anomalies. DBSCAN requires two parameters: the size of neighborhood ε and the minimum number of points required to form a cluster N_{min} .

The algorithm starts with an arbitrary feature vector x that has not been checked. The number of feature vectors $N_\varepsilon(x)$ contained in the ε -neighborhood of x is found and compared to N_{min} :

$$\begin{cases} \text{If } N_\varepsilon(x) < N_{min}, \text{ then } x \text{ is labeled as noise,} \\ \text{If } N_\varepsilon(x) \geq N_{min}, \text{ then } x \text{ is a part of a cluster.} \end{cases} \quad (13)$$

Vectors marked as noise might later be discovered as a part of another vector ε -environment and hence be made a part of a cluster. If a vector is found to be a part of a cluster, its ε -neighborhood is also part of that cluster. After that, each point y contained in the ε -neighborhood is checked. If y is density-reached from x with respect to ε and N_{min} , it is added to the cluster. Vector y is density-reachable from x with respect to ε and N_{min} , if there is a chain of points x_1, x_2, \dots, x_m , where $x_1 = x$ and $x_m = y$, such that $\forall i \in \{1, 2, \dots, m-1\}$ the two following conditions are satisfied:

$$\begin{cases} d^E(x_i, x_{i+1}) \leq \varepsilon, \\ N_\varepsilon(x_i) \geq N_{min}, \end{cases} \quad (14)$$

where $d^E(x_i, x_{i+1})$ is the Euclidean distance between x_i and x_{i+1} . The cluster is built when all vectors density-reachable from x have been found. Then, a new unvisited vector is processed, leading to a discovery of a further cluster or noise. All points which remain cluster-less after the algorithm is finished are classified as anomalies.

In addition to the discovered anomalies, DBSCAN can find arbitrarily-shaped clusters and does not require to know the number of clusters in the dataset a priori. DBSCAN requires just two parameters that should be optimally chosen. There are several studies devoted to this problem [16], [17]. In this study, ε is defined as the maximal value of pair distance between feature vectors of the training set X :

$$\varepsilon = \max_{x_i, x_j \in X} d^E(x_i, x_j). \quad (15)$$

After that, N_{min} is calculated as a minimal value of neighbors of each feature vector of the training set taking into account that the size of the neighborhood ε is defined by (15):

$$N_{min} = \min_{x \in X} N_\varepsilon(x). \quad (16)$$

DBSCAN parameters calculated according to (15) and (16) guarantee that all feature vectors extracted from user agents of the training set participate in the model of normal user behavior. In order to classify feature vector x corresponding to a user agent of a new HTTP request, DBSCAN with selected parameters is applied to the data set consisting of x and all feature vectors of the training set. If x remains cluster-less after DBSCAN has been applied, it is classified as anomaly. In other words, if x is not density-reached from any point of the training set, it corresponds to an intrusive HTTP request.

D. Aggregated time bin statistics

Entropy values of different parameter distributions can have different scales. In order to standardize the entropy vectors of the training set z-score normalization is used [20]. Using this approach entropy values of the j -th parameter are normalized based on their mean μ_j and standard deviation σ_j :

$$E_{ij}^z = \frac{E_{ij} - \mu_j}{\sigma_j}, \quad (17)$$

where E_{ij}^z is the new value of E_{ij} . These new values are distributed with standard normal distribution, with zero mean and unit standard deviation.

In order to analyze HTTP requests in a new time bin an entropy vector is extracted and normalized by using the mean and standard deviation values found in the training phase. This normalized vector e^z is classified as an anomaly if

$$\|e^z\|_2 > \max_{i \in T} \|E_i^z\|_2, \quad (18)$$

where T is the set of time bins analyzed in the training set. In this case, requests received during this new bin are supposed to be analyzed further.

IV. EXPERIMENTAL RESULTS

The demonstration of the proposed intrusion detection approach is carried out using Realistic Global Cyber Environment (RGCE) which has been created to be equivalent to the Internet meaning that the structures are as similar as possible. RGCE contains a network operator and the Internet Service Providers (ISPs) with their core services. These structures enable realistic organization and ISP environments and therefore real threats and attack vectors. RGCE is isolated from production networks, which allows one to use vulnerabilities and attacks without a risk of contaminating production networks. Within RGCE, traffic is automatically generated based on realistic traffic patterns that simulate or replicate end user traffic.

In this research, a web server has been installed in the RGCE environment. In order to test the proposed methods of intrusion detection, normal network traffic and special attacks against the web server have been generated. The traffic has been generated during a five-day time period. On the first day, there is only legitimate traffic, whereas the traffic gathered for the next four days consists of legitimate traffic mixed with several attacks against the web server. These attacks include scanning attacks, DoS attacks, bruteforce attacks and various targeted attacks. The web server's HTTP logs gathered during these five days are then used to evaluate the performance of the proposed detection scheme.

First, the search for anomalous HTTP requests is carried out. For this purpose, we form three training and three testing datasets which include feature vectors extracted from requested web resources, values of HTTP queries attributes and user agents, respectively. Each training set is generated based on HTTP requests gathered on the first day and, therefore, does not contain any intrusions. The testing sets include several vectors corresponding to anomalous HTTP requests. These anomalies are caused by attacks such as SQL injections, directory traversals, cross-site scripting attempts and double encoding attacks.

The detection of HTTP requests which contain injections into web resource parameters is performed with the help of SVDD. Feature vectors from these parameters are extracted by using n -gram models described in Section II when $n \in \{1, 2, 3\}$. Figure 1 shows the dependence between false positive and true positive rates for different n -gram models. As one can see, for low values of false positive rate, SVDD shows the best results in terms of true positive rate when 1-gram model is applied and feature vectors are calculated by using (2). When the method parameters are chosen optimally, 97.96 percent of intrusive requests are detected while the number of false alarms is 0.52 percent. Resulting from this, anomalous web resources in HTTP requests are detected with the accuracy rate equal to 99.20 percent.

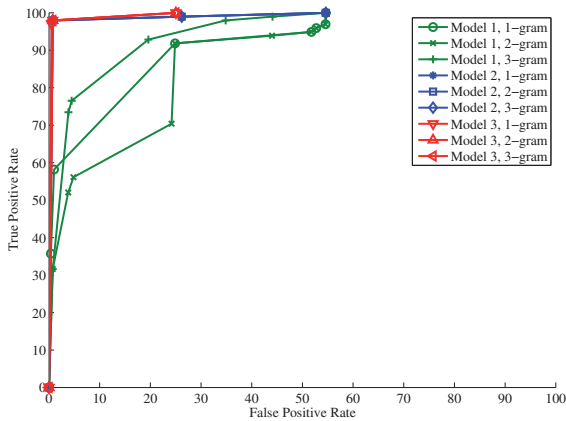


Fig. 1. The dependence between false positive and true positive rates of the detection of anomalous web resources by SVDD for different n -gram models.

After this, k -means is applied to feature vectors extracted from attribute values. As previously, three n -gram models with different values of n have been used to extract these vectors. The dependence between false positive and true positive rates is shown in Figure 2. It can be noticed that the application of 1-gram and either equation (2) or (3) for the features extraction allows one to detect all (100 %) injections into HTTP attribute values. Thus, even such simple technique as k -means can be successfully used for HTTP attacks detection if the features extraction and the parameters selection are carried out in a proper way.

Anomalies within user agent values are classified by DBSCAN. In Figure 3, dependencies between false positive and true positive rates for different n -gram models are compared. As one can notice, for zero false positive rate the best results in terms of true positive rate (95.45%) are provided when 1-gram model is applied and feature vectors are calculated by using (2). Such approach allows us to classify user agent values with the accuracy equal to 97.5. Although the detection accuracy is still high, it is much less than in cases of other HTTP request parameters. This can be explained by the fact that the traffic gathered on the first day contains only 9 unique user agent values, which is not enough for the construction of a normal user behavior model.

We compared the performance of the discussed methods with other outlier detection techniques: Self-Organizing Map

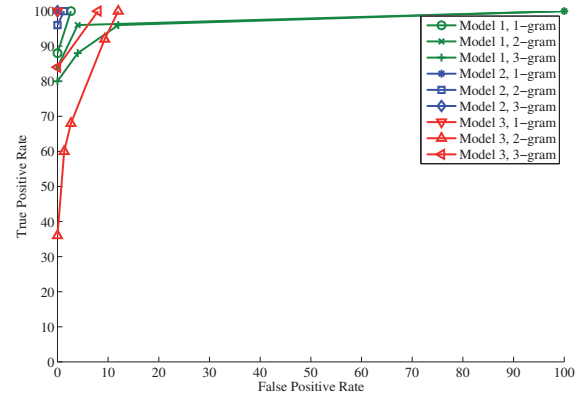


Fig. 2. The dependence between false positive and true positive rates of the detection of anomalous attributes by k -means for different n -gram models.

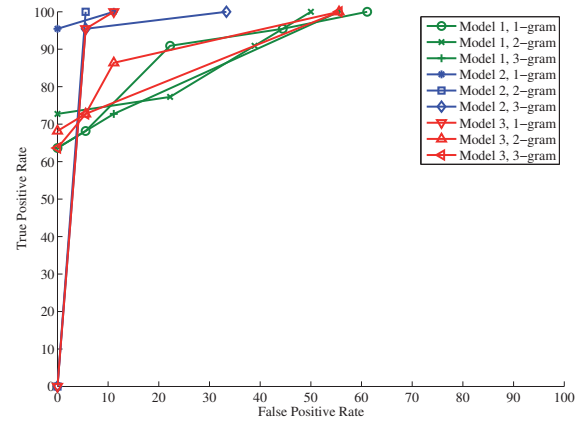


Fig. 3. The dependence between false positive and true positive rates of the detection of anomalous user agents by DBSCAN for different n -gram models.

(SOM) and Local Outlier Factor (LOF). In order to extract features, different n -gram models are applied. Those models for which the detection accuracy is the highest are selected. The comparison results are shown in Table I, where these highest accuracy values are listed. As one can see, the methods based on SVDD, k -means and DBSCAN show better results and outperform other techniques in terms of detection accuracy. We can also see that the calculation of feature vectors by (2) allows us to reach the highest or one of the highest accuracy rates. Furthermore, 1-gram model is supposed to be applied if the aim is to minimize the number of false alarms.

TABLE I. DETECTION ACCURACY OF DIFFERENT ANOMALY DETECTION METHODS FOR DETECTING INTRUSIVE HTTP REQUESTS.

Algorithm	Web resources	Attribute values	User agents
SVDD	99.2 %	98 %	90 %
K -means	99.15 %	100 %	77.5 %
DBSCAN	98.75 %	98 %	97.5 %
SOM	98.7 %	99 %	87.5 %
LOF	98.2 %	95 %	97.5 %

Finally, the traffic gathered is considered as a time series. A sliding time window is used to define the overlapping time bins. For each time bin, the required parameters are extracted

and entropy vectors are calculated and normalized. Figure IV shows normalized entropy values for the traffic gathered during the first and the fourth days.

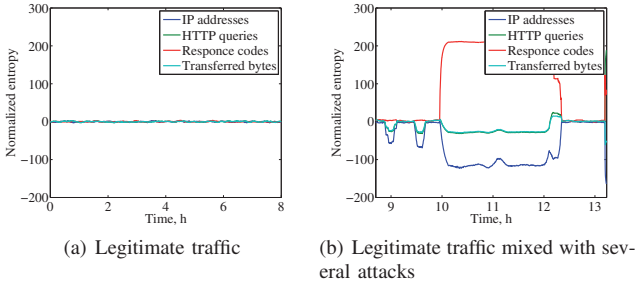


Fig. 4. Normalized entropy values calculated with the time window of 60 seconds sliding with the step equal to 10 seconds.

The first day contains only legitimate traffic, whereas during the fourth day, several active scans and password bruteforce attempts are carried out by attackers (see Table II).

TABLE II. SCANS AND PASSWORD BRUTEFORCE ATTEMPTS DURING THE FOURTH DAY.

Attack	Time
Password bruteforce by Hydra	08:57 - 09:11
Password bruteforce by Hydra	09:28 - 09:41
Database takeover by Sqlmap	09:58 - 12:21
Active scan by Owasp	12:07 - 12:21
Forced browse by Owasp (dirbuster)	13:06 - 13:14

The detection of these attacks is tested for different sliding time windows. The values of the detection accuracy are compared in Table III.

TABLE III. DETECTION ACCURACY OF THE ENTROPY-BASED METHOD FOR DIFFERENT TIME WINDOWS.

Time window	Time step	Attacks detected	Detection accuracy in the time step
10 minutes	1 minute	5 of 5	98.35 %
10 minutes	10 seconds	5 of 5	98.76 %
10 minutes	1 second	5 of 5	98.29 %
1 minute	10 seconds	5 of 5	99.24 %
1 minute	1 second	5 of 5	98.70 %

As one can see, when the size of the window is big enough, all scans and bruteforce attacks can be detected with the help of the proposed mechanism. The value of time step is supposed to be selected as short as possible for the earlier detection of attacks.

V. CONCLUSION

In this study, we consider the use of the analysis of HTTP logs in the detection of network intrusions. First, a training set of HTTP requests which does not contain any attacks is analyzed. Once all relevant information has been extracted from the logs, several anomaly detection schemes are applied to describe the model of normal user behavior. This model is then used to detect network attacks as deviations from the norms. Data generated in a demonstration phase using RGCE environment was extremely suitable for the study modeling real Internet traffic that contains legitimate traffic mixed with several attacks. The simulations based on this data show that the method results in a higher accuracy rate compared to

other data mining techniques. In the future, we are planning to continue the use of the anomaly-detection approaches to detect web intrusions. We are going to concentrate on the improvement of the performance of methods in terms of the detection accuracy and real time detection of more complicated web-based attacks.

REFERENCES

- [1] S. Mukkamala and A. Sung. A comparative study of techniques for intrusion detection. Proc. of the 15th IEEE International Conference Tools with Artificial Intelligence, pp. 570–577, 2003.
- [2] A. Nguyen-Tuong, S. Guarnieri, D. Greene, J. Shirley and D. Evans. Automatically Hardening Web Applications Using Precise Tainting. Proc. of the 20th IFIP International Information Security Conference (SEC), pp. 372–382, 2005.
- [3] S. Axelsson. Research in intrusion-detection systems: a survey. Department of Computer Engineering, Chalmers University of Technology, Goteborg, Sweden, Technical Report. pp. 98–117, 1998.
- [4] D. Gollmann. Computer Security. Wiley, 2nd edition, 2006.
- [5] T. Sipola, A. Juvonen and J. Lehtonen. Anomaly detection from network logs using diffusion maps. Springer, Engineering Applications of Neural Networks, IFIP Advances in Information and Communication Technology, pp. 172–181, 2011.
- [6] W. K. G. Fan. An adaptive anomaly detection of WEB-based attacks. Proc. of the 7th International Conference on Computer Science & Education (ICCSE), pp. 690–694, 2012.
- [7] M. Zolotukhin, T. Hämäläinen, A. Juvonen. Online Anomaly Detection by Using N-gram Model and Growing Hierarchical Self-Organizing Maps. Proc. of the 8th International Wireless Communications and Mobile Computing Conference (IWCMC), pp. 47–52, 2012.
- [8] J. Wang, M. Zhang, X. Yang, K. Long and C. Zhou. HTTP-sCAN: Detecting HTTP-flooding attack by modeling multi-features of web browsing behavior from noisy dataset. Proc. of the 19th Asia-Pacific Conference on Communications (APCC), pp. 677–682, 2013.
- [9] Apache 2.0 Documentation (2011). <http://www.apache.org/>.
- [10] A. Klein. Detecting and Preventing HTTP Response Splitting and HTTP Request Smuggling Attacks at the TCP Level. Tech. Note. <http://www.securityfocus.com/archive/1/408135>, Aug. 2005.
- [11] C. Y. Suen. n-Gram Statistics for Natural Language Understanding and Text Processing. IEEE Tran. on Pattern Analysis and Machine Intelligence, , Vol. PAMI-1, Is. 2, pp. 164–172, 1979.
- [12] T. Hirsimäki, J. Pyllkkonen and M. Kurimo. Importance of High-Order N-Gram Models in Morph-Based Speech Recognition. IEEE Tran. on Audio, Speech, and Language Processing, Vol. 17, Is. 4, pp. 724–732, 2009.
- [13] S. Lloyd. Least squares quantization in pcm. IEEE Trans. on Inf. Theor., Vol. 28 Is. 2, pp. 129–137, 2006.
- [14] D. Pham, S. Dimov and C. Nguyen. Selection of k in k-means clustering. Proc. of IMechE Mechanical Engineering Science, pp. 103–119, 2005.
- [15] M. Ester, H. Kriegel, S. Jörg, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. AAAI Press, pp. 226–231, 1996.
- [16] J. Kim. The Anomaly Detection by Using DBSCAN Clustering with Multiple Parameters. Proc. of ICISA, pp. 1–5, 2011.
- [17] Smiti, A. DBSCAN-GM: An improved clustering method based on Gaussian Means and DBSCAN techniques. Proc. of International Conference on Intelligent Engineering Systems (INES), pp. 573–578, 2012.
- [18] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda and B. Schölkopf. An introduction to kernel-based learning algorithms. IEEE Transactions on Neural Networks, Vol. 12, Is. 2, pp. 181–201, 2001.
- [19] D.M.J. Tax and R.P.W. Duin. Support Vector Data Description. Machine Learning, Vol. 54, Is. 1, pp. 45–56, 2004.
- [20] C. Saranya and G. Manikandan. A Study on Normalization Techniques for Privacy Preserving Data Mining. International Journal of Engineering and Technology (IJET), Vol. 5, Is. 3, pp. 2701–2704, 2009.

PII

**ANALYSIS OF APPROACHES TO INTERNET TRAFFIC
GENERATION FOR CYBER SECURITY RESEARCH AND
EXERCISE**

by

Tero Kokkonen, Timo Hämäläinen, Marko Silokunnas, Jarmo Siltanen, Mikhail
Zolotukhin, Mikko Neijonen 2015

Lecture Notes in Computer Science, vol. 9247, pp. 254-267

Reproduced with kind permission of Springer International Publishing AG,
Copyright © 2015, Springer International Publishing AG.

Analysis of Approaches to Internet Traffic Generation for Cyber Security Research and Exercise

Tero Kokkonen^{1,2(✉)}, Timo Hämäläinen², Marko Silokunnas¹, Jarmo Siltanen¹, Mikhail Zolotukhin², and Mikko Neijonen¹

¹ Institute of Information Technology,
JAMK University of Applied Sciences, Jyväskylä, Finland
{tero.kokkonen,marko.silokunnas,jarmo.siltanen,
mikko.neijonen}@jamk.fi

² Department of Mathematical Information Technology,
University of Jyväskylä, Jyväskylä, Finland
{timo.t.hamalainen,mikhail.m.zolotukhin}@jyu.fi
tero.t.kokkonen@student.jyu.fi

Abstract. Because of the severe global security threat of malwares, vulnerabilities and attacks against networked systems cyber-security research, training and exercises are required for achieving cyber resilience of organizations. Especially requirement for organizing cyber security exercises has become more and more relevant for companies or government agencies. Cyber security research, training and exercise require closed Internet like environment and generated Internet traffic. JAMK University of Applied Sciences has built a closed Internet-like network called Realistic Global Cyber Environment (RGCE). The traffic generation software for the RGCE is introduced in this paper. This paper describes different approaches and use cases to Internet traffic generation. Specific software for traffic generation is created, to which no existing traffic generation solutions were suitable.

Keywords: Internet traffic generation · Cyber security research and exercise · Cyber security · Network security

1 Introduction

The JAMK University of Applied Sciences has built a closed Internet-like network called Realistic Global Cyber Environment (RGCE). RGCE mimics the real Internet as closely as possible and contains most services found within the real Internet, from tier 1 Internet Service Providers (ISP) to small local ISPs and even individual home and corporate ISP clients. The fact that RGCE is completely isolated from the Internet allows RGCE to use accurate GeoIP information for all IP addresses within RGCE. This allows the creation of exercises or research cases where the attackers and the defenders are seemingly in different parts of the world and any device (real or virtual) will assume that it is actually operating within the real Internet. RGCE also contains various web services found in the real Internet [1, 2].

Due to the fact that RGCE is isolated from the real Internet, RGCE does not contain background user traffic of its own. This poses a problem: how can you realistically train for a scenario where your public services are being attacked by an unknown party, and the attack traffic is concealed within normal user traffic if there is no normal user traffic? This is the basic problem to be solved in order to efficiently use RGCE for cyber security exercises or research.

Traffic generation has an important role when characterizing behaviour of the Internet. Behaviour of the real Internet consists of the rapid changes of the network, network traffic and user behaviour as well as the variables of characterization vary from the traffic links and protocols to different users or applications [3]. In addition changing nature of connections in Internet is influenced by the behaviour of the users, which determines the page level, and the connection level correlation that should be included to the traffic generation models [4]. According to the study [4] this is neglected by the scientific literature.

There are two fundamental approaches to Internet-like traffic generation, trace-based generation and analytical model-based generation. In trace-based generation the content and the timings of the captured real traffic are retransmitted and in analytical model-based generation the traffic is generated based on the statistical models [5, 6].

Due to increasing amount of traffic, applications and users deep analysis of real Internet traffic is essential for planning and managing networks [7]. Deep analysis of real Internet traffic also gives an efficient viewpoint for realizing the extensive processes of the Internet [8]. Thus the deep analysis of the real network traffic can be used for developing Internet traffic generation software using realistic traffic patterns from both humans and machines.

In this paper the Internet traffic generation software is introduced. First, the requirements, existing solutions and different approaches for traffic generation are presented. Then the developed solution is introduced and evaluated.

2 Found Requirements

The main purpose of developed Internet traffic generation software is to generate user traffic for the cyber security exercises conducted within RGCE. To meet the requirements for cyber security exercises the Internet traffic generation software was implemented according to the following self-generated requirements:

- Centralized control; the system shall have a single point of control and the control mechanism shall enable the generation of a large volume of traffic with minimal user interaction.
- Ability to generate legitimate traffic; the generated traffic shall adhere to the generated protocol.
- Ability to generate meaningful traffic on several layers of the OSI model; the system shall be able to generate meaningful traffic on OSI layers 3-6. This shall include IP, TCP, HTTP and other application layer protocols.

- Ability to generate attack traffic; the system shall be able to generate traffic for various attacks commonly encountered on the Internet. Examples of such attacks include SYN flood, NTP and DNS amplification DDoS attacks.
- Generated traffic shall look like real Internet traffic; the traffic shall be as indistinguishable as possible from real traffic for both humans and machines.
- Ability to make the traffic look like it is coming from anywhere within RGCE; it shall be possible to deploy parts of the system to various parts of RGCE to make the geolocation information look realistic.
- Generated traffic shall not be a replay; replaying previously recorded traffic would make it easy to distinguish generated traffic from normal user traffic, unless the recorded captures are of significant length.
- Generated traffic shall work with existing servers; the system shall be able to use normal, non-modified servers as targets for traffic generation. A simple example would be HTTP: the system shall be able to generate legitimate non-identical requests to a given HTTP server, with varying HTTP headers and make those requests at human-like intervals.
- The system shall be highly autonomous; the system shall be able to recover from errors without human intervention as much as possible. The system shall be able to generate traffic without human intervention for extended periods of time.

3 Existing Solutions for Traffic Generation

There are a number of proprietary and open source tools available for Internet-like traffic generation, such as TG Traffic Generator [9], NetSpec [10], Netperf [11], Packet Shell [12] and D-ITG [13, 14]. A detailed listing and analysis of available tools can be found from the study [5]. Those mentioned tools approach the problem from the viewpoint of workload generation through statistical models. Their goal is to generate repeatable workloads for networks and monitoring tools.

Such tools suffer from the fact that they are often implemented on top of non-real-time operating systems (OS). This causes their behaviour to be un-deterministic due to various scheduling decisions made by the OS as introduced in study [15]. Performance of D-ITG is also analysed in [14]. Netbed has a different viewpoint, it is a tool for integrating three experimental environments: network emulator, network simulator and real networks [16].

Developed Internet traffic generation software avoids many of above-mentioned problems, mainly because the goal is not in the generation of realistic workloads, but rather in meaningful payloads and good integration with existing off-the-shelf products with minimal customization.

4 Approaches

There are different approaches to Internet traffic generation with their pros and cons. These described approaches were analysed for the development of Internet traffic generation software.

4.1 Network Layer Traffic Generation

Generating traffic on the network layer is a simple approach to traffic generation. It is trivial to implement using, for example, Linux raw sockets [17], and can be implemented for both IPv4 and IPv6.

This approach works by generating a large number of IP packets with randomized payloads. The use of Linux raw sockets also allows the source IP address of the packet to be spoofed, which allows a single machine to simulate a huge number of individual hosts. The machine sending the IP packets could be considered to be the default gateway for a large organization, such as university or a company.

An example system could work by requiring a definition of a range of source IP addresses to use (e.g. 10.0.0.1-10.0.0.255 for IPv4) and then generating a large number of IP packets with the source field set to one of the IP addresses within the source range.

The generated packets are only meaningful when analysed on the IP layer. If the requirements for the generated traffic are such that the traffic has to be meaningful on higher layers (e.g. TCP), this approach is not suitable without a considerable amount of effort. This means that implementing a custom TCP stack on top of Linux raw sockets is required. The only benefit over regular Linux TCP sockets [18] is the ability to spoof source IP addresses for individual TCP segments [19].

Various analytical model-based network traffic generation tools utilize this approach. Such tools put emphasis on IP traffic characteristics (e.g. packet size and timing), rather than the transmitted data itself [13, 9, 10].

This is not feasible for the purposes of Internet traffic generation within RGCE (see Section 2). But it is relevant for testing various other aspects of network performance.

4.2 Transport Layer Traffic Generation

Using existing TCP stacks found in operating systems to handle the TCP connections significantly reduces the complexity of the implementation but makes IP spoofing [20] difficult. It is still possible to use a single machine to simulate a larger amount of hosts by using IP aliasing.

An elementary approach to traffic generation on the transport layer would be to utilize TCP stack provided by the operating system. This greatly simplifies the implementation of the traffic generator, as the OS TCP stack will take care of retransmission and other TCP details. As a downside, this approach does not allow for much control over the generated traffic characteristics.

As with the network layer approach (Section 4.1) this method works as long as meaningful exchanges on higher layers of the OSI model are not required (e.g. HTTP). It is possible to overcome this problem for the simplest of cases, such as creating multiple identical HTTP requests and always expecting an identical reply. More complex transmissions are also possible to implement but in most cases it would be more straightforward to just implement the approach described in Section 4.6

4.3 Replaying Traffic

When considering approaches to Internet traffic generation, replaying PCAP files [21] is a rather natural option. Typically, traffic replay aims to generate repeatable workloads for systems under test [6]. This is achieved by replaying recorded data [22] or synthesizing [23] traffic traces and then replaying them through the network. Tcpreplay [22] is existing software solution that is able to replay captured TCP traffic from files.

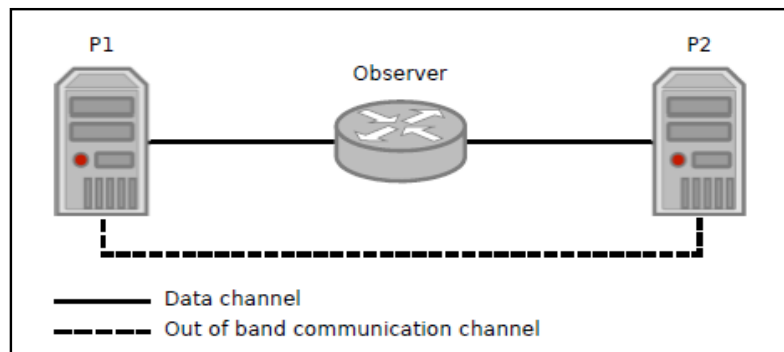


Fig. 1. Simple traffic replaying environment

It is necessary to use Out of band communication channel (Fig. 1) if the orchestration should not interfere with the system under test. Orchestration could include communicating the roles, timing, and bandwidth quotas associated with the replay [5].

In order to make this approach work, some processing is required for the PCAP file:

- Filtering out all unnecessary data streams. Unless the PCAP file is captured with the intention of replaying it, it is likely that the file contains a lot of unnecessary packets.
- Compiling the payload bytes from the TCP segments. Sending individual TCP segments from the PCAP file is not a feasible approach because network conditions are very likely to differ between the recording environment and the replaying environment. When the bytes are properly extracted and sent over the operating system's TCP stack the implementation does not have to concern itself with TCP details. It also makes it easier to detect and handle networking problems in the replaying environment.
- Constructing an intermediate presentation of the PCAP file that describes what to send and what to receive for each participant of the conversation.

It is worth noting that replaying PCAP files is only feasible for reliable transport layer protocols (e.g. TCP and SCTP). While it is possible to just extract the sent UDP (or other unreliable transport level protocols) packets and resend them, it will require extra steps to ensure that the packets get to their destination due to the nature of UDP [24]. There are two approaches to overcome this problem:

- Protocol awareness. The system needs to be aware of the protocol it is replaying and in case of lost packets mimic the simulated protocol’s behaviour in such situations (if any). This requires considerable effort to duplicate the protocol’s functionality and the solution starts to resemble the approach detailed in section 4.4.
- Out of band communication channel. An out of band communication could be utilized to transfer information about sent and received packets between participants. While this approach makes sure that all packets get delivered, it does not reliably reproduce the simulated protocol, because it is acceptable to lose packets in some UDP based protocols.

The following subsections will detail the out of band communication channel approach and its limitations. It is worth noting that the out of band communication channel must use a reliable transport layer protocol, such as TCP. The out of band communication channel also introduces additional latency to the replaying caused by TCP.

4.4 Replaying in a Reliable Network

It is assumed in Fig. 2 that the replaying environment does not suffer from packet loss, thus introducing no unexpected side effects.

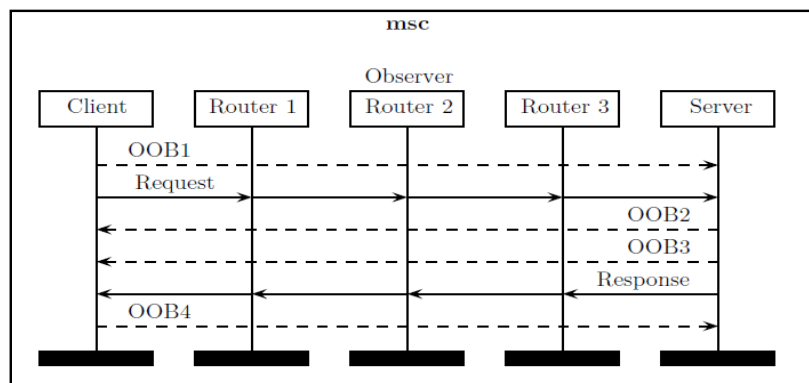


Fig. 2. No network problems

Replaying in a reliable network is processed as follows: client notifies Server through the OOB channel that it is about to send a request, client sends the actual request, server notifies client that it received the request, server notifies client that it is about to send a response, server sends the response and finally client notifies server that it received the response successfully.

This scenario works as expected and does not introduce any additional side effects; the observer sees a single request and a single response and thus cannot tell the traffic apart from the real traffic.

4.5 Replaying in an Unreliable Network

The fact that the out of band communication channel introduces some reliability features to the system can cause the observer to see responses without requests, this can be seen from Fig. 3.

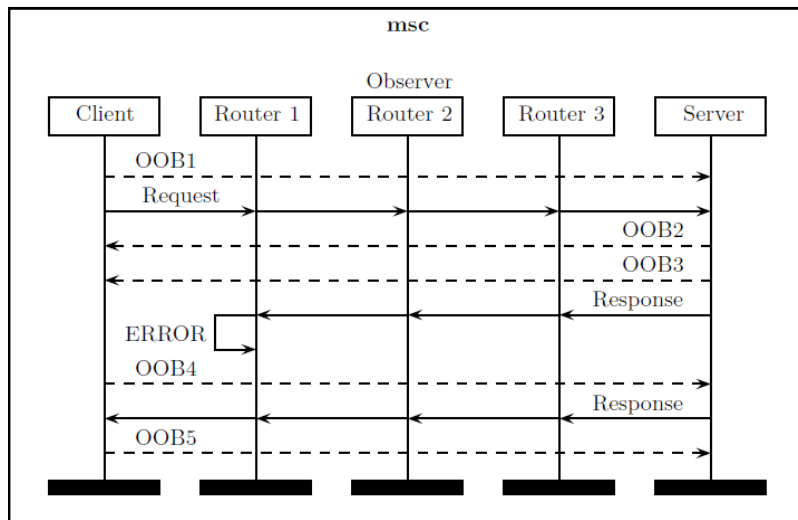


Fig. 3. Network problems

Replaying in an unreliable network is processed as follows: client notifies server through the OOB channel that it is about to send a request, client sends the actual request, server notifies client that it received the request, server notifies client that it is about to send a response, server sends the response, router 1 fails to deliver the packet, client notifies Server that it did not receive a response, server resends the response and Client notifies the server that it received the response successfully.

In the case of a network failure the observer observes multiple identical responses without corresponding requests. This allows an observer familiar with the protocol in question, to conclude that this traffic is not authentic.

Even though replaying PCAP files is problematic for protocols that are implemented on top of unreliable transport protocols, it is still robust for protocols utilizing reliable transport protocols. But still this approach cannot be used with existing servers and will end up repeating the same conversation over and over again, thus not fulfilling the requirements listed in section 2.

4.6 Simulating Clients

Simulating full clients for Internet traffic generation offers a flexible solution to traffic generation, as it allows fine-grained control over the generated traffic and the depth of simulation. This approach does not allow precise traffic generation on packet level or control of the various packet characteristics that are available in other traffic generation solutions, such as Inter Departure Time (IDT) and Packet Size (PS).

This approach can fulfil the requirements listed in section 2; it can be used with any server and the generated traffic is sufficiently diverse. It is also possible to

implement very specific types of traffic (e.g. deliberately broken TCP traffic). If the client simulation is sufficiently sophisticated, it is very difficult for the server and observers to distinguish it from the realistic clients.

It is worth noting, that this approach requires a significant amount of effort, as it is difficult to create a single solution that could simulate multiple clients and protocols in a convincing manner. This means that the system will require multiple protocol specific modules.

5 Implemented Solution

Implemented solution aims to generate traffic that looks meaningful to a human observer. It was decided to implement Internet traffic generation software using the full client simulation approach. Solution consists of a hierarchical network of nodes. The network forms a tree like structure. The network forms an opt-in botnet, where each individual node is a host.

5.1 Terminology

The network consists of three different node types: King, Slavemaster and Botmaster. Bots are not nodes (hosts), but are run on the same host as the Botmaster.

King is the root node of the tree. King acts as a bridge between the UI and the rest of the network. The UI is running on a webserver on this node. Both Slavemasters and Botmasters can connect to King. Every message sent into the network by the user passes through King.

Slavemaster connects to the King or another Slavemaster and acts as a router between nodes. Slavemasters can connect to other Slavemasters and thus the depth of the tree representing the network can be arbitrary. Slavemasters have full knowledge of the tree underneath themselves. When a Slavemaster receives a message it checks the message recipient. If the recipient is the Slavemaster it broadcasts that message to all of its children, who then broadcast it to their children and so on. If the recipient is one of the descendants of the Slavemaster message is forwarded towards it.

Botmaster is a leaf node of the tree. Botmasters are charged with performing the actual traffic generation. Botmasters run one or more Bots. Multiple Bots can be running simultaneously. Botmaster receives messages and status updates from its Bots and forwards them to King, which will then update the UI accordingly. In the current implementation Bots are ran in the same process as the Botmaster.

Bot handles the actual traffic generation. Each Bot is tasked with generating traffic of a certain type (e.g. HTTPBot generates HTTP traffic). If a Bot encounters an error it sends a notification to the UI about it.

5.2 Implementation

Current implementation of the system contains traffic generation profiles for various protocols and services, such as HTTP, SMTP, DNS, FTP, NTP, IRC, Telnet, SSH, CHARGEN and ICMP. Each protocol or service is capable of containing different

profiles. For example there are five different bots for HTTP protocol: HTTPBot mimics an user that is browsing the internet, SlowlorisBot performs the slowloris HTTP DoS attack, SlowPOSTBot performs the slow POST HTTP DoS attack, HTTPAuthBot repeatedly attempts to authenticate using HTTP Basic Auth and HTTPDDoSBot repeats the same HTTP request continuously.

Implementation is done for GNU/Linux using the Go programming language [25]. Each node of the system (King, Slavemaster and Botmaster) has its own binaries. In the current implementation Bots are ran in the same process as the Botmaster, but this is required to change in the future development. Go was chosen as the implementation language due to the fact that it has native support for coroutines (called goroutines in Go) and easy interfacing with C programming language.

Go also provides a way to facilitate communication between goroutines using channels, which are derived from Hoare's CSP [26]. The first few versions of our traffic generation software utilized the C interface significantly, but the current version contains no C code. The need for interfacing with C reduced as the Go ecosystem grew and more libraries became available. Most of the C code in the early versions was related to utilizing raw sockets to conduct IP spoofing, which can now be achieved using Go.

Bots are run inside the Botmaster as goroutines. A Bot can contain multiple goroutines. Naturally, the Botmaster contains goroutines that are not Bots as well, such as goroutines related to communication with the rest of the network.

Bots communicate with the Botmaster using Go's channels. Botmasters, Slavemasters and King communicate between each other using a custom text based protocol implemented on top of TCP.

The UI is implemented as a single-page web application served by a web server running on the King. The UI communicates with the King using a custom protocol on top of WebSockets [27]. The King acts as a translator between the WebSocket protocol and the protocol used by the rest of the network. User interface (UI) of developed Internet traffic generation software can be seen in Fig. 4.

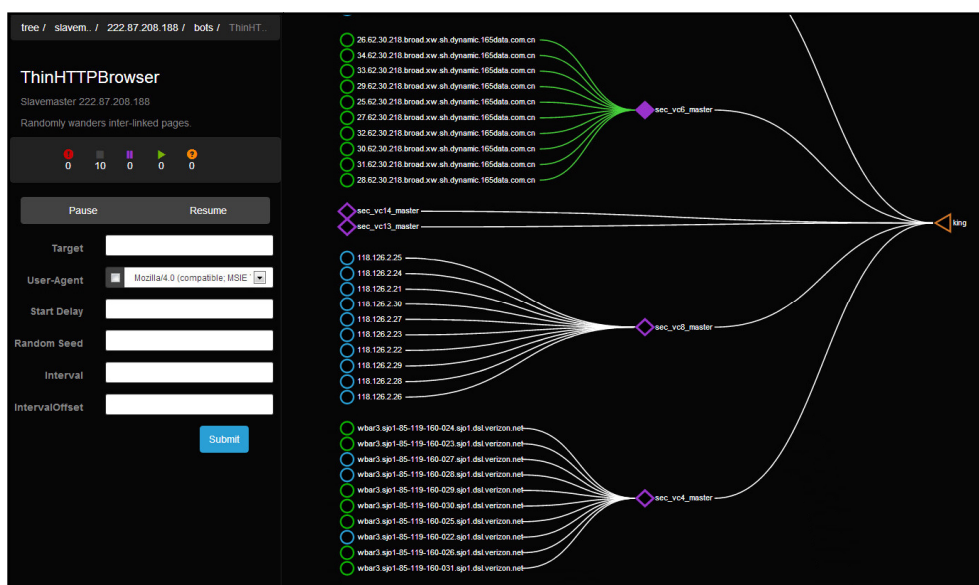


Fig. 4. UI of developed Internet traffic generation software

5.3 Evaluation

The evaluation for the developed solution is studied as follows. There were three different networks and three different amount of data generation Botmasters chosen. The different networks chosen were Localhost, Local Area Network (LAN) and Internet. The webserver with webpage including text pictures and links to 30 sub-pages was installed and HTTPBots browsed the contents of that webserver on each network cases. The LAN was in the JAMK University of Applied Sciences and the Internet scenario was between Netherlands and Finland (the webserver located in the Netherlands). The amount of Botmasters was 5, 25 and 125. The network data was captured on both sides, server side and client side. Time period of every single capture is 30 minutes long.

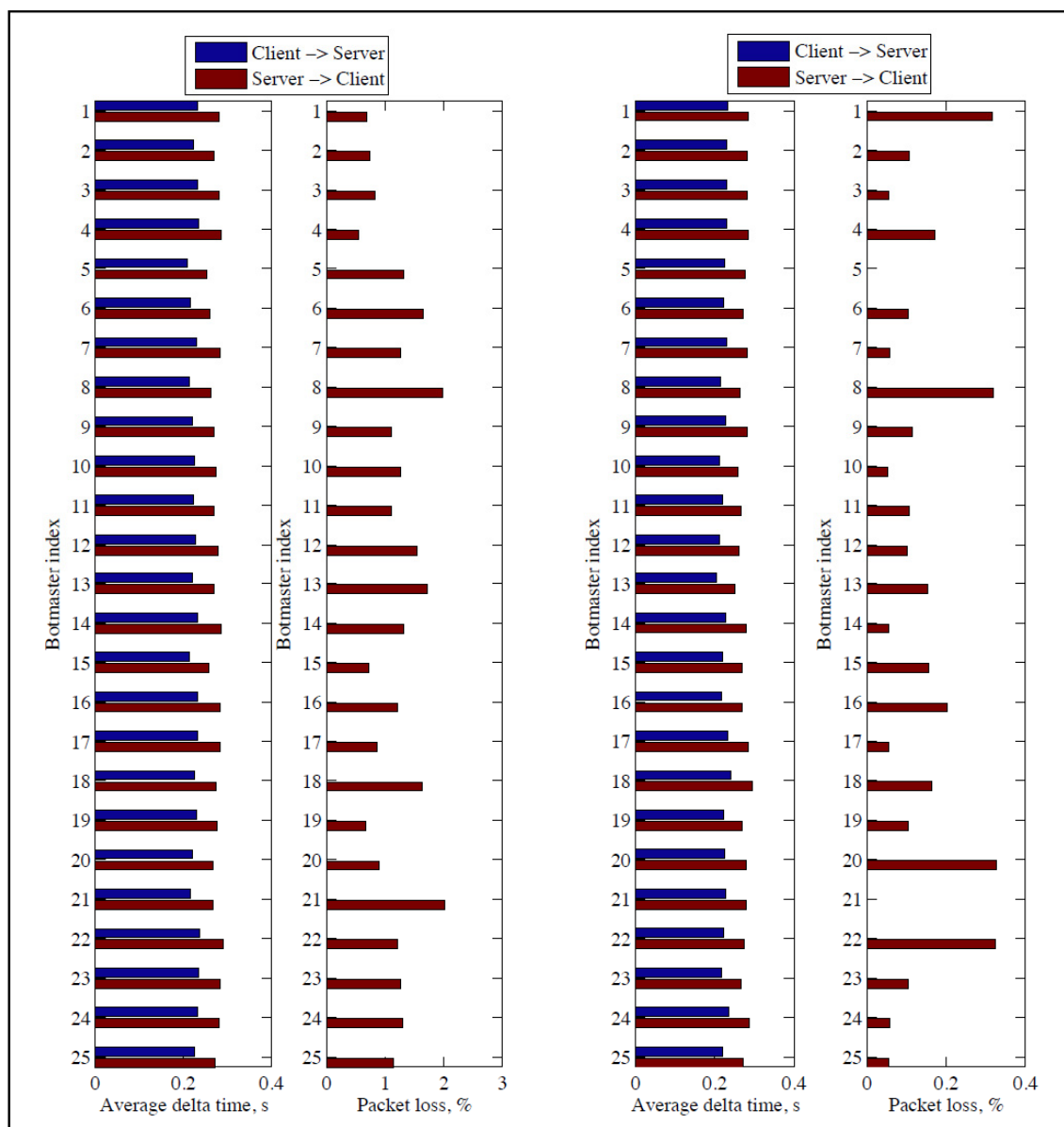


Fig. 5. Left: Internet 25 Botmasters, average delta time (s) and packet loss (%) Right: LAN 25 Botmasters, average delta time (s) and packet loss (%)

Evaluation data was generated using the HTTPBot, which mimics a browser by first downloading an HTML page from the targeted HTTP server. HTTPBot downloads all images, JavaScripts and CSS files referenced in the HTML document. Once all of the files are downloaded, the HTTPBot searches for a link to another page within the same domain or another domain. A link is chosen at random and the same process is repeated again.

The network traffic (PCAP data) was captured from the client and server side of the connection and also log data from the server was collected.

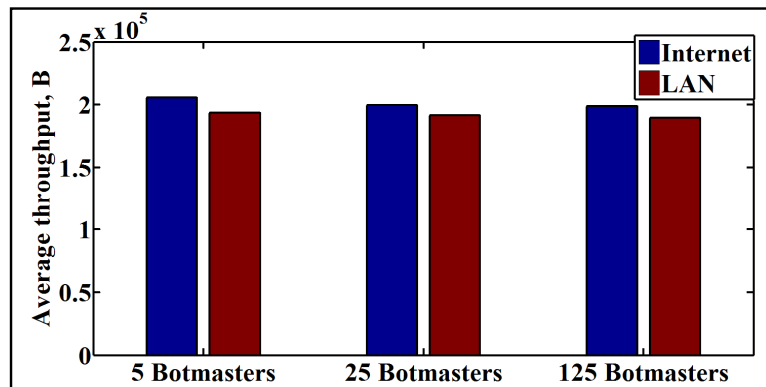


Fig. 6. Average throughput for generated traffic

Fig. 5 shows packet loss (%) and average delta time for 25 Botmasters in Internet and LAN. Average delta time in those figures is an average time difference between sent packets from the same source during the same conversation. Fig. 6 shows average throughput from all clients to server in all measured cases captured in client sides of the connections.

Evaluation shows that developed system is scalable and capable of producing significant amount of traffic. Scalability was proofed using different amount of Botmasters and different network topologies. In all tested network topologies the generated traffic behaves as expected based on the calculated characteristics.

Since Internet traffic generation software is designed for conducting research in network security and cyber attack detection, it is also capable to produce different sorts of attacks e.g. Denial of Service DoS/DDoS attacks based on volumetric traffic, resource exhausting or exploits and also bruteforce attacks.

Developed solution is also tested in the National Cyber Security Exercises organized by Finnish Defence Forces [28, 29]. Initial version of Internet traffic generation software is also being used for Internet traffic generation in an anomaly detection study [2].

All of those experiments show that developed traffic generation solution can be used to generate different kind of data patterns, and it is appropriate for different kind of cyber security analysis for example for big scale National cyber security exercises.

5.4 Lessons Learned

The generation of the Internet traffic is extremely important for research and development of cyber security. For example research and development of Anomaly Detection algorithms or Intrusion Detection Systems requires an environment with realistic legitimate background traffic and design made attacks [30, 2]. Generation of Internet traffic has an important role in cyber security exercises and training.

There were some lessons learned from the use cases that caused extra development for the data generation software. OOB communication for control traffic is very important when generating lot of traffic (e.g. HTTPDDoSBot). Generated data might block the outgoing data and if the command communication data uses the same interface it is also blocked. That causes situation where one Bot blocks the Botmaster out from the network. Another lessons learned that required changes for development was CPU bound meaning that if there is Bot doing resource intensive processing it might harm the whole process and block the Botmaster out of communication. Bots that are CPU resource intensive (e.g. SYN flood Bot) cannot have permission to generate traffic as fast as they are capable of processing, thus there must be a limit e.g. 5000 packets/second/Bot.

6 Conclusion

In this study, approaches to realistic Internet traffic generation for cyber security research and exercise were considered. First requirements for traffic generation were analysed. After that different solutions and approaches were described. Finally suitable approach was chosen and developed Internet traffic generation software was introduced. As a conclusion it can be said that the developed Internet traffic generation software met the requirements and it is suitable for modelling the Internet traffic as a part of the cyber security research and exercise. Requirements for next phase were found and in future the development of those requirements are planned to execute. The deployment of the several Botmasters shall be automated and replaying of PCAP data (limited only for TCP) between Botmasters shall also be developed.

References

1. JAMK University of Applied Sciences, Jyväskylä Security Technology (JYVSECTEC), Realistic Global Cyber Environment (RGCE). <http://www.jyvsectec.fi/en/rgce/>
2. Zolotukhin, M., Hämäläinen, T., Kokkonen, T., Siltanen, J.: Analysis of HTTP requests for anomaly detection of web attacks. In: 2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing, pp. 406–411, August 2014
3. Floyd, S., Paxson, V.: Difficulties in Simulating the Internet. *IEEE/ACM Trans. Netw.* **9**(4), 392–403 (2001)
4. Casilari, E., Gonzblez, F.J., Sandoval, F.: Modeling of HTTP traffic. *Communications Letters, IEEE* **5**(6), 272–274 (2001)

5. Botta, A., Dainotti, A., Pescapè, A.: A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks (Elsevier)* **14**(15), 3531–3547 (2012)
6. Hong, S.-S., Wu, S.: On interactive internet traffic replay. In: Valdes, A., Zamboni, D. (eds.) *RAID 2005. LNCS*, vol. 3858, pp. 247–264. Springer, Heidelberg (2006)
7. Pries, R., Wamser, F., Staehle, D., Heck, K., Tran-Gia, P.: On traffic characteristics of a broadband wireless internet access. In: *Next Generation Internet Networks, NGI 2009*, pp. 1–7, July 2009
8. Li, T., Liu, J., Lei, Z., Xie, Y.: Characterizing service providers traffic of mobile internet services in cellular data network. In: *2013 5th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, vol. 1, pp. 134–139, August 2013
9. The University of Southern California (USC-ISI), The Information Sciences Institute, TG Traffic Generation Tool. <http://www.postel.org/tg/>
10. The University of Kansas, The Information and Telecommunication Technology Center (ITTC), NetSpec Tool. <http://www.ittc.ku.edu/netspec/>
11. Netperf. <http://www.netperf.org/netperf/>
12. pksh -the Packet Shell. <http://tecsiel.it/pksh/index.html>
13. Università degli Studi di Napoli “Federico II”, D-ITG, Distributed Internet Traffic Generator. <http://traffic.comics.unina.it/software/ITG/>
14. Angrisani, L., Botta, A., Miele, G., Vadursi, M.: An experimental characterization of the internal generation cycle of an open-source software traffic generator. In: *2013 IEEE International Workshop on Measurements and Networking Proceedings (M N)*, pp. 74–78, October 2013
15. Botta, A., Dainotti, A., Pescapè, A.: Do You Trust Your Software-Based Traffic Generator. *IEEE Communications Magazine*, 158–165 (2010)
16. White, B., et al.: An integrated experimental environment for distributed systems and networks. In: *Proceedings of the 5th Symposium on Operating Systems Design and Implementation, 2002*. USENIX Association, December 2002
17. Kleen, A.: Linux Programmer’s Manual RAW(7). <http://www.manpages.info/linux/raw.7.html>
18. Kleen, A., Singhvi, N., Kuznetsov’s, A.: Linux Programmer’s Manual TCP(7). <http://www.manpages.info/linux/tcp.7.html>
19. Postel, J.: Transmission Control Protocol. RFC 793 (INTERNET STANDARD). Updated by RFCs 1122, 3168, 6093, 6528. Internet Engineering Task Force, September 1981. <http://www.ietf.org/rfc/rfc793.txt>
20. Tanase, M.: IP Spoofing: An Introduction. *The Security Blog*, March 2003. <http://www.symantec.com/connect/articles/ip-spoofing-introduction>
21. WireShark Wiki, Libpcap File Format. <http://wiki.wireshark.org/Development/Libpcap/FileFormat/>
22. Tcpreplay. <http://tcpreplay.synfin.net/>
23. Khayari, R.E.A., Rücker, M., Lehman, A., Musovic, A.: ParaSynTG: a parameterized synthetic trace generator for representation of WWW traffic. In: *SPECTS (2008)*, pp. 317–323, June 16–18, 2008
24. Postel, J.: User Datagram Protocol. RFC 768 (INTERNET STANDARD). Internet Engineering Task Force, August 1980. <http://www.ietf.org/rfc/rfc768.txt>
25. The GO Programming Language. <https://golang.org/>
26. Hoare, C.A.R.: Communicating Sequential Processes. *Communications of the ACM* **21**(8), 666–677 (1978)

27. Fette, I., Melnikov, A.: WebSocket Protocol. RFC 6455 (INTERNET STANDARD). Internet Engineering Task Force, December 2011. <http://www.ietf.org/rfc/rfc6455.txt>
28. Ministry of defense press release May 8, 2013. Cyber Security Exercise in Jyväskylä, May 13–17, 2013. Kyberturvallisuusharjoitus Jyväskylässä, May 13–17, 2013. http://www.defmin.fi/ajankohtaista/tiedotteet/2013/kyberturvallisuusharjoitus_jyvaskylassa_13.-17.5.2013.5502.news
29. Finnish Defence Forces Press Release, June 3, 2014, Performance of Cyber Security is developed by co-operation between Government Authorities and University. Kybersuorituskykyä hiotaan viranomaisten ja korkeakoulujen yhteistyöllä. <http://www.fdf.fi/wcm/su+puolustusvoimat.fi/pv.fi+staattinen+sivusto+su/puolustusvoimat/tiedotteet/kybersuorituskyky+hiotaan+viranomaisten+ja+korkeakoulujen+yhteistyolla>
30. Luo, S., Marin, G.A.: Realistic Internet traffic simulation through mixture modeling and a case study. In: Proceedings of the 2005 Winter Simulation Conference, pp. 2408–2416, December 4–7, 2005

PIII

**DATA MINING APPROACH FOR DETECTION OF DDOS
ATTACKS UTILIZING SSL/TLS PROTOCOL**


by

Mikhail Zolotukhin, Timo Hämäläinen, Tero Kokkonen, Antti Niemelä, Jarmo
Siltanen 2015

Lecture Notes in Computer Science, vol. 9247, pp. 274-285

Reproduced with kind permission of Springer International Publishing AG,
Copyright © 2015, Springer International Publishing AG.

Data Mining Approach for Detection of DDoS Attacks Utilizing SSL/TLS Protocol

Mikhail Zolotukhin¹() , Timo Hämäläinen¹, Tero Kokkonen²,
Antti Niemelä², and Jarmo Siltanen²

¹ Department of Mathematical Information Technology, University of Jyväskylä,
Jyväskylä, Finland

{mikhail.m.zolotukhin,timo.t.hamalainen}@jyu.fi

² JAMK University of Applied Sciences, Jyväskylä, Finland
{tero.kokkonen,antti.niemela,jarmo.siltanen}@jamk.fi,
tero.t.kokkonen@student.jyu.fi

Abstract. Denial of Service attacks remain one of the most serious threats to the Internet nowadays. In this study, we propose an algorithm for detection of Denial of Service attacks that utilize SSL/TLS protocol. These protocols encrypt the data of network connections on the application layer which makes it impossible to detect attackers activity based on the analysis of packet payload. For this reason, we concentrate on statistics that can be extracted from packet headers. Based on these statistics, we build a model of normal user behavior by using several data mining algorithms. Once the model has been built, it is used to detect DoS attacks. The proposed framework is tested on the data obtained with the help of a realistic cyber environment that enables one to construct real attack vectors. The simulations show that the proposed method results in a higher accuracy rate when compared to other intrusion detection techniques.

Keywords: Network security · Intrusion detection · DoS attack · Data mining · Anomaly detection

1 Introduction

Due to the fact that Internet has become the major universal communication infrastructure, it is also subject to attacks in growing numbers and varieties. One of the most serious threats to the Internet nowadays is Denial of Service (DoS) attacks [2]. This kind of attack disables the network servers using lots of messages which need response and consumes the bandwidth of the network or the resource of the system. Because it is difficult for an attacker to overload the targets resources from a single computer, modern DoS attacks are launched via a large number of distributed attacking hosts in the Internet. Such distributed DoS (DDoS) attacks can force the victim to significantly downgrade its service

performance or even stop delivering any service [1]. Moreover, DDoS attacks are more complex and harder to prevent compared to conventional DoS attacks.

Traditional DDoS attacks are carried out at the network layer, e.g. ICMP flooding, SYN flooding, and UDP flooding. The purpose of these attacks is to consume the network bandwidth and deny service to legitimate users of the victim systems. This type of attack has been well studied recently and different schemes have been proposed to protect the network and equipment from such bandwidth attacks [3–7]. For this reason, attackers shift their offensive strategies to application-layer attacks. Application-layer DoS attacks may focus on exhausting the server resources such as Sockets, CPU, memory, disk bandwidth, and I/O bandwidth. Unlike network-layer DoS attacks, application-layer attacks do not necessarily rely on inadequacies in the underlying protocols or operating systems. They can be performed by using legitimate requests from legitimately connected network machines. The most popular application-layer DoS attacks are HTTP page flooding and low-rate DoS attacks [1].

The problem of application-layer DoS and DDoS attacks detection is of great interest nowadays. For example, in [8], an anomaly detector based on hidden semi-Markov model is proposed to describe the spatial-temporal patterns of normal users and to detect application-layer DDoS attacks for popular websites. Study [9] proposes an advanced entropy-based scheme, which divides variable rate DDOS attacks into different fields and treats each field with different methods. Paper [10] considers detection of slow DoS attacks by analyzing specific spectral features of network traffic over small time horizons. In [11], authors detect application-layer DDoS attacks by constructing a random walk graph based on sequences of web pages requested by each user. Paper [12] shows a novel detection technique against HTTP-GET attacks, based on Bayes factor analysis and using entropy-minimization for clustering.

Despite the rising interest to the detection of application DDoS attacks, most of the current researches concentrate on various HTTP DDoS attacks. In this study, we propose an algorithm for the detection of DDoS attacks utilizing SSL/TLS protocol [13]. These protocols encrypt the data of network connections in the application layer which makes it impossible to detect attacker's activity based on the analysis of packet's payload. For this reason, we concentrate on statistics that can be extracted from packet headers. Based on these statistics, we build a model of normal user behavior by using several data mining algorithms. Once the model has been built, it is used to detect DoS attacks.

The rest of the paper is organized as follows. Extraction of feature vectors from network packets is considered in Section 2. Section 3 introduces a scheme that uses data mining techniques to build a model of normal user behavior and subsequently detects DoS attacks. In Section 4, we present some numerical results to evaluate the algorithm proposed and compare it with certain analogues. Finally, Section 5 draws the conclusions and outlines future work.

2 Problem Formulation

We consider a web server that provides several services working in two application layer protocols: HTTP and HTTPS. Outgoing and incoming traffic of this server is captured during some time period $[T_s, T_e]$. There is no guarantee that the traffic captured is free of attacks. For this reason, we aim to investigate captured traffic and discover behavior patterns of normal users. In this study, it is assumed that the most part of the traffic captured is normal. In real world, this can be achieved by filtering the traffic with the help of a signature-based intrusion detection system [14]. Once normal behavior patterns have been discovered, these patterns can be used to analyze network traffic and detect DoS and DDoS attacks against the web server in online mode.

We concentrate on attack detection based on the analysis of statistics that can be extracted from packet headers. For this purpose, for each packet, we extract the following information: time stamp, IP address and port of the source, IP address and port of the destination, protocol, packet size, window size, sequence number, acknowledgment number, time to live and TCP flags.

Packets with some common properties passing a monitoring point in a specified time interval can be combined into flows. As a rule, these common properties include IP address and port of the source and IP address and port of the destination. Thus, for each packet we also extract the index of the network traffic flow this packet belongs to.

3 Algorithm

In order to detect network flows related to DoS and DDoS attacks we apply an algorithm that can be divided into two main steps. First, we apply an anomaly detection algorithm in order to find time intervals when an attack takes place. After that, the traffic sent during these time intervals is analyzed in more details to detect flows related to the attack.

3.1 Detection of Network Anomalies

In order to find time intervals which contain traffic anomalies we consider the network traffic as time series. For this reason, the analyzed time period $[T_s, T_e]$ is divided into equal overlapping time bins of length ΔT by points $T_s + \frac{t}{w}\Delta T$, where $t = \{w, w+1, \dots, w\frac{T_e-T_s}{\Delta T} - 1\}$. The length of each time bin ΔT should be picked in such a way that it contains enough information to detect anomalies. Moreover, the value of w should be big enough for the earlier detection of attacks.

For each resulting time interval the following features are calculated:

1. Source IP address sample entropy,
2. Source port sample entropy,
3. Destination IP address sample entropy,
4. Destination port sample entropy,

5. Total number of flows,
6. Average flow duration,
7. Average number of packets in one flow,
8. Average size of packets,
9. Average size of TCP window for packets.

Features 7 – 9 are extracted separately for packets sent from source to the destination and packets sent from the destination to the source.

Sample entropy allows one to capture the degree of dispersal or concentration of the parameter's distribution. Let us assume that in the t -th time interval the i -th parameter has n_i^t unique values which appear with frequencies $p_{i1}^t, \dots, p_{in_i^t}^t$. In this case, sample entropy E_i^t for the i -th parameter in the t -th time interval is defined as follows:

$$E_i^t = - \sum_{k=1}^{n_i^t} p_{ik}^t \log_2 p_{ik}^t. \quad (1)$$

The model of normal user behavior is built by calculating chi-square values. Chi-square values are used to find out how much the observed values of a particular given sample are different from the expected values of the distribution [15, 16]. Chi-square value for the t -th time interval is calculated as follows:

$$\chi_t^2 = \sum_{i=1}^{n_x} \frac{(x_i^t - \mu_i)^2}{\mu_i}, \quad (2)$$

where n_x is the number of features (in our case it is equal to 12), x_i^t is the value of the i -th feature for the t -th time interval and μ_i is the mean value of the i -th feature during the analyzed time period $[T_s, T_e]$. Chi-square value in (2) is similar to the traditional chi-square statistic for independence tests.

Once chi-square values have been calculated, some filtering can be applied to remove outliers and noise to build the model of normal user behavior. As proposed in study [17], we define the following distance function:

$$d(\chi_{t_1}^2, \chi_{t_2}^2) = p(\chi_{t_1}^2 \text{ is normal}) - p(\chi_{t_2}^2 \text{ is normal}), \quad (3)$$

where $p(x \text{ is normal})$ is the probability that value x is normal and can be found as follows:

$$p(x \text{ is normal}) = \begin{cases} \frac{\sigma_{\chi^2}^2}{(x - \mu_{\chi^2})^2}, & \text{if } x \geq \mu_{\chi^2} + \sigma_{\chi^2}, \\ 1, & \text{if } x < \mu_{\chi^2} + \sigma_{\chi^2}, \end{cases} \quad (4)$$

where μ_{χ^2} and $\sigma_{\chi^2}^2$ are the mean and the variance of chi-square values, respectively. In this case, the distance d between a normal pattern and an outlier pattern is expected to be higher than the distance d between two normal patterns or two outlier patterns.

It is easy to divide all the chi-square values into two clusters, i.e. normal values and outliers, by using single-linkage clustering algorithm. This algorithm

belongs to a class of agglomerative hierarchical clustering methods. In the beginning of the algorithm, each chi-square value forms a cluster, i.e. the number of clusters is equal to the number of chi-square values, and every cluster consists of only one element. At each iteration, the algorithm combines those two clusters which are the least distant from each other. The distance $d(C_i, C_j)$ between two clusters C_i and C_j is defined as the minimal distance between two chi-square values of these clusters, such that one value is taken from each cluster:

$$d(C_i, C_j) = \min_{\chi_t^2 \in C_i, \chi_\tau^2 \in C_j} (d(\chi_t^2, \chi_\tau^2)). \quad (5)$$

The algorithm stops when the required number of clusters is formed. In our case, this number is equal to two: one cluster for normal values C_n and another one for outliers C_o . All outliers are removed from the model and all normal values are used for detecting anomalies.

The χ^2 statistic approximately follows a normal distribution according to the central limit theorem [18], regardless of the distribution that each of the extracted features follows. If we make the assumption that values of cluster C_n resemble a normal distribution, approximately 99.7% of all chi-square values should fall within three standard deviations of the mean value of cluster C_n .

In order to classify network traffic during the recent time interval, we discover necessary features for this time interval and calculate chi-square value χ^2 . Network traffic at this time interval is classified as anomalous if

$$\chi^2 > \bar{\mu}_{\chi^2} + \alpha \bar{\sigma}_{\chi^2}, \quad (6)$$

where $\bar{\mu}_{\chi^2}$ and $\bar{\sigma}_{\chi^2}$ are the mean and the standard deviation of chi-square values from cluster C_n , and parameter $\alpha \geq 3$. Thus, we can find the time interval when an attack takes place.

3.2 Detection of Intrusive Flows

In order to build a model for detection of network flows related to a DDoS attack we consider time intervals during which traffic is classified as normal. For each flow in these time intervals, the following information is extracted:

1. Average, minimal and maximal size of packets,
2. Average, minimal and maximal size of TCP window,
3. Average, minimal and maximal time since the previous packet,
4. Average, minimal and maximal time to live,
5. Percentage of packets that have TCP flag SYN,
6. Percentage of packets that have TCP flag ACK,
7. Percentage of packets that have TCP flag PSH,
8. Percentage of packets that have TCP flag RST,
9. Percentage of packets that have TCP flag FIN.

We extract these features separately for both directions: from the source to the destination and from the destination to the source. Thus, the i -th flow is presented as as a feature vector y_i of length $n = 34$.

Values of vectors y_i can have different scales. In order to standardize the feature vectors of the training set max-min normalization is used. Max-min normalization performs a linear alteration on the original data so that the values are normalized within the given range [19]. In this paper, we map vectors y_i to range $[0, 1]$. To map a value y_{ij} of an attribute $(y_{1j}, y_{2j}, \dots, y_{nj})$ from range $[\min_{1 \leq i \leq n} y_{ij}, \max_{1 \leq i \leq n} y_{ij}]$ to range $[0, 1]$, the computation is carried out as follows

$$z_{ij} = \frac{y_{ij} - \min_{1 \leq i \leq n} y_{ij}}{\max_{1 \leq i \leq n} y_{ij} - \min_{1 \leq i \leq n} y_{ij}}, \quad (7)$$

where z_{ij} is the new value of y_{ij} in the required range.

The model of normal behavior can be found with the help of density-based spatial clustering of applications with noise (DBSCAN). DBSCAN is a powerful density-based clustering algorithm, which is often used for detecting outliers. It discovers clusters in the training dataset starting from the estimated density distribution of feature vectors [20].

DBSCAN requires two parameters: the size of neighborhood ε and the minimum number of points required to form a cluster N_{min} . The algorithm starts with an arbitrary feature vector z that has not been checked. The number of feature vectors $N_\varepsilon(z)$ contained in the ε -neighborhood of z is found and compared to N_{min} :

$$\begin{cases} \text{If } N_\varepsilon(x) < N_{min}, \text{ then } z \text{ is labeled as noise,} \\ \text{If } N_\varepsilon(x) \geq N_{min}, \text{ then } z \text{ is a part of a cluster.} \end{cases} \quad (8)$$

Vectors marked as noise might later be discovered as a part of another vector ε -environment and hence be made a part of a cluster. If a vector is found to be a part of a cluster, its ε -neighborhood is also part of that cluster. After that, each point \bar{z} contained in the ε -neighborhood is checked. If \bar{z} is density-reached from z with respect to ε and N_{min} , it is added to the cluster. Vector \bar{z} is density-reachable from z with respect to ε and N_{min} , if there is a chain of points z_1, z_2, \dots, z_m , where $z_1 = z$ and $z_m = \bar{z}$, such that $\forall i \in \{1, 2, \dots, m-1\}$ the two following conditions are satisfied:

$$\begin{cases} d(z_i, z_{i+1}) \leq \varepsilon, \\ N_\varepsilon(z_i) \geq N_{min}, \end{cases} \quad (9)$$

where $d(z_i, z_{i+1})$ is the Euclidean distance between z_i and z_{i+1} . The cluster is built when all vectors density-reachable from z have been found. Then, a new unvisited vector is processed, leading to a discovery of a further cluster or noise. As a rule, all points which remain cluster-less after the algorithm is finished are classified as anomalies. Since, we assume that all flows in time intervals considered are normal we consider all cluster-less points as clusters which contain only one point. Figure 1 shows an example of the application of DBSCAN, with $N_{min} = 3$ and $\varepsilon = 0.25$ (radius of each circle).

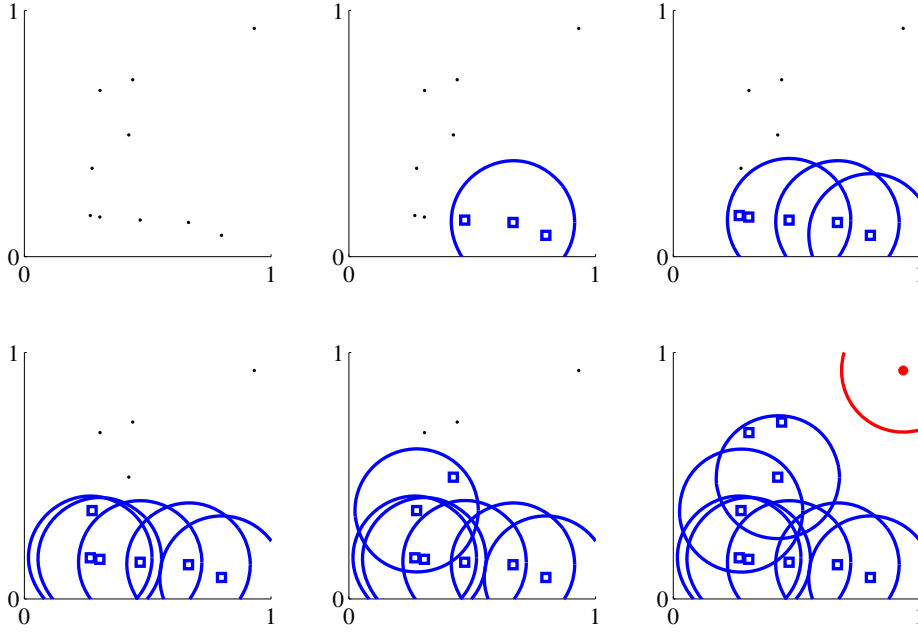


Fig. 1. An example of the application of DBSCAN

Thus, DBSCAN can find arbitrarily-shaped clusters and does not require to know the number of clusters in the dataset a priori. DBSCAN requires just two parameters that should be optimally chosen: the size of neighborhood ε and the minimum number of points required to form a cluster N_{min} . There are several studies devoted to this problem [21, 22]. In this study, N_{min} is selected to be equal to 10% of the total number of flows in time intervals that have been classified as normal, and ε is equal to the average Euclidean distance between feature vectors corresponding to these flows.

Once the clustering has been completed, the maximal pairwise Euclidean distance is calculated for each cluster. Let us denote the maximal pairwise distance of elements of the i -th cluster C_i as m_i :

$$m_i = \max_{z_j, z_k \in C_i} d(z_j, z_k). \quad (10)$$

For each cluster-less point, m_i is selected to be equal to the minimal value of maximal pairwise distances of elements of all clusters. In order to classify a new flow in the recent time interval, all necessary features are extracted from this flow into vector z . After this, the cluster (or cluster-less point) which contains the vector which is the least distant from z is found:

$$i^* = \arg \min_{z_j \in C_i} d(z, z_j). \quad (11)$$

This flow is classified as an attack if the distance between z and the least distant vector is greater than the corresponding maximal pairwise distance:

$$\min_{z_j \in C_{i^*}} (d(z, z_j)) > m_{i^*}. \quad (12)$$

After this, the traffic that corresponds to intrusive flows can be blocked to prevent the development of the attack.

4 Numerical Simulations

Realistic Global Cyber Environment (RGCE) was used for the DoS/DDoS data generation. RGCE is closed Internet-like environment developed and hosted by JAMK University of Applied Sciences. As one of the main features, RGCE executes real IP-addresses and geolocations [23,24]. A web server serving a static main page with some text and a picture through HTTPS (SSL/TLS) was installed as a part of the RGCE infrastructure. After that, RGCE was used to generate both legitimate user traffic and DoS/DDoS traffic to this web server. RGCE data generation software uses botnet architecture where the bots are controlled by botmasters [23]. The bots were distributed in RGCE with different global IP-addresses that also simulates global geological distribution inside the RGCE. Two different types of bots were used for the data generation: bots that generated legitimate HTTPS traffic and bots that generated non-legitimate HTTPS traffic (DOS/DDoS) to the test web server. The bot that generated legitimate traffic crawled through the targeted web-site and generated requests for found content and followed found links. The bot that generated the non-legitimate traffic created multiple requests to the targeted single web-site page. The resulting data set contains mainly HTTPS traffic as the test web server communicated with the clients only through encrypted protocol. The non-encrypted traffic (HTTP) in the dataset are the initial handshakes between the clients (bots) and the test web server before the encrypted channel was created. All the traffic was captured as PCAP-files [25] for the numerical analysis.

To test attack detection algorithms we consider one of such PCAP-files. This file contains 80 minutes of traffic or 429202 traffic flows. To build a model of normal user behavior we use the training set which contains 8 minutes or 10% of traffic. This traffic is free of attacks but it contains few outliers. The testing set contains remaining 72 minutes or 90% of the traffic. This set contains DDoS attack from two different subnets. The attack starts at 8:53.19 and ends at 68:37.2. The length of time bin ΔT was selected to be equal to five seconds, whereas parameter w is equal to 5. In this case, each time interval contained enough information to detect anomalies every second.

Figure 4 shows features extracted for each second of the time period considered. On this figure, blue line represents normal traffic whereas red line corresponds to the DDoS attack. Chi-square values calculated for each second for the features extracted are presented on Figure 4. As one can see, chi-square values corresponding to time intervals when the attack takes place are higher. In order to find time intervals that contain anomalous traffic we apply the method described in the previous section. Parameter α is selected to be equal to 5. As a result, time of the start and time of the end of the attack can be defined up to the size of time window which is equal to 1 second in our simulation.

We compare the performance of DBSCAN with other clustering and outlier detection techniques: K-means, K-Nearest Neighbors (KNN), Support Vector

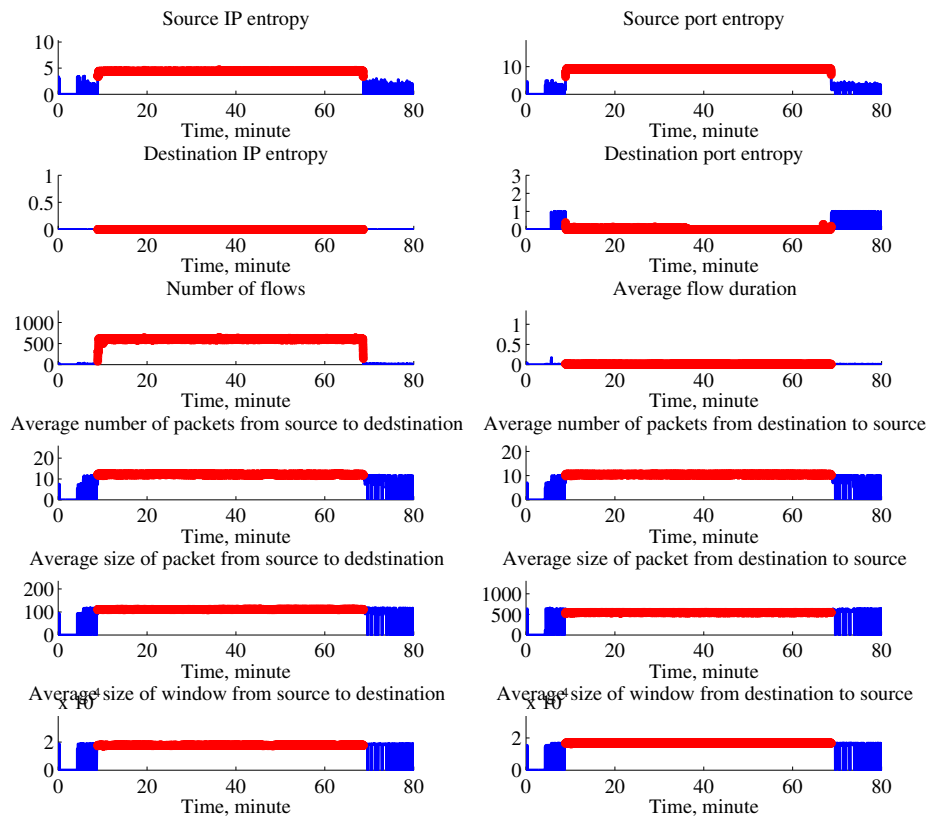


Fig. 2. Features extracted for each time interval to find anomalous network traffic (blue line - normal traffic, red line - traffic during the attack).

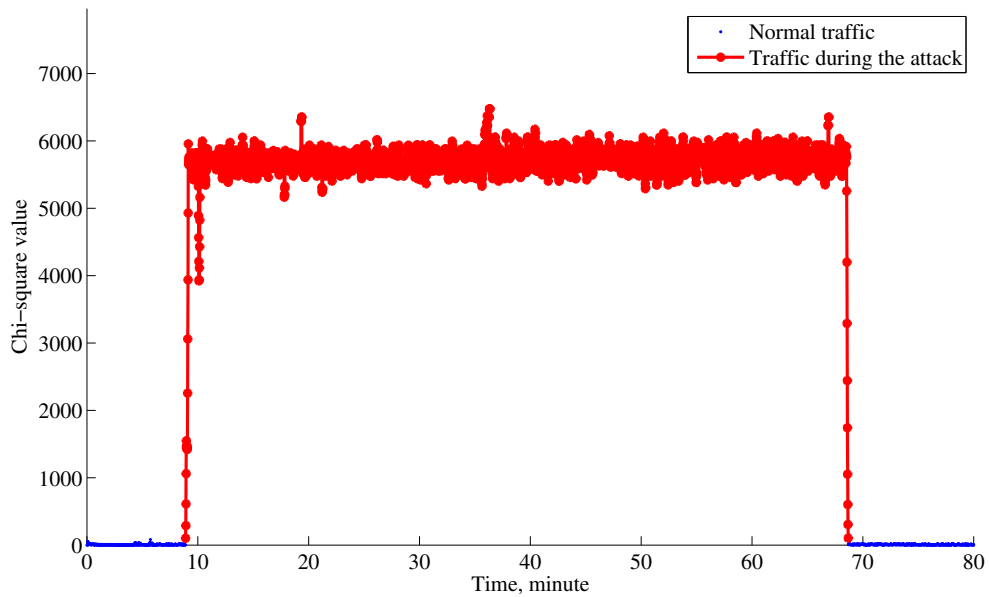


Fig. 3. Chi-square values of features extracted.

Data Description (SVDD) and Self-organizing Map (SOM). To evaluate the performance of each algorithm, the following characteristics are calculated in our tests:

- True positive rate – the ratio of the number of correctly detected anomalous samples to the total number of anomalous samples in the testing set
- False positive rate – the ratio of the number of normal samples classified as anomalous to the total number of normal samples in the testing set
- Detection accuracy – the ratio of the total number of normal samples detected as normal and anomalies detected as anomalies to the total number of samples in the testing set.

Figure 4 shows the dependence between false positive and true positive rates for different detection methods and different parameters. To compare the accuracy of the methods, we select optimal parameters of the methods based on the training set. The comparison results are listed in Table 1. As one can notice, all methods are able to detect all intrusive conversations ($TPR = 100\%$). However, SVDD gives the worst results with the highest number of false alarms ($FPR = 6.0627\%$). DBSCAN outperforms other methods in terms of accuracy (Accuracy = 99.9993%) and number of false alarms ($FPR = 0.0697\%$). DBSCAN's FPR equal to 0.0697% corresponds to only one flow in the testing set which is normal but is classified as an attack. This false alarm can be explained by the fact that

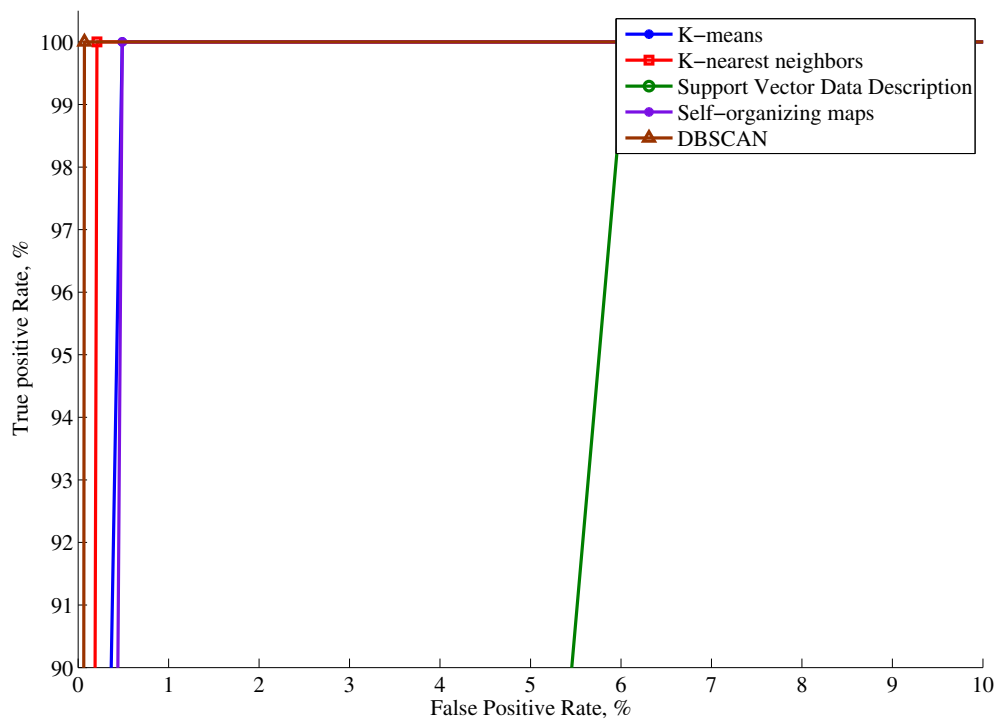


Fig. 4. Dependence between false positive and true positive rates for different detection methods and different parameters.

Table 1. Intrusion detection accuracy of different detection methods.

Algorithm	TPR	FPR	Accuracy
K-means	100 %	0.4878 %	99.9951 %
KNN	100 %	0.2091 %	99.9979 %
SVDD	100 %	6.0627 %	99.9390 %
SOM	100 %	0.4878 %	99.9951 %
DBSCAN	100 %	0.0697 %	99.9993 %

the number of conversations in the training set is very low and probably not enough for building the accurate model of normal user behavior.

5 Conclusion

In this paper, the scheme for detection of Denial of Service attacks that utilize SSL/TLS protocol is proposed. The scheme is based on the analysis of statistics extracted from packet headers. Based on these statistics, we build a model of normal user behavior by calculating Chi-square values and clustering flows with DBSCAN. Once the model has been built, it is used to detect DDoS attacks. The method is tested on the data obtained with the help of a realistic cyber environment. The simulation results show that the scheme proposed allows to detect all intrusive flows with very low number of false alarms. In the future, we are planning to improve the algorithm in terms of the detection accuracy, and test it with a dataset which contains traffic captured during several days.

References

1. Durcekova, V., Schwartz, L., Shahmehri, N.: Sophisticated denial of service attacks aimed at application layer. In: ELEKTRO, pp. 55–60 (2012)
2. Gu, Q., Liu, P.: Denial of Service Attacks. Handbook of Computer Networks: Distributed Networks, Network Planning, Control, Management, and New Trends and Applications, vol. 3. John Wiley & Sons (2008)
3. Peng, T., Leckie, K.R.M.C.: Protection from distributed denial of service attacks using history-based IP filtering. In: Proc. of IEEE International Conference on Communications, vol. 1, pp. 482–486 (2003)
4. Limwiwatkul, L., Rungsawangr, A.: Distributed denial of service detection using TCP/IP header and traffic measurement analysis. In: Proc. of IEEE International Symposium on Communications and Information Technology, vol. 1, pp. 605–610 (2004)
5. Yuan, J., Mills, K.: Monitoring the macroscopic effect of DDoS flooding attacks. IEEE Tran. Dependable and Secure Computing **2**(4), 324–335 (2005)
6. Chen, R., Wei, J.-Y., Yu, H.: An improved grey self-organizing map based dos detection. In: Proc. of IEEE Conference on Cybernetics and Intelligent Systems, pp. 497–502 (2008)
7. Ke-Xin, Y., Jian-Qi, Z.: A novel DoS detection mechanism. In: Proc. of International Conference on Mechatronic Science, Electric Engineering and Computer (MEC), pp. 296–298 (2011)

8. Xie, Y., Yu, S.-Z.: Monitoring the Application-Layer DDoS Attacks for Popular Websites. *IEEE/ACM Transactions on Networking* **17**(1), 15–25 (2008)
9. Zhang, J., Qin, Z., Ou, L., Jiang, P., Liu, J., Liu, A.: An advanced entropy-based DDOS detection scheme. In: *Proc. of International Conference on Information Networking and Automation (ICINA)*, vol. 2, pp. 67–71 (2010)
10. Aiello, M., Cambiaso, E., Mongelli, M., Papaleo, G.: An on-line intrusion detection approach to identify low-rate DoS attacks. In: *Proc. of International Carnahan Conference on Security Technology (ICCST)*, pp. 1–6 (2014)
11. Xu, C., Zhao, G., Xie, G., Yu, S.: Detection on application layer DDoS using random walk model. In: *Proc. of IEEE International Conference on Communications (ICC)*, pp. 707–712 (2014)
12. Chwalinski, P., Belavkin, R., Cheng, X.: Detection of application layer DDoS Attacks with clustering and bayes factors. In: *Proc. of IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 156–161 (2013)
13. Dierks, T., Rescorla, E.: The transport layer security (TLS) protocol. IETF RFC 4346 (2006)
14. Gollmann, D.: *Computer Security*, 2nd edn. Wiley (2006)
15. Ye, N., Borrer, C.M., Parmar, D.: Scalable Chi-Square Distance versus Conventional Statistical Distance for Process Monitoring with Uncorrelated Data Variables. *Quality and Reliability Engineering International* **19**(6), 505–515 (2003)
16. Muraleedharan, N., Parmar, A., Kumar, M.: A flow based anomaly detection system using chi-square technique. In: *Proc. of the 2nd IEEE International Advance Computing Conference (IACC)*, pp. 285–289 (2010)
17. Corona, I., Giacinto, G.: Detection of server-side web attacks. In: *Proc of JMLR: Workshop on Applications of Pattern Analysis*, pp. 160–166 (2010)
18. Johnson, R., Wichern, D.: *Applied Multivariate Statistical Analysis*. Prentice-Hall, Upper Saddle River (1998)
19. Saranya, C., Manikandan, G.: A Study on Normalization Techniques for Privacy Preserving Data Mining. *International Journal of Engineering and Technology (IJET)* **5**(3), 2701–2704 (2013)
20. Ester, M., Krieger, H., Jörg, S., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise, pp. 226–231. AAAI Press (1996)
21. Kim, J.: The anomaly detection by using DBSCAN clustering with multiple parameters. In: *Proc. of the ICISA*, pp. 1–5 (2011)
22. Smiti, A.: DBSCAN-GM: an improved clustering method based on gaussian means and DBSCAN techniques. In: *Proc. of the IEEE 16th International Conference on Intelligent Engineering Systems (INES)*, pp. 573–578 (2012)
23. Jyvsectec-rgce - homepage. <http://www.jyvsectec.fi/en/rgce/>
24. Zolotukhin, M., Hämäläinen, T., Kokkonen, T., Siltanen, J.: Analysis, of http requests for anomaly detection of web attacks. In: *Proc. of the 12th IEEE International Conference on Dependable, Autonomic and Secure Computing*, pp. 406–411 (2014)
25. WireShark Wiki, Libpcap File Format. <http://wiki.wireshark.org/Development/LibpcapFileFormat/>

PIV

**ONLINE DETECTION OF ANOMALOUS NETWORK FLOWS
WITH SOFT CLUSTERING**

by

Mikhail Zolotukhin, Timo Hämäläinen, Tero Kokkonen, Jarmo Siltanen 2015

7th International Conference on New Technologies, Mobility and Security,
NTMS 2015, Paris, France, July 27-29, 2015, Proceedings

Reproduced with kind permission of IEEE,
Copyright © 2015, IEEE.

Online Detection of Anomalous Network Flows with Soft Clustering

Mikhail Zolotukhin, Timo Hämäläinen
Department of Mathematical Information Technology
University of Jyväskylä, Jyväskylä, Finland
email: {mikhail.m.zolotukhin, timo.t.hamalainen}@jyu.fi

Tero Kokkonen, Jarmo Siltanen
Institute of Information Technology
JAMK University of Applied Sciences, Jyväskylä, Finland
email: {tero.kokkonen, jarmo.siltanen}@jamk.fi

Abstract—In this study, we apply an anomaly-based approach to analyze traffic flows transferred over a network to detect the flows related to different types of attacks. Based on the information extracted from network flows a model of normal user behavior is discovered with the help of several clustering techniques. This model is then used to detect anomalies within recent time intervals. Since this approach is based on normal user behavior, it can potentially detect zero-day intrusions. Moreover, such a flow-based intrusion detection approach can be used in high speeds since it is based on information in packet headers, and, therefore, has to handle a considerably lesser amount of data. The proposed framework is tested on the data obtained with the help of a realistic cyber environment (RGCE) that enables one to construct real attack vectors. The simulations show that the proposed method results in a higher accuracy rate when compared to other intrusion detection techniques.

I. INTRODUCTION

While the number of the Internet users and high-bandwidth services offered to them is growing, the number of observed network attacks are increasing exponentially [1]. The research community, aiming to timely detection of intruders and prevention of damage, has developed a growing interest in intrusion detection [2]. Despite of the rapid development of new tools and strategies to detect network attacks, signature-based detection remains the most popular approach in commercial intrusion detection [3]. This approach is based on comparing new data with a knowledge base of known intrusions and, consequently, cannot recognize new attacks. On the other hand, an anomaly-based intrusion detection system (IDS) compares input data with a model of normal user behavior and marks a significant deviation from this model as an anomaly [4], [5], [6], [7], [8], [9]. As a result, such systems can potentially also detect attacks that have never been seen before.

A traditional IDS inspects every packet received by the computer or detected in the network in which the system has been deployed. The recent rise in the amount of traffic and the increase in line speed put a heavy computational load and resource consumption on such traditional payload-based IDSs [10]. For this reason, nowadays, researchers try to solve the problem of intrusion detection in high-speed networks by flow-based traffic analysis [6], [10], [2]. A flow is a group of IP packets, with some common properties, passing a monitoring point in a specified time interval [7]. Flow-based monitoring is based on information in packet headers, and, compared to payload-based IDSs, flow-based IDSs have to handle a considerably lower amount of data [6]. However, since flow measurements provide only an aggregated view of the data

transferred over the network and between hosts in terms of number of packets and bytes, they cannot reach the accuracy of the payload-based approach. Thus, a flow-based IDS is not expected to completely substitute a payload-based system, but can be used along with it to allow early detection in environments in which payload-based inspection is not scalable [6].

The problem of flow-based network anomaly detection is of great interest nowadays. For example, [7] proposes a flow-based anomaly detection system, which is based on a Multi-Layer Perceptron and Gravitational Search Algorithm for optimizing interconnection weights between the layers. The system, having been trained with a flow-based data set, can classify benign and malicious flows and do it with very high accuracy rate. In [8], the proposed IDS operates on network flows and uses one-class support vector machine for their analysis. In contrast to traditional anomaly detection systems, the system is trained with malicious rather than with benign network data. Systems proposed in [7] and [8] consider the network traffic as a set of non-interconnected flows and, therefore, cannot manage to detect types of attacks that are only visible when the network traffic is analyzed over time. For this reason, researchers consider traffic as time series, which are a powerful tool to describe network traffic evolution patterns. For example, [11] describes the creation of an anomaly-based intrusion detection system that analyzes flow data through a firewall protecting a controlled network environment. The study suggests the use of statistical measures, such as standard deviation and interquartile range, to develop traffic forecasts based upon this flow data. Paper [12] presents a functional extension for both NetFlow and IPFIX flow exporters. This extension allows for timely intrusion detection and mitigation of large flooding attacks. Study [13] proposes a novel approach of network anomaly detection method based on chi-square mechanism through transport layer protocol behavior analysis. In [14], the authors present an anomaly detection system using a seven-dimensional flow analysis based on the improved Holt-Winters forecasting method on the traffic characterization of each one of the different analyzed dimensions.

This study combines these two approaches together. First, we consider network traffic as a time series. For each time interval in a series, we extract various per-flow information. Based on this information, flows transferred during each time interval are divided into clusters. After that, distributions of the resulting clusters are compared against each other to find time bins which contain anomalous network traffic. Finally, traffic

transferred during "anomalous" time bins is analyzed and the flows related to the attack are discovered.

In order to cluster flows inside each time bin, we apply a soft clustering approach. In soft clustering, a set of membership levels is associated with each data sample. These levels indicate the probability that the data sample belongs to a particular cluster. Various soft clustering methods are used by researchers for flow-based intrusion detection. For example, [15] proposes to detect anomalous network flows based on a fuzzy c-means clustering algorithm. In [16], an adaptive flow clustering method based on fuzzy logic is introduced, and it is demonstrated that this method prevents memory and CPU resources from becoming exhausted during various flooding attacks. Paper [17] presents an intrusion detection technique based on fuzzy-neural networks and used with a k-means clustering algorithm and radial support vector machine.

The rest of the paper is organized as follows. Extraction of feature vectors from network flows is considered in Section II. Section III introduces a scheme that uses data mining techniques to build a model of normal user behavior and subsequently detects intrusive flows. In Section IV, we present some numerical results to evaluate the algorithm proposed and compare it with certain analogues. Finally, Section V draws the conclusions and outlines future work.

II. FEATURE EXTRACTION

We consider the network traffic as time series. For this reason, the analyzed time period $[T_s, T_e]$ is divided into equal overlapping time bins of length ΔT by points $T_s + \frac{t}{w}\Delta T$, where $t = \{w, w+1, \dots, w\frac{T_e-T_s}{\Delta T} - 1\}$. The length of each time bin ΔT should be picked in such a way that it contains enough information to detect anomalies. Moreover, the value of w should be big enough for the earlier detection of attacks.

We concentrate on the intrusion detection based on the analysis of network traffic flows. A flow is a group of IP packets with some common properties passing a monitoring point in a specified time interval. In this study, we assume that these common properties include the IP address and port of the source and IP address and port of the destination, while the time interval corresponds to one of the bins defined above. Thus, all packets which have the same source's and destination's IP address and port are considered as one network traffic flow. Flow measurements provides an aggregated view of traffic information and drastically reduce the amount of data to be analyzed.

For each flow at each time bin we extract the following per-flow information: time of the start, IP address and port of the source, IP address and port of the destination, the number of packets transferred from the source to the destination, the flow duration in seconds, the maximal, minimal and average size of packets transferred during this flow (see Table I). We divide vector f_i^t of features extracted from the i -th traffic flow at the t -th time bin into two parts:

$$f_i^t = (x_i^t, y_i^t), \quad (1)$$

where

$$\begin{aligned} x_i^t &= (I_i^{src,t}, P_i^{src,t}, I_i^{dst,t}, P_i^{dst,t}), \\ y_i^t &= (N_i^t, B_i^{max,t}, B_i^{min,t}, B_i^{avg,t}, D_i^t). \end{aligned} \quad (2)$$

TABLE I. FEATURES EXTRACTED FROM THE i -TH TRAFFIC FLOW AT THE t -TH TIME BIN

Feature	Description
$I_i^{src,t}$	IP address of the source
$P_i^{src,t}$	Source port
$I_i^{dst,t}$	IP address of the destination
$P_i^{dst,t}$	Destination port
N_i^t	Number of packets
$B_i^{max,t}$	Maximal packet size
$B_i^{min,t}$	Minimal packet size
$B_i^{avg,t}$	Average packet size
D_i^t	Flow duration

In addition, the total number of flows n^t presented at the t -th time bin is taken into account.

III. ALGORITHM

Let us assume that there is a special data set of which it is known that it contains only legitimate network traffic. We can use this data as a training set in order to build a model of normal user behavior. This model is then applied to find attacks in online mode.

In order to detect network flows related to different types of attacks, we apply a clustering algorithm to flows transferred during each time bin. After that, distributions of clusters are compared to find time bins which contain network traffic anomalies caused by an attack. Finally, traffic transferred during anomalous time bins is analyzed and flows related to the attack are discovered.

A. Flow clustering

The clustering technique we applied in this study is based on Expectation Maximization (EM) algorithm for the Gaussian mixture-density parameter estimation problem [18]. We assume that the second parts of flow feature vectors $Y^t = \{y_i^t\}_{i=1}^{n^t}$ are distributed according to the following probabilistic model:

$$p(y_i^t | \Theta^t) = \sum_{k=1}^M \alpha_k^t p_k^t(y_i^t | \mu_k^t, \Sigma_k^t), \quad (3)$$

where M is the number of Gaussians which can be determined by the Bayesian Information Criterion [19], $p(y_i^t | \Theta^t)$ is probability density function of $\Theta^t = (\alpha_1^t, \dots, \alpha_M^t, \mu_1^t, \dots, \mu_M^t, \Sigma_1^t, \dots, \Sigma_M^t)$, $\alpha_1^t, \dots, \alpha_M^t$ are parameters such as $\sum_{k=1}^M \alpha_k^t = 1$, and $p_k^t(y_i^t | \mu_k^t, \Sigma_k^t)$ is the density function of the k -th Gaussian with mean value μ_k^t and covariance matrix Σ_k^t , which can be calculated as follows:

$$p_k^t(y_i^t | \mu_k^t, \Sigma_k^t) = \frac{e^{-\frac{1}{2}(y_i^t - \mu_k^t)^T (\Sigma_k^t)^{-1} (y_i^t - \mu_k^t)}}{(2\pi)^{5/2} |\Sigma_k^t|^{1/2}}. \quad (4)$$

It is assumed that there are unobserved data items $\bar{Y}^t = \{\bar{y}_i^t\}_{i=1}^{n^t}$ whose values inform us which component density generates each data sample y_i^t . For each $i \in (1, \dots, n^t)$, $\bar{y}_i^t \in (1, \dots, M)$ and $\bar{y}_i^t = k$ if the i -th sample was generated by the k -th mixture component.

To find the optimal distribution parameters Θ^t , the EM algorithm applies iteratively the expectation and maximization

steps. During the expectation step of iteration j , the following function is introduced:

$$Q(\Theta^t, (\Theta^t)^{(j-1)}) = E(\log p(Y^t, \bar{Y}^t | \Theta^t) | Y^t, (\Theta^t)^{(j-1)}), \quad (5)$$

which is the expected value of the complete-data log-likelihood $\log(p(Y^t, \bar{Y}^t | \Theta^t))$ with respect to the unknown data \bar{Y}^t given the observed data Y^t and the current parameter estimates $(\Theta^t)^{(j-1)} = ((\alpha_1^t)^{(j-1)}, \dots, (\alpha_M^t)^{(j-1)}, (\mu_1^t)^{(j-1)}, \dots, (\mu_M^t)^{(j-1)})$ and $(\Sigma_1^t)^{(j-1)}, \dots, (\Sigma_M^t)^{(j-1)}$. It is assumed that $(\Theta^t)^{(0)}$ is a random vector. Function $Q(\Theta^t, (\Theta^t)^{(j-1)})$ can be reduced to the following form:

$$Q(\Theta^t, (\Theta^t)^{(j-1)}) = \sum_{k=1}^M \sum_{i=1}^{n^t} \log(\alpha_k^t) p(k | y_i^t, (\Theta^t)^{(j-1)}) + \sum_{k=1}^M \sum_{i=1}^{n^t} \log(p_k^t(y_i^t | \mu_k^t, \Sigma_k^t)) p(k | y_i^t, (\Theta^t)^{(j-1)}), \quad (6)$$

where $p(k | y_i^t, (\Theta^t)^{(j-1)})$ is the probability that the value of the Gaussian-selector is k , given the observation y_i^t , and the mixture parameters are $(\Theta^t)^{(j-1)}$. This probability can be estimated as follows:

$$p(k | y_i^t, (\Theta^t)^{(j-1)}) = \frac{(\alpha_k^t)^{(j-1)} p_k^t(y_i^t | (\mu_k^t)^{(j-1)}, (\Sigma_k^t)^{(j-1)})}{\sum_{k=1}^M (\alpha_k^t)^{(j-1)} p_k^t(y_i^t | (\mu_k^t)^{(j-1)}, (\Sigma_k^t)^{(j-1)})}. \quad (7)$$

During the maximization step of the j -th iteration, the expectation function $Q(\Theta^t, (\Theta^t)^{(j-1)})$ is maximized to find the next estimation $(\Theta^t)^{(j)}$ of distribution Θ^t :

$$(\Theta^t)^{(j)} = \operatorname{argmax}_{\Theta^t} (Q(\Theta^t, (\Theta^t)^{(j-1)})). \quad (8)$$

This maximization is carried out by introducing Lagrangian multipliers and calculating partial derivatives. As a result, estimates of distribution parameters can be calculated as follows:

$$\begin{aligned} (\alpha_k^t)^{(j)} &= \frac{1}{n^t} \sum_{i=1}^{n^t} p(k | y_i^t, (\Theta^t)^{(j-1)}), \\ (\mu_k^t)^{(j)} &= \frac{\sum_{i=1}^{n^t} y_i^t p(k | y_i^t, (\Theta^t)^{(j-1)})}{\sum_{i=1}^{n^t} p(k | y_i^t, (\Theta^t)^{(j-1)})}, \\ (\Sigma_k^t)^{(j)} &= \frac{\sum_{i=1}^{n^t} p(k | y_i^t, (\Theta^t)^{(j-1)}) (y_i^t - (\mu_k^t)^{(j)}) (y_i^t - (\mu_k^t)^{(j)})^T}{\sum_{i=1}^{n^t} p(k | y_i^t, (\Theta^t)^{(j-1)})}. \end{aligned} \quad (9)$$

The algorithm proceeds by using the newly-derived parameters as the estimation for the next iteration. These two steps are repeated as necessary. Each iteration is guaranteed to increase the log-likelihood, and the algorithm is guaranteed to converge to a local maximum of the likelihood function.

Thus, for data sample y_i^t the EM algorithm calculates the probability $p(k, y_i^t, \Theta^t)$ that the sample belongs to the k -th distribution under model Θ^t . We define the probability P_{ij}^t of flows $f_i^t = (x_i^t, y_i^t)$ and $f_j^t = (x_j^t, y_j^t)$ belonging to the same cluster as follows:

$$P_{ij}^t = (1 - d_H(x_i^t, x_j^t)) \sum_{k=1}^M p(k, y_i^t, \Theta^t) \times p(k, y_j^t, \Theta^t), \quad (10)$$

where $d_H(x_i^t, x_j^t)$ is the Hamming distance between vectors x_i^t and x_j^t , which is defined as the percentage of elements that differ. Thus, if two flows have the same source or destination IP or port, the probability that these flows belong to the same cluster increases.

Finally, in order to obtain flow clusters, we apply a single-linkage clustering algorithm [20]. This algorithm belongs to a class of agglomerative hierarchical clustering methods. In the beginning of the algorithm, each feature vector in the training dataset forms a cluster, i.e. the number of clusters is equal to the number of feature vectors, and every cluster consists of only one element. During the algorithm iterations, these clusters are sequentially combined into larger clusters. At each iteration, the algorithm combines those two clusters which are the least distant from each other. The distance $d(C_l, C_k)$ between two clusters C_l and C_k is defined as follows:

$$d(C_l, C_k) = \min_{f_i^t \in C_l, f_j^t \in C_k} (P_{ij}^t). \quad (11)$$

The algorithm stops when the required number of clusters n_C is formed. We form the same number of clusters for all time bins.

B. Detection of network anomalies

The way feature vectors are distributed across clusters during an attack differs markedly from the vector distribution corresponding to legitimate traffic. Thus, we can define whether a computer or network system is under attack during the current time interval.

Let us consider training set Ω which contains only legitimate network traffic. For each time bin in the training set, the proposed flow clustering algorithm is applied. After that, for each resulting cluster the number of feature vectors contained in the cluster is counted. We introduce histogram vector $h^t = (h_1^t, h_2^t, \dots, h_{n_C}^t)$ which consists of these numbers sorted in descending order. Once histogram vector h^t has been calculated for each time bin t from the training set Ω , matrix H is built using these vectors. The t -th row $H_t = (H_{t1}, \dots, H_{tn_C})$ of this matrix corresponds to the histogram vector calculated for the t -th time bin:

$$H_t = (h_1^t, h_2^t, \dots, h_{n_C}^t). \quad (12)$$

After that, for each row H_t , we calculate the Mahalanobis distance $d_M(H_t, H)$ between this row and matrix H . The Mahalanobis distance measures how many standard deviations away H_t is from the mean of H . This distance is zero if H_t is at the mean of H , and grows as H_t moves away from the mean. If we denote the mean of H as $\mu^H = (\mu_1^H, \mu_1^H, \dots, \mu_{n_C}^H)$ and covariance matrix of H as C^H , then $d_M(H_t, H)$ can be defined as follows:

$$d_M(H_t, H) = \sqrt{(H_t - \mu^H)^T C^H (H_t - \mu^H)}. \quad (13)$$

Once the training has been completed and histogram matrix H has been defined, the model can be used to classify the network traffic during the recent time bin. As previously, for each flow in this time bin, we extract the necessary features. After that, the clustering algorithm is used and histogram vector h which represents the distribution of feature vectors is obtained. In order to classify network traffic in this time

bin, Mahalanobis distance $d_M(h, H)$ between h and training matrix H is calculated. Histogram vector h is classified as an anomaly if

$$d_M(h, H) > \frac{|\Omega| + 1}{|\Omega|} \max_{t \in \Omega} d_M(H_t, H), \quad (14)$$

where $|\Omega|$ is the number of time bins in the training set. If an anomaly is detected during the time bin considered, traffic transferred at this time bin is analyzed to detect intrusive flows.

C. Detection of intrusive flows

In order to find clusters that contain flows related to attacks during the recent time bin, we compare clusters $\{c_1, \dots, c_{n_C}\}$ obtained during this time bin and clusters obtained during the training phase. Let us denote the distance $d(C_j^t, C_k^t)$ between clusters C_j^t and C_k^t as the distance between centers of these clusters. The center of cluster C_k^t is calculated as mean values of y_i^t , $i \in C_k^t$. Flows of cluster c_j are classified as intrusive if one of the following conditions is met:

$$\begin{aligned} \min_{t \in \Omega, 1 \leq k \leq n_C} d(c_j, C_k^t) &> \frac{|\Omega| + 1}{|\Omega|} \max_{\tau, t \in \Omega, 1 \leq l, k \leq n_C} d(C_l^\tau, C_k^t), \\ |c_j| &> \frac{|\Omega| + 1}{|\Omega|} \max_{t \in \Omega} |C_{k^{t*}}^t|, \quad k^{t*} = \arg \min_{1 \leq k \leq n_C} d(c_j, C_k^t). \end{aligned} \quad (15)$$

where $|c_j|$ is the number of flows in cluster c_j .

IV. NUMERICAL SIMULATIONS

The experimental part of the research was done using an isolated Realistic Global Cyber Environment (RGCE) and Internet data generation software developed by JAMK University of Applied Sciences. RGCE is an Internet-like network comprising Internet Service Providers and main web services such as Domain Name Service, social media sites, news sites and search engines. RGCE is isolated from the Internet, allowing the user to practice with accurate GeoIP information, malware and attacks, without threat of contamination [21], [22]. Also, because of the isolation, the traffic within RGCE is automatically generated using special software which relies on the same functional principles as botnets and allows one to generate realistic traffic patterns of end users [21]. RGCE was used for National Cyber Security Exercises by Finnish Defense Forces [23]. RGCE has shown its capability for being a realistic Internet-like environment. It offers very sophisticated and realistic but isolated capability for training, exercise and research.

A web server with known vulnerabilities was installed in the RGCE environment. Outgoing and incoming traffic of this server was captured in order to test the proposed algorithm. The traffic was captured during four days. There was only legitimate traffic generated during the first day. However, the traffic generated during the next three days contained legitimate traffic as well as various forms of network intrusions, including scanning, brute-force attempts and targeted attacks.

All network packets gathered during these four days were combined into traffic flows. The length of time bin ΔT was selected to be equal to one minute, whereas parameter w is equal to 60. In this case each time bin contained enough

information to detect anomalies every second. The network traffic generated during the first day was used to build a model of normal user behavior. This model was then applied to the data corresponding to the next three days. To evaluate the performance of the proposed technique, the following characteristics were calculated in our tests: True Positive Rate (TPR), False Positive Rate (FPR) and detection accuracy.

At the first stage, time bins that contain anomalous network traffic are detected. For this reason, the clustering algorithm proposed in Section III is applied to each time bin and cluster histogram vectors are calculated. The feature vector distribution corresponding to the legitimate traffic differs from distributions obtained for various sorts of attack. As a result, the Mahalanobis distance for the histogram vector corresponding to a time bin which contains intrusive flows will be greater than distances corresponding to normal time bins.

Figure 1 shows the detection accuracy of the proposed algorithm calculated for different values of parameter n_C . As one can see, for any value of this parameter, the algorithm shows good results in terms of accuracy. However, in our simulations, when the value of parameter n_C is between 10 and 15, the detection accuracy of the method is maximal and exceeds 99%. This can be explained by the fact that in the dataset used the number of different tuples source-destination in the majority of time bins lies in this interval.

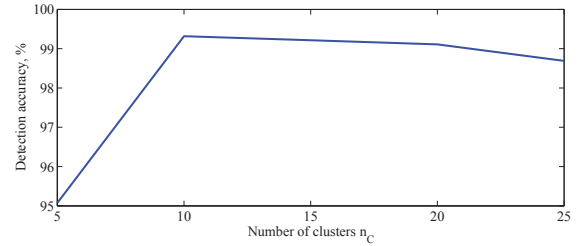


Fig. 1. Detection accuracy of the algorithm proposed for different values of parameter n_C .

We compared the performance of the proposed technique for the detection of "anomalous" time bins with other flow-based intrusion detection schemes: Flow Anomaly Detection System (FADS) [11], Exponentially Weighted Moving Average (EWMA) [12] and Chi-square Detection Mechanism [13]. Figure IV shows the dependence between false positive and true positive rates for different flow-based intrusion detection methods and different parameters. The detection accuracies

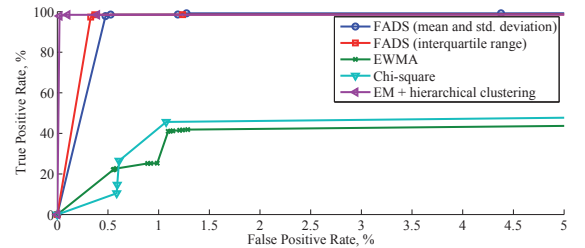


Fig. 2. The dependence between false positive and true positive rates for the first step of the algorithm compared to analogues.

of the methods are listed in Table II. Here we compared the method proposed when parameter n_C is equal to 10. As one

TABLE II. INTRUSION DETECTION ACCURACY OF THE PROPOSED ALGORITHM COMPARED TO ANALOGUES.

Method	TPR	FPR	Accuracy
FADS (mean and standard deviation)	97.84 %	0.48 %	99.10 %
FADS (interquartile range)	97.31 %	0.33 %	99.08 %
EWMA	22.17 %	0.55 %	79.94 %
Chi-square	14.65 %	0.59 %	76.99 %
EM + hierarchical clustering	98.39 %	0.39 %	99.32 %

can notice, the proposed algorithm based on soft clustering and analysis of histograms outperforms the analogues. However, detection accuracy of the proposed algorithm is not equal to 100%. It is caused by the fact that the method could not detect several password brute-force attempts when the number of these attempts during the time interval considered was low. On the other hand, in this case, it would be very difficult for the attacker to guess the password and get administrator privileges.

Finally, the algorithm for the detection of intrusive flows proposed in this study was compared with other clustering techniques: single linkage clustering, K-means, K-Nearest Neighbors (KNN) and Self Organizing Map (SOM). For this comparison, only time bins that contain anomalous traffic were used. The sum of Euclidean and Hamming distances was selected as a similarity metric. The comparison results are presented in Table III.

TABLE III. INTRUSION DETECTION ACCURACY OF THE PROPOSED ALGORITHM COMPARED TO OTHER CLUSTERING TECHNIQUES.

Method	TPR	FPR	Accuracy
KNN	99.56 %	11.19 %	98.48 %
Single linkage clustering	99.10 %	9.45 %	98.93 %
K-means	99.80 %	5.72 %	99.65 %
SOM	99.92 %	3.21 %	99.84 %
EM + hierarchical clustering	99.88 %	0 %	99.88 %

V. CONCLUSION

In this study, we apply an anomaly-based detection approach to find network traffic flows which are related to different types of attacks. We consider network traffic as a time series and extract various per-flow information for each time interval in a series. Based on this information, flows transferred during each time interval are divided into groups based on the EM algorithm and hierarchical clustering. After that, the distributions of resulting clusters are compared against each other to find the time bins that contain anomalous network traffic. Finally, traffic transferred during these time bins is analyzed and flows related to the attack are discovered. The proposed framework is tested on the data obtained with the help of a realistic cyber environment that enables one to construct real attack vectors. The simulations show that the proposed method results in a higher accuracy rate than other intrusion detection techniques. In the future, we are planning to focus on improving the accuracy of the method and developing algorithms for the detection of more sophisticated attacks.

REFERENCES

- [1] "Cert coordination center <http://www.cert.org/certcc.html>," Jul. 2008.
- [2] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, "An overview of ip flow-based intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 12, no. 3, pp. 343–356, 2010.
- [3] D. Gollmann, *Computer Security*. Wiley, 2nd edition, 2006.
- [4] R. Chitrakar and H. Chuanhe, "Anomaly based intrusion detection using hybrid learning approach of combining k-medoids clustering and naive bayes classification," in *Proceedings of the 8th Int. Conf. on Wireless Communications, Networking and Mobile Computing*, 2012, pp. 1–5.
- [5] M. Sato, H. Yamaki, and H. Takakura, "Unknown attacks detection using feature extraction from anomaly-based ids alerts," in *Proceedings of the 12th IEEE/IPSJ International Symposium on Applications and the Internet (SAINT)*, 2012, pp. 273–277.
- [6] A. Sperotto and A. Pras., "Flow-based intrusion detection," in *Proceedings of IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2011, pp. 958–963.
- [7] Z. Jadidi, V. Muthukkumarasamy, E. Sithirasanen, and M. Sheikhan, "Flow-based anomaly detection using neural network optimized with gsa algorithm," in *Proceedings of 33rd IEEE International Conference on Distributed Computing Systems Workshops*, 2013, pp. 76–81.
- [8] P. Winter, E. Hermann, and M. Zeilinger, "Inductive intrusion detection in flow-based network data using one-class support vector machines," in *Proceedings of 4th International Conference on New Technologies, Mobility and Security (NTMS)*, 2011, pp. 1–5.
- [9] M. Zolotukhin, T. Hämäläinen, and A. Juvenon, "Growing hierarchical self-organizing maps and statistical distribution models for online detection of web attacks," *Springer. Web Information Systems and Technologies. Lecture Notes in Business Information Processing*, vol. 140, pp. 281–295, 2013.
- [10] Winter, "Inductive intrusion detection in flow-based network data using one-class support vector machines, msc thesis," 2010.
- [11] M. Chapple, T. Wright, and R. Winding, "Flow anomaly detection in firewalled networks," in *Proceedings of Securecomm and Workshops*, 2006, pp. 1–6.
- [12] R. Hofstede, V. Bartos, A. Sperotto, and A. Pras, "Towards real-time intrusion detection for netflow and ipfix," in *Proceedings of the 9th International Conference on Network and Service Management (CNSM)*, 2013, pp. 227–234.
- [13] N. Muraleedharan, A. Parmar, and M. Kumar, "A flow based anomaly detection system using chi-square technique," in *Proceedings of the 2nd IEEE International Advance Computing Conference*, 2010, pp. 285–289.
- [14] M. de Assis, J. Rodrigues, and M. L. P. Junior, "A novel anomaly detection system based on seven-dimensional flow analysis," in *Proceedings of IEEE Global Communications Conference (GLOBECOM)*, 2013, pp. 735–740.
- [15] L. Xie, Y. Wang, L. Chen, and G. Yue, "An anomaly detection method based on fuzzy c-means clustering algorithm," in *Proceedings of the 2nd Int. Symp. on Networking and Network Security*, 2010, pp. 89–92.
- [16] S. Song and C. Zhixiong, "Adaptive network flow clustering," in *Proceedings of IEEE International Conference on Networking, Sensing and Control*, 2007, pp. 596–601.
- [17] A. M. Chandrasekhar and K. Raghuvver, "Intrusion detection technique by using k-means, fuzzy neural network and svm classifiers," in *Proceedings of IEEE International Conference on Networking, Sensing and Control*, 2013, pp. 1–7.
- [18] J. Bilmes, "A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models," Technical Report TR-97-021, ICSI, 1997.
- [19] C. Fraley and A. Raftery, "How many clusters? which clustering method? answers via model-based cluster analysis," *The Computer Journal*, vol. 41, pp. 578–588, 1998.
- [20] M. Rafsanjani, Z. Varzaneh, and N. Chukanlo, "A survey of hierarchical clustering algorithms," *The Journal of Mathematics and Computer Science*, vol. 5, no. 3, pp. 229–240, 2012.
- [21] "Jyvsectec-rgce - homepage," <http://www.jyvsectec.fi/en/rgce/>.
- [22] M. Zolotukhin, T. Hämäläinen, T. Kokkonen, and J. Siltanen, "Analysis of http requests for anomaly detection of web attacks," in *Proceedings of the 12th IEEE International Conference on Dependable, Autonomic and Secure Computing*, 2014, pp. 406–411.
- [23] "Ministry of defence press release 8.5.2013, kyberurvallisuusuharjoitus jyvaskylässä 13.-17.5.2013," http://www.defmin.fi/ajankohtaista/tiedotteet/2013/kyberurvallisuusuharjoitus_jyvaskylassa_13.-17.5.2013.5502.news, 2013.

PV

**MODEL FOR SHARING THE INFORMATION OF CYBER
SECURITY SITUATION AWARENESS BETWEEN
ORGANIZATIONS**

by

Tero Kokkonen, Jari Hautamäki, Jarmo Siltanen, Timo Hämäläinen 2016

23rd International Conference on Telecommunications, ICT 2016, Thessaloniki,
Greece, May 16-18, 2016, Proceedings

Reproduced with kind permission of IEEE,
Copyright © 2016, IEEE.

Model for Sharing the Information of Cyber Security Situation Awareness between Organizations

Tero Kokkonen, Jari Hautamäki, Jarmo Siltanen

Institute of Information Technology
JAMK University of Applied Sciences
Jyväskylä, Finland
email: {jari.hautamaki, tero.kokkonen,
jarmo.siltanen}@jamk.fi

Timo Hämäläinen

Department of Mathematical Information Technology
University of Jyväskylä
Jyväskylä, Finland
email: timo.t.hamalainen@jyu.fi

Abstract—Exchanging of Situation Awareness information is extremely important for organizations in order to survive as part of the cyber domain. The situation Awareness is required for decision making and for an early warning of upcoming threats. Situation Awareness and the security information in the cyber domain differ from the kinetic domain. Because of that, Situation Awareness has different requirements and use cases, for example when considering time or geographical distances. There is always a risk when sharing security information due to the classified nature of the information. It might contain information of weaknesses or vulnerabilities of the organization, and if used wrongly it jeopardizes the continuity of the business or mission. The model introduced in this paper for creating information sharing topologies enables sharing of classified security related information between multiple organizations with the lowest possible risks levels.

Keywords—Cyber Security; Situation Awareness; Common Operational Picture; Sharing Situation Awareness; STIX; TAXII

I. INTRODUCTION

Nowadays almost all systems and data are digitalized and connected using data networks forming so-called cyber domain. Thus it is very important to know the situation and risk level of your own assets both in civilian and military domains and also both in public and independent organizations or individuals. There exist plenty of different cyber threats or attacks that are able to affect business continuity or an ongoing mission. If the awareness of those threats is shared between organizations it could be used as an early warning and preparation for new threats. In this study, the model for sharing the cyber security situation information between organizations based on the risk level of sharing this kind of sensitive information is developed and demonstrated.

Firstly this paper describes the situation awareness and decision making definitions and concentrates on cyber security related situation awareness. After that the requirements for sharing the situation awareness information and the common standards are described. Finally, the developed model for sharing cyber security situation awareness information is described and analysed and also the future work is discussed.

II. SITUATION AWARENESS AND DECISION MAKING

Situation Awareness (SA) has an important role in decision making in dynamic environments. Civil, commercial and especially military aviation has the longest tradition of using SA in decision making; however, accurate SA is extremely important in many other domains also. In fact, with inappropriate SA even the trained decision maker will make the wrong decisions. [1]

As described in [2] and [3] there are plenty of different definitions for SA. Endsley states that “*Situation awareness is the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future*” [1].

SA depends on the person's competence and understanding of the specific task related events and phenomena [1], [3], [4] and the situation picture such as Common Operational Picture (COP) is a presentation of all the task related information available for the person [4]. In many cases the information originates from a large amount of data and multiple sensor feeds which are automatically fused using multi-sensor data fusion processed, for example according to JDL model by the US Joint Directors of Laboratories Data Fusion Sub-Group [3], [5], [6], [7].

The dominant model for command and control (C2) is the OODA-loop of John Boyd [8]. The OODA-loop model is originally generated for fighter combats; however, it has later been developed for general model of decision making for winning and losing [8]. OODA decision making loop is a process of Observation-Oriented-Decision-Action. SA is about having an understanding of what is happening and what will happen in the near future and it includes three processes: Perception (perceive, gather data) – Comprehension (understand, create mental model) – Projection (think ahead, update the model) [1], [3], [9].

SA has a temporal nature and SA is almost always related to time (past, current and future). It can be said that relevant SA improves efficiency or effectiveness of decision-making.

III. SITUATION AWARENESS IN CYBER SECURITY

Finland's Cyber Security Strategy defines cyber domain as interdependent, multipurpose electronic data processing environment and according to EU's Cyber Security Strategy Cyber Security includes safeguards and actions used to protect both the civilian and military cyber domains. [10], [11]

It is extremely important to achieve appropriate SA about own cyber domain and perceive possible security issues, indicators of compromises (IOC) or risk statuses. With appropriate cyber SA the decision makers are able to make the right decisions for achieving cyber resilience and ensure continuity of their operation or business.

The strategic guidelines of Finland's Cyber Security Strategy state one goal to improve the situation awareness of different actors by sharing information of vulnerabilities, disturbances and their effects. There is also threat assessment and prediction of cyber domain included requiring an analysis of the different statuses for example political, military, social or economic [10].

US-CERT provides document Cybersecurity Questions for CEOs to instruct leadership collaboration about cyber security risk management with essential cyber security risk management concepts where one of those concepts is "Maintain situational awareness of cyber threats" including timely detection of cyber incidents, awareness of organization specific threats and vulnerabilities and relevant impacts for business [12].

Military commanders are used to have COP for decision making in the kinetic environment; however, even if the cyber domain is realized as an operational domain there is no versatile Cyber Common Operating picture (CCOP) [13]. The cyber domain and physical world are different, for example, time and location diverge between physical world and cyber space [3], [13], [14]. In physical world there are plenty of specific sensors and signal processing but in the cyber space sensors and processing techniques are different. Sensors in cyber domain are for example intrusion detection systems (IDS), log file analysis or antivirus/malware protection systems and SA is achieved at the lower (technical) levels; however, higher level SA analysis is non-automated manpowered work [14]. For example, for making decisions for cognitive networks there must be awareness of previous states, current configuration and resources available [15].

IV. SHARING OF SITUATIONAL AWARENESS

As stated in [12] identifying and responding to incidents is more efficient if organization is sharing threat information with partners. In military there are standards for sharing information of kinetic environment between different stakeholders for example Tactical Data Links. One example of Tactical Data Links is The Joint Tactical Information Distribution System (JTIDS) Link16 the high-capacity, ECM-resistant, secure data link for voice and data [16]. In C2 systems the shared information can be owned by different stakeholders with different geographical locations, different missions or even

different security clearances [5]. Good real life example of that is the usage of Air Situation Data Exchange (ASDE) during the Euro Championship 2012 where NATO Programming Centre (NPC) supported Poland in the air situation data exchange with national organizations [17].

In the foreword of [18] it is stated that "*information sharing is one of the most heard suggested solutions for increasing cyber resilience*". In many cases there are noticed requirements for collaboration and changing of the cyber security information between organizations for achieving better cyber resiliency, maintaining business continuity or cyber-incident response capability [18], [19], [20], [21].

The information exchange should be done between devices in a machine-to-machine (M2M) level according to four phases: information collection, transmission of filtered information, analysis of information and operations executed based on the analysis [22].

The building blocks for cyber security information sharing are introduced in [18]. NATO Communications and Information Agency has developed Cyber Security Data Exchange and Collaboration Infrastructure (CDXI) concept and the high level requirements of the CDXI concept have been identified in [19]. Also the Center for Strategic and International Studies (CSIS) outline recommendations for future policy and legislation about sharing the cyber threat information [20].

Even though the security information sharing architectures have been implemented it is also important to train, test and evaluate processes. Cyber security exercises contain complex information sharing requirements between exercise teams [23]. JAMK University of Applied Sciences has developed Cyber Range called Realistic Global Cyber Environment (RGCE) for utilizing cyber security research and exercises in realistic Internet-like environment [24], [25], [26]. RGCE environment and the Cyber Security exercises can be used for testing the models in realistic environments.

Mitre Corporation has developed standards called Structured Threat Information eXpression (STIX™) and Trusted Automated eXchange of Indicator Information (TAXII™) for describing and collaborating cyber threat information in a standardized and structured manner [21], [27], [28], [29]. STIX ontology based situation assessment framework is presented in [27] and as a result it has been mentioned that the mechanism performed well; however, it has some limitations that will be improved in the future. Both STIX and TAXII have been transitioned to OASIS Advanced open standards for the information society [30].

V. STIX AND TAXII

STIX allows structured expression for threat information and supports following cyber threat management use cases (Fig. 1): analysing cyber threats, specifying indicator patterns, managing response activities and sharing the information of cyber threat. [28]

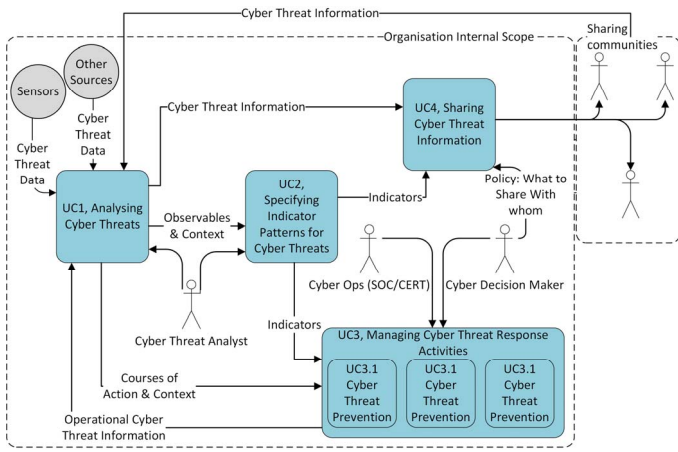


Fig. 1. STIX use cases, quoted from [28]

The STIX architecture consists of eight constructs and all of those constructs have been utilized for the XML schema. The constructs are: Observable, Indicator, Incident, TTP (Tactics, Techniques, and Procedures), ExploitTarget, CourseOfAction, Campaign and ThreatActor. [28]

Batch of services and message exchanges enabling sharing of cyber threat information between organizations are defined in TAXII and are intended to be used by information producers, consumers and developers. TAXII supports three different threat information sharing architectures (Fig. 2) hub-and-spoke, peer-to-peer, and source-subscriber without being bound to any specific format or protocol. [29]

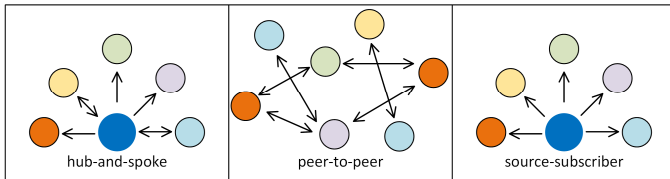


Fig. 2. TAXII threat sharing models, quoted from [29]

Security information sharing is one of the most critical issues for organizations to increase the defence capability against security threats [29]. As seen on Fig. 2, TAXII allows construction of complex information sharing topologies between different actors, whereas cyber threat information can be shared efficiently and formalized using STIX data structures.

VI. DEVELOPED MODEL AND DEMONSTRATION

STIX and TAXII were decided to be used in this study because those are reasonable new standards and emerging to be widely used. We developed a minimum risk model for sharing cyber security threat information between organizations for achieving better SA in cyber domain.

Information sharing requires secure, high capacity infrastructure that supports large set of use cases, policies and practices to common use [29].

Organizations are assumed to share processed security information which could be classified. There is always a risk for sharing this kind of information that the shared information is abused. Information sharing requires trusted relationship between organizations or parties, and a trusted relationship can be accessed several ways. This research implements a model for STIX structured information sharing using TAXII service topology definition based on the risk level of sharing information between organizations. Using this model direct connections with higher risk levels can be reduced and safer information sharing communities produced. In this research it is assumed that all organizations are responsible for defining the risk level for the sharing based on shared information and connected organization. The risk level values are required to have the same scale. Organization is not required to share information to all the other organizations; only to the trusted ones with the risk level estimation.

There are similar challenges if the purpose is to find the shortest paths between nodes in a graph. Zhao and White describes new framework for security information sharing based on defining threat alert levels of community [31]. Hernandez-Ardieta, Tapiador and Suarez-Tangil proposes a model of information sharing communities as directed graphs where nodes representing community members and edges modelling sharing relationships among them [32].

Edsger Dijkstra has invented a way to find the shortest path within a graph. In this study the shortest path tree method called Dijkstra algorithm is used for finding the lowest risks for information sharing between organizations [33].

Let's assume a real life scenario of security information sharing community according to Fig. 3. In this scenario there are three different countries where the national CERT acts as the highest security information sharing organizations. The next level security information sharing organizations are Internet Service Providers (ISP) and the lowest level of information sharing organizations are enterprises. In this scenario two ISPs operate in the country 1 and one ISP in the country 2. Every node represents an organization with individual TAXII service for information sharing in STIX data structures. Every link between nodes has a risk level value based on the risk estimation of shared sensitive information and the organization attached and risk level values between [1 (minimum risk), 20 (high risk)]. If the risk level is unacceptable there is no link between nodes. For example, sharing information from an enterprise to another in a different country comes with a higher risk level defined. There are also cases that enterprises do not share information to enterprises in another country but shared information shall flow through ISP or CERT level to enterprises in another country.

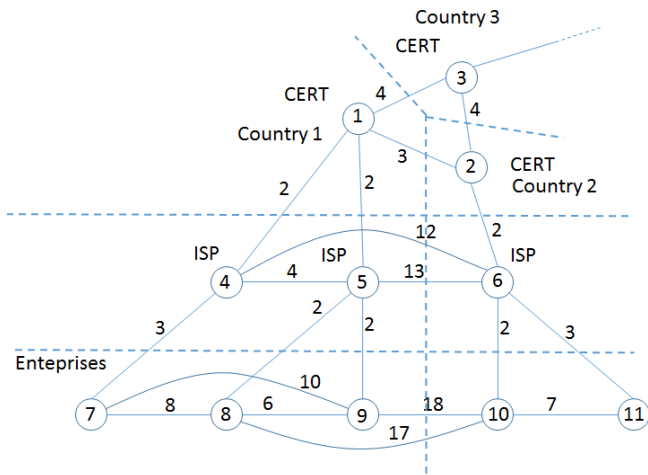


Fig. 3. Security information sharing community with risk values

The topology of information sharing community with the lowest risk level has been calculated with Dijkstra's algorithm [33]. According to [34] Dijkstra's algorithm presented as pseudo code for network G with N nodes and distances (risk values) D_{ij} for edges $(i,j \in N)$. Where S is a start node and P is labelled set of nodes. The shortest path from S to other nodes j can be found as follows:

- Start $P = \{S\}$, $D_S=0$ and $D_j=d_{sj}$ for $j \in N$ ($j \neq S$)
- Phase 1, find the closest node $i \notin P$ where $D_i = \min D_j$ and $j \notin P$. Set $P = P \cup \{i\}$, if P contains all nodes stop, else go to Phase 2
- Phase 2, update labels for $j \notin P$, set $D_j = \min[D_j, D_i + d_{ij}]$, go to Phase 1

Now the information sharing topology for the real life scenario can be implemented as in Fig. 4.

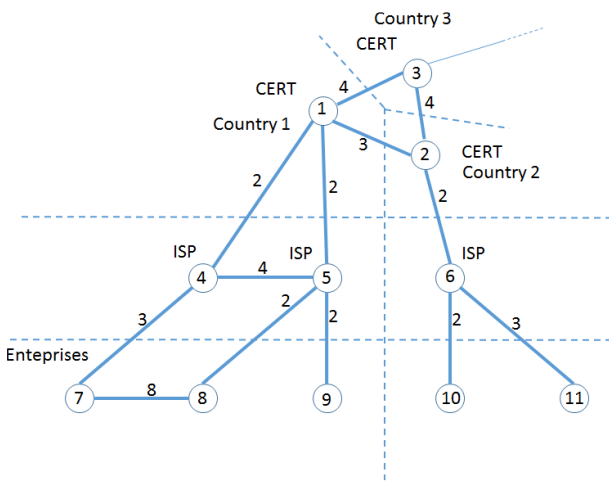


Fig. 4. The calculated topology with the lowest risk level

Using Fig. 4 it can be seen that even if there are no direct connections between all the enterprises and there is a high risk level between enterprises in different countries, the security information flow goes from every node to another with

minimum risk implementation. Based on this topology security information sharing community is able to configure TAXII services with minimum risk.

VII. CONCLUSION

Using our model for creating information sharing topology for STIX and TAXII based infrastructure it is possible to share classified security related information between multiple organizations with minimum risks. The model could be used inside one country or internationally between different organization levels. Both STIX and TAXII have libraries available with different programming languages e.g. Python. It is easy to develop different information sharing scenarios and test this topology model in different use cases or scenarios. The most difficult part of the information sharing in real life will be gaining the trust between different organizations for sharing sensitive security information with classified nature. As a next step the model shall be tested in a real environment and scenario. As a future work the model is planned to be implemented and tested as a part of cyber security exercise in RGCE. There is also the requirement to formulate the definition for risk level calculation and improve the model to have one-way links instead of bidirectional links between nodes. Besides of using Dijkstra's algorithm, there could also be comparison of different algorithms for example Genetic Algorithm (GA) or A* for creating of complex information sharing topologies. With the current model there is only risk level as a parameter for weight of the link, as a future work there should be research on using more parameters, for example, information validity or latency.

ACKNOWLEDGMENT

This work was partially funded by the Regional Council of Central Finland/Council of Tampere Region and European Regional Development Fund/Leverage from the EU 2014-2020 as part of the JYVSECTEC Center project of JAMK University of Applied Sciences Institute of Information Technology.

REFERENCES

- [1] M. Endsley, "Toward a Theory of Situation Awareness in Dynamic Systems", *Human Factors: The Journal of the Human Factors and Ergonomics Society*, March 1995, vol. 37, no. 1 32-64
- [2] Royal Aeronautical Society, Human Factors group, "Summary of the various definitions of Situation Awareness", <http://www.raes-hfg.com/crm/reports/sa-defns.pdf>
- [3] G. Tadda and J. Salerno, "Overview of Cyber Situation Awareness", *Cyber Situational Awareness, Issues and Research*, *Advances in Information Security*, Volume 46, Springer, 2010, pp 15-35
- [4] R. Kuusisto, T. Kuusisto and L. Armistead, "Common Operational Picture, Situation Awareness and Information Operations", *Proceedings of the 4th European Conference on Information Warfare and Security*. Glamorgan, UK, 2005, pp. 175-185
- [5] J. Preden, L. Motus, R. Pahtma, and M. Meriste, "Data Exchange for Shared Situation Awareness," *IEEE International Multi-Disciplinary Conference in Cognitive Methods in Situation Awareness and Decision Support (CogSIMA) 2012*, pp.198-201, 6-8 March 2012
- [6] E. Azimirad and J. Haddadnia, "The Comprehensive Review On JDL Model In Data Fusion Networks: Techniques and Methods", (*IJCSIS*) *International Journal of Computer Science and Information Security*, Vol. 13, No. 1, January 2015

- [7] O. Baud, P. Gomord, N. Honoré, L. Ostorero, O. Taupin and P. Tubery, "Multi Sensor Data Fusion Architectures for Air Traffic Control Applications", *Sensor and Data Fusion* pp. 103-122, February 2009
- [8] B. Brehmer, "The Dynamic OODA Loop: Amalgamating Boyd's OODA Loop and the Cybernetic Approach to Command and Control", *Proceedings of 10th International Command and Control Research and Technology Symposium (ICCRTS) The Future of C2*, 2005
- [9] Airbus Flight Operations Briefing Notes, Human Performance, Enhancing Situational Awareness. http://www.airbus.com/fileadmin/media_gallery/files/safety_library_items/AirbusSafetyLib_-FLT_OPS-HUM_PER-SEQ06.pdf
- [10] Secretariat of the Security Committee, "Finland's Cyber Security Strategy", Government Resolution 24.1.2013
- [11] Joint communication to the European Parliament, the Council, the European Economic and Social Committee and the Committee of the Regions, "Cybersecurity Strategy of the European Union: An Open, Safe and Secure Cyberspace", JOIN(2013) 1 final, Brussels, 7.2.2013
- [12] U.S Department of Homeland Security, United States Computer Emergency Readiness Team US-CERT, "Cybersecurity Questions for CEOs", <https://www.us-cert.gov/sites/default/files/publications/DHS-Cybersecurity-Questions-for-CEOs.pdf>
- [13] G. Conti, J. Nelson, D. Raymond. "Towards a Cyber Common Operating Picture", 5th International Conference on Cyber Conflict (CyCon 2013), NATO CCDCOE Publications, Tallinn, 2013
- [14] P. Barford, M. Dacier, T. Dietterich, M. Fredrikson, J. Giffin, S. Jajodia, S. Jha, J. Li, P. Liu, P. Ning, X. Ou, D. Song, L. Strater, V. Swarup, G. Tadda, C. Wang and J. Yen "Cyber SA: Situational Awareness for Cyber Defense", *Cyber Situational Awareness, Issues and Research, Advances in Information Security, Volume 46*, Springer, 2010, pp 3-13
- [15] A. Kärkkäinen, "Developing cyber security architecture for military networks using cognitive networking", Aalto University publication series, Doctoral Dissertations 170/2015
- [16] UK Ministry of Defence, "Joint Air Defence", Joint Warfare Publication JWP 3-63, Second edition, 31 December 2012
- [17] NATO Programming Centre (NPC), Air Situation Data Exchange (ASDE), [https://npc.ncia.nato.int/Pages/Air-Situation-Data-Exchange-\(ASDE\).aspx](https://npc.ncia.nato.int/Pages/Air-Situation-Data-Exchange-(ASDE).aspx), site referred 8 January 2016
- [18] E. Luijck and A. Kernkamp, "Sharing Cyber Security Information Good Practice Stemming from the Dutch Public-Private-Participation Approach", *Global Conference on Cyber Space 2015*
- [19] L. Dandurand and O. Serrano Serrano, "Towards Improved Cyber Security Information Sharing, Requirements for a Cyber Security Data Exchange and Collaboration Infrastructure (CDXI)", 5th International Conference on Cyber Conflict (CyCon 2013), NATO CCDCOE Publications, Tallinn, 2013
- [20] D. Zheng and J. Lewis, "Cyber Threat Information Sharing, Recommendations for Congress and the Administration", Center for Strategic and International Studies (CSIS), A Report of the CSIS Strategic Technologies Program, March 2015
- [21] K. Mephram, G. Ghinea, P. Louvieris and N. Clewley, "Dynamic Cyber-Incident Response", 6th International Conference on Cyber Conflict (CyCon 2014), NATO CCDCOE Publications, Tallinn, 2014
- [22] J. Linné, K. Majewski, M. Salminen and R. Samani, "Cyber Security for Decision Makers", Docendo, 2015
- [23] N. Wilhelmson and T. Svensson, "Handbook for planning, running and evaluating information technology and cyber security exercises", Swedish National Defence College, Center for Asymmetric Threats Studies (CATS), 2014
- [24] JYVSECTEC Jyväskylä Security Technology, RGCE Realistic Global Cyber Environment, <http://jyvsectec.fi/en/rgce/>, site referred January 8 2016
- [25] M. Zolotukhin, T. Hamalainen, T. Kokkonen and J. Siltanen: "Online Detection of Anomalous Network Flows with Soft Clustering", The Seventh International Conference on New Technologies, Mobility and Security, NTMS, 2015 Paris, France
- [26] M. Zolotukhin; T. Hamalainen; T. Kokkonen; A. Niemela; J. Siltanen, "Data Mining Approach for Detection of DDoS Attacks Utilizing SSL/TLS Protocol", The 15th International Conference on Next Generation Wired/Wireless Advanced Networks and Systems, NEW2AN 2015, August 26-28, 2015 St. Petersburg, Russia
- [27] L. Shan and M. Kokar, "A Situation Assessment Framework for Cyber Security Information Relevance Reasoning," 18th International Conference on Information Fusion (Fusion 2015), pp.1459-1466, 6-9 July 2015
- [28] S. Barnum, "Standardizing Cyber Threat Intelligence Information with the Structured Threat Information eXpression (STIX™)", Version 1.1, Revision 1, February 20 2014, <http://stixproject.github.io/getting-started/whitepaper/>, site referred December 20 2015
- [29] J. Connolly, M Davidson and C. Schmidt, "The Trusted Automated eXchange of Indicator Information (TAXII™)", May 2 2014, <http://taxiiproject.github.io/getting-started/whitepaper/>, site referred December 20 2015
- [30] OASIS Cyber Threat Intelligence (CTI) TC, https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=cti, site referred January 7 2016
- [31] W. Zhao and G. White, "A Collaborative Information Sharing Framework for Community Cyber Security", IEEE Conference on Technologies for Homeland Security (HST), pp. 457-462, 2012
- [32] J. Hernandez-Ardieta, J. Tapiador and G. Suarez-Tangil, "Information Sharing Models for Cooperative Cyber Defence", 5th International Conference on Cyber Conflict (CyCon 2013), NATO CCDCOE Publications, Tallinn, 2013
- [33] E. Dijkstra, "A Note on Two Problems in Connexion with Graphs", *Numerische Mathematik* 1: 269–271, 1959
- [34] X. Li, G Li, S. Zhang, "Routing Space Internet Based on Dijkstra's Algorithm," *First Asian Himalayas International Conference on Internet (AH-ICI 2009.)*, pp.1-4, 2009

PVI

**INCREASING WEB SERVICE AVAILABILITY BY DETECTING
APPLICATION-LAYER DDOS ATTACKS IN ENCRYPTED
TRAFFIC**

by

Mikhail Zolotukhin, Timo Hämäläinen, Tero Kokkonen, Jarmo Siltanen 2016

23rd International Conference on Telecommunications, ICT 2016, Thessaloniki,
Greece, May 16-18, 2016, Proceedings

Reproduced with kind permission of IEEE,
Copyright © 2016, IEEE.

Increasing Web Service Availability by Detecting Application-Layer DDoS Attacks in Encrypted Traffic

Mikhail Zolotukhin, Timo Hämäläinen

Department of Mathematical Information Technology
University of Jyväskylä, Jyväskylä, Finland
email: {mikhail.m.zolotukhin, timo.t.hamalainen}@jyu.fi

Tero Kokkonen, Jarmo Siltanen

Institute of Information Technology
JAMK University of Applied Sciences, Jyväskylä, Finland
email: {tero.kokkonen, jarmo.siltanen}@jamk.fi

Abstract—Nowadays, zero-day Denial-of-Service (DoS) attacks become frighteningly common in high-speed networks due to constantly increasing number of vulnerabilities. Moreover, these attacks become more sophisticated, and, therefore, they are hard to detect before they damage several networks and hosts. Due to these reasons, real-time monitoring, processing and network anomaly detection must be among key features of a modern DoS prevention system. In this paper, we present a method which allows us to timely detect various denial-of-service attacks against a computer or a network system. We focus on detection of application-layer DoS attacks that utilize encrypted protocols by applying an anomaly-detection-based approach to statistics extracted from network packets. Since network traffic decryption can violate ethical norms and regulations on privacy, the detection scheme proposed analyzes network traffic without its decryption. The scheme includes the analysis of conversations between a web server and its clients, the construction of a model of normal user behavior by dividing these conversations into clusters and the examination of distribution of these conversations among the resulting clusters with the help of the stacked auto-encoder which belongs to a class of deep learning algorithms. Conversations of clients that deviate from those normal patterns are classified as anomalous. The proposed technique is tested on the data obtained with the help of a realistic cyber environment.

I. INTRODUCTION

Denial-of-Service (DoS) attacks are one of the most serious threats to the Internet nowadays. According to the report published by Kaspersky Lab [1], in the first quarter of 2015, there were 23 095 large DDoS attacks targeting 12 281 unique web resources in 76 countries. DoS attacks aim to disable a computer or network system using lots of messages which need responses consuming the bandwidth or other resources of the system. Since it is difficult for an attacker to overload the targets resources from a single computer, modern DoS attacks are launched via a large number of distributed attacking hosts in the Internet. Such distributed DoS (DDoS) attacks can force the victim to significantly downgrade its service performance or even stop delivering any service [2]. Designed to elude

This work was partially funded by the Regional Council of Central Finland/Council of Tampere Region and European Regional Development Fund/Leverage from the EU 2014–2020 as part of the JYVSECTEC Center project of JAMK University of Applied Sciences Institute of Information Technology.

detection by today’s most popular cyber security tools, these attacks can quickly incapacitate a targeted business, costing victims millions of dollars in lost revenue and productivity.

The purpose of traditional DDoS attacks carried out at the network layer is to consume the network bandwidth and deny service to legitimate users of the victim systems. This type of attack has been well studied recently and different schemes have been proposed to protect the network and equipment from such bandwidth attacks [3], [4], [5]. Nowadays, application-layer DDoS attacks are becoming more and more widespread [2], [7], [10]. Such attacks may focus on exhausting the server resources such as Sockets, CPU, memory, disk bandwidth, and I/O bandwidth. Unlike network-layer DoS attacks, application-layer attacks do not necessarily rely on inadequacies in the underlying protocols or operating systems. They can be performed by using legitimate requests from legitimately connected network machines. This makes application-layer DDoS attacks undetectable for signature-based intrusion detection systems (IDSs).

Anomaly-based approach is a promising solution for detecting and preventing application-layer DDoS attacks. Such approach learns the features of event patterns which form normal user behavior, and, by observing patterns that deviate from the established norms detects when an intrusion has occurred. Thus, systems which use the anomaly detection approach are modeled according to normal user behavior and, therefore, are able to detect behavior-based denial-of-service attacks.

Attacks that involve the use of HTTP protocol is the most prevalent application-layer denial-of-service attack type nowadays. Depending on the level of their sophistication HTTP-based DDoS attacks can be grouped into the following three major categories: trivial, intermediate and advanced [10]. During a trivial DDoS attack, each bot participating in the attack sends one or a limited number of unrelated HTTP attacks towards the target site. This type of attacks includes such well-known attacks as Slowread and Slowpost [8]. In the case of an intermediate attack, bots generate random sequences of browser-like requests of web-pages with all of their embedded content making the attack traffic indistinguishable from the

regular human traffic. It is predicted that advanced DoS attacks will rise in popularity in the future. These attacks will consist of sequences of HTTP requests which are carefully chosen so as to better mimic the browsing behavior of regular human users.

The problem of anomaly-based detection of application-layer DoS and DDoS attacks that utilize HTTP protocol is of great interest nowadays. For example, paper [6] analyzes application-layer DDoS attacks against a HTTP server with the help of hierarchical clustering of user sessions. Study [7] shows a novel detection technique against HTTP-GET attacks, based on Bayes factor analysis and using entropy-minimization for clustering. In study [8], detection of slow HTTP attacks is carried out by analyzing specific spectral features of network traffic over small time horizons. In [9], authors detect application-layer DDoS attacks by constructing a random walk graph based on sequences of web pages requested by each user. Finally, study [10] proposes the next-generation system for application-layer DDoS defense by modeling network traffic to dynamic web-domains as a data stream with concept drift.

Most of the current studies devoted to HTTP-based DDoS attack detection concentrate on the analysis of information extracted from network packets' payload which includes web resource requested, HTTP request method, session ID and other parameters. However, it remains unclear how to detect DDoS attacks utilizing protocols that encrypt the data of network connections in the application layer. In this case, it is impossible to detect attackers' activity based on the analysis of packets' payload without decrypting it [12]. However, in most of the cases, network traffic decryption violates complex sets of laws and regulations on privacy, along with a high risk of conflict with web service users. For this reason, the detection of DDoS attacks is supposed to be carried out with the help of statistics that can be extracted mostly from network packet headers.

In this research, we concentrate on timely detection of trivial and intermediate DoS and DDoS attacks against a computer or network system. We design an algorithm that allows us to detect application-layer DDoS threats that utilize encrypted protocols. For this purpose, we focus on anomaly-detection-based approach applied to statistics mostly extracted from network packet headers. The algorithm includes analysis of conversations between a web server and its clients, dividing them into groups by applying several clustering techniques and defining the model of normal user behavior. Conversations of clients that deviate from those normal patterns are classified as anomalous. In addition, groups of conversations initiated by one client to the same destination socket during some short time interval are classified based on the reconstruction error generated by the stacked auto-encoder. The proposed method is tested on the data obtained with the help of a realistic cyber environment that enables one to construct real attack vectors.

The rest of the paper is organized as follows. Extraction of feature vectors from network flows is considered in Section II. Section III introduces a scheme that uses clustering techniques

to build a model of normal user behavior and sheds light on detection of trivial and more complicated intermediate DoS attacks. In Section IV, we present some numerical results to evaluate the algorithm proposed. Finally, Section V draws the conclusions and outlines future work.

II. FEATURE EXTRACTION AND STANDARDIZATION

Let us consider a web server that provides several services working in two application layer protocols: HTTP and HTTPS. Outgoing and incoming traffic of this server is captured during some time period $[T_s, T_e]$. In this study, it is assumed that the traffic captured during this time period is free of attack. We aim to investigate captured traffic and discover behavior patterns of normal users. Once normal behavior patterns have been discovered, these patterns can be used to analyze network traffic and detect DoS and DDoS attacks against the web server in online mode. For this purpose, the analyzed time period $[T_s, T_e]$ is divided into equal non-overlapping time intervals of length ΔT . The length of each time interval ΔT should be picked in such a way that allows one to detect attacks timely.

We concentrate on the intrusion detection based on the analysis of network traffic flows. A flow is a group of IP packets with some common properties passing a monitoring point in a specified time interval. In this study, we assume that these common properties include the IP address and port of the source and IP address and port of the destination. The time interval in this case is considered to be $[T_s, T_e]$. In other words, when analyzing a traffic flow extracted in time interval $[T_s + i\Delta T, T_s + (i+1)\Delta T]$, we take into account all packets of this flow transferred during previous time intervals: $[T_s + (i-1)\Delta T, T_s + (i)\Delta T]$, $[T_s + (i-2)\Delta T, T_s + (i-1)\Delta T]$, etc. Resulting flow measurements provide us an aggregated view of traffic information and drastically reduce the amount of data to be analyzed. After that, two flows such as the source socket of one of these flows is equal to the destination socket of another flow and vice versa are found and combined together. This combination is considered as one conversation between a client and the server.

A conversation can be characterized by following four parameters: source IP address, source port, destination IP address and destination port. For each such conversation at each time interval, we extract the following information:

- 1) duration of the conversation
- 2) number of packets sent in 1 second
- 3) number of bytes sent in 1 second
- 4) maximal, minimal and average packet size
- 5) maximal, minimal and average size of TCP window
- 6) maximal, minimal and average time to live (TTL)
- 7) percentage of packets with different TCP flags: FIN, SYN, RST, PSH, ACK and URG
- 8) percentage of encrypted packets with different properties: handshake, alert, etc

Features of types 2–8 are extracted separately for packets sent from the client to the server and from the server to the client. It is worth to mention that here we do not take into account time intervals between subsequent packets of the same flow. Despite

the fact, that increasing of these time intervals is a good sign of a DDoS attack, taking them into consideration leads to the significant increasing of the number of false alarms. It is caused by the fact, that when the server is under attack it cannot reply to legitimate clients timely as well, and, therefore, legitimate clients look like attackers from this point of view.

Values of the extracted feature vectors can have different scales. In order to standardize the feature vectors, max-min normalization is used. Since all network traffic captured in the time interval considered is assumed to be legitimate, all the resulting normalized feature vectors can be used to reveal normal user behavior patterns and detect behavioral anomalies.

III. ALGORITHM

The detection algorithm proposed in this study includes three steps. First, a model of normal user behavior is built by dividing extracted feature vectors into clusters. Recent conversations that deviate from extracted normal patterns are found in order to discover trivial DoS attacks. Finally, the examination of distributions of the conversations among the resulting clusters allows us to classify more advanced attacks.

A. Conversation Clustering

Once all relevant features have been extracted and normalized, the resulting feature vectors can be used to determine the model of normal user behavior. For this purpose, we divide these vectors into several groups by applying a clustering algorithm. Each such group is supposed to consist of objects that are in some way similar between themselves and dissimilar to objects of other groups. Clustering allows us to discover hidden patterns presented in the dataset to represent a data structure in a unsupervised way. There are many different clustering algorithms which can be categorized based on the notation of a cluster. The most popular categories include hierarchical clustering algorithms [13], centroid-based clustering algorithms [16] and density-based clustering algorithms [17].

Each cluster calculated represents a specific class of traffic in the network system under inspection. For example, one such class can include conversations between a web server and its clients which use one of web resources of the server. Since the traffic is encrypted it is impossible to define what web resource these clients request. However, since it is assumed that traffic being clustered is legitimate, we can state that each cluster describes a normal user behavior pattern.

B. Detection of Trivial DoS Attacks

To detect trivial DoS attacks we extract necessary features from a new conversation and classify the resulting feature vector according to the clusters found. If this vector does not belong to any of the clusters, the corresponding conversation is labeled as intrusive and it is supposed to be blocked by the server.

For hierarchical clustering algorithms, to define whether a new vector belongs to a cluster or not, we calculate the minimal distance between this vector and vectors-members of the cluster. If this minimal distance is greater than a predefined

threshold, this vector does not belong to the cluster. This threshold T_i for the i -th cluster is calculated based on vectors of the training set which belong to this cluster: $T_i = \mu_i^n + \gamma\sigma_i^n$, where μ_i^n is the average distance between two neighboring vectors of this cluster and σ_i^n is the standard deviation of these distance values.

For centroid-based clustering methods, to define whether a new vector belongs to a cluster or not, we calculate the distance between this vector and the cluster center. If the distance between the new vector and the cluster center is greater than a predefined threshold, this vector does not belong to the cluster. This threshold T_i for the i -th cluster is calculated based on vectors of the training set which belong to this cluster: $T_i = \mu_i^c + \alpha\sigma_i^c$, where μ_i^c is the average distance between the center and vectors of this cluster and σ_i^c is the standard deviation of these distance values.

For density-based clustering algorithms, one option to define whether a new vector belongs to a cluster or not is based on the density-reachability of this vector from a point of the cluster. Vector z is density-reachable from y with respect to ε and N_{min} , if there is a chain of points y_1, y_2, \dots, y_m , where $y_1 = y$ and $y_m = z$, such that $\forall i \in \{1, 2, \dots, m-1\}$, $d^E(y_i, y_{i+1}) \leq \varepsilon$ and $N_\varepsilon(y_i) \geq N_{min}$, where $d^E(y_i, y_{i+1})$ is distance between y_i and y_{i+1} . Parameters ε and N_{min} can be selected based on vectors of the training set as follows: ε is the average distance between vectors of the cluster and N_{min} is the minimal number of points such that each point of the cluster is density-reachable from another point of the cluster.

C. Detection of Intermediate DoS Attacks

The technique proposed in Section III-B can help to detect trivial DoS attacks when conversations between an attacker and the web server deviate from normal user behavior patterns. However, if the attacker is able to mimic properly the browsing behavior of a regular human user, conversations related to this attack might belong to one of the clusters of the normal behavior model and, therefore, remain undetected. In this case, the way how feature vectors are distributed across clusters should be taken into consideration. This vector distribution during an attack can differ markedly from the vector distribution corresponding to legitimate traffic. Thus, we can define whether a computer or network system is under attack during the current time interval, and, moreover, find clients responsible for initiating conversations related to the attack.

For this purpose, we group all conversations which have the same source IP address, destination IP address and destination port together and analyze each such group separately. Such approach is in-line with studies [6], [7], [9] mentioned in Section I. Those studies analyze sequences of conversations (requests) belonging to one HTTP session. In our case, since the session ID cannot be extracted from encrypted payload, we focus on conversations initiated by one client to the destination socket during some short time interval. We can interpret a group of such conversations as a rough approximation of the user session.

For each such group, the percentage of feature vectors contained in each cluster is counted. We introduce histogram vector $h^{it} = (h_1^{it}, h_2^{it}, \dots, h_{n_C}^{it})$ which consists of these percentage values. Here n_C is the number of clusters revealed and h_j^{it} is the number of feature vectors of the i -th group (source IP address, destination IP address, destination port) in the training set belonging to the j -th cluster at the t -th time interval divided by the total number vectors of the i -th group at this time interval. Once histogram vector h^{it} has been calculated for each conversation group i and each time interval t , matrices H^i are built using these vectors. The t -th row $H^{it} = (H_1^{it}, \dots, H_{n_C}^{it})$ of matrix H^{it} corresponds to the histogram vector $(h_1^{it}, h_2^{it}, \dots, h_{n_C}^{it})$ calculated for the i -th conversation group at t -th time interval.

In this study, we propose a method for analyzing the resulting histogram vectors based on stacked auto-encoder [18] belonging to a class of deep learning algorithms [19]. Deep learning involves models which try to hierarchically learn deep features of input data with neural networks which are deeper than three layers. First, the network is initialized via unsupervised training and then tuned in a supervised manner. This approach allows us to learn high-level features from low-level ones and formulate proper features for pattern classification. As a result, the model allows to generate more abstract features that are invariant to local changes of the input.

A standard Auto-Encoder (AE) consists of one visible input layer, one hidden layer and one reconstruction layer. The reconstruction layer has the same size as the input layer. The objective of the AE to learn how to reproduce input vectors as outputs. This process includes two steps: encoding and decoding. The i -th value of vector h encoded as h^e can be calculated as $h_i^e = f(\sum_{j=1}^{n^{input}} w_{ij}^e h_j + b_i^e)$, where f is an activation function, n^{input} is the number of neurons on the input layer, w_{ij}^e is the weight of the connection between the i -th neuron of the hidden layer and the j -th neuron of the input layer and b_i^e is the i -th value of the bias vector on the input layer. Similarly, the i -th value of decoded vector h^d can be defined as $h_i^d = f(\sum_{j=1}^{n^{hidden}} w_{ij}^d h_j^e + b_i^d)$, where n^{hidden} is the number of neurons on the hidden layer, w_{ij}^d is the weight of the connection between the i -th neuron of the output layer and the j -th neuron of the hidden layer and b_i^d is the i -th value of the bias vector on the hidden layer.

The goal of training is to minimize the mean square error $E_h = \frac{1}{2} \sum_{i=1}^n (h_i - h_i^d)^2$ between input vector h and its reconstruction h^d . For training the network, AE can utilize a supervised learning technique called back-propagation. This technique implies changing connection weights after processing each feature vector h based on the amount of error E_h in the output. Using the gradient descent algorithm, changes in each weight Δw_{ij}^l and bias Δb_i^l ($l \in \{e, d\}$) can be found as follows:

$$\Delta w_{ij}^l = -\eta \frac{\partial E_h}{\partial w_{ij}^l}, \quad \Delta b_i^l = -\eta \frac{\partial E_h}{\partial b_i^l}, \quad (1)$$

where η is a parameter defining the learning rate and partial derivatives $\frac{\partial E_h}{\partial w_{ij}^l}$ and $\frac{\partial E_h}{\partial b_i^l}$ for $l \in \{e, d\}$ are calculated as

follows:

$$\frac{\partial E_h}{\partial w_{ij}^l} = h_j^e \delta_i^l, \quad \frac{\partial E_h}{\partial b_i^l} = \delta_i^l. \quad (2)$$

Here parameters δ_i^e and δ_i^d measure how much the i -th neuron is responsible for an error in the output, and they can be obtained as

$$\delta_i^d = (h_i^d - h_i) f', \quad \delta_i^e = \left(\sum_{j=1}^{n^{hidden}} w_{ij}^d \delta_i^d \right) f'. \quad (3)$$

If an AE can reconstruct original input perfectly, it means that hidden layer stores enough information of the input. In order to minimize information loss, auto-encoders can be stacked. Stacked auto-encoder can be trained in a greedy layer-wise fashion. The training of a SAE starts with the training of the first layer AE on raw input vectors. Once the first layer AE has been trained, we can calculate vectors which consist of activation of the hidden neurons of the first AE. These vectors can be used to train the second layer AE. This procedure is repeated for subsequent layers by using the output of each layer as input for the subsequent layer. Thus, parameters of each layer are trained individually while freezing parameters for the remainder of the model. Once all AEs have been trained, fine-tuning using back-propagation can be applied to improve the results by tuning the parameters of all layers at the same time.

Detection of anomalous histogram vectors described above is based on the assumption that data has variables correlated with each other and can be embedded into a lower dimensional subspace in which normal samples and anomalous samples appear significantly different [20]. For this purpose, a stacked auto-encoder is trained and fine-tuned with vectors of histogram matrix H^i . After this, for the t -th row H^{it} of this matrix we calculate its reconstruction error $E^{it} = \sqrt{\sum_{j=1}^{n_C} (H_j^{it} - \hat{H}_j^{it})^2}$, where \hat{H}^{it} is the reconstruction of H^{it} when applying the SAE. These reconstruction error values are used to calculate threshold $T^E = \mu_i^E + \omega \sigma_i^E$, where μ_i^E is the mean value of reconstruction error values calculated for feature vectors of the i -th group contained in the training set, and σ_i^E is the standard deviation of these values.

Let us consider a client which initiates several connections of type i during the recent time interval. After we classify these connections according to clusters obtained during the training, histogram vector h is calculated. Finally, this vector is encoded and decoded with the corresponding SAE and its reconstruction error E is found. If E is greater than threshold T_i^E , vector h is classified as an anomaly and connections of the client are considered as an attack. As one can see, the scheme proposed cannot define which connections of the client are normal and which connections have bad intent. However, this scheme allows one to find the attacker and what web service he attempts to attack.

IV. NUMERICAL SIMULATIONS

Our simulation examples are implemented with the help of Realistic Global Cyber Environment (RGCE) [14]. RGCE is

a closed environment, which mimics the structures and the user traffic of the real Internet that allow one to use real IP addresses and GeoIP information. Traffic within RGCE is automatically generated using special software which relies on the same functional principles as botnets. RGCE allows us to generate realistic traffic patterns of end users. Thus, this environment is capable to simulate very sophisticated and realistic scenarios that can be used for training, exercise and research [15], [14].

For this research, we model a fictitious service provider in RGCE. This provider hosts and defends a web server providing several web shop services. Communication between the web shop server and its clients is carried out with encrypted HTTPS protocol. Traffic of web shop clients is generated by both human users and automated user bots. Length of the simulation scenario is approximately 2 hours. In the scenario, there are 55 web shop users with different IP addresses and GeoIP locations. Several users have the web shop as a target for their attacks. The attackers use different sorts of DDoS attacks, including Slowloris, Slowpost and more advanced types of DDoS attacks. As in the real world, attackers scan the target beforehand in order to select suitable offensive methods. Moreover, attackers can perform different attack types during the same time period to increase their chances of success.

First 12 minutes of the dataset (or 10% of the traffic) are considered as the training set since there are no attacks during that period of time. We find all conversations initiated by the web shop clients during this training period. The size of time step is selected to be equal to 1 second for timely detection of attacks. Once the necessary features are extracted from the conversations, we divide them into clusters. In our simulations, we used the following clustering algorithms: single-linkage clustering algorithm, k-means, fuzzy c-means, self-organizing map (SOM) and DBSCAN.

After that, we look for anomalous conversations in the rest of the dataset as it is described in Section III-B. Figures 1 and 2 show how True Positive Rate depends on False Positive Rate when detecting Slowloris and Slowpost attacks correspondingly with the help of different clustering algorithms with different parameters. As one can notice, four

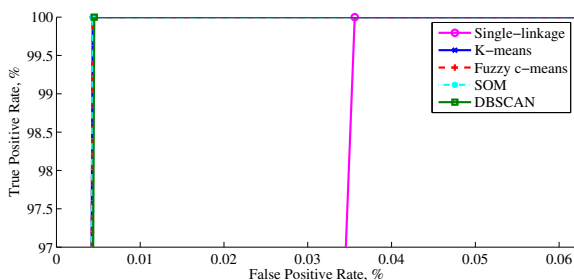


Fig. 1. ROC curve for detection of Slowloris.

clustering methods are able to detect all conversations related to Slowloris (TPR = 100%) with extremely low number of false alarms varying from 0.004% to 0.005%. However, single-

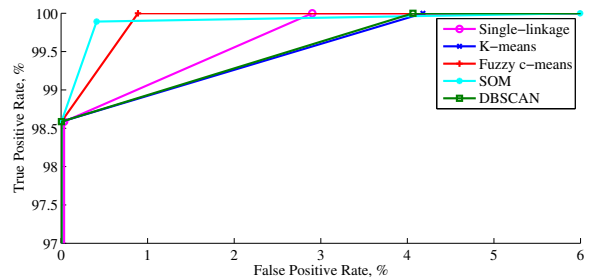


Fig. 2. ROC curve for detection of Slowpost.

linkage clustering algorithm gives the worst results with the highest number of false alarms (FPR = 0.0356%). In the case of Slowpost attack detection, when false positive rate is below 0.036% true positive rate is very high (TPR \approx 98%) for all methods tested. For bigger FPR values (FPR < 1%), fuzzy c-means and SOM outperform other methods in terms of true positive rate. If the false positive rate is allowed to be high (\approx 6%), all conversations related to Slowpost can be detected (TPR = 100%).

Tables I and II show TPR, FPR and the accuracy of detection of these trivial DoS attacks for the cases when clustering parameters are selected in an optimal way, i.e. when the detection accuracy is maximal. As one can see, almost all conversations related to slow HTTP/HTTPS attacks can be properly detected, while the number of false alarms remains very low.

TABLE I
SLOWLORIS DETECTION ACCURACY

Method	TPR	FPR	Accuracy
Single-linkage	100 %	0.0356 %	99.9644 %
K-means	100 %	0.0043 %	99.9957 %
Fuzzy c-means	100 %	0.0043 %	99.9957 %
SOM	100 %	0.0043 %	99.9957 %
DBSCAN	100 %	0.0045 %	99.9955 %

TABLE II
SLOWPOST DETECTION ACCURACY

Method	TPR	FPR	Accuracy
Single-linkage	98.587 %	0.0356 %	99.9619 %
K-means	98.587 %	0.0043 %	99.9931 %
Fuzzy c-means	98.587 %	0.0043 %	99.9931 %
SOM	98.587 %	0.0043 %	99.9931 %
DBSCAN	98.587 %	0.0045 %	99.9929 %

In order to detect attackers who are able to mimic properly the browsing behavior of a regular human user, the method described in Section III-C is applied. As it was mentioned above, the method cannot define which connections of the attacker are normal and which connections have bad intent. Therefore, we focus on detection of the attacker, its target and time interval when the attack takes place. In order to evaluate the technique proposed, we apply k-means, fuzzy c-means and SOM for clustering conversations. These algorithms are

selected since they allow us to obtain the best results in terms of trivial DoS attack detection as it is shown in Tables I and II. The number of hidden layers for the SAE is selected between 1 and 5. The number of neurons on each hidden layer is selected in such a way that it is less than the number of neurons on the previous layer.

Figure 3 shows how True Positive Rate depends on False Positive Rate when detecting intermediate DDoS attacks with the scheme proposed in this research. As one can notice, when

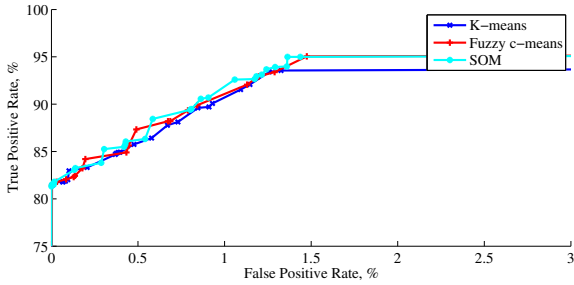


Fig. 3. ROC curve for detection of intermediate DDoS attacks.

FPR is 0, the SAE can detect more than 80% of conversation groups initiated by an attacker during some short time interval. For bigger FPR values ($FPR > 1\%$), the SAE which analyzes histogram matrices calculated by SOM outperforms other methods in terms of true positive rate.

Table III shows TPR, FPR and the accuracy of intermediate DDoS attacks detection. As one can see, these attacks can be properly detected. To decrease the FPR, instead of analyzing raw reconstruction error, we can consider its moving average over several recent time intervals. This allows us to reduce the amount of false alarms, but at the same time, this will most likely increase the delay between the time moment when a DDoS attack starts and time moment when this attack is detected. Thus, we have to choose between a low level of the false positive rate and timely detection of the attack.

TABLE III
INTERMEDIATE DDoS ATTACK DETECTION ACCURACY

Method	TPR	FPR	Accuracy
K-means	93.5569 %	1.3261 %	96.1058 %
Fuzzy c-means	95.051 %	1.475 %	96.7815 %
SOM	94.9906 %	1.3633 %	96.8068 %

V. CONCLUSION AND FUTURE WORK

In this research, we aimed to timely detect trivial and intermediate application-layer DoS attacks in encrypted network traffic by applying an anomaly-detection-based approach to statistics extracted from network packets. The scheme included the separation of conversations between a web server and its clients into clusters and the analysis of how conversations initiated by one client to the server during some short time interval are distributed among these clusters. The proposed technique was tested on the data obtained with the help of RGCE, a realistic cyber environment that generated realistic

traffic patterns of end users. As a result, almost all conversations related to DoS attacks were properly detected, while the number of false alarms remained very low. In the future, we are planning to improve the algorithm in terms of the detection accuracy, and test it with a bigger dataset which contains traffic captured during several days. In addition, we will focus on the simulation of advanced DDoS attacks in RGCE and detection of these attacks by applying anomaly-based approach.

REFERENCES

- [1] Kaspersky Lab. Statistics on botnet-assisted DDoS attacks in Q1 2015. <https://www.slideshare.net/KasperskyLabGlobal/statistics-on-botnet-assisted-ddos-attacks-in-q1-2015>, May 29, 2015.
- [2] V. Durcekova, L. Schwartz and N. Shahmehri. Sophisticated Denial of Service attacks aimed at application layer. *ELEKTRO*, pp. 55–60, 2012.
- [3] J. Yuan and K. Mills, Monitoring the macroscopic effect of DDoS flooding attacks. *IEEE Tran. Dependable and Secure Computing*, Vol. 2(4), pp. 324–335, 2005.
- [4] R. Chen, J.-Y. Wei and H. Yu. An improved Grey Self-Organizing Map based DOS detection. *Proc. of IEEE Conference on Cybernetics and Intelligent Systems*, pp. 497–502, 2008.
- [5] Y. Ke-xin and Z. Jian-qi. A novel DoS detection mechanism. *Proc. of International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)*, pp. 296–298, 2011.
- [6] C. Ye, K. Zheng and C. She. Application layer ddos detection using clustering analysis. *Proc. of the 2nd International Conference on Computer Science and Network Technology (ICCSNT)*, pp. 1038–1041, 2012.
- [7] P. Chwalinski, R. Belavkin, R. and X. Cheng. Detection of Application Layer DDoS Attacks with Clustering and Bayes Factors. *Proc. of IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 156–161, 2013.
- [8] M. Aiello, E. Cambiaso, M. Mongelli and G. Papaleo. An On-Line Intrusion Detection Approach to Identify Low-Rate DoS Attacks. *Proc. of International Carnahan Conference on Security Technology (ICCSST)*, pp. 1–6, 2014.
- [9] C. Xu, G. Zhao, G. Xie and S. Yu. Detection on application layer DDoS using random walk model. *Proc. of IEEE International Conference on Communications (ICC)*, pp. 707–712, 2014.
- [10] D. Stevanovic and N. Vlajic. Next Generation Application-Layer DDoS Defences: Applying the Concepts of Outlier Detection in Data Streams with Concept Drift. *Proc. of the 13th International Conference on Machine Learning and Applications*, pp. 456–462, 2014.
- [11] T. Dierks and E. Rescorla. The transport layer security (TLS) protocol. *IETF RFC 4346*, 2006.
- [12] Gartner - 2015 - Cybercriminals Hiding in SSL Traffic. White paper, Venafi, 2015.
- [13] M. Rafsanjani, Z. Varzaneh and N. Chukanlo. A survey of hierarchical clustering algorithms. *The Journal of Mathematics and Computer Science*, Vol. 5(3), pp. 229–240, 2012.
- [14] T. Kokkonen, T. Hämäläinen, M. Silokunnas, J. Siltanen, M. Neijonen. Analysis of Approaches to Internet Traffic Generation for Cyber Security Research and Exercise. *Proc. of the 15th Int. Conference on Next Generation Wired/Wireless Networking (NEW2AN)*, pp. 254–267, 2015.
- [15] M. Zolotukhin, T. Hämäläinen, T. Kokkonen and J. Siltanen. Online detection of anomalous network flows with soft clustering. *Proc. of the 7th International Conference on New Technologies, Mobility and Security (NTMS)*, pp. 1–5, 2015.
- [16] F. Gullo and A. Tagarelli. Uncertain Centroid Based Partitional Clustering of Uncertain Data. *VLDB Endow*, Vol. 5(7), pp. 610–621, 2012.
- [17] W.-K. Loh and Y.-H. Park. A Survey on Density-Based Clustering Algorithms. *Lecture Notes in Electrical Engineering*, Vol. 280, pp. 775–780, 2014.
- [18] Y. Chen, Z. Lin, X. Zhao, G. Wang and Y. Gu. Deep Learning-Based Classification of Hyperspectral Data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, Vol. 7(6), pp. 2094–2107, 2014.
- [19] G. Hinton and R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science* 28, Vol. 313(5786), pp. 504–507, 2006.
- [20] M. Sakurada and T. Yairi. Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction. *Proc. of the 2nd Workshop on Machine Learning for Sensory Data Analysis (MLSDA)*, pp. 4–11, 2014.

PVII

**ARCHITECTURE FOR THE CYBER SECURITY SITUATIONAL
AWARENESS SYSTEM**

by

Tero Kokkonen 2016

Lecture Notes in Computer Science, vol. 9870, pp. 294-302

Reproduced with kind permission of Springer International Publishing AG,
Copyright © 2016, Springer International Publishing AG.

Architecture for the Cyber Security Situational Awareness System

Tero Kokkonen^(✉)

Institute of Information Technology, JAMK University of Applied Sciences,
Jyväskylä, Finland
tero.kokkonen@jamk.fi

Abstract. Networked software systems have a remarkable and critical role in the modern society. There are critical software systems in every business area. At the same time, the amount of cyber-attacks against those critical networked software systems has increased in large measures. Because of that, the cyber security situational awareness of the own assets plays an important role in the business continuity. It should be known what is the current status of the cyber security infrastructure and own assets and what it will be in the near future. For achieving such cyber security situational awareness there is need for the Cyber Security Situational Awareness System. This study presents the novel architecture of the Cyber Security Situational Awareness System. The study also presents the use case of threat mitigation process for such Cyber Security Situational Awareness System.

Keywords: Cyber security · Situational awareness · Multi sensor data fusion · Situational awareness information sharing · Early warning

1 Introduction

Situational awareness and early warning capability is extremely important for command and control of the own assets or making decisions related to the mission or business. Military aviation has a long history of using command and control systems with situational awareness generated by multi sensor information that could also be shared from the systems of other organisations. There are similar requirements for situational awareness in the cyber domain. Sensor feed from multiple different sensors should be fused automatically and visualised for the decision maker. Additionally, the information of known cyber threats should be shared with other organisations.

The terms situational awareness and situation awareness are mixed in the literature and used for describing the same phenomenon. In this paper the term situational awareness is used because situational awareness is considered to describe the phenomenon more accurately.

As stated in [1] real time cyber security situational awareness and data exchange are required in several strategic guidelines of different countries, for example in Finland's Cyber Security Strategy [1, 2]. A systematic literature review [1] indicates that there are several studies related to situational awareness in cyber domain; however, it is still stated in [3] that there is no solution for Cyber Common Operating Picture (CCOP).

This paper proposes state of the art architecture for the Cyber Security Situational Awareness System including a multi sensor data fusion component and data exchange with trusted partner organisations. The paper also presents the use case process for the Cyber Security Situational Awareness System and threat mitigation. The paper consists of a comprehensive set of reference literature and research papers as the background of the study. First, the Cyber Security Situational Awareness is discussed and the Data Fusion process is described. Also, the interfaces are presented, and the requirements for Human Machine Interface and data visualisation are analysed, followed by the description of the proposed architecture and finally, the conclusion with proposed items for further work is presented.

2 Cyber Security Situational Awareness

Endsley specifies one of the most used definitions of situational awareness (or as stated in the original reference situation awareness) as in the volume of time and space gathering information and elaborating understanding of what is happening and prediction of what will happen in the near future [1, 4]. From the point of view of Cyber Security Situational Awareness System, it means that there is multi sensor information available indicating what is happening, there is the capability for analysing such information, and there is also capability for making predictions what will happen in the near future.

As stated in [5] there are three types of information needed for situational awareness in cyber security: information of computing and network components (own assets), threat information, and information of mission dependencies. According to [6] there are four components of situational awareness: Identity (organisation's goals, structure, decisions making processes and capabilities), Inventory (hardware and software components), Activity (past and present activity of own cyber assets), and Sharing (both inbound and outbound). Paper [7] proposes a framework that consists of real-time monitoring, anomaly detection, impact analysis, and mitigation strategies (RAIM). The U. S. Army Innovation Challenge for Cyber Situational Awareness covers analytics, data storage, and visualisation of networks, assets, open-source information, user activity, and threats [8].

It is important to notice that there is a large and increasing number of systems, devices and cyber security applications or sensors in the organisation network providing data to be analysed. Analysing that increasing amount of information requires high computational power [9]. Data fusion is a recognised technique in surveillance and the security systems used for merging the scattered surveillance and status information as integrated totality. For example, paper [10] introduces data fusion for intrusion detection information.

3 Multi Sensor Data Fusion

The data fusion is defined as “*the process of combining data to refine state estimates and predictions*” [11]. The dominant data fusion model is JDL model by the US Joint Directors of Laboratories Data Fusion Sub-Group. In the JDL model the fusion process

is divided into different levels. Originally, there were levels 0–4. Nowadays, there are levels 0–6 which can be described for the cyber domain as follows [11–16]:

- Level 0 (Data Assessment). Cyber security sensor feed to the system.
- Level 1 (Object Assessment). Identification of cyber entities for example services, devices, physical network connections or information flows and the properties of those entities.
- Level 2 (Situation Assessment). State of the systems in cyber domain. Combining, for example information of software versions, vulnerabilities or patches installed.
- Level 3 (Impact Assessment). Information related to an ongoing attack or threat, indicating the damage and mitigation actions or incident response required or already done.
- Level 4 (Process Refinement/Resource Management). Management of cyber sensors. Selection of used sensors, configuration of sensor settings and definition of the reliability score of each sensor.
- Level 5 (User Refinement/Knowledge Management). Human Machine Interface (HMI) providing access to control each layer of fusion. An important part of that level is effective visualisation of information to the user.
- Level 6 (Mission Management). Determination of mission objectives and policy for supporting decision making.

Giacobe presents an application of the JDL data fusion process model for cyber security utilising JDL levels 0–5 [14], and paper [15] introduces adapted national level JDL data fusion model for levels 0–5.

Paper [16] divides multi sensor data fusion algorithms under four main categories: Fusion of imperfect data, Fusion of correlated data, Fusion of inconsistent data, and Fusion of disparate data. There are several mathematical algorithms under those four categories. For example, [17] utilises Support Vector Machines (SVMs) as the fusion algorithm for network security situational awareness, and paper [18] proposes a Hierarchical Network Security Situation Assessment Model (HNSSAM) with DS data fusion for cyber security. Spatiotemporal event correlation is used for anomaly detection and for network forensics in study [19].

4 Interfaces

The proposed architecture includes several types of input information for data fusion supporting all the levels of JDL Data Fusion process. Because of that, the data fusion engine should implement several different data fusion algorithms chosen to support data fusion of such data. Following interfaces are proposed for the architecture.

4.1 Sensor Information

Input interfaces for the information from the cyber security sensor feeds such as information from anomaly based or signature based Intrusion Detection Systems (IDS), Intrusion Prevention Systems (IPS), firewalls, antivirus systems, log file analyser, authentication alarms etc.

4.2 Own Assets Status Information

Input interfaces for the information of the systems in the cyber domain. All the entities and their properties should be identified as well as their status and configuration information. Includes also the information of the sensors with their status and configuration information. Some of the systems are able to automatically inform their status and configuration information. Otherwise, the user will update the status information using HMI. If the service is under attack, the impact assessment status information is most likely input to the system by user. Additionally, the spare parts of the physical devices should be input to the system.

4.3 Analysis Information

The analysed impact assessment information about an ongoing attack or threat; caused damage, information of attacker, what are the used attack methods, what are the countermeasures, present and past mitigation activities or incident response activities, and the result of those activities. The analysis information also consists of Indicators Of Compromise (IOC) information and open source intelligence information originated, for example from social media, news or CERT-bulletins concerning systems in the use or the business area represented. Such open source intelligence information might offer early warning information about incoming threats or information needed for incident response. Paper [20] states that pure technical data is just a part of bigger situational awareness fused with intelligence information.

Certain policies or objectives that should be noticed as part of the Situational Awareness and decision-making information are input as part of the analysis information. The analysis information is input to the system both automatically and using HMI.

4.4 Sharing the Information

Information sharing is one of the most critical elements in cyber security. If there is a trusted network of other organisations and there is the capability to share information with those organisations, there is much more information available for the data fusion. With shared information there are requirements for filtering the information before sharing it according to the company policy. All the information cannot be shared because of the confidentiality of the security information. Inbound data should also be analysed and the reliability score assigned.

In the case of simultaneously ongoing data fusion and data sharing processed the origin of the information should be indicated because of the data-loops. If the information is shared (outbound) to any organisation of the information sharing community and after while the same information is shared back (inbound) from any organisation of the information sharing community, there is a data-loop. Data-loops produce problems with the data fusion algorithms. If the origin of the information is indicated and data fusion algorithm notices that inbound information originates from itself, such information should be perceived in the fusion process.

There are standards called Structured Threat Information eXpression (STIX™) [21] and Trusted Automated eXchange of Indicator Information (TAXII™) [22] for

exchanging cyber threat information. The information sharing community using such standards could be formed as described in paper [23].

5 HMI and Visualisation Layer

HMI should propose access to modify and add information for all the layers of fused data as described earlier in 3 and 4. It should also visualise the information efficiently for the user to obtain the scattered information more understandable format.

The visualisation part of the Cyber Security Situational Awareness System offers the Cyber Common Operating Picture to the user. The required data is in the system, the question is how to visualise that data to the user, especially for the decision maker who might not have deep technical background and knowhow. Many tools in cyber security are only for special purpose and certain data, not for integration of several types of data and without interoperability with other tools [24]. The authors of paper [25] used attack graphs for visualisation and ArcSight was used for visualisation in [26]. The paper [27] focuses on visualisation of threat and impact assessment.

The cyber domain is complex and there is plenty of different information available. The main conclusion for the visualisation problem is that there should be different visualisation tools and techniques for different purposes and for different user roles. Visualisation tools for high level decision makers are totally different to the tools for the analyst. Using case studies, the authors of paper [28] emphasise the potential for several different visualisation tools.

A solution for visualisation problem would be the usage of common symbols. Paper [29] suggests usage of military symbols, for example defined in standard MIL-STD-2525 [30]. Such standards should be extended for cyber domain, for example a military symbol for pending identity could mean a new incident in cyber domain. The common symbols should be defined and adopted as global standard for cyber security.

6 Proposed Architecture

The proposed novel architecture for the Cyber Security Situational Awareness System includes data fusion engine according to 3, interfaces described in 4, as well as HMI and Visualisation layer described in 5. Because there is plenty of different information from different sources the information needs to be normalised. The block diagram of the proposed architecture is presented in Fig. 1.

The ultimate goal for such systems is that described functionalities are as automatic as possible; however, there is analyst operator required for controlling the data fusion, controlling the sensors, and adding analysis information to the system. For example, cyber security sensors might produce false alarms and the data fusion might help with the false alarms by fusing the information from multiple sources; however, the analyst operator is required to analyse the sensor feed and maybe configure the sensors or indicating to the system that false alarms are occurring. Also, if there is a real incident ongoing, the analyst operator is capable of inputting the case related additional information to the system. The possible process for situational awareness and threat mitigation using the proposed architecture is presented in Fig. 2.

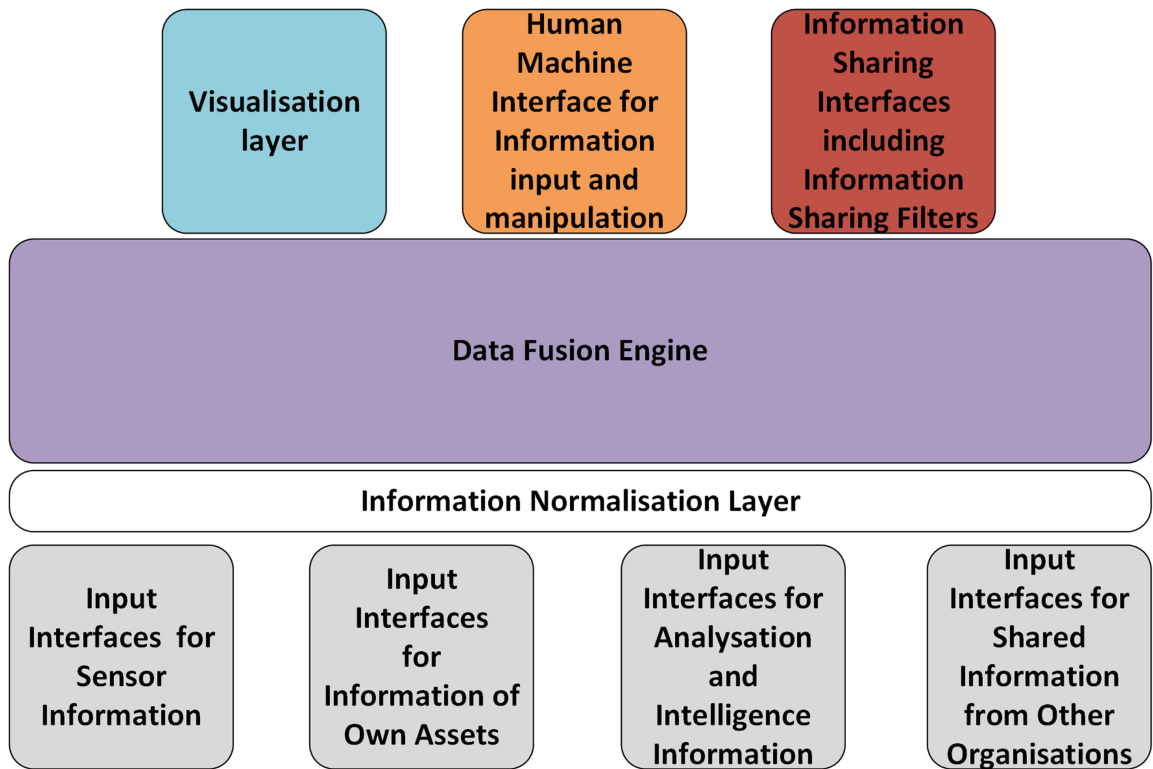


Fig. 1. Block diagram of proposed architecture

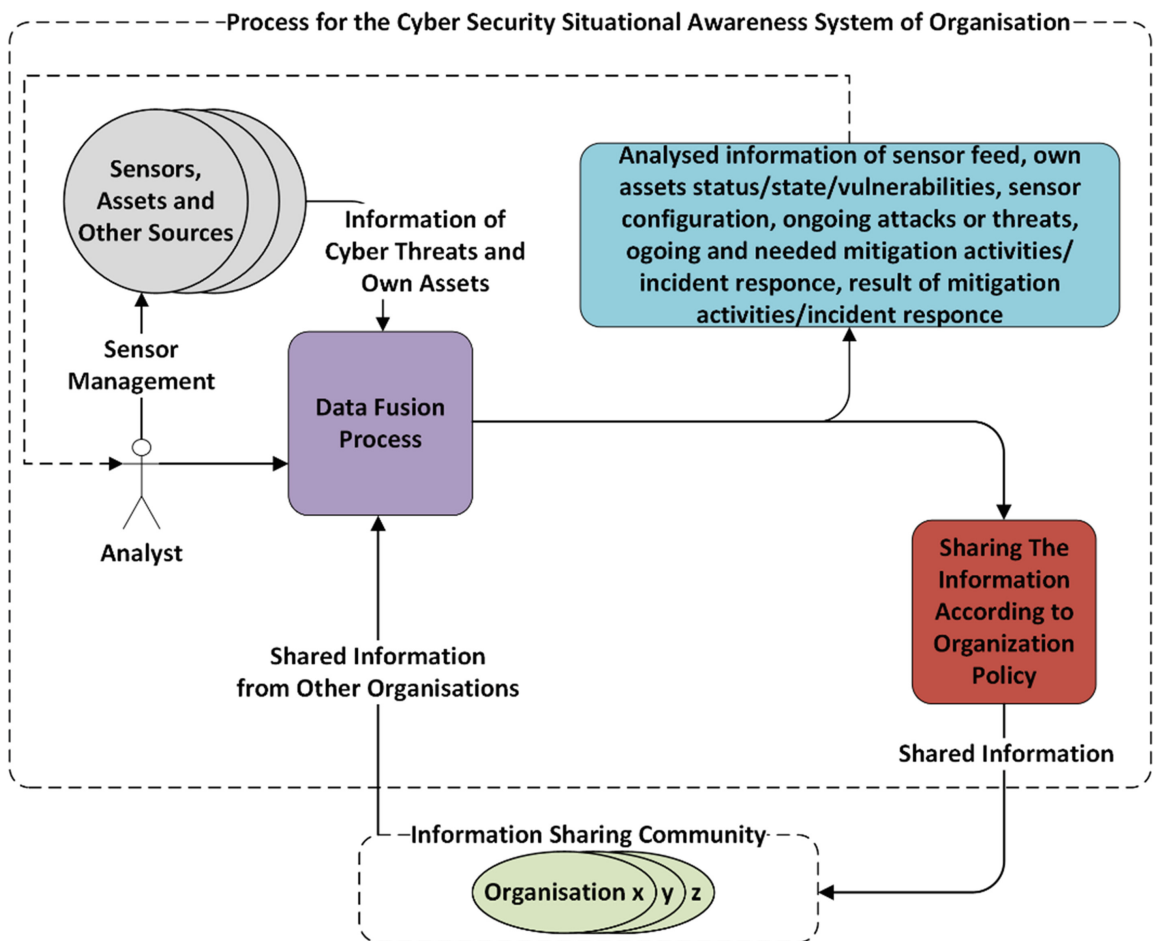


Fig. 2. Use case process for the cyber security situational awareness system

The proposed architecture represents the state of the art system in the domain of cyber security situational awareness systems utilising both data fusion engine and data exchange mechanisms at the same time. It also provides capability for implementation of all the levels of JDL data fusion process. Even the highest levels could be implemented and input to the system as a part of the Situational Awareness. The Visualisation could be deployed in layers for supporting the totally different requirements of different user roles, for example decision maker compared to analyst.

Detailed system requirements for the Cyber Security Situational Awareness System can be derived using proposed architecture. There are requirements for the data fusion engine according to 3, interfaces according to 4, HMI and Visualisation layer according to 5 and use case process presented in Fig. 2.

Developing such a system as product for the operational use requires detailed design and a great deal of software development. There are plenty of technical difficulties for developing such a system. Data models of the input information might be one of those. Some devices and sensors use standardised data models and protocols and some might use proprietary models. Some information is human made and some is automatically generated, the problem comes with the human made information, is it always without errors and structured correctly. Similar problems exist with many other integrated systems and can be solved using standardisation and structured data formats. Initial versions should be implemented with certain sensors and data feeds and extended gradually.

Processing a large amount of data could require lot of computational power; however, during the exact design of the system it could be divided into different nodes. The visualisation should be tested deeply with different user roles. There is a global requirement for common standardisation of visualisation symbols in cyber domain. Visualisation should be implemented layer by layer for different users and use cases.

7 Conclusion

The study proposes novel architecture for the Cyber Security Situational Awareness System. It includes the process for using such a system for achieving the cyber resilience of the business or mission. The proposed architecture includes both multi sensor fusion process and information exchange process which both are required for achieving proper situational awareness of the cyber security infrastructure and own assets. The architecture utilises all the levels of JDL data fusion model. Pure technical situational awareness could be enriched, for example using open source intelligence information, impact analysis information, information of incident response actions and certain policies of the organisation. The proposed architecture could be used both in government and industry organisations for state of the art Cyber Security Situational Awareness System.

The next steps for the study are developing a proof of concept system using the proposed architecture, testing different multi sensor data fusion algorithms for the proposed architecture and visualising the situational awareness of a complex distributed network system. Developed proof of concept system could be used for functional

evaluation of the theoretical architecture proposed in this study. Additionally, automatic threat mitigation based on situational awareness would be an interesting domain of research and development.

Acknowledgment. This work was funded by the Regional Council of Central Finland/Council of Tampere Region and European Regional Development Fund/Leverage from the EU 2014–2020 as part of the JYVSECTEC Center project of JAMK University of Applied Sciences Institute of Information Technology.

References

1. Franke, U., Brynielsson, J.: Cyber situational awareness - a systematic review of the literature. *Comput. Secur.* **46**, 18–31 (2014)
2. Secretariat of the Security Committee: Finland's Cyber Security Strategy. Government Resolution, 24 January 2013
3. Conti, G., Nelson, J., Raymond, D.: Towards a cyber common operating picture. In: *Proceedings of the 5th International Conference on Cyber Conflict (CyCon)*. NATO CCDCOE Publications, Tallinn (2013)
4. Endsley, M.: Toward a theory of situation awareness in dynamic systems. *Hum. Factors: J. Hum. Factors Ergon. Soc.* **37**(1), 32–64 (1995)
5. The MITRE Corporation: Cybersecurity, Situation Awareness. <https://www.mitre.org/capabilities/cybersecurity/situation-awareness/>. Accessed 23 May 2016
6. The Industrial Control System Information Sharing and Analysis Center (ICS-ISAC): Situational Awareness Reference Architecture (SARA). <http://ics-isac.org/blog/sara/>. Accessed 23 May 2016
7. Ten, C.W., Manimaran, G., Liu, C.C.: Cybersecurity for critical infrastructures: attack and defense modeling. *IEEE Trans. Syst. Man Cybern. - Part A: Syst. Hum.* **40**(4), 853–865 (2010)
8. Keller, J.: Army cyber situational awareness innovation challenge focuses on cyber threats at brigade level. In: *Military & Aerospace Electronics*, 18 November 2015. <http://www.militaryaerospace.com/articles/2015/11/army-cyber-threats.html>. Accessed 23 May 2016
9. Yu, W., Xu, G., Chen, Z., Moulema, P.: A cloud computing based architecture for cyber security situation awareness. In: *Proceedings of the IEEE Conference on Communications and Network Security (CNS)*, National Harbor, MD, pp. 488–492 (2013)
10. Bass, T.: Intrusion detection systems and multisensor data fusion. *Commun. ACM Mag.* **43**(4), 99–105 (2000)
11. Steinberg, A., Bowman, C., White, F.: Revisions to the JDL data fusion model. In: *SPIE Proceedings, Sensor Fusion: Architectures, Algorithms, and Applications III*, vol. 3719, pp. 430–441 (1999)
12. Azimirad, E., Haddadnia, J.: The comprehensive review on JDL model in data fusion networks: techniques and methods. *Int. J. Comput. Sci. Inf. Secur. (IJCSIS)* **13**(1) (2015)
13. Blasch, E., Steinberg, A., Das, S., Llinas, J., Chong, C., Kessler, O., Waltz, E., White, F.: Revisiting the JDL model for information exploitation. In: *Proceedings of the 16th International Conference on Information Fusion (FUSION)*, Istanbul, pp. 129–136 (2013)
14. Giacobe, N.: Application of the JDL data fusion process model for cyber security. In: *SPIE Proceedings, Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications*, vol. 7710, p. 77100R, 28 April 2010

15. Swart, I., Irwin, B., Grobler, M.: MultiSensor national cyber security data fusion. In: Proceedings of the 10th International Conference on Cyber Warfare and Security (ICCWS), pp. 320–328 (2015)
16. Khaleghi, B., Khamis, A., Karray, F.O., Razavi, S.N.: Multisensor data fusion: a review of the state-of-the-art. *Inf. Fusion* **14**(1), 28–44 (2013)
17. Liu, X., Wang, H., Liang, Y., Lai, J.: Heterogeneous multi-sensor data fusion with multi-class support vector machines: creating network security situation awareness. In: Proceedings of the Sixth International Conference on Machine Learning and Cybernetics, Hong Kong, pp. 2689–2694 (2007)
18. Zhanga, Y., Huang, S., Guob, S., Zhu, J.: Multi-sensor data fusion for cyber security situation awareness. In: Proceedings of the 3rd International Conference on Environmental Science and Information Application Technology (ESIAT 2011). *Procedia Environ. Sci.* **10**, 1029–1034 (2011)
19. Xie, Y.: A spatiotemporal event correlation approach to computer security. Doctoral Dissertation, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA, USA (2005)
20. Kornmaier, A., Jaouën, F.: Beyond technical data - a more comprehensive situational awareness fed by available Intelligence Information. In: Proceedings of the 6th International Conference on Cyber Conflict (CyCon). NATO CCDCOE Publications, Tallinn (2014)
21. Barnum, S.: Structured Threat Information eXpression (STIX™). Version 1.1, Revision 1, 20 February 2014. <http://stixproject.github.io/getting-started/whitepaper/>. Accessed 24 May 2016
22. Connolly, J., Davidson, M., Schmidt, C.: Trusted Automated eXchange of Indicator Information (TAXII™), 2 May 2014. <http://taxiiproject.github.io/getting-started/whitepaper/>. Accessed 24 May 2016
23. Kokkonen, T., Hautamäki, J., Siltanen, J., Hämäläinen, T.: Model for sharing the information of cyber security situation awareness between organizations. In: Proceedings of the 23rd International Conference on Telecommunications (ICT), Thessaloniki, Greece (2016)
24. Fink, G., North, C., Endert, A., Rose, S.: Visualizing cyber security: usable workspaces. In: Proceedings of the 6th International Workshop on Visualization for Cyber Security (VizSec), Atlantic City, NJ, pp. 45–56 (2009)
25. Jajodia, S., Noel, S., Kalapa, P., Albanese, M., Williams, J.: Cauldron mission-centric cyber situational awareness with defense in depth. In: Proceedings of the Military Communications Conference (MILCOM), Baltimore, MD, pp. 1339–1344 (2011)
26. Briesemeister, L., Cheung, S., Lindqvist U., Valdes, A.: Detection, correlation, and visualization of attacks against critical infrastructure systems. In: Proceedings of the 8th Annual Conference on Privacy, Security and Trust, Ottawa, Canada (2010)
27. Nusinov, M.: Visualizing threat and impact assessment to improve situation awareness. Thesis, Rochester Institute of Technology (2009)
28. Hall, P., Heath, C., Coles-Kemp, L.: Critical visualization: a case for rethinking how we visualize risk and security. *J. Cybersecur.* **1**(1), 93–108 (2015)
29. Grégoire, M., Beaudoin, L.: Visualisation for network situational awareness in computer network defence. In: Visualisation and the Common Operational Picture. RTO-MP-IST-043 (2005)
30. U.S Department of Defence Interface Standard, Joint Military Symbolology: MIL-STD-2525D, 10 June 2014

PVIII

**WEIGHTED FUZZY CLUSTERING FOR ONLINE DETECTION
OF APPLICATION DDOS ATTACKS IN ENCRYPTED
NETWORK TRAFFIC**

by

Mikhail Zolotukhin, Tero Kokkonen, Timo Hämäläinen, Jarmo Siltanen 2016

Lecture Notes in Computer Science, vol. 9870, pp. 326-338

Reproduced with kind permission of Springer International Publishing AG,
Copyright © 2016, Springer International Publishing AG.

Weighted Fuzzy Clustering for Online Detection of Application DDoS Attacks in Encrypted Network Traffic

Mikhail Zolotukhin¹(), Tero Kokkonen², Timo Hämäläinen¹,
and Jarmo Siltanen²

¹ Department of Mathematical Information Technology, University of Jyväskylä,
Jyväskylä, Finland

{mikhail.m.zolotukhin,timo.t.hamalainen}@jyu.fi

² Institute of Information Technology, JAMK University of Applied Sciences,
Jyväskylä, Finland

{tero.kokkonen,jarmo.siltanen}@jamk.fi

Abstract. Distributed denial-of-service (DDoS) attacks are one of the most serious threats to today's high-speed networks. These attacks can quickly incapacitate a targeted business, costing victims millions of dollars in lost revenue and productivity. In this paper, we present a novel method which allows us to timely detect application-layer DDoS attacks that utilize encrypted protocols by applying an anomaly-based approach to statistics extracted from network packets. The method involves construction of a model of normal user behavior with the help of weighted fuzzy clustering. The construction algorithm is self-adaptive and allows one to update the model every time when a new portion of network traffic data is available for the analysis. The proposed technique is tested with realistic end user network traffic generated in the RGCE Cyber Range.

Keywords: Network security · Intrusion detection · Denial-of-service · Anomaly detection · Fuzzy clustering

1 Introduction

Distributed denial-of-service (DDoS) attacks have become frighteningly common in modern high-speed networks. These attacks can force the victim to significantly downgrade its service performance or even stop delivering any service by using lots of messages which need responses consuming the bandwidth or other resources of the system [1]. Since it is difficult for an attacker to overload the target's resources from a single computer, DDoS attacks are launched via a large number of attacking hosts distributed in the Internet. Although traditional

network-based DDoS attacks have been well studied recently [2–4], there is an emerging issue of detection of DDoS attacks that are carried out on the application layer [5–9]. Unlike network-layer DDoS attacks, application-layer attacks can be performed by using legitimate requests from legitimately connected network machines which makes these attacks undetectable for signature-based intrusion detection systems (IDSs). Moreover, DDoS attacks may utilize protocols that encrypt the data of network connections in the application layer. In this case, it is hard to detect attacker’s activity without decrypting web users network traffic and, therefore, violating their privacy.

Anomaly-based approach is probably the most promising solution for detecting and preventing application-layer DDoS attacks because this approach is based on discovering normal user behavioral patterns and allows one to detect even zero-day attacks. Attacks that involve the use of HTTP protocol is the most prevalent application-layer DDoS attack type nowadays [7] due to the protocol popularity and high number of vulnerabilities in this protocol. For these reasons, many recent studies have been devoted to the problem of anomaly-based detection of application-layer DDoS attacks that utilize HTTP protocol. For example, paper [5] analyzes application-layer DDoS attacks against a HTTP server with the help of hierarchical clustering of user sessions. Study [6] shows a novel detection technique against HTTP-GET attacks, based on Bayes factor analysis and using entropy-minimization for clustering. In [7], authors propose the next-generation application-layer DDoS defense system based on modeling network traffic to dynamic web-domains as a data stream with concept drift. In [8], authors detect application-layer DDoS attacks by constructing a random walk graph based on sequences of web pages requested by each user. In [13], we propose an algorithm for detection of trivial and intermediate application-layer DoS attacks in encrypted network traffic based on clustering conversations between a web server and its clients and analysis of how conversations initiated by one client during some short time interval are distributed among these clusters.

In this study, we improve our technique for anomaly-based detection of application-layer DDoS attacks that utilize encrypted protocols proposed in [13]. The technique relies on the extraction of normal behavioral patterns and detection of samples that significantly deviate from these patterns. For this purpose, we analyze the traffic captured in the network under inspection. It is assumed that the most part of the traffic captured during the system training is legitimate. This can be achieved by filtering the traffic with the help of a signature-based intrusion detection system. Unfortunately, the method presented in [13] requires the entire training dataset to be stored in memory, and, therefore, it cannot be applied when the amount of network traffic is really huge. The improvement proposed in this study allows one to update the normal user behavior model every time when a new portion of network traffic is available for the analysis. As a result, it requires significantly less amounts of computing and memory resources and can be successfully employed for prevention of DDoS attacks in high-speed networks.

The rest of the paper is organized as follows. Extraction of the most relevant feature vectors from network traffic is considered in Sect. 2. Fuzzy clustering algorithms are presented in Sect. 3. Section 4 describes the DDoS attack detection technique based on weighted fuzzy clustering. In Sect. 5, we evaluate the performance of the technique proposed. Finally, Sect. 6 draws the conclusions and outlines future work.

2 Feature Extraction

We concentrate on the analysis of network traffic flows in SSL/TLS traffic transferred over TCP protocol as the most popular reliable stream transport protocol. A flow is a group of IP packets with some common properties passing a monitoring point in a specified time interval. In this study, we assume that these common properties include IP address and port of the source and IP address and port of the destination. The length of each such time interval should be picked in such a way that allows one to detect attacks timely. When analyzing a traffic flow extracted in some time interval, we also take into account all packets of this flow transferred during previous time intervals. Resulting flow measurements provide us an aggregated view of traffic information and drastically reduce the amount of data to be analyzed. After that, two flows such as the source socket of one of these flows is equal to the destination socket of another flow and vice versa are found and combined together. This combination is considered as one conversation between a client and a server.

A conversation can be characterized by following four parameters: source IP address, source port, destination IP address and destination port. For each such conversation at each time interval, we extract the following information:

1. duration of the conversation
2. number of packets sent in 1 second
3. number of bytes sent in 1 second
4. maximal, minimal and average packet size
5. maximal, minimal and average size of TCP window
6. maximal, minimal and average time to live (TTL)
7. percentage of packets with different TCP flags: FIN, SYN, RST, etc.

Features of types 2–7 are extracted separately for packets sent from the client to the server and from the server to the client. It is worth to mention that here we do not take into account time intervals between subsequent packets of the same flow. Despite the fact, that increasing of these time intervals is a good sign of a DDoS attack, taking them into consideration leads to the significant increasing of the number of false alarms. It is caused by the fact, that when the server is under attack it cannot reply to legitimate clients timely as well, and, therefore, legitimate clients look like attackers from this point of view.

Values of the extracted feature vectors can have different scales. In order to standardize the feature vectors, max-min normalization is used. Extracted feature vectors are recalculated as follows:

$$x_{ij} = \frac{x_{ij} - \min_{1 \leq i \leq n} x_{ij}}{\max_{1 \leq i \leq n} x_{ij} - \min_{1 \leq i \leq n} x_{ij}}. \quad (1)$$

As a result, new value of feature x_{ij} is in range $[0, 1]$. We denote vectors that contain minimum and maximum feature values as x_{min} and x_{max} respectively.

3 Theoretical Background

In this section, we present theoretical background for the DDoS attack detection algorithm presented in the next section.

3.1 Fuzzy C-Means

Let us consider data set $X = \{x_1, x_2, \dots, x_n\}$ where x_j is a feature vector of length d . We are aiming to divide these vectors into c clusters. Fuzzy c-means is a method of clustering which allows one vector x_j to belong to two or more clusters [10]. It is based on minimization of the following objective function:

$$J = \sum_{i=1}^c \sum_{j=1}^n u_{ij}^m \|v_i - x_j\|^2, \quad (2)$$

where $m > 1$ is a fuzzification coefficient, u_{ij} is the degree of membership of the j -th feature vector x_j to the i -th cluster and v_i is the center of this cluster. This objective function can be minimized by iteratively calculating the cluster centers as follows:

$$v_i = \frac{\sum_{j=1}^n u_{ij}^m x_j}{\sum_{j=1}^n u_{ij}^m}, \quad (3)$$

where

$$u_{ij} = \frac{1}{\sum_{k=1}^c \left(\frac{\|v_i - x_j\|}{\|v_k - x_j\|} \right)^{2/(m-1)}}. \quad (4)$$

The clustering algorithm can be described as follows:

1. Initialize the membership matrix $U = \{u_{ij}\}$ in such a way that $u_{ij} \in (0, 1)$ for $\forall i \in \{1, \dots, c\}$ and $\forall j \in \{1, \dots, n\}$ and $\sum_{i=1}^c u_{ij} = 1$.
2. Calculate fuzzy cluster centers v_i for $i \in \{1, \dots, c\}$ using (3).
3. Compute the objective function with (2).
4. Compute new U using (4) and go back to step 2.

The algorithm stops if at some iteration the improvement of the objective function over previous iteration is below a certain threshold, or the maximum number of iterations is reached.

3.2 Weighted Fuzzy C-Means

One of the biggest drawbacks of the fuzzy c-means algorithm is that it requires to store all feature vectors in memory before the clustering algorithm starts. For this reason, this algorithm cannot be applied in the case when the dataset to be clustered is huge. One solution of this problem is dividing the dataset into subsets of data so that the size of each subset does not exceed the amount of memory resources of the processing system. In this case, the clustering result of the first subset can be summarized as weighted centroids. These centroids can be then used as weighted points when clustering vectors of the second and subsequent subsets. This technique is often used for clustering data streams. In this case, each subset consists of data vectors that have arrived for processing at the recent time window.

Weighted fuzzy c-means [11, 12] relies on the principle described above. Let us consider data set $X = \{x_1, x_2, \dots, x_n\}$. In addition, there is a set of weighted centroids $\{v_1, \dots, v_c\}$ defined based on previous subsets. We are aiming to update these centroids based on vectors of subset X . The objective function minimized by the weighted fuzzy c-means can be defined as follows:

$$J = \sum_{i=1}^c \sum_{j=1}^{c+n} u_{ij}^m w_j \|v_i - x_j\|^2. \quad (5)$$

where w_j is the weight of the j -th point. Cluster centers can be found as

$$v_i = \frac{\sum_{j=1}^{c+n} w_j u_{ij}^m x_j}{\sum_{j=1}^{c+n} w_j u_{ij}^m}. \quad (6)$$

Thus, the weighted fuzzy c-means algorithm for clustering a dataset that consists of T subsets X^t , $t \in \{1, \dots, T\}$ can be formulated as follows:

1. Apply fuzzy c-means to vectors of the first subset X^1 to find centroids v_i and calculate weights for the resulting centroids:

$$w_i = \sum_{j=1}^n u_{ij} \quad (7)$$

2. Import vectors of the next subset and calculate membership matrix $U = \{u_{ij}\}$ where $i \in \{1, \dots, c\}$ and $j \in \{1, \dots, c+n\}$ as follows:

$$u_{ij} = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j \text{ and } j \in \{1, \dots, c\}, \\ \frac{1}{\sum_{k=1}^c \left(\frac{\|v_i - x_j\|}{\|v_k - x_j\|} \right)^{2/(m-1)}}, & \text{if } j \in \{c+1, \dots, c+n\}. \end{cases} \quad (8)$$

3. For each vector x_j of the new subset, assign weight equal to 1 and recalculate centroid weights as follows:

$$w'_i = \begin{cases} \sum_{j=1}^{c+n} u_{ij} w_j, & \text{if } i \in \{1, \dots, c\}, \\ 1, & \text{if } i \in \{c+1, \dots, c+n\}. \end{cases} \quad (9)$$

where $w_j = 1$, $\forall j \in \{c+1, \dots, n\}$.

4. Update cluster centroids v_i , $i \in \{1, \dots, c\}$ by substituting weight values w_i in (6) with new weights w'_i .
5. Compute the objective function by substituting new weights and centroids in (5).
6. Compute new U using (4) and go back to step 3.
7. If, at some iteration, the improvement of the objective function (5) over previous iteration is below a certain threshold, or the maximum number of iterations is reached, stop modifying centroids and move to step 2.

The algorithm stops once all subsets have been imported and analyzed.

4 DDoS Attack Detection Algorithm

In this section, we formulate our DDoS attack anomaly-based detection algorithm. The algorithm relies on constructing a normal user behavior model and detecting patterns that deviate from the expected norms. We consider two versions of obtaining the normal user behavior model: offline and online. Offline training algorithm can only be applied to a small training dataset that can be stored in memory with the resulting model that cannot be modified afterwards. Online version of the training algorithm allows one to rebuild the normal user behavior model based on feature vectors arrived in the recent time interval. As a result, it requires significantly less amounts of memory since there is no need to store all feature vectors extracted.

4.1 Offline Training Algorithm

In study [13], we presented the general description of our DDoS attack detection approach. This approach relies on the training stage that can be divided into two main parts. First, all conversations extracted from the network traffic are clustered. After that, for each particular user, distributions of conversations among the resulting clusters are analyzed. In this subsection, we formulate the training algorithm for the case of fuzzy clustering in more details.

Let us consider the set of standardized feature vectors $X = \{x_1, \dots, x_n\}$ extracted from a training set of network traffic conversations. We apply fuzzy c-means clustering algorithm to obtain cluster centroids $V^x = \{v_1^x, \dots, v_c^x\}$ and partition matrix $U^x = \{u_{ij}^x\}$. Clustering allows us to discover hidden patterns presented in the dataset to represent a data structure in a unsupervised way. Each cluster centroid calculated represents a specific class of traffic in the network system under inspection. For example, one such class can include conversations between a web server and clients which request some web page of this server. Since the traffic may be encrypted it is not always possible to define what web page these clients request. However, since it is assumed that traffic being clustered is mostly legitimate, we can state that each cluster centroid describes a normal user behavior pattern.

To evaluate how much a feature vector is similar to the revealed normal patterns or their combination a reconstruction criterion can be considered [14]. The reconstruction of x_j is defined as follows:

$$\bar{x}_j = \frac{\sum_{i=1}^c (u_{ij}^x)^m v_i^x}{\sum_{i=1}^c (u_{ij}^x)^m}. \quad (10)$$

We can calculate the reconstruction error E of this vector as

$$E(x_j) = \|x_j - \bar{x}_j\|^2. \quad (11)$$

For the detection purposes, we store the average value μ^x of reconstruction errors calculated for feature vectors contained in the training set X as well as the standard deviation of these error values σ^x .

After that, we group all conversations which are extracted in certain time interval and have the same source IP address, destination IP address and destination port together and analyze each such group separately. Such approach is in-line with other studies devoted to the problem of application-based DDoS attacks detection [5,6,8]. Those studies analyze sequences of conversations (requests) belonging to one HTTP session. In our case, since the session ID cannot be extracted from encrypted payload, we focus on conversations initiated by one client to the destination socket during some short time interval. We can interpret a group of such conversations as a rough approximation of the user session.

Let us consider these user session approximations $S = \{s_1, \dots, s_N\}$ found in the training set. In [13], we introduced histogram vectors each element of which was equal to the percentage of feature vectors contained in each particular conversation cluster. In this study, we extend this approach for the case of fuzzy clustering by introducing session membership matrix $A = \{a_{ij}\}$ with $i \in \{1, \dots, N\}$ and $j \in \{1, \dots, n\}$ such that

$$a_{ij} = \begin{cases} 1, & \text{if } x_j \in s_i, \\ 0, & \text{if } x_j \notin s_i \end{cases} \quad (12)$$

We consider new feature matrix

$$Y = A(U^x)^T. \quad (13)$$

Element y_{ij} of this matrix is directly proportional to the average probability that a conversation from the i -th ‘‘session’’ belongs to the j -th cluster of conversations.

Once new feature vectors have been extracted, fuzzy c-means clustering algorithm is applied to obtain cluster centroids $V^y = \{v_1^y, \dots, v_C^y\}$ and partition matrix $U^y = \{u_{ij}^y\}$. Similarly to the clusters of conversations, each new cluster centroid represents a specific class of traffic in the network under inspection. For example, one such centroid can relate to clients that use some web service in similar manner. As previously, we consider each resulting cluster centroid as a normal user behavior pattern, because it is assumed that traffic captured during the training is mostly legitimate. We calculate the reconstruction error for

each of these new feature vectors and store its average value μ^y and standard deviation σ^y .

As a result, once the training has been completed, the following model of normal user behavior is obtained:

$$M_{offline} = \{x_{min}, x_{max}, V^x, \mu^x, \sigma^x, V^y, \mu^y, \sigma^y\}. \quad (14)$$

4.2 Online Training Algorithm

The biggest drawback of the algorithm described above is that it requires to store all feature vectors X and Y in memory until the moment the clustering is completed. In this subsection, we improve our technique by proposing an online training algorithm that allows to rebuild the model of normal user behavior every time when a new portion of network traffic is available for the analysis.

Let us assume that the network traffic is captured during several short time intervals and, for each time interval, there is a set of feature vectors extracted from conversations between users. We are aiming to build the model of normal user behavior under the condition that there is only room for storing one such set of vectors in memory. For this purpose, we consider feature vectors extracted during the first time interval. The offline version of the training algorithm can be applied to these vectors to obtain x_{min} , x_{max} , V^x , μ^x , σ^x , V^y , μ^y and σ^y . In addition, we calculate weights $w^x = (w_1^x, \dots, w_c^x)$ and $w^y = (w_1^y, \dots, w_c^y)$ of cluster centroids V^x and V^y by using partition matrices U^x and U^y respectively as shown in (7). Moreover, we calculate the following matrix H :

$$H = V^y(e^T w^x)^{-1}, \quad (15)$$

where e is vector of length c each element of which is equal to 1. The following model of normal user behavior is obtained after the traffic captured during the first time interval is analyzed:

$$M_{online} = \{x_{min}, x_{max}, V^x, w^x, \nu^x, \mu^x, \sigma^x, H, V^y, w^y, \nu^y, \mu^y, \sigma^y\}, \quad (16)$$

where ν^x and ν^y are correspondingly the total number of conversations and the total number of user sessions analyzed.

Once a new portion of network traffic has been captured, we extract necessary features from conversations to form new set $X = \{x_1, \dots, x_n\}$. Let us denote vectors x_{min} and x_{max} obtained during the previous training iteration as x_{min}^{old} and x_{max}^{old} . New vectors x_{min} and x_{max} can be found as follows:

$$\begin{aligned} x_{min,j} &= \min \{x_{min,j}^{old}, \min_{1 \leq i \leq n} x_{ij}\}, \\ x_{max,j} &= \max \{x_{max,j}^{old}, \max_{1 \leq i \leq n} x_{ij}\}. \end{aligned} \quad (17)$$

After that, cluster centroids V^x are recalculated:

$$v_{ij}^x = \frac{(x_{max,j}^{old} - x_{min,j}^{old})v_{ij}^x + x_{min,j}^{old} - x_{min,j}}{x_{max,j} - x_{min,j}}. \quad (18)$$

Next, we apply weighted fuzzy c-means (starting from step 2) to vectors from X taking into consideration recalculated cluster centroids V^x and their weights w^x . Once new cluster centroids and their weights have been defined, we calculate reconstruction error for each vector in X as shown in (11) and find its mean value $\bar{\mu}^x$ and standard deviation $\bar{\sigma}^x$. New values of μ^x and σ^x can be found as follows:

$$\begin{aligned}\mu^x &= \frac{\nu^x \bar{\mu}^x + n \mu^{x,old}}{\nu^x + n}, \\ \sigma^x &= \sqrt{\frac{\nu^x (\bar{\sigma}^x)^2 + n (\sigma^{x,old})^2 + \nu^x (\mu^{x,old} - \mu^x)^2 + n (\mu^{x,old} - \mu^x)^2}{\nu^x + n}}.\end{aligned}\quad (19)$$

After that, new value ν^x is recalculated as

$$\nu^x = \nu^{x,old} + n. \quad (20)$$

The second part of the model is also updated. First, new cluster centroids V^y are obtained as

$$V^y = H w^x. \quad (21)$$

After that, feature matrix $Y = \{y_1, \dots, y_N\}$ for user session approximations is found (13). Weighted fuzzy c-means is applied to matrix Y taking into account recalculated centroids V^y and their weights w^y . As a result, new cluster centroids with updated weights are obtained. Average value μ^y and standard deviation σ^y of reconstruction errors for vectors in Y are recalculated in the same manner as it is shown in (19). The total number of user sessions is updated as

$$\nu^y = \nu^{y,old} + N. \quad (22)$$

Finally, new matrix H is calculated as shown in (15).

As a result, the entire model of normal user behavior is updated based on new vectors from X . The training is finished, once the traffic captured during each time interval has been taken into account.

4.3 Attack Detection

To detect a trivial DoS attack we extract necessary features from a new conversation and calculate the reconstruction error e^x for the resulting feature vector x according to centroids V^x . We classify this conversation as an intrusion if

$$e^x > \mu^x + \alpha^x \sigma^x, \quad (23)$$

where α^x is the parameter that is configured during tuning the detection system.

If a client initiates several connections during the recent time interval, we calculate a partition matrix for these connections and find the corresponding vector y . After that, the reconstruction error e^y is defined based on centroids V^y . This client is classified as an attacker if

$$e^y > \mu^y + \alpha^y \sigma^y. \quad (24)$$

As previously, parameter α^y is supposed to be configured during the system validation. Although, in this case, we cannot define which connections of the client are normal and which connections have bad intent, we are able to find the attacker and what web service he attempts to attack. Moreover, this technique allows one to detect more sophisticated types of DDoS attacks.

5 Numerical Simulations

The proposed technique is tested with network traffic generated in Realistic Global Cyber Environment (RGCE) [15]. A web shop server is implemented in RGCE to serve as a target of three different DDoS attacks carried out by several attackers. Communication between the server and its clients is carried out with encrypted HTTPS protocol. An IDS prototype that relies on the proposed technique is implemented in Python. The resulting program analyzes arriving raw packets, combines them to conversations, extracts necessary features from them, adds the resulting feature vectors to the model in the training mode and classifies those vectors in the detection mode. The IDS is trained with the traffic that does not contain attacks by using offline and online training algorithms proposed. Once the training has been completed, several attacks are performed to evaluate true positive rate (TPR), false positive rate (FPR) and accuracy of the algorithms.

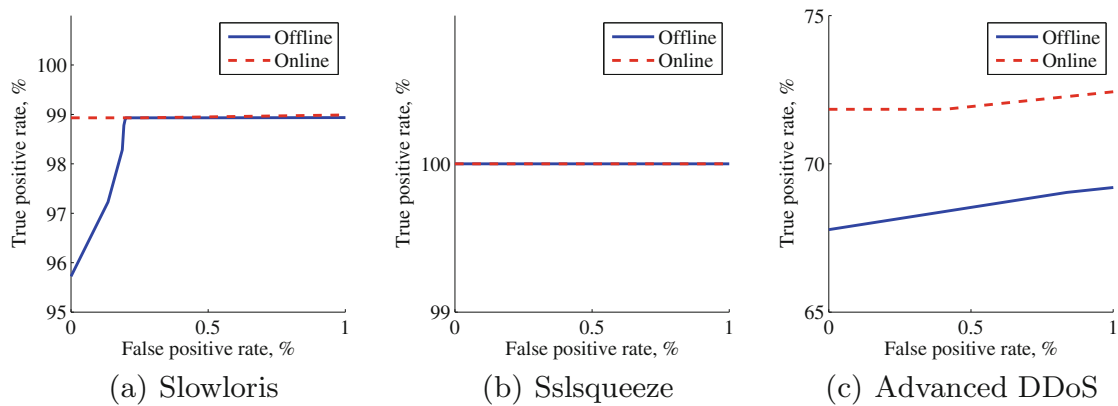


Fig. 1. ROC curves for detection of different DDoS attacks.

First DDoS attack tested is Slowloris that is usually classified as a trivial application-layer DDoS attack. During this attack, each attacker tries to hold its connections with the server open as long as possible by periodically sending subsequent HTTP headers, adding to-but never completing-the requests. As a result, the web server keeps these connections open, filling its maximum concurrent connection pool, eventually denying additional connection attempts from clients. Figure 1(a) shows how TPR depends on FPR when detecting Slowloris for different values of parameter α^x . As one can notice, for low values of FPR both variants of the algorithm are able to detect almost all conversations related to Slowloris (TPR \approx 99%).

However, if conversations related to a DDoS attack are close enough to the cluster centroids V^x of the normal behavior model, they will remain undetected. In this case, vectors of feature matrix Y should be taken into consideration. In order to test the second part of the model, Sslsqueeze attack is performed. In this case, attackers send some bogus data to the server instead of encrypted client key exchange messages. By the moment the server completes the decryption of the bogus data and understands that the data decrypted is not a valid key exchange message, the purpose of overloading the server with the cryptographically expensive decryption has been already achieved. As can be seen at Fig. 1(b) this attack is detected with 100 % accuracy.

Finally, we carry out a more advanced DDoS attack with the attackers that try to mimic the browsing behavior of a regular human user. During this attack several bots request a random sequence of web resources from the server. Since all those requests are legitimate, the corresponding conversations are classified as normal. However, the analysis of feature vectors Y calculated for user sessions allow us to detect the most part of the attacking attempts. Figure 1(c) shows how TPR depends on FPR for different values of parameter α^y when we try to detect this advanced DDoS attack. As one can see, for low values of FPR about 70 % of attacking attempts can be detected.

Table 1 shows the accuracy of detection of these three DDoS attacks for the cases when parameters α^x and α^y are selected in an optimal way, i.e. when the detection accuracy is maximal. As one can see, almost all attacks can be properly detected, while the number of false alarms remains very low.

Table 1. Detection accuracy

Training algorithm	Slowloris	Sslsqueeze	Advanced DDoS
Offline	99.113 %	100 %	68.619 %
Online	99.223 %	100 %	71.837 %

6 Conclusion

In this research, we aimed to timely detect different sorts of application-layer DDoS attacks in encrypted network traffic by applying an anomaly-detection-based approach to statistics extracted from network packets. Our method relies on the construction of the normal user behavior model by applying weighted fuzzy clustering. The proposed online training algorithm allows one to rebuild this model every time when a new portion of network traffic is available for the analysis. Moreover, it does not require a lot of computing and memory resources to be able to work even in the case of high-speed networks. We tested our technique on the data obtained with the help of RGCE Cyber Range that generated realistic traffic patterns of end users. As a result, almost all DDoS

attacks were properly detected, while the number of false alarms remained very low. In the future, we are planning to improve the algorithm in terms of the detection accuracy, and test it with a bigger dataset which contains real end user traffic captured during several days. In addition, we will focus on the simulation of more advanced DDoS attacks and detection of these attacks by applying our anomaly-based approach.

Acknowledgments. This work is partially funded by the Regional Council of Central Finland/Council of Tampere Region and European Regional Development Fund/Leverage from the EU 2014–2020 as part of the JYVSECTEC Center project of JAMK University of Applied Sciences Institute of Information Technology.

References

1. Durcekova, V., Schwartz, L., Shahmehri, N.: Sophisticated denial of service attacks aimed at application layer. In: ELEKTRO, pp. 55–60 (2012)
2. Yuan, J., Mills, K.: Monitoring the macroscopic effect of DDoS flooding attacks. *IEEE Trans. Dependable Secure Comput.* **2**(4), 324–335 (2005)
3. Chen, R., Wei, J.-Y., Yu, H.: An improved grey self-organizing map based DOS detection. In: Proceedings of IEEE Conference on Cybernetics and Intelligent Systems, pp. 497–502 (2008)
4. Ke-xin, Y., Jian-qi, Z.: A novel DoS detection mechanism. In: Proceedings of International Conference on Mechatronic Science, Electric Engineering and Computer (MEC), pp. 296–298 (2011)
5. Ye, C., Zheng, K., She, C.: Application layer ddos detection using clustering analysis. In: Proceedings of the 2nd International Conference on Computer Science and Network Technology (ICCSNT), pp. 1038–1041 (2012)
6. Chwalinski, P., Belavkin, R.R., Cheng, X.: Detection of application layer DDoS attacks with clustering and bayes factors. In: Proceedings of IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 156–161 (2013)
7. Stevanovic, D., Vljajic, N.: Next generation application-layer DDoS defences: applying the concepts of outlier detection in data streams with concept drift. In: Proceedings of the 13th International Conference on Machine Learning and Applications, pp. 456–462 (2014)
8. Xu, C., Zhao, G., Xie, G., Yu, S.: Detection on application layer DDoS using random walk model. In: Proceedings of IEEE International Conference on Communications (ICC), pp. 707–712 (2014)
9. Ndibwile, J., Govardhan, A., Okada, K., Kadobayashi, Y.: Web Server protection against application layer DDoS attacks using machine learning and traffic authentication. In: Proceedings of IEEE 39th Annual Computer Software and Applications Conference (COMPSAC), vol. 3, pp. 261–267 (2015)
10. Dunn, J.: A fuzzy relative of the ISODATA process, and its use in detecting compact well-separated clusters. *J. Cybern.* **3**(3), 32–57 (1973)
11. Hore, P., Hall, L., Goldgof, D.: A fuzzy c means variant for clustering evolving data streams. In: Proceedings of IEEE International Conference on Systems, Man and Cybernetics, pp. 360–365 (2007)
12. Wan, R., Yan, X., Su, X.: A weighted fuzzy clustering algorithm for data stream. In: Proceedings of ISECS International Colloquium on Computing, Communication, Control, and Management, vol. 1, pp. 360–364 (2008)

13. Zolotukhin, M., Hämäläinen, T., Kokkonen, T., Siltanen, J.: Increasing web service availability by detecting application-layer DDoS attacks in encrypted traffic. In: Proceedings of the 23rd International Conference on Telecommunications (ICT) (2016)
14. Izakian, H., Pedrycz, W.: Anomaly detection in time series data using a fuzzy c-means clustering. In: Proceedings of Joint IFSA World Congress and NAFIPS (IFSA/NAFIPS) Annual Meeting, pp. 1513–1518 (2013)
15. Kokkonen, T., Hämäläinen, T., Silokunnas, M., Siltanen, J., Neijonen, M.: Analysis of approaches to internet traffic generation for cyber security research and exercise. In: Proceedings of the 15th International Conference on Next Generation Wired/Wireless Networking (NEW2AN), pp. 254–267 (2015)

PIX

**ON APPLICATION-LAYER DDOS ATTACK DETECTION IN
HIGH-SPEED ENCRYPTED NETWORKS**

by

Mikhail Zolotukhin, Tero Kokkonen, Timo Hämäläinen, Jarmo Siltanen
Accepted Oct/2016

International Journal of Digital Content Technology and its Applications
(JDCTA)

Reproduced with kind permission of Advanced Institute of Convergence
Information Technology (AICIT),
Copyright © 2016, AICIT.

On Application-Layer DDoS Attack Detection in High-Speed Encrypted Networks

¹Mikhail Zolotukhin, ²Tero Kokkonen, ³Timo Hämäläinen, ⁴Jarmo Siltanen

^{1,2,3}*Department of Mathematical Information Technology, University of Jyväskylä
Jyväskylä, Finland
{mikhail.m.zolotukhin, timo.t.hamalainen}@jyu.fi, tero.t.kokkonen@student.jyu.fi*

^{2,4}*Institute of Information Technology, JAMK University of Applied Sciences
Jyväskylä, Finland
{tero.kokkonen, jarmo.siltanen}@jamk.fi*

Abstract

Application-layer denial-of-service attacks have become a serious threat to modern high-speed computer networks and systems. Unlike network-layer attacks, application-layer attacks can be performed by using legitimate requests from legitimately connected network machines which makes these attacks undetectable for signature-based intrusion detection systems. Moreover, the attacks may utilize protocols that encrypt the data of network connections in the application layer making it even harder to detect attacker's activity without decrypting users network traffic and violating their privacy. In this paper, we present a method which allows us to timely detect various application-layer attacks against a computer network. We focus on detection of the attacks that utilize encrypted protocols by applying an anomaly-detection-based approach to statistics extracted from network packets. Since network traffic decryption can violate ethical norms and regulations on privacy, the detection method proposed analyzes network traffic without decryption. The method involves construction of a model of normal user behavior by analyzing conversations between a server and clients. The algorithm is self-adaptive and allows one to update the model every time when a new portion of network traffic data is available. Once the model has been built, it can be applied to detect various types of application-layer denial-of-service attacks. The proposed technique is evaluated with realistic end user network traffic generated in our virtual network environment. Evaluation results show that these attacks can be properly detected, while the number of false alarms remains very low.

Keywords: *Network Security, Intrusion Detection, Denial of Service, Anomaly Detection, Traffic Clustering*

1 Introduction

The Internet has become the major universal communication infrastructure. Unfortunately, it is also subject to cyber-attacks in growing numbers and varieties. Denial-of-service (DoS) attacks have become frighteningly common in modern high-speed networks. On a daily basis, hundreds of websites are hit by networks of infected machines which flood them with junk data making them inaccessible for regular users [1] [2] [3]. In general, DoS attacks aim to disable a computer or network system using lots of messages which need responses consuming the bandwidth or other resources of the system.

Since it is difficult for an attacker to overload the targets resources from a single computer, DoS attacks are often launched via a large number of distributed attacking hosts in the Internet. Such distributed DoS (DDoS) attacks can force the victim to significantly downgrade its service performance or even stop delivering any service [4]. Designed to elude detection by today's most popular tools, these attacks can quickly incapacitate a targeted business, costing victims in lost revenue and productivity.

Traditional DDoS attacks such as ICMP flooding and SYN flooding are carried out at the network layer. The purpose of these attacks is to consume the network bandwidth and deny service to legitimate

users of the victim systems. This type of attack has been well studied recently and different schemes have been proposed to protect the network and equipment from such bandwidth attacks [5][6][7]. For this reason, attackers shift their offensive strategies to application-layer attacks. Application-layer DDoS attacks may focus on exhausting the server resources such as Sockets, CPU, memory, disk bandwidth, and I/O bandwidth. Unlike network-layer DoS attacks, application-layer attacks do not necessarily rely on inadequacies in the underlying protocols or operating systems. They can be performed by using legitimate requests from legitimately connected network machines. This makes application-layer DDoS attacks so difficult to detect and prevent.

Anomaly-based approach is a promising solution for detecting and preventing application-layer DDoS attacks. Such approach learns the features of event patterns which form normal behavior, and, by observing patterns that deviate from the established norms (anomalies), detects when an intrusion has occurred. Thus, systems which use the anomaly detection approach are modeled according to normal user-behavior and, therefore, are able to detect zero-day attacks, i.e. intrusions unseen previously. The problem of anomaly-based detection of application-layer DoS and DDoS attacks is of great interest nowadays [8][9][10][11][12][13][14][15] [16] [17].

In this study, we focus on the detection of the attacks that involve the use of HTTP protocol since it is the most prevalent application-layer denial-of-service attack type nowadays [2]. Study [10] divides HTTP-based DDoS attacks into three categories based on the level of their sophistication. The first category is trivial DDoS attacks, during which each bot participating in the attack sends one or a limited number of unrelated HTTP attacks towards the target site. This type of attacks includes such well-known attacks as Sslsqueeze [18] and Slowloris [8]. The second category includes attacks that are carried out by bots generating random sequences of browser-like requests of web-pages with all of their embedded content making the attack traffic indistinguishable from the regular human traffic. The last category contains more advanced DoS attacks that are predicted to rise in popularity in the future. These advanced attacks will consist of sequences of HTTP requests which are carefully chosen so as to better mimic the browsing behavior of regular human users.

Despite the rising interest to the detection of application-layer DDoS attacks utilizing HTTP protocol, most of the current researches concentrate on the analysis of information extracted from network packet payload which includes web resource requested, request method, session ID and other parameters. However, nowadays many DDoS attacks are utilizing secure protocols such as SSL/TLS that encrypt the data of network connections in the application layer which makes it impossible to detect attacker activity based on the analysis of packets' payload without decrypting it [19]. For this reason, the detection of DDoS attacks is supposed to be carried out with the help of statistics that can be extracted mostly from network packet headers.

Another challenge in the problem of application-layer DDoS attacks detection is distinguishing these attacks from flash crowds. Flash crowds are large surges of legitimate traffic which occur on popular web sites when thousands of requests access to the web servers over the relatively short period of time. Flash crowds are quite similar with DDoS attacks in terms of network anomaly and traffic phenomenon. They even can cause a web site or target to slow down its service for users or even temporarily shut down due to the significant increase of traffic [20].

It is also critical to implement a DDoS detection system which is capable to function effectively in computer networks that have high traffic and high-speed connectivity [21]. Moreover, since the mitigation of damage from a DDoS attack relies on its timely detection, the detection process is supposed to take place in an online mode. Nowadays, high-speed computer networks and systems deal with thousands traffic flows per second resulting in the data rate of several Gbps. For this reason, the construction of the normal user behavior model and the detection of an anomalous activity in a high-speed network requires considerable amounts of memory and computing resources. Thus, the problem of proper management of these resources is one of the most important challenges when designing a DDoS detection and prevention system.

In this study, we propose an application-layer DDoS attack detection scheme that meets the requirements discussed above. First, our method relies on the extraction of normal user behavior patterns and detection of anomalies that significantly deviate from these patterns. This allows us to detect even attacks that are carried out with legitimate requests from legitimately connected network machines. Moreover, the scheme proposed operates with information extracted from packet headers and therefore can be applied in secure protocols that encrypt the data of network connections without its decrypting. Finally, the normal user behavior model is obtained with the help of a data stream

clustering algorithm that allows us to continuously update the model within memory and time restrictions. In order to evaluate our scheme, we implement a DDoS detection system prototype that employs the algorithm proposed. In addition, we create a virtual network environment that allows us to generate some realistic end user network traffic and different sorts of DDoS attacks. Simulation results show that these attacks can be properly detected, while the number of false alarms remains very low.

The rest of the paper is organized as follows. Problem formulation and related work are discussed in Section 2. Extraction of the most relevant features from network traffic is considered in Section 3. Section 4 describes our approach of DDoS attacks detection in encrypted network traffic. The implementation of this approach in the context of high-speed networks is presented in Section 5. In Section 6, we evaluate the performance of the technique proposed. Finally, Section 7 draws the conclusions and outlines future work.

2 Problem formulation

We concentrate on the detection of application-layer DDoS attacks in SSL/TLS traffic transferred over TCP protocol as the most popular reliable stream transport protocol. We consider a network system that consists of several web servers that provide various services to their end users by utilizing this protocol. Outgoing and incoming traffic of these servers is captured and analyzed in order to detect and prevent potential attacks. The analysis process can be divided into two main phases: training and detection. During the training phase, we aim to investigate the traffic and discover behavior patterns of normal users. It is assumed that the most part of the traffic captured during this training phase is legitimate. In real world, this can be achieved by filtering the traffic with the help of a signature-based intrusion detection system. Once normal user behavior patterns have been discovered, these patterns can be used to analyze network traffic and detect DDoS attacks against the servers in online mode.

3 Related work

The anomaly-based detection approach is often applied for the application-layer DDoS attack detection. Study [15] proposes an advanced entropy-based scheme, that analyzes the distribution of captured packets among network flows. The method allows to detect variable rate DDoS attacks, divide them into different fields and treat each field with different methods. Paper [14] proposes to cluster user sessions based on number of requests in the session, request rate, average popularity of objects in the session and average transition probability of objects in the session. Hierarchical clustering is applied to separate attack sessions from normal ones. Paper [9] shows a novel detection technique against HTTP-GET attacks that relies on clustering of user sessions based on minimizing entropy of requests in the sessions belonging to the same cluster. Bayes factor analysis is then used to detect attacking hosts. Paper [8] considers detection of slow DoS attacks by analyzing numbers of packets received by a web server over small time horizons. The detection is carried out by using two spectral metrics: average of these packet numbers and the mutual information of the fast Fourier transform applied to the number of packets received at the current time horizon and at the previous one. In [11], authors model a normal user browsing behavior by constructing a random walk graph based on sequences of web pages requested by each user. Once the random walk model has been trained, the users subsequent page request sequence is predicted based on page transition probabilities calculated. After this, an attacker can be detected by calculating the similarity between the predicted page request sequence and observed sequence in the subsequent observation period. Study [10] proposes the next-generation system for application-layer DDoS defense by modeling network traffic to dynamic web-domains as a data stream with concept drift. An outlier detection is carried out by using the normalized length of the longest common subsequence similarity metric applied to chronological browsing sequences visited Web pages during a user session. In [12], different features are constructed for each user session to differentiate between an attacker and a normal user. These features include the number of HTTP requests made by a user in a particular time slot, the number of unique URL requests, numbers of successful, redirected and invalid requests, and several others. Once all necessary features are calculated, logistic regression is used for modeling a normal user browsing behavior for detecting the application layer DDoS attack traffic. Study [13] proposes to search for abstract features for each user

session with the help of a stacked auto encoder. After that, a logistic regression classifier is used to find network traffic related to a DDoS attack.

As one can notice, all of these studies propose to detect application-layer DDoS attacks by monitoring network packet payload. However, it remains unclear how to detect attacks in encrypted traffic. In this study, this problem is solved by modeling normal user behavior based on clustering feature vectors extracted from packet headers. Traffic clustering without using packet payload information is a crucial domain of research nowadays due to the rise in applications that are either encrypted or tend to change port consecutively. For example, study [22] uses k-means and model-based hierarchical clustering based on the maximum likelihood in order to group together network flows with similar characteristics. In [23], authors classify encrypted traffic with supervised learning algorithm C4.5 that generates a decision tree using information gain, semi-supervised k-means and unsupervised multi-objective genetic algorithm (MOGA).

In modern high-speed networks, large amounts of flow data are generated continuously at an extremely rapid rate. For this reason, it is not possible to store all the data in memory, which makes algorithms such as k-means and other batch clustering algorithms inapplicable to the problem of traffic flow clustering [24].

Data stream clustering algorithms is probably the most promising solution for this problem. Most stream clustering algorithms summarize the data stream using special data structures: cluster features, core sets, or grids. After performing the data summarization step, data stream clustering algorithms obtain a data partition via an offline clustering step [25]. Cluster feature trees and micro-cluster trees are employed to summarize the data in such algorithms as BIRCH [26], CluStream [27] and ClusTree [28]. The StreamKM++ algorithm [29] summarizes the data stream into a tree of coresets that are weighted subsets of points that approximate the input data. Grid-based algorithms such as DStream [30] and DGClust [31] partition the feature space into grid cells, each of which is representing a cluster. Approximate clustering algorithms such as streaming k-means [32] first choose a subset of the points from the stream, ensuring that the selected points are as distant from each other as possible, and then execute k-means on the data subset. Approximate stream kernel k-means algorithm [24] uses importance sampling to sample a subset of the data stream, and clusters the entire stream based on each data points similarity to the sampled data points in real-time.

4 Feature extraction

In order to extract features that are necessary for building a normal user behavior model and detecting outliers, we consider a portion of network traffic transferred in the computer system under inspection in some very short time window. The length of this time window should be picked in such a way that allows one to detect attacks timely.

The method proposed in this study is based on the analysis of network traffic flows. A flow is a group of IP packets with some common properties passing a monitoring point in a specified time interval. These common properties include transport protocol, the IP address and port of the source and IP address and port of the destination. As it was mentioned in the previous section, in this study, we concentrate on the traffic transferred over TCP. The time interval is considered to be equal to the time window defined previously. Moreover, when analyzing a traffic flow extracted in the current time window, we take into account all packets of this flow transferred during previous time windows. Resulting flow measurements provide us an aggregated view of traffic information and drastically reduce the amount of data to be analyzed. After that, two flows such as the source socket of one of these flows is equal to the destination socket of another flow and vice versa are found and combined together. This combination is considered as one conversation between a client and a server.

A conversation can be characterized by following four parameters: source IP address, source port, destination IP address and destination port. For each such conversation at each time interval, we extract the following information:

- 1) duration of the conversation
- 2) number of packets sent in 1 second
- 3) number of bytes sent in 1 second
- 4) maximal, minimal and average packet size
- 5) maximal, minimal and average size of TCP window
- 6) maximal, minimal and average time to live (TTL)

- 7) percentage of packets with different TCP flags: URG, ACK, PSH, RST, SYN and FIN
- 8) percentage of encrypted packets with different properties: handshake, alert, etc.

Features of types 2–8 are extracted separately for packets sent from the client to the server and from the server to the client. Features of types 2–7 are extracted from packet headers whereas a value of feature 8 can be found in packet payload even though it is encrypted.

It is worth to mention that here we do not take into account time intervals between subsequent packets of the same flow. Despite the fact, that increasing of these time intervals is a good sign of a DDoS attack, taking them into consideration leads to the significant increasing of the number of false alarms. It is caused by the fact, that when the server is under attack it cannot reply to legitimate clients timely as well, and, therefore, legitimate clients look like attackers from this point of view.

Values of the extracted feature vectors can have different scales. In order to standardize the feature vectors, max-min normalization is used. Max-min normalization performs a linear alteration on the original data so that the values are normalized within the given range. For the sake of simplicity, we map vectors to range $[0,1]$. Since all network traffic captured during the training stage is assumed to be legitimate, all the resulting standardized feature vectors can be used to reveal normal user behavior patterns and detect behavioral anomalies.

To map a value x_{ij} of the j -th attribute with values (x_{1j}, x_{2j}, \dots) from range $x_{ij} \in [x_{min,j}, x_{max,j}]$ to range $z_{ij} \in [0, 1]$ the computation is carried out as follows:

$$z_{ij} = \frac{x_{ij} - x_{min,j}}{x_{max,j} - x_{min,j}}. \quad (1)$$

5 Detection approach

In order to be able to classify application-layer DDoS attacks, we propose an anomaly-detection-based system that relies on the extraction of normal behavioral patterns during the training followed by the detection of samples that significantly deviate from these patterns. For this purpose, we analyze the traffic captured in the network under inspection with the help of several data mining techniques.

5.1 Training

Once all relevant features have been extracted and standardized, we divide resulting feature vectors into several groups by applying a clustering algorithm. Each such group is supposed to consist of objects that are in some way similar between themselves and dissimilar to objects of other groups. Clustering allows us to discover hidden patterns presented in the dataset to represent a data structure in an unsupervised way. There are many different clustering algorithms including hierarchical clustering algorithms, centroid-based clustering algorithms and density-based clustering algorithms. Each cluster calculated represents a specific class of traffic in the network system under inspection. For example, one such class can include conversations between a web server and clients which request the same web page of this server. Since the traffic may be encrypted it is not always possible to define what web page these clients request. However, since it is assumed that traffic being clustered is mostly legitimate, we can state that each cluster describes a normal user behavior pattern.

After that, we group all conversations which are extracted in certain time interval and have the same source IP address, destination IP address and destination port together and analyze each such group separately. Such approach is in-line with other studies devoted to the problem of application-based DDoS attacks detection [9][11][14]. Those studies analyze sequences of conversations (requests) belonging to one HTTP session. In our case, since the session ID cannot be extracted from encrypted payload, we focus on conversations initiated by one client to the destination socket during some short time interval. We can interpret a group of such conversations as a rough approximation of the user session.

For each such group of conversations, we obtain a sequence of numbers that are labels of the clusters found to which these conversations belong. An n -gram model is applied to extract new features from each such sequence. An n -gram is a sub-sequence of n overlapping items (characters, letters, words, etc.) from a given sequence. The n -gram models are used in speech recognition [33] and language

processing [34]. Thus, the n -gram model transforms each user session to a sequence of n -labels. Then the frequency vector is built by counting the number of occurrences of each n -label in the analyzed session. The length of the frequency vector is k^n , where k is the number of conversation clusters.

Once new feature vectors have been extracted, a clustering algorithm can be applied to divide these vectors into groups. Similarly, to the clusters of conversations, each cluster of n -gram vectors represents a specific class of traffic in the network under inspection. For example, one such cluster can include clients that use some web service in similar manner. As previously, we consider each resulting cluster as a normal user behavior pattern, because it is assumed that traffic captured during the training is mostly legitimate. Thus, the normal user behavior model consists of the clusters of two types: clusters of conversations and clusters of user sessions, that directly depend on the conversation clusters.

5.2 Detection

Once the training has been completed, the system is able to detect network intrusions. To detect a trivial DoS attack we extract necessary features from a new conversation and classify the resulting feature vector according to the clusters found. If this vector does not belong to any of the clusters, the corresponding conversation is labeled as intrusive and it is supposed to be blocked by the server.

For example, for centroid-based clustering methods, to define whether a new vector belongs to a cluster or not, we calculate the distance between this vector and the cluster center. If the distance between the new vector and the cluster center is greater than a predefined threshold, this vector does not belong to the cluster. This threshold T for some cluster can be calculated based on vectors of the training set which belong to this cluster: $T = \mu + \gamma\sigma$, where μ is the average distance between the center and vectors of this cluster, σ is the standard deviation of these distance values and γ is some numeric parameter tuned during the detection system validation.

The anomalous conversations found allow us to detect trivial DDoS attacks. However, if the attacker is able to mimic the browsing behavior of a regular human user, conversations related to this attack might belong to one of the clusters of the normal behavior model and, therefore, remain undetected. In this case, n -gram statistics should be taken into consideration. Vectors obtained with n -gram model during an attack can differ markedly from vectors corresponding to legitimate traffic. Thus, we can define whether a computer or network system is under attack during the current time interval, and, moreover, find clients responsible for initiating conversations related to the attack.

Let us consider a client which initiates several connections of certain type during the recent time interval. After we classify these connections according to clusters of conversations obtained during the training, the n -gram model is applied to transform this client session into new feature vector. If this vector does not belong to any of the session clusters extracted during the training, then this new vector is classified as an anomaly and all connections of the client are considered as an attack. As one can see, in this case, we cannot define which connections of the client are normal and which connections have bad intent. However, this scheme allows us to find the attacker and what web service he attempts to attack. After that, the attacker can be black-listed and a more sophisticated approach can be applied to analyze the conversations initiated by this attacker in more details.

6 Implementation

In this section, we discuss how the approach described above can be implemented to protect a web service in a high-speed encrypted network. The most challenging part of the implementation is related to the training stage, since there can be huge volumes of network traffic generated continuously at an extremely rapid rate and there is no possibility to store all features extracted from this traffic in memory. For this reason, a data stream clustering algorithm can be applied to conversations between users and the web service. However, in this case, conversation clusters may change every time a new portion of traffic has arrived. In turn, this leads to modifications of n -gram vectors representing user sessions calculated during previous time windows. These modifications are supposed to be made before session clusters are updated with new data extracted from this portion of the traffic.

6.1 Clustering conversations

Let us consider conversations between clients and the web service that take place in the current time window. We propose to cluster feature vectors extracted from these conversations by constructing an array of centroids that summarizes the data partition [35] [36].

We consider feature vectors $X^t = \{x_1^t, \dots, x_c^t\}$ extracted from n_c^t conversations during the t -th time window and standardized vectors $Z^t = \{z_1^t, \dots, z_c^t\}$ that are obtained from raw vectors X^t with the help of max-min standardization using values $x_{max,j}^t$ and $x_{min,j}^t$. In order to find k centroids for these vectors, we can apply standard iterative refinement technique. First, k centroids are supposed to be initiated. It can be carried out by randomly choosing k feature vectors from Z^t . However, there is a more efficient way to select initial centroids by following kmeans++ procedure [37] [38]:

- 1) Choose an initial center m_1^t centroid uniformly at random from Z^t .
- 2) Choose the next center m_i^t , selecting $m_i^t = z_j^t \in Z^t$ with probability v_j^t :

$$v_j^t = \frac{\min_{l \in \{1, \dots, i-1\}} d(m_l^t, z_j^t)}{\sum_{h=1}^{n_c^t} \min_{l \in \{1, \dots, i-1\}} d(m_l^t, z_h^t)}, \quad (2)$$

where $j = \{1, \dots, n_c^t\}$.

- 3) Repeat the previous step $k - 1$ times to select k initial centroids.

Once initial centroids m_1^t, \dots, m_k^t have been selected, we iteratively update the centroids as follows:

- 1) Assign each feature vector to the nearest centroid:

$$p_i^t = \{z \in Z^t: d(z, m_i^t) = \min_l d(z, m_l^t)\}. \quad (3)$$

- 2) For each partition p_i^t , find a new centroid:

$$m_i^t = \frac{1}{|p_i^t|} \sum_{z \in p_i^t} z. \quad (4)$$

These two steps are repeated until there are no longer changes in partitions during the assignment step.

Let us denote the raw vector that corresponds to standardized vector z as $x(z)$. For each resulting partition p_i^t , we store in memory its centroid

$$\mu_i^t = \frac{1}{|p_i^t(z)|} \sum_{z \in p_i^t(z)} x(z), \quad (5)$$

the number of feature vectors contained in partition p_i^t

$$w(p_i^t) = |p_i^t|, \quad (6)$$

and sum of squared features in these vectors

$$\zeta(p_i^t) = \sum_{z \in p_i^t} x^2(z), \quad (7)$$

where $x^2(z) = (x^2(z_1), x^2(z_2), \dots)$. It is worth noting that despite we use standardized vectors for clustering, we store statistics calculated for raw vectors. In addition, we store vectors $x_{min,j}^t$ and $x_{max,j}^t$ used for the standardization.

We calculate all the partitions for τ consecutive time windows $t \in \{1, 2, \dots, \tau\}$, where value of τ is defined by the memory constraints. In order to compress $\tau \times k$ resulting partitions into new k clusters, first, we calculate minimal $\bar{x}_{min,j}^\tau$ and maximal $\bar{x}_{max,j}^\tau$ feature values:

$$\begin{aligned} \bar{x}_{min,j}^\tau &= \min_{t \in \{1, 2, \dots, \tau\}} x_{min,j}^t, \\ \bar{x}_{max,j}^\tau &= \max_{t \in \{1, 2, \dots, \tau\}} x_{min,j}^t. \end{aligned} \quad (8)$$

These values are used to standardize centroids μ_i^t into vectors m_i^t for $t \in \{1, 2, \dots, \tau\}$ and $i \in \{1, 2, \dots, k\}$.

We obtain new k centroids with the technique similar to the one described above. The only difference is that we take into account the numbers of feature vectors assigned with each centroid:

- 1) Choose an initial center \bar{m}_1^τ centroid uniformly at random from m_j^t where $t \in \{1, 2, \dots, \tau\}$ and $j \in \{1, 2, \dots, k\}$.
- 2) Choose the next center \bar{m}_j^τ , selecting $\bar{m}_j^\tau = m_j^t$ with probability v_j^t :

$$v_j^t = \frac{w(p_j^t) \min_{l \in \{1, \dots, i-1\}} d(\bar{m}_l^\tau, m_j^t)}{\sum_{t=1}^\tau \sum_{h=1}^k w(p_h^t) \min_{l \in \{1, \dots, i-1\}} d(\bar{m}_l^\tau, m_h^t)}. \quad (9)$$

- 3) Repeat the previous step $k - 1$ times to select k initial centroids.

After that, new centroids are updated as follows:

- 1) Assign each old centroid m_j^t to the nearest new centroid \bar{m}_i^τ :

$$\bar{p}_i^\tau = \{m_j^t : d(m_j^t, \bar{m}_i^\tau) = \min_l d(m_j^t, \bar{m}_l^\tau)\}. \quad (10)$$

- 2) For each partition \bar{p}_i^τ , find a new centroid:

$$\bar{m}_i^\tau = \frac{1}{\sum_{m(z) \in \bar{p}_i^\tau} w(z)} \sum_{m(z) \in \bar{p}_i^\tau} w(z)z. \quad (11)$$

These two steps are again repeated until there are no longer changes in partitions during the assignment step.

For each resulting partition \bar{p}_i^τ , we store in memory its centroid

$$\bar{\mu}_i^\tau = \frac{1}{\sum_{m(z) \in \bar{p}_i^\tau} w(z)} \sum_{m(z) \in \bar{p}_i^\tau} w(z)x(z), \quad (12)$$

the number of feature vectors associated with this centroid

$$w(\bar{p}_i^\tau) = \sum_{m(x) \in \bar{p}_i^\tau} w(x), \quad (13)$$

and sum of squared features in these vectors

$$\varsigma(\bar{p}_i^\tau) = \sum_{m(x) \in \bar{p}_i^\tau} \varsigma(x). \quad (14)$$

In addition, we substitute values $x_{min,j}^t$ and $x_{max,j}^t$ for $t \in \{1, 2, \dots, \tau\}$ with vectors $\bar{x}_{min,j}^\tau$ and $\bar{x}_{max,j}^\tau$ used for the standardization.

Once $\tau \times k$ partitions p_i^t have been compressed to new k partitions \bar{p}_i^τ , information about the old partitions can be removed from the memory. After that, the algorithm continues in the same manner with finding partitions for the next $\tau - 1$ time windows $t \in \{\tau + 1, \tau + 2, \dots, 2\tau - 1\}$ and combining them with partitions \bar{p}_i^τ to obtain new k partitions.

In order to reduce the amount of computing resources required, this clustering procedure can be substituted with streaming k -means approximation proposed in [38]. However, from our numerical simulations, we notice, that the clustering algorithm used converges in just few iterations. To guarantee that the clustering is completed during the current time window, the number of iterations is recommended to be artificially limited.

6.2 Clustering sessions

We consider a group of conversations that are extracted at time window t and have the same source IP address, destination IP address and destination port. As mentioned in the Section 4 we interpret this group as a rough approximation of the user session. Once all connections at this time window have been divided into k partitions, for each user session, we obtain an n -gram vector of size k^n . Let us denote the new feature matrix as $Y^t = \{y_1^t, \dots, y_{n_s^t}^t\}$, where n_s^t is the number of different sessions at time window t .

We apply the same partition algorithm to new feature vectors in order to obtain K session centroids. As previously, for each resulting partition P_i^t , in addition to its centroid $M_i^t = m(P_i^t)$, we store in memory the number of feature vectors associated with this centroid:

$$w(P_i^t) = |P_i^t|. \quad (15)$$

However, instead of just sums of squared features in vectors assigned to a centroid, we calculate and store matrix $S(P_i^t)$ of size $k^n \times k^n$, the (j, l) -th element $S_{jl}(P_i^t)$ of which is calculated as follows:

$$S_{jl}(P_i^t) = \sum_{y \in P_i^t} y_j y_l, \quad (16)$$

where $j, l \in \{1, \dots, k^n\}$.

Once connection partitions for τ consecutive time windows $t \in \{1, 2, \dots, \tau\}$ have been calculated and $\tau \times k$ resulting partitions p_i^t have been compressed to new k connection clusters \bar{p}_i^τ , information stored for each session partition P_i^t is supposed to be updated. It is caused by the fact that connection clusters have been changed which leads to the modifications in all the n -gram vectors. For this purpose, we introduce function $f(j, p_i^t, \bar{p}_i^\tau)$ with $j \in \{1, \dots, k^n\}$, such that if the j -th n -gram contains label l and partition p_i^t is assigned to new partition \bar{p}_i^τ , the function returns the index that corresponds to the n -gram which is obtained from the j -th gram by substituting label l with label i .

Let us assume that conversation partition p_i^τ contains some of the partitions obtained at time window τ :

$$p_{i_1}^\tau, \dots, p_{i_q}^\tau \in \bar{p}_i^\tau. \quad (17)$$

It is worth noting that partition \bar{p}_i^τ can also contain partitions from other time windows, but they do not affect vectors Y^t at this point.

If the j -th gram contains label i , the j -th component of the i -th session centroid M_i^t is modified as follows:

$$M_{ij}^t = \sum_{a=1}^q M_{i, f(j, p_{i_a}^t, \bar{p}_i^\tau)}^t. \quad (18)$$

Similarly, elements of matrix $S(P_i^t)$ are modified:

$$S_{jl}(P_i^t) = \sum_{a=1}^q S_{f(j, p_{i_a}^t, \bar{p}_i^\tau), l}(P_i^t), \quad l \in \{1, \dots, k^n\}, \quad (19)$$

$$S_{lj}(P_i^t) = \sum_{a=1}^q S_{l, f(j, p_{i_a}^t, \bar{p}_i^\tau)}(P_i^t), \quad l \in \{1, \dots, k^n\}.$$

If the j -th gram contains label $i_a \in \{1, \dots, i_q\}$ and does not contain label i , the j -th component of the i -th session centroid M_i^t and elements of matrix $S(P_i^t)$ become equal to zero:

$$M_{ij}^t = 0 \quad (20)$$

and

$$S_{jl}(P_i^t) = 0, l \in \{1, \dots, k^n\}, \quad (21)$$

$$S_{lj}(P_i^t) = 0, l \in \{1, \dots, k^n\}.$$

The rest of the components do not change. Thus, we update information about session partitions to represent modifications in n -grams caused by the compression of connection clusters.

Once session partitions P_i^t , where $t \in \{1, \dots, \tau\}$ and $i \in \{1, \dots, K\}$, have been updated, these $\tau \times K$ partitions are compressed to new K clusters with the same technique as the one was applied for connections. For each resulting partition \bar{P}_i^τ , we store in memory its centroid

$$m(\bar{P}_i^\tau) = \bar{M}_i^\tau, \quad (22)$$

the number of feature vectors associated with this centroid

$$w(\bar{P}_i^r) = \sum_{m(x) \in \bar{P}_i^r} w(x), \quad (23)$$

and matrix $S(\bar{P}_i^r)$, the (j, l) -th elements $S_{jl}(\bar{P}_i^r)$ of which is defined as follows:

$$S_{jl}(\bar{P}_i^r) = \sum_{m(x) \in \bar{P}_i^r} S_{jl}(x). \quad (24)$$

6.3 Attack detection

The final model of normal user behavior consists of minimal $x_{min,j}$ and maximal $x_{max,j}$ feature values, centroids $\mu_i = m(p_i)$, numbers of vectors assigned $w_i = w(P_i)$ and sums of squared feature values $\zeta_i = \zeta(P_i)$ for k connection partitions p_1, \dots, p_k and centroids $M_i = m(P_i)$, numbers of vectors assigned $W_i = w(P_i)$ and matrices $S_i = S(P_i)$ for K session partitions P_1, \dots, P_K .

First, we recalculate conversation centroids and sums of squared feature values according to values $x_{min,j}$ and $x_{max,j}$ used for the standardization:

$$m_{ij} = \frac{\mu_{ij} - x_{min,j}}{x_{max,j} - x_{min,j}}, \quad (25)$$

$$S_{ij} = \frac{\zeta_{ij} + w_i(x_{min,j}^2 - 2x_{min,j}\mu_{ij})}{(x_{max,j} - x_{min,j})^2},$$

where $i \in \{1, \dots, k\}$.

For each partition p_i we calculate radius r_i and diameter ψ_i in a similar manner they are calculated for cluster features in [26]:

$$r_i = \sqrt{\frac{e^T s_i}{w_i} - m_i^T m_i}, \quad (26)$$

$$\psi_i = \sqrt{\frac{2w_i e^T s_i - 2w_i^2 m_i^T m_i}{w_i(w_i - 1)}},$$

where e is vector of the same length as s_i each element of which is equal to 1.

Similarly, for each partition P_i we calculate radius R_i and diameter Ψ_i as follows:

$$R_i = \sqrt{\frac{E^T S_i^{diag}}{W_i} - M_i^T M_i}, \quad (27)$$

$$\Psi_i = \sqrt{\frac{2W_i E^T S_i^{diag} - 2W_i^2 M_i^T M_i}{W_i(W_i - 1)}},$$

where S_i^{diag} is the vector which consists of diagonal elements of matrix S_i and E is vector of the same length as S_i^{diag} each element of which is equal to 1.

If the distance d between standardized feature vector x extracted from a new connection and the closest centroid $m_{i(x)}$ is greater than the following linear combination of $r_{i(x)}$ and $\psi_{i(x)}$:

$$d(x, m_{i(x)}) > r_{i(x)} + \alpha \psi_{i(x)}, \quad (28)$$

then this connection is classified as an attack.

Similarly, if the distance d between feature vector y extracted from a new session and the closest centroid $M_{i(y)}$ is such that:

$$d(y, M_{i(y)}) > R_{i(y)} + \beta \Psi_{i(y)}, \quad (29)$$

then this user session is classified as an attack. Parameters $\alpha > 0$ and $\beta > 0$ are supposed to be tuned during the model validation stage in order to guarantee the highest detection accuracy.

7 Algorithm evaluation

In order to evaluate the detection approach proposed, first, we briefly overview our virtual network environment used to generate some realistic end user network traffic and various DDoS attacks. Then, we evaluate the cost of the clustering on the training data. Finally, we present results of the detection of three different DDoS attacks.

7.1 Test environment and data set

To test the DDoS detection scheme proposed in this study, a simple virtual network environment is designed. The environment botnet consists of a command and control (C2) center, a web server, and several virtual bots. C2 stores all necessary information about bots in a data base and allows to control bots by specifying the traffic type, the pause between two adjacent sessions and the delay between connections in one session. The web server has several services including a web bank website, file storage, video streaming service and few others. Each bot is a virtual machine with running a special program implemented in Java that allows the bot to receive commands from C2 and generate some traffic to the web server. It is worth noting that all the traffic is transferred by using encrypted SSL/TLS protocol.

In this research, we concentrate on the analysis of incoming and outgoing traffic of the bank web site that allows a client to log in and do some bank operations, see Figure 1. In order to generate a normal bank user traffic, we specify several scenarios that each bot follows when using the site. Each scenario consists of several actions following each other. These actions include logging in to the system by using the corresponding user account, checking the account balance, transferring some money to another account, checking the result of the transaction, logging out of the system, and some other actions. Each action corresponds to requesting a certain page of the bank service with all of its embedded content. Pauses between two adjacent actions are selected in a way similar to a human user behavior. For example, checking the account's balance usually takes only a couple of seconds, whereas filling in information to transfer money to another account may take much longer time.

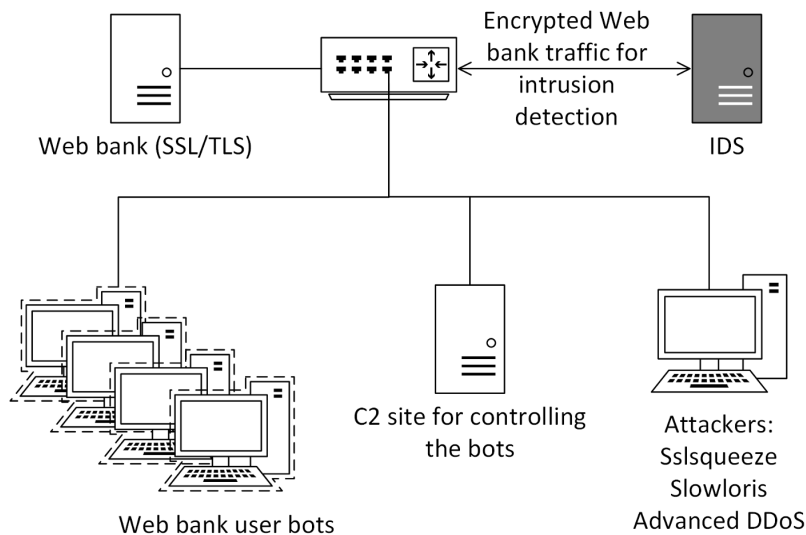


Figure 1. Test setup

In addition to the normal traffic, several attacks are performed against the bank web service. First DDoS attack tested is Sslsqueeze. During this attack, attackers send some bogus data to the server instead of encrypted client key exchange messages. By the moment the server completes the decryption of the bogus data and understands that the data decrypted is not a valid key exchange message, the purpose of overloading the server with the cryptographically expensive decryption has been already achieved.

The second attack is Slowloris. In the case of this attack, each attacker tries to hold its connections with the server open as long as possible by periodically sending subsequent HTTP headers, adding to-but never completing-the requests. As a result, the web server keeps these connections open, filling its maximum concurrent connection pool, eventually denying additional connection attempts from clients. Moreover, we carry out a more advanced DDoS attack with the attackers that try to mimic the browsing behavior of a regular human user. During this attack several bots request sequences of web pages with all of their embedded content from the service. Unlike the normal user behavior, these sequences are not related to each other by any logic but generated randomly.

Finally, an intrusion detection system (IDS) prototype that relies on the proposed technique is implemented in Python. The resulting program analyzes arriving raw packets, combines them to conversations, extracts necessary features from them, implants the resulting feature vectors to the model in the training mode and classifies those vectors in the detection mode. The IDS is trained with the traffic that does not contain attacks by using the online training algorithm proposed. After that, the system tries to find conversations and clients that are related to the attacks specified above.

The resulting program analyses arriving raw packets, combines them to conversations, extracts necessary features from them, implants the resulting feature vectors to the model in the training mode (Figure 2) and classifies those vectors in the detection mode (Figure 3).

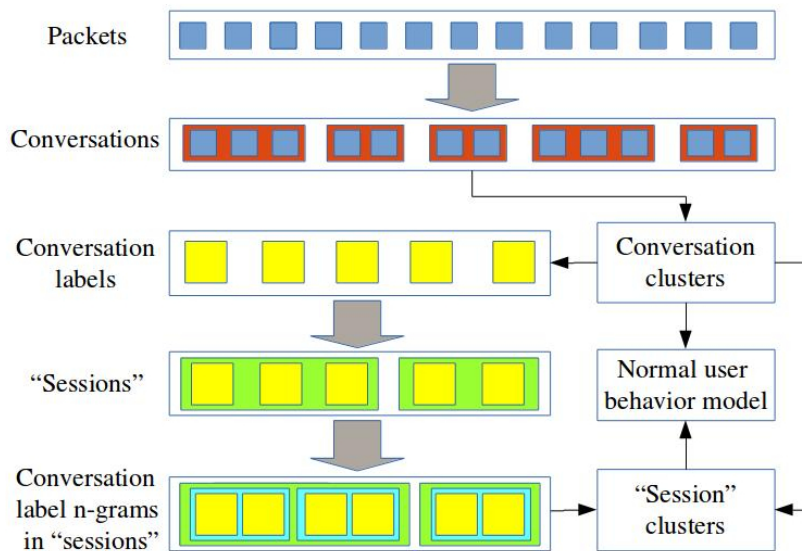


Figure 2. IDS in the training mode

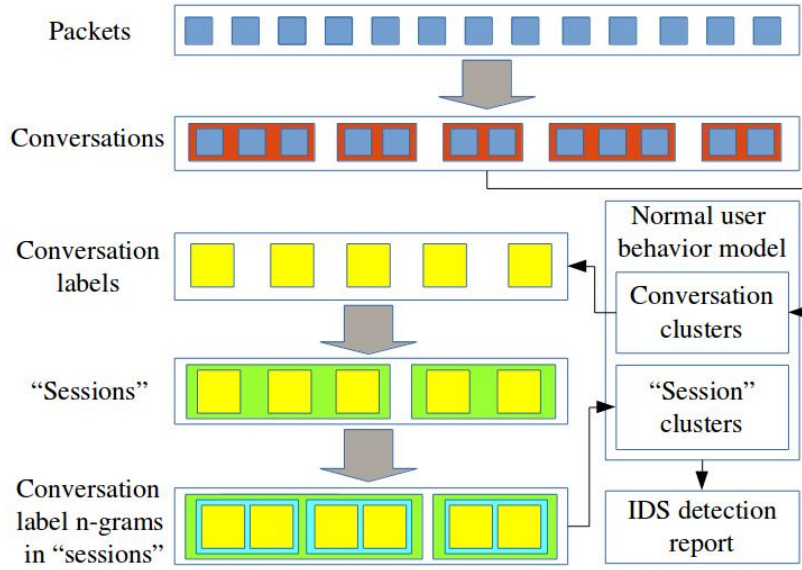


Figure 3. IDS in the detection mode

7.2 Clustering evaluation

In order to evaluate the clustering scheme, we generate a portion of normal user traffic as described above. The duration of the simulation is 10 minutes, during which 45 bots communicate with the web bank server. The pause between two adjacent sessions varies from 15 to 45 seconds. To train the system we use the time window of length 5 seconds. Cluster centroids in the model are compressed every 10 time windows.

In order to evaluate the clustering scheme, we calculate the average cost C of dividing the resulting set of vectors X into clusters with centers m as follows:

$$C = \frac{1}{|X|} \sum_{x \in X} \min_{y \in m} d(x, y). \quad (30)$$

First, we compare the cost of clustering vectors that represent conversations by offline and online approaches. The comparison results for different numbers of clusters are presented in Figure 4. As one can see, the cost of clustering these vectors online is comparable with the cost in the offline case and the difference between costs reduces when the number of clusters grows. When the number of clusters is equal or greater than 15, the difference between costs is only about 1.4%.

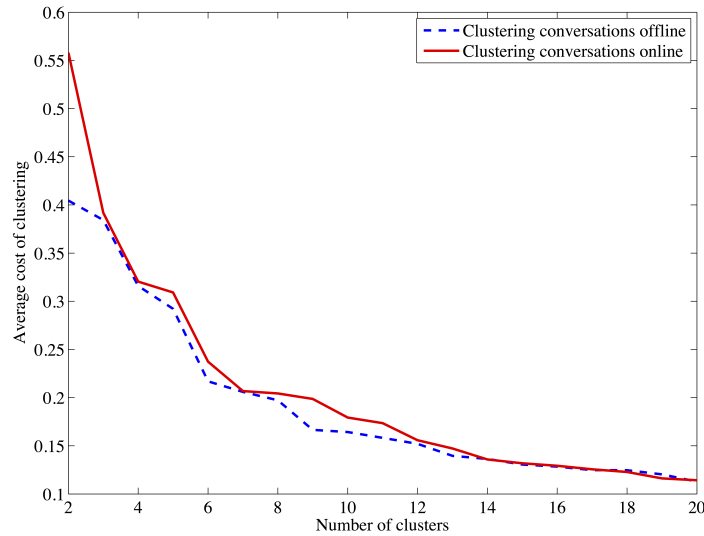


Figure 4. Average cost of clustering conversations

Similar results are obtained for n -gram vectors that represent normal user session approximations. It is worth noticing that, for session clusters built in the online mode, n -grams are constructed based on the clustering conversation vectors in online mode, and correspondingly, for session clusters obtained in offline mode, n -grams are constructed based on the clustering conversation vectors in offline mode. The comparison results for different numbers of clusters are presented in Figure 5. As one can notice, the cost of clustering in the online case is again comparable with the offline approach.

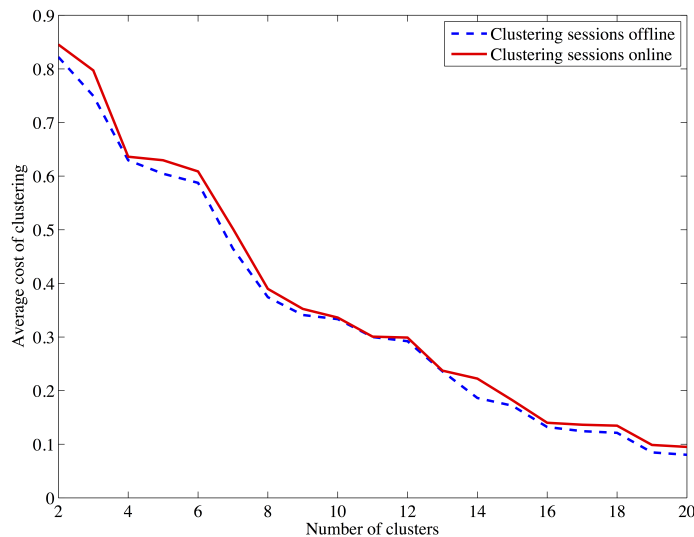


Figure 5. Average cost of clustering "sessions"

7.3 Detection accuracy

In order to evaluate attack detection accuracy, we employ the training dataset described above. The IDS is trained with the traffic that does not contain attacks by using the training algorithm proposed. Once the training has been completed, several attacks are performed to evaluate true positive rate

(TPR), false positive rate (FPR) and accuracy of the algorithms. Since one of the most important drawbacks of an anomaly-based detection system is high numbers of false alarms, we are only interested in results when the false positive rate is below 1%.

We expect that trivial DDoS attacks such as Sslsqueeze and Slowloris can be detected by analyzing feature vectors extracted from conversations. Since almost every conversation between a client and the web server takes few milliseconds, the time window size is selected to be 1 second. Since, we could not find any other anomaly-based attack detection method for the encrypted network traffic, we compare our results with the offline version of the algorithm proposed in this study. In [17], we compared several well-known batch clustering algorithms for the problem of trivial DDoS attack detection. It turned out, that such algorithms as k-means, self-organizing maps and fuzzy c-means allow one to detect slow HTTP attacks with high accuracy. In this study, we use k-means combined with k-means++ as the offline version of our approach.

Figure 6 shows how TPR depends on FPR when detecting Sslsqueeze for different numbers of conversation clusters in the model of normal user behavior and different values of parameter α . As one can notice, for low values of FPR both variants of the algorithm are able to detect almost all conversations related to the attack. In particular, we are able to detect 98.5% of intrusive conversations with no false alarms and 99.9% of such conversations with FPR equal to 0.8%.

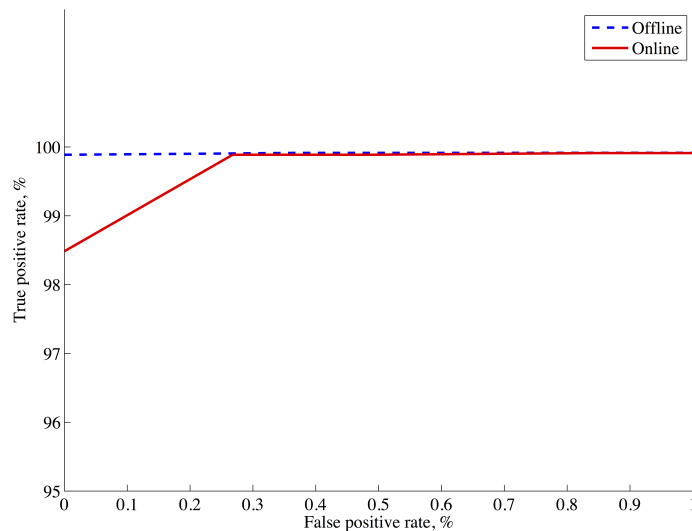


Figure 6. Dependence of true positive rate on false positive rate of Sslsqueeze detection for different clustering parameters

Dependency of TPR on FPR when detecting Slowloris is presented in Figure 7. As previously, different numbers of conversation clusters in the model of normal user behavior and different values of parameter α are used to obtain the results. As one can see, both variants of the algorithm are able to detect almost all conversations related to the attack (TPR > 99%) without false alarms.

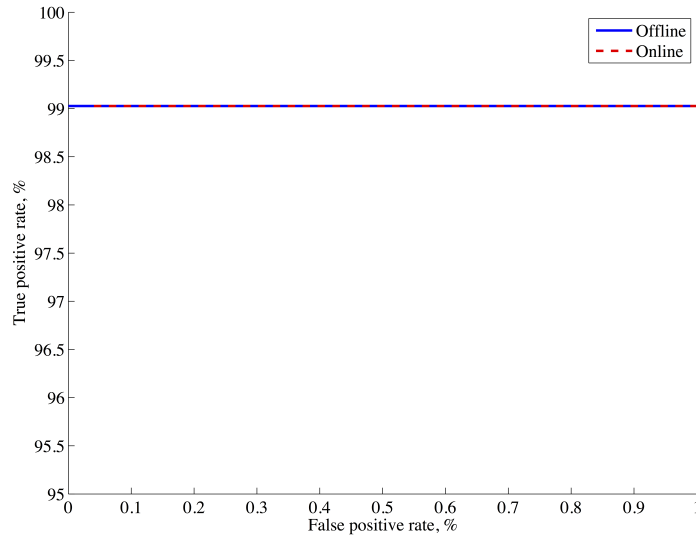


Figure 7. Dependence of true positive rate on false positive rate of Slowloris detection for different clustering parameters

Finally, we carry out a more advanced DDoS attack with the attackers that try to mimic the browsing behavior of a regular human user. During this attack several bots request a random sequence of web resources from the server. Since all those requests are legitimate, the corresponding conversations are classified as normal. However, the analysis of n -gram vectors that represent approximations of user sessions allow us to detect the most part of the attacking attempts. In this simulation, 2-gram model is applied. Figure 8 shows how TPR depends on FPR for different values of the time window size. When the time window duration is only 1 second, there is no possibility to detect the attack since the number of conversations in all user sessions is not enough to distinguish a legitimate user's sessions from attacker's ones. However, when the size of the time window grows, normal user sessions become more and more distinguishable from the sessions related to the attack. As one can see, when the time window size is 4 seconds, almost all intrusive sessions can be detected (TPR = 97%) without false alarms. When the time window size is 8 seconds, all sessions related to the attack can be detected (TPR = 100%) with few false alarms (FPR = 0.93%).

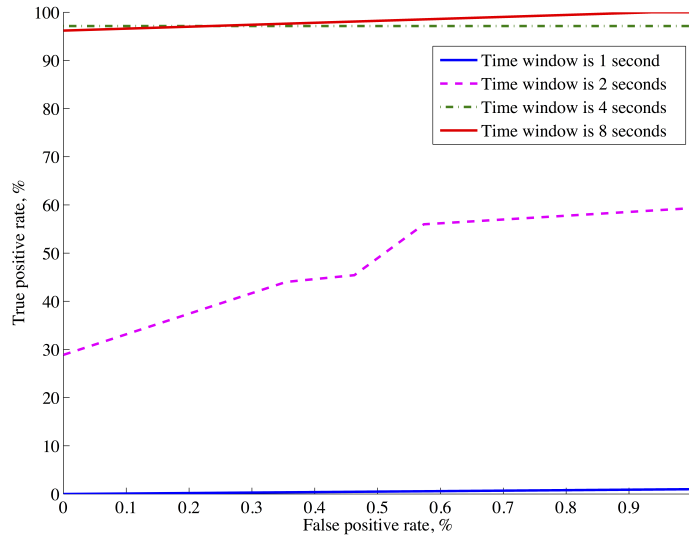


Figure 8. Dependence of true positive rate on false positive rate of more advanced DDoS attack detection for different time window sizes and clustering parameters

Table 1 shows the accuracy of detection of these three DDoS attacks for the cases when clustering parameters are selected in an optimal way, i.e. when the detection accuracy is maximal.

Table 1. Detection accuracy

Attack	Time window size			
	1 second	2 seconds	4 seconds	8 seconds
Sslsqueeze	99,89 %			
Slowloris	99,77 %			
Intermediate DDoS	63,29 %	85,21 %	98,52 %	99,34 %

7.4 Discussion

The detection approach proposed in this study allows us to detect trivial as well as more advanced DDoS attacks. However, the detection accuracy strongly depends on the clustering parameters selected for the model of normal user behavior. For this reason, these parameters should be carefully selected. For example, the number of conversation clusters should depend on the number of web pages of the web service considered. For completely different pages there should be different clusters whereas similar pages can be combined to the same cluster. Moreover, since clustering vectors that represent user sessions requires analysis of matrices of size $k^n \times k^n$, where k is the number of conversation clusters and n is the number of grams in the n -gram model, value of k should be limited depending on available memory. For the same reason, number of grams in the n -gram model should be small. As we can see from the results presented above, 2-gram model allows us to detect almost all intrusions. The number of "session" clusters should depend on the number of different scenarios how the web service can be used by some user. Parameters α and β are recommended to lie between 0 and 1. If these parameters are close to zero, the number of true positives is the highest. When these parameters grow, the number of false alarms decreases. Finally, the size of time window should depend on the average duration of a normal user session in order to increase the accuracy of the detection of intermediate DDoS attacks.

It is also worth to notice, that despite our approach allows us to distinguish attacks from the normal user traffic with high accuracy, the traffic we label as normal would be classified as an advanced DDoS

attack by [10], because this traffic consists of sequences of HTTPS requests which are carefully chosen so as to better mimic the browsing behavior of regular human users. For this reason, in the future, we are going to test our method on some real end user traffic.

8 Conclusion

In this research, we considered the problem of timely detection of different sorts of application-layer DDoS attacks in encrypted high-speed network traffic by applying an anomaly detection-based approach to statistics extracted from network packets. Our method relies on the construction of the normal user behavior model by applying a data stream clustering algorithm. The online training scheme proposed allows one to rebuild this model every time when a new portion of network traffic is available for the analysis. Moreover, it does not require a lot of computing and memory resources to be able to work even in the case of high-speed networks. In addition, an IDS prototype that relies on the proposed technique was implemented. This prototype was used to test our technique on the data obtained with the help of our virtual network environment that generated realistic traffic patterns of end users. As a result, almost all DDoS attacks were properly detected, while the number of false alarms remained very low.

In the future, we are planning to improve the algorithm in terms of the detection accuracy, and test it with a bigger dataset which contains real end user traffic captured during several days. In addition, we will focus on the simulation of more advanced DDoS attacks and detection of these attacks by applying our anomaly-based approach.

9 Acknowledgment

This work was partially funded by the Regional Council of Central Finland/Council of Tampere Region and European Regional Development Fund/Leverage from the EU 2014-2020 as part of the JYVSECTEC Center project of JAMK University of Applied Sciences Institute of Information Technology.

10 References

- [1] Kaspersky Lab, “Statistics on botnet-assisted DDoS attacks in Q1 2015”, 2015, <https://securelist.com/blog/research/70071/statistics-on-botnet-assisted-ddos-attacks-in-q1-2015/>, accessed 26.8.2016.
- [2] Radware, “2015-2016 Global Application & Network Security Report”, 2016, https://www.radware.com/ert-report-2015/?utm_source=security_radware&utm_medium=promo&utm_campaign=2015-ERT-Report, accessed 26.8.2016.
- [3] Verisign, “Distributed Denial of Service Trends Report”, 2016, https://www.verisign.com/en_US/security-services/ddos-protection/ddos-report/index.xhtml, accessed 26.8.2016.
- [4] V. Durcekova, L. Schwartz, and N. Shahmehri, “Sophisticated Denial of Service Attacks Aimed at Application Layer”, In Proceedings of the 9th International Conference ELEKTRO, pp 55–60, 2012.
- [5] R. Chen, J. Wei and H. Yu, “An improved Grey Self-Organizing Map based DOS detection”, 2008 IEEE Conference on Cybernetics and Intelligent Systems, Chengdu, 2008, pp. 497-502.
- [6] Y. Ke-xin and Z. Jian-qi, “A Novel DoS Detection Mechanism”, In Proceedings of International Conference on Mechatronic Science, Electric Engineering and Computer (MEC), pp 296–298, 2011.
- [7] J. Yuan and K. Mills, “Monitoring the Macroscopic Effect of DDoS Flooding Attacks”, IEEE Tran. on Dependable and Secure Computing, vol. 2, pp 324–335, 2005.
- [8] M. Aiello, E. Cambiaso, M. Mongelli, and G. Papaleo, “An On-Line Intrusion Detection Approach to Identify Low-Rate DoS Attacks”, In Proceedings of International Carnahan Conference on Security Technology (ICCST), pp 1–6, 2014.

- [9] P. Chwalinski, R. R. Belavkin, and X. Cheng, "Detection of Application Layer DDoS Attacks with Clustering and Bayes Factors", In Proceedings of IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp 156–161, 2013.
- [10] D. Stevanovic and N. Vljajic, "Next Generation Application-Layer DDoS Defences: Applying the Concepts of Outlier Detection in Data Streams with Concept Drift", In Proceedings of the 13th International Conference on Machine Learning and Applications, pp 456–462, 2014.
- [11] C. Xu, G. Zhao, G. Xie, and S. Yu, "Detection on Application Layer DDoS using Random Walk Model", In Proceedings of IEEE International Conference on Communications (ICC), pp 707–712, 2014.
- [12] S. Yadav and S. Selvakumar, "Detection of Application Layer DDoS Attack by Modeling User Behavior Using Logistic Regression", In Proceedings of the 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions), pp 1–6, 2015.
- [13] S. Yadav and S. Selvakumar. "Detection of Application Layer DDoS Attack by Feature Learning Using Stacked Autoencoder", In Proceedings of International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT), pp 261–266, 2016.
- [14] C. Ye, K. Zheng, and C. She, "Application layer DDoS detection using clustering analysis", In Proceedings of the 2nd International Conference on Computer Science and Network Technology (ICCSNT), pp 1038–1041, 2012.
- [15] J. Zhang, Z. Qin, L. Ou, P. Jiang, J. Liu, and A. Liu, "An Advanced Entropy-Based DDOS Detection Scheme", In Proceedings of International Conference on Information Networking and Automation (ICINA), pp 67–71, 2010.
- [16] M. Zolotukhin, T. Hämäläinen, T. Kokkonen, A. Niemelä, and J. Siltanen. Data mining approach for detection of ddos attacks utilizing SSL/TLS protocol. Springer Lecture Notes in Computer Science, vol. 9247, pp 274–285, 2015.
- [17] M. Zolotukhin, T. Hämäläinen, T. Kokkonen, and J. Siltanen. Increasing web service availability by detecting application-layer ddos attacks in encrypted traffic. In Proceedings of the 23rd International Conference on Telecommunications (ICT), pp 1–6, 2016.
- [18] M. Trojnara, "Sslsqueeze 1.0", SSL service load generator proof of concept, 2011, <http://pastebin.com/AgLQzL6L>, accessed 26.8.2016.
- [19] Gartner, "Are Cybercriminals hiding in SSL traffic", White paper sponsored by venafi, 2015, <https://citrixready.citrix.com/content/dam/ready/partners/ve/venafi/venafi-trustforce/Gartner%20-%20202015%20-%20Cybercriminals%20Hiding%20in%20SSL%20Traffic.pdf>, accessed 26.8.2016.
- [20] W. Bulajoul, A. James, and M. Pannu, "Network intrusion detection systems in high-speed traffic in computer networks", In Proceedings of the 10th IEEE International Conference on Business Engineering (ICEBE), pp 168–175, 2013.
- [21] K. Li, W. Zhou, P. Li, J. Hai, and J. Liu, "Distinguishing DDoS Attacks from Flash Crowds Using Probability Metrics", In Proceedings of the 3rd International Conference on Network and System Security (NSS), pp 9–17, 2009.
- [22] U. Chaudhary, I. Papapanagiotou, and M. Devetsikiotis, "Flow Classification Using Clustering And Association Rule Mining". In Proceedings of the 15th IEEE International Workshop on Computer Aided Modeling, Analysis and Design of Communication Links and Networks (CAMAD), pp 76–80, 2010.
- [23] D. Arndt and A. N. Zincir-Heywood, "A Comparison of Three Machine Learning Techniques for Encrypted Network Traffic Analysis", In Proceedings of IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), pp 107–114, 2011.
- [24] R. Chitta, R. Jin, and A. Jain, "Stream Clustering: Efficient Kernel-based Approximation using Importance Sampling", In Proceedings of IEEE International Conference on Data Mining Workshop (ICDMW), pp 607–614, 2015.
- [25] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. de Carvalho, and J. Gama, "Data Stream Clustering: A Survey", ACM Computing Surveys, vol. 46, article 13, pp 13:1–13:31, 2013.
- [26] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: A New Data Clustering Algorithm and Its Applications", Data Mining and Knowledge Discovery, vol. 1, no. 2, pp 141–182, 1997.

- [27] C. Aggarwal, J. Han, J. Wang, and P. Yu, “A framework for clustering evolving data streams”, In Proceedings of Conference on Very Large Data Bases (VLDB03), vol. 29, pp 81–92, 2003.
- [28] P. Kranen, I. Assent, C. Baldauf, and T. Seidl, “The ClusTree: indexing micro-clusters for anytime stream mining”, *Knowl. Inf. Syst.*, vol. 29, issue 2, pp 249–272, 2011.
- [29] M. Ackermann, M. Märtens, C. Raupach, K. Swierkot, C. Lammersen, and C. Sohler. “StreamKM++: A clustering algorithm for data streams”, *ACM J. Exp. Algor.* vol. 17, no. 2, article 2.4, 30 pages, May 2012.
- [30] Y. Chen and L. Tu, “Density-Based Clustering for Real-Time Stream Data”, In Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 133–142, 2007.
- [31] J. Gama, P. Rodrigues, and L. Lopes, “Clustering distributed sensor data streams using local processing and reduced communication”, *Intell. Data Anal.*, vol. 15, issue 1 pp 3–28, 2011.
- [32] M. Shindler, A. Wong, and A. W. Meyerson, “Fast and Accurate k-means For Large Datasets”, In Proceedings of the Conference on Neural Information Processing Systems, pp 2375–2383, 2011.
- [33] T. Hirsimäki, J. Pylkkönen, and M. Kurimo, “Importance of High-Order N-Gram Models in Morph-Based Speech Recognition”, in *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, no. 4, pp. 724-732, May 2009.
- [34] C. Y. Suen, “n-Gram Statistics for Natural Language Understanding and Text Processing”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 164-172, April 1979.
- [35] P. Domingos and G. Hulten. A general method for scaling up machine learning algorithms and its application to clustering. In Proceedings of the 8th International Conference on Machine Learning, pp 106–113, 2001.
- [36] R. Shah, S. Krishnaswamy, and M. Gaber. Resource-aware very fast k-means for ubiquitous data stream mining. In Proceedings of the 2nd International Workshop on Knowledge Discovery in Data Streams, Held in Conjunction with the 16th European Conference on Machine Learning, 2005.
- [37] D. Arthur and S. Vassilvitskii, K-means++: The Advantages of Careful Seeding, Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms. pp 1027-1035, 2007.
- [38] V. Braverman, A. Meyerson, R. Ostrovsky, A. Roytman, M. Shindler, and B. Tagiku. Streaming k-means on well-clusterable data. In Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, pp 26–40, 2011.