Riku Petteri Lehkonen

Improving backdoor remote communication detection with covert channel feature analysis

Master's Thesis in Information Technology

May 16, 2024

University of Jyväskylä

Faculty of Information Technology

Author: Riku Petteri Lehkonen

Contact information: ripelehk@student.jyu.fi

Supervisors: Tapio Frantti, Tommi Mikkonen, and Reijo Savola

Title: Improving backdoor remote communication detection with covert channel feature analysis

Työn nimi: Takaovien etäviestinnän havaitsemisen parantaminen salaisen kanavan ominaisuuksien analyysin avulla

Project: Master's Thesis

Study line: Software and Telecommunication Technology

Page count: 80+0

Abstract: This research introduces a novel framework, which supports the detection of covert timing channels by detailing generic covert timing channel features. The framework is developed using the design science research methodology and has been validated by generating and detecting a simple covert timing channel. Furthermore, the study reviews the existing landscape of covert channels and determines that majority of research on covert channel detection is very protocol and environment dependant, thus there is lack of flexibility for them to function as a generic detection technique. The absence of effective generic detection techniques makes it challenging to implement covert timing channel detection in practice, such as for backdoor detection. The research also highlights the insufficiency in current covert channel detection research, since it is not approached holistically through all the system elements. The resulting framework offers features for generic covert timing channel detection developers, which may be used for classifiers. Enhancing the development of detectors for defense evasion techniques supports in improving backdoor remote communication detection.

The research was supported by Business Finland (grant number 10/31/2022) and the University of Jyväkylä.

Keywords: backdoor, cybersecurity, covert channel, persistent threat

Suomenkielinen tiivistelmä: Tässä tutkimuksessa esitellään uusi kehys, joka tukee ajoituspeitekanavien havaitsemista määrittelemällä niiden yleiset ominaisuudet. Kehys on kehitetty käyttäen suunnittelututkimus tutkimusmenetelmää, ja se on validoitu luomalla ja havaitsemalla yksinkertainen ajoituspeitekanava. Lisäksi tutkimuksessa tarkastellaan olemassa olevia peitekanavatutkimuksia ja todetaan, että suurin osa peitekanavien havaitsemista koskevista tutkimuksista on hyvin protokollasta ja ympäristöstä riippuvaisia, joten ne eivät ole riittävän joustavia toimimaan yleisinä havaitsemistekniikoina. Tehokkaiden yleisten havaitsemistekniikoiden puuttuminen tekee ajoituspeitekanavien havaitsemisen toteuttamisesta haastavaa käytännössä, kuten takaovien havaitsemiseksi. Tutkimuksessa korostetaan myös nykyisen peitekanavien havaitsemista koskevan tutkimuksen riittämättömyyttä, koska sitä ei lähestytä kokonaisvaltaisesti kaikkien järjestelmäelementtien kautta. Tutkimuksen tuloksena syntynyt kehys tarjoaa yleistettäviä ominaisuuksia ajoituspeitekanavien havaitsemisenmenetelmien kehittäjiä varten, joita voidaan käyttää luokittelijoissa. Puolustuksen välttämistekniikoiden havaitsemisen kehittämisen parantaminen tukee takaovien etäviestinnän havaitsemista.

Tutkimusta ovat tukeneet Business Finland (apuraha nro 10/31/2022) ja Jyväkylän yliopisto.

Avainsanat: takaovi, kyberturvallisuus, peitekanava, jatkuva uhka

List of Figures

List of Tables

Contents

1 Introduction

In the context of cybersecurity, a backdoor refers to any attack that is used to bypass normal authentication to allow remote access (Jain and Parashu 2017). This can extend to individual hardware, operating system, software, and network, which are part of an ever larger portion of our every-day lives due to digitalisation (Yadav and Rao 2015). Bypassing authentication in this way leads to an access to sensitive information and exposes access for further possible damage in the system. In this sense, backdoors function as a foothold to the system for possible persistency and expansion, which is hard to detect and can appear on multiple levels.

Backdoors that are used for foothold often require some form of commanding or are used to exfiltrate information from the system. This type of communication is against the system's security policy and is restricted with wardens, such as firewalls in a networked environment. To bypass these restrictions, backdoor remote communication often use different information hiding techniques, where covert channels are the most prominent ones. Covert channels in a networked environment uses protocols, such as Hypertext Transfer Protocol (HTTP), Dynamic Name Server (DNS), Transmission Control Protocol (TCP) and Precision Time Protocol (PTP), to communicate between a sender and a receiver covertly. This type of communication is difficult to manually recognise and often requires machine learning solutions that are prone to overlearn a pattern and are often easily circumvented with a minor variation for the hiding method (Zillien and Wendzel 2023).

This covert elevated access leads to more widespread damages and is part of cyber attacks, which are becoming more prominent yearly. The average cost of a data breach in the United States in 2022 was 9.44 million USD, while a global one was less than half of that at 4.35 million USD ("Cost of a data breach 2022" 2022). Healthcare industry experiences especially high costs in damages due to the sensitivity of their data, and their costs have gone up by 42% since 2020 ("Cost of a data breach 2022" 2022). According to the Cybersecurity Ventures' article in 2020, cybercrime will cost the world 10.5 trillion USD annually by 2025, up from the previously reported 3 trillion USD in 2015 ("Cybercrime To Cost The World $10.5 Trillion Annually By 2025" 2020). To combat this, companies and countries need to increase their funding to protect themselves against these attacks, and Statista esti-

mates that worldwide information security services spending will increase to 76 billion USD by 2023 (”Worldwide information security services spending from 2017 to 2023” 2022).

However, attacks targeting critical infrastructure of a country extends the threat beyond monetary value to the livelihoods and safety of the people. For example, Finland is considered the most digitalized country in EU based on Digital Economy and Society Index (DESI) 2022 report (”Finland in the Digital Economy and Society Index” 2022), which means successful attacks on its digitalised infrastructure will lead to a greater negative impact to citizens lives and society's function. Improving backdoor detection capabilities would increase trust and resilience of a digitalized society, while also reducing monetary and trust related costs.

While backdoors are difficult to detect, they have a common factor in networked environments for using remote access to communicate with the malicious attacker. These backdoors often implement defense evasion techniques, such as covert channels, to avoid the detection of occurring communication. Proposed solution focuses on improving the detection of possible covert communication through feature analysis. Due to the importance of digitalised critical infrastructure, detection improvements focus on covert timing channel communication, since they are more protocol independent than storage channels and therefore are usable in both IT and OT environments. This is crucial, since there is currently ongoing IT/OT convergence in critical infrastructure.

The goal of this study is therefore to improve the detection of backdoors through improvements in covert channel communication detection. This is approached through two research questions: (1) Can the initial foothold's backdoor communication detection be improved? and (2) Can the detection of defense evasion techniques employed by backdoors be improved through generic covert timing channel feature analysis?

This thesis is split into two portions: the literature review and the constructive research. The constructive portion is conducted by creating a design science artifact. For the literature review, the following initial databases are used: ACM Digital Library, Google Scholar, IEEE Xplore, Scopus, and JYKDOK, the electronic library of the university of Jyväskylä. To search for prior research, a number of combinations of keywords are used. The initial keywords used for the searches were “backdoor”, “cybersecurity”, “detection”, “persistent

threat", "covert channel" and "cyber kill chain". From these initial searches, a pool of possible articles was formed based on the contents of the abstracts, introductions, and conclusions of the articles.

Additionally, backward and forward search function of Google Scholar was used to search for more articles, which supplemented the initial pool. The articles used in the thesis are chosen based on their relevance to the subject, as well as where they were published, their publication year, and how often they have been referenced by others. In some cases, seminars, reports, and presentations were included based on the author's authority.

In chapter two, backdoors and remote communication are introduced and described through common attack model phases. Chapter three details how network covert channels are defined and their overlaps within information hiding, provides information on the history, definitions, taxonomy, and their limitations. Chapter four focuses on network inter-arrival time covert timing channels, with focus on the definition and some of the current existing detection methods. Chapter five provides information on how the constructive research was conducted. Chapter six contains more detailed information on how the design science artifact was created and evaluated. Results and discussion of the study are detailed in chapter seven. The last chapter is the conclusion, which is the chapter eight.

2 Remote Access and Backdoors in IT/OT environments

Operational Technology (OT) is critical to the function of various industries, as it is used from manufacturing to utilities. Similarly, countries' critical infrastructure are built upon OT systems, which raises the importance of OT system cybersecurity. With the accelerating convergence between Information Technology (IT) and OT systems, there are new fundamental cybersecurity risks involved in OT environments that did not exist previously. Successful cyberattack on an OT system can have varying consequences from equipment damage to production shutdowns, and even threats to human safety. (Dhirani, Armstrong, and Newe 2021) Therefore, securing critical infrastucture in particular is considered as a part of national security, and it is a point of focus within European Union in the upcoming NIS 2 directive. (Eckhardt and Kotovskaia 2023)

Backdoors can be defined as opening a possible interface into secured systems that bypasses the normal authentication process. While the backdoor itself can be benign, they are often seen as a vulnerability and as a feature that intentionally compromises a platform (Thomas and Francillon 2018). In similar fashion, remote access opens additional access points to the system that may be used remotely, but unlike backdoors, it accesses secured systems through normal authentication processes to certify the authority for the connection.

The combination of backdoors and remote access may create additional system vulnerabilities in the form of enabling remote connections without the necessary authority levels. Often times Advanced Persistent Threat (APT) attacks communicate with their established footholds through backdoors and remote connections to enable them to command their previously installed malware or exfiltrate data from the system. Defense mechanisms are continuously developed in an attempt to detect backdoors, but the attackers are developing their methods in parallel to circumvent the current detection methods.

In this chapter we will go over definitions for backdoors and remote access in the IT/OT environment, how these methods are used in APT attacks, and where in the attack senquence these methods may be used covertly in intrusion kill chains.

2.1 Remote Access and Control in IT/OT Environment

The COVID-19 pandemic caused many organizations to shift to remote work, which includes companies that rely on OT systems. This has lead to increased use of remote access pathways, which opens new vulnerabilities to attack vectors. Specifically, the risk of unauthorized access and data breaches have risen significantly, due to employees accessing systems remotely. (Peterson 2023) While securing remote access technologies is a strong focus in the industry, the highest occurring initial attack vector in their OT/control system incidents in 2021 was remote access (Bristow 2021).

Remote access is often discussed in the context of a networked environment, where the most common types of remote access include dial-up and wireless connections. Remote access is therefore an access to organizational systems or processes that are acting on behalf of users, which communicate through external networks such as the internet. (NIST 2020)

Organizational systems contain various levels of sensitive information, which should not be accessible by everyone. Organizations implement access controls, which are processes that accept or deny requests for access. This control may be implemented through a single or multiple access control mechanisms which authenticate the user, such as password-based authentication, possession-based authentication or biometric authentication. An authentication system that requires more than one distinct or unique authentication mechanism for successful authentication is called multifactor authentication (MFA) (NIST 2020).

Authenticated user is given a level of access to the system based on the organizations access control policy. Access control policies are high-level requirements that specify how access is managed, who may access information, and under what circumstances this information may be accessed (Hu, Kuhn, Yaga, et al. 2017). Access control mechanisms may implement multiple policies individually or simultaneously. A common security design used in conjuction with access control policies is Principle of Least Privilege (PoLP), where organizations grant users and processes only the rights necessary to accomplish their assigned tasks and responsibilities (NIST 2020). This design helps to prevent and audit system misuse, such as in the case of misused remote access connections.

2.2 Remote Backdoors

In cybersecurity a backdoor refers to a method that is used to bypass normal authentication process to gain access into a target system (Jain and Parashu 2017). Backdoors may be theoretically implemented at various levels into a system. This could be in the form of a dedicated program, as a hardware component or even embedded as part of another program. (Thomas and Francillon 2018) Therefore the embedding process may occur at various stages, such as during system's development, manufacturing or upkeep cycle, which means that the system is theoretically never completely safe from a backdoor attack.

The detection of the embedded backdoors is often performed by manually reverse engineering a program binary or by monitoring suspicious system events, such as abnormal system logs, network traffic logs, or host logs to notice the use of backdoors (Thomas and Francillon 2018). While backdoors are a known problem, detecting them is difficult due to dormant backdoors lacking activity for their detection, the possible obfuscation of genuinely malicious files enabling backdoors, and possible legitimate backdoors that may also be used illegitimately. These legitimate backdoors are often seen as a vulnerability, but they may exist as fail-safe mechanisms to implement required software configuration updates without explicit user authentication, or simply as accidental leftovers from their "debug" functionality during development cycle (Thomas and Francillon 2018).

Recent advancements in machine learning methodologies, such as the incorporation of deep neural networks (DNN) as classifiers in decision-making processes (such as processing anomalous logs), have unveiled a fresh target for backdoor attacks. Specifically, the training process and datasets used in machine learning methods. (Papernot et al. 2016; Li et al. 2022) The developed attacks are capable of corrupting machine learning training processes where the trained model will act normally on benign samples, but when backdoor is triggered by attacker-specified trigger patterns, it will have its predictions maliciously and consistently changed. In the case of data exfiltration, this could be used to hide the misuse of a remote connections and its normally detectable indicators of compromise (IoC) and indicators of attack (IoA). (Xiang, Miller, and Kesidis 2019) Therefore, the system that was developed to secure another system is instead repurposed as its vulnerability.

2.3 Advanced persistent threat and intrusion kill chains

Advanced Persistent Threat (APT) actors possess more sophisticated levels of expertise, significant resources and time to facilitate their attacks. Because of this, APT actors are sometimes used as an euphemism for state-sponsored espionage groups (Lemay et al. 2018). The targets of APT actors, unlike typical cyber-criminals, are not necessarily financially driven. Similar to nation-state/sponsored actors, the objectives often include establishing and extending footholds within IT or OT infrastructure of the targeted organizations for exfiltration of information, and undermining or impending critical aspects of system. Similarly, APT actors may simply want to remain hidden to act out these objectives at a more opportune moment in future. This creates an unique issue where a malware corrupted system might contain APT attack foothold in their backups, which prevents recovery from an occurring attack. APT actors often utilize multiple attack vectors, such as cyber and/or physical, and evolve their attack tactics during the stay in the system. This may include repositioning to continue carrying out the objectives, adapt to defenders attempts to resist it, and to maintain the level of interaction needed to carry out the objectives. APT attacks are therefore naturally resilient to detection and prevention, which makes it difficult for organizations to defend themselves against them. (NIST 2020; Bahrami et al. 2019)

To defend against APT attacks, their tactics, techniques and procedures (TTP) have been summarised into Intrusion Kill Chains that describe the attack lifecycle (Bahrami et al. 2019). Most prominently used models are Cyber-Kill-Chain (CKC) (Hutchins, Cloppert, Amin, et al. 2011) and MITRE ATT&CK (Strom et al. 2018), which have been extended to form more specific framework based intrusion kill chains, such as IoT-Kill-Chain (IoTKC) (Haseeb, Mansoori, and Welch 2020), ICS Cyber-Kill-Chain (Assante and Lee 2015) and ICS MITTRE ATT&CK (Alexander, Belisle, and Steele 2020) models or combined to cover more accurately all phases of the attack lifecycle with Unified-Kill-Chain (UKC) (Pols and Berg 2017). These models may be used to break down complex attacks into sequential stages, which helps analysts study and solve attacks stage-by-stage, and create possible mitigation strategies for each of the stages (Bahrami et al. 2019). Defenders aim to detect an occurring APT attack at the earliest stage possible to minimise the damage and to prevent a long-term foothold in the system. However, there is an issue with using attacker perspective models

while defending, since parts of the attack lifecycle occur outside of defenders territory and therefore are not detectable through traditional means. There is research towards addressing this issue, such as by Villalón-Huerta, Gisbert, and Ripoll-Ripoll (2022) where they have extended CKC for a defender's perspective based kill chain in the form of Security Operation Center (SOC) Critical Path, which is formed from the mandatory sequence of actions to detect and enable neutralization of a threat. However, cybersecurity defense industry have already adapted attacker perspective progress models in their defensive operations, and changing established processes might have too many overhead costs for them to be considered (Knerler, Parker, and Zimmerman 2022). Additionally, effective measurement of SOC performance is difficult, which increases the resistance security vendors may have towards implementing new practices at a larger scale, since the scalability and performance of the solution are similarly difficult to measure. This issue is exacerbated by the fact that many guidelines on SOCs are written by security vendors, which may lead to additional bias against external solutions. (Vielberth et al. 2020)

2.3.1 Cyber Kill Chain

Lockheed Martin Cyber Kill Chain is a seven phase intrusion kill chain, which is formed from reconnaissance, weaponization, delivery, exploitation, installation, command and control (C2), and actions on objective phases (Hutchins, Cloppert, Amin, et al. 2011). It is developed by Lockheed Martin Corporation in 2011, by adapting "kill chain" concept used in military environments that uses stages to outline the structure of an attack.

In *reconnaissance* the attacker gathers information about the potential target, such about an individual or an organization, which may be used to select a possible target and aid in later phases of the attack (Hutchins, Cloppert, Amin, et al. 2011). This may be in the form of target identification, selection or profiling, and is done through publicly available data, such as social networks, network tracing tools and mailing lists. Yadav and Rao (2015) note that reconnaissance may be done passively, where the defender cannot realistically perceive the information gathering process, since it occurs outside the controlled premise, or actively, where the profiling is far deeper and it comes into contact with the target's controlled perimiter. For example, reconnaissance with social networks is passive, whereas port scanning

the target system is active, which may be detected with network intrusion detection systems (NIDS).

Weaponization phase focuses on coupling malicious payload with deliverable file, such as coupling a remote access malware with Adobe Portable Document Format (PDF) (Hutchins, Cloppert, Amin, et al. 2011). This phase also includes designing a penetration plan, which is formed from the previous information gathered in the reconnaissance phase to enable delivery of the malware. Similarly, this process occurs outside of the defender's perimeter, since the payload is not delivered to the target.

Delivery is the first part of cyber kill chain where the attacker will leave traces for certain, which is why the attack is often performed with anonymous services or compromised websites and email accounts. In most cases this phase requires some form of user interactions like downloading and executing malicious files or visiting a malicious website. Zero-click attack is a sophisticated class of attacks, where a device can be compromised remotely without any user interaction. Most prominently known attack of this type of an attack is Pegasus spyware. (Younis et al. 2022) Recoinnassance phase enables target selection or motivation, which helps to entice the user interaction needed for delivery. Common delivery mechanisms are email attachments, phishing attacks, drive by download and USB/removal media. (Yadav and Rao 2015)

After delivery to victim host, *exploitation* triggers attacker's code, which often targets an application or operation system (OS) vulnerability. Exploit may also leverage an OS feature to auto-execute code or target the user themselves. (Hutchins, Cloppert, Amin, et al. 2011) Exploit's goal is to operate silently, since anti-viruses or similar security mechanisms may prevent its function (Yadav and Rao 2015).

Installation process creates a foothold from which to further advance the cyber attack from. This could be in the form of installing a remote access trojan or backdoor, which allows persistence in the system. (Hutchins, Cloppert, Amin, et al. 2011) APT attacks' installation process contains checks and resilience features to improve the success of installation and to protect the attackers with anonymous installation (Yadav and Rao 2015).

Command and control (C2) compromised hosts must usually beacon outside to receive fur-

ther instructions and commands. APT malware especially requires manual interaction rather than function automatically, which is done through internet controller servers to create C2 channels. (Hutchins, Cloppert, Amin, et al. 2011) These C2 channels are often obfuscated and hidden through techniques such as covert channels to prevent the communication traces occurring within the system. (Li, Song, and Yang 2022)

Actions on objectives is where intruder takes actions to achieve their initial objective, which is done in CKC after all the six previous phases (Hutchins, Cloppert, Amin, et al. 2011). This could be in the form of data exfiltration, where data is collected, encrypted and extracted; attacks on the data integrity or availability; or use the system as a hop point for further compromise and lateral movement (Yadav and Rao 2015).

2.3.2 Other intrusion chains and models

While CKC has remained a prominently known and used kill chain, it is critiqued to be perimeter and malware focused (Pols and Berg 2017). Malware-focused thinking is due to CKC considering that the attacker is an outsider, when in reality attack may occur through an insider. This leads to the attack bypassing certain steps in the kill chain, making defense built upon it insufficient at detecting the attack. (Liu et al. 2018) Another issue with a higher level adversary model, such as CKC, is that it is lacking in its description of actions that the APTs perform inside the network perimeter within a target organization (Strom et al. 2018). While the last four phases of CKC occur inside, there are actions on the objective that would require separation to individual phases, if the kill chain should be used in developing defensive actions effectively (Pols and Berg 2017).

These issues have lead to other intrusion kill chains and TTP models, from which the most known ones are MITRE ATT&CK and Unified Kill Chain. MITRE ATT&CK in particular is a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations. It functions as a mid-level adversary model which ties low level concepts such as exploits, vulnerability databases and models with high level models, such as CKC. Therefore it consolidates and provides a more complete description of what attacker does inside or while embedded in a computer network. (Strom et al. 2018) However, MITRE

ATT&CK does not function as a sequential chain and requires analysts to manually create the TTP chain by selecting occurred tactics and techniques from within the MITRE ATT&CK model (Pols and Berg 2017). Furthermore, Pols and Berg (2017) argue that the techniques cannot be assigned exclusively to specific tactics because they are frequently used by multiple tactics and across multiple phases of an attack chain, which increases the difficulty of creating TTP chains.

To address the issues with CKC and MITRE ATT&CK, Pols and Berg (2017) developed Unified Kill Chain based on previous TTP models and intrusion kill chains. UKC expands or separates the previously established phases into 18 consecutive phases as seen in Table 1. The expansion of attack phases include attack vectors and paths within the targeted organisation, thus avoiding CKC's issue with perimeter focus, in addition to expanding MITRE ATT&CK model by including phases such as pivoting and social engineering. Additionally, it may still be used to create attack and actor specific kill chains, but with lesser ambiguity of where an individual technique may land in tactics in comparison to MITRE ATT&CK model. The main advantage from UKC compared to CKC is the addition of cyclical nature and repetition occurring in APT attacks. CKC approaches attack phases as stages of consecutive actions that lead from Initial Foothold to Network Propagation, and lastly onto Actions on Objective, whereas UKC suggests that these actions function as iterative loops, that may eventually lead to the next stage. This behaviour more accurately describes the behaviour of the attacker and thus aids defensive plans as well. (Lehto 2022)

In this thesis the sequential phases of an APT attack exist to add context for when a covert channel may occur within an attack chain, and thus they are discussed through the high-level model of CKC. However, as CKC suffers from lack of internal network tactics, certain tactics from UKC will be referred when needed.

Table 1: The Unified Kill Chain by Pols and Berg (2017)

Phase	**Description**
Reconnaissance	Researching, identifying and selecting targets using active or passive reconnaissance.

Table 1: The Unified Kill Chain by Pols and Berg (2017)

Weaponization	Preparatory activities aimed at setting up the infrastructure required for the attack.
Delivery	Techniques resulting in the transmission of a weaponized object to the targeted environment.
Social Engineering	Techniques aimed at the manipulation of people to perform unsafe actions.
Exploitation	Techniques to exploit vulnerabilities in systems that may, amongst others, result in code execution.
Persistence	Any access, action or change to a system that gives an attacker persistent presence on the system.
Defense Evasion	Techniques an attacker may specifically use for evading detection or avoiding other defenses.
Command and Control	Techniques that allow attackers to communicate with controlled systems within a target network.
Pivoting	Tunneling traffic through a controlled system to other systems that are not directly accessible.
Discovery	Techniques that allow an attacker to gain knowledge about a system and its network environment.
Privilege Escalation	The result of techniques that provide an attacker with higher permissions on a system or network.
Execution	Techniques that result in execution of attacker-controlled code on a local or remote system.
Credential Access	Techniques resulting in the access of, or control over, system, service or domain credentials.
Lateral Movement	Techniques that enable an adversary to horizontally access and control other remote systems.
Collection	Techniques used to identify and gather information from a target network prior to exfiltration.

Table 1: The Unified Kill Chain by Pols and Berg (2017)

Exfiltration	Techniques that result or aid in an attacker removing files and information from a target network.
Target Manipulation	Techniques aimed at manipulation of the target system to achieve the objective of the attack.
Objectives	Socio-technical objectives of an attack that are intended to achieve a strategic goal.

2.4 Hiding APT network communication

One way that APT actors improve their resilience to remain and to establish a foothold in the target system is through backdoors. These backdoors then function as a foothold for command and control techniques that allow attackers to communicate with controlled systems within a target network (Pols and Berg 2017). Backdoors are by nature difficult to detect, but active backdoors can leave various tracks in the system that may be monitored to detect anomalous behaviour. In the case of remote accessed backdoors, these tracks are often detected from network communication, since they are most often performed through remote network access. Additionally, if the attacker has multiple footholds within the target system, communicating from foothold to foothold leaves additional tracks inside the target's protected network. Attacker is therefore incentivised to developed and implement methods to remain hidden and obscure footholds' communication. One method to do so is with network covert channels, where information transfer or communication is hidden within normal network traffic through steganography between two collaborating partners (Yadav and Rao 2015). A malware capable of using steganography, and thus any malware using covert channels, is called stegomalware (Caviglione 2021).

Network covert channels function primarily during CKC's C2 and Actions on Objectives phases, where installation of malware is already performed, and thus collaborating parties exist. Covert channels may also be utilised to propagate malware to an already infected system by transferring the malicious payload covertly, coordinating attacks within a target system through covert remote commands, and exfiltrate data from a highly secure section of

the system to a lower secure section or even outside of the target's internal network (Mazurczyk and Wendzel 2017; Li, Song, and Yang 2022). In the context of UKC, network covert channel may be utilised in defense evasion, command and control, pivoting, lateral movement and exfiltration. While covert channels function as a defense evasion tactic by hiding communication and information, it is often used in unison with other techniques in other tactics, which is similar to how content injection might occur during initial access and command and control.

3 Network Covert Channels

This chapter focuses on examining network covert channels, what are their distinguishing features and how are they categorised. To achieve this, it is paramount to also discuss the differing views on the terminology within information hiding and the topics surrounding covert channels. Through this discussion, the framing of network covert channels is possible without the commonly occurring uncertainty over terminology's synonyms and overlaps (Mazurczyk et al. 2016).

Information hiding (Petitcolas, Anderson, and Kuhn 1999) is the discipline that falls within *communications security* and it focuses on preventing the detection of transfer of information. Within the field of information hiding exists *covert channels* that are established between a covert sender and covert receiver to transfer information stealthily, alongside *steganography* and *anonymity* (Petitcolas, Anderson, and Kuhn 1999). Covert channels can be further divided into subtypes based on the environment they are develop in or the unifying pattern of the channel, such as Single-Host Covert Channels, Network Covert Channels, Subliminal Channels and Out-of-Band Covert Channels (Carrara and Adams 2016b). While in this thesis we focus on network covert channels, it is important to discuss the overlap of steganography, network covert channels and subliminal channels, since the terms are sometimes used incorrectly as synonyms (Carrara and Adams 2016a). The topics and categories within information hiding are highlighted in Figure 1.

Covert channels' definition is still continually revised and there are differing views on what is the exactly accurate definition for it, which is why it is important to define it from the beginning. The term *covert channel* was originally coined by Lampson (1973) as "communication channel that is not intended for information transfer at all". Kemmerer (1983) extended this later to define that covert channels arise from "the use of an entity not normally viewed as a data object to transfer information". While Lampson's definition appears in the context of a high security level process leaking information covertly to a lower security level process that is denied access to the information otherwise, his and Kemmerer's definition focus on the misuse of the channel or entity contrary to its original purpose, rather than the circumvention of security policies. These definitions therefore do not appropriately place

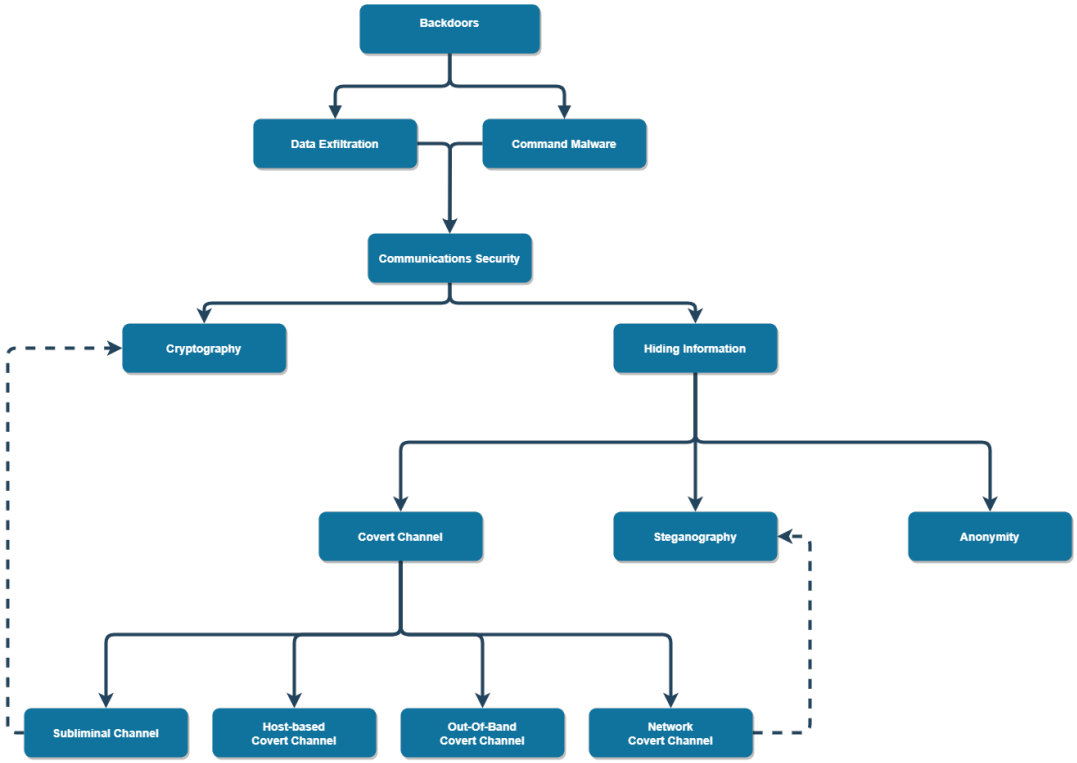


Figure 1: Covert channels in relation to surrounding research areas. Highlighting the subliminal channel's connection to cryptography and network covert channel's connection to steganography with dotted lines.

covert channels within communications security, which is noted in Trusted Computer System Evaluation Criteria (TCSEC) "Light Pink Book" (Gligor 1994). Girling (1987) defines covert channels as a "transfer of information in a way that would normally be contrary to a network's security policy", which would place it within communications security. However, Carrara and Adams (2016a) note that while this definition is applicable to networked environments, it is too narrowly focused on distributed systems to be generally accepted. In addition to this, it does not take into account intention in the transfer of the information.

The most commonly used definition was released by Trusted Computer System Evaluation Criteria (TCSEC) in their "Orange Book" where covert channels were defined as "any communication channel that can be exploited by a process to transfer information in a manner that violates the system's security policy" (Latham 1986), which was further established as the prominent definition through Gligor's TCSEC "Light Pink Book" (Gligor 1994). How-

ever, this would define all side-channels as covert channels, when side-channel research is focused on the unintentional leakage of information and only the receiver is interested in successful transmission (Carrara and Adams 2016a). Side-channels however may be utilised to create covert channels; the acoustic electronic switching noise in computers when varying its cooling fans or hard disk spinning speed could carry 1/0 information to an audio receiver (Shao, Islam, and Ren 2020).

Most recent noteworthy definition of covert channels is from National Insitute of Security and Standards (NIST) Special Publication (SP) 800-53 rev.5, where they define covert channels as "an unintended or unauthorized intra-system channel that enables two cooperating entities to transfer information in a way that violates the system's security policy but does not exceed the entities' access authorizations." (NIST 2020). Their definition builds upon the previous definitions to create a clear distinction between covert channels and the other terms surrounding it, such as steganography, subliminal channels and side-channels. It also defines that the privacy policy violation occurs within accepted access authorization. There is however a lack of definition for the scope of intra-system channels by NIST and therefore it is not completely certain whether their definition extends perfectly for out-of-band covert channels, which often times use covert receivers outside of the covert sender's normal internal system structure. Nevertheless, we will use NIST's definition of covert channels in this thesis due to it describing covert channels most accurately and its applicability to network covert channels.

Although the exact definition of covert channels is not necessarily completely agreed upon, the scenario describing it is. Simmons' prisoner's problem (Simmons 1984a) is set as a information hiding problem where two prisoners are cooperating in trying to devise an escape plan where they must communicate through a warden in written messages on paper. Warden allows the communication in hopes of finding out their plans through them inadvertently revealing it in their messages. Prisoners are called Alice and Bob who are sender and receiver respectively, while the warden is called Wendy. Simmons' prisoner's problem is visualised in Figure 2. The message, such as the scenario's piece of paper with messages, is described as an *overt channel*, and may be altered by Wendy. The goal for the prisoners is thus to embed a covert channel into the overt channel for hidden communication. Additionally, the

overt channel limits the size of the covert channel since the hidden message cannot be bigger than the message it is hidden into.

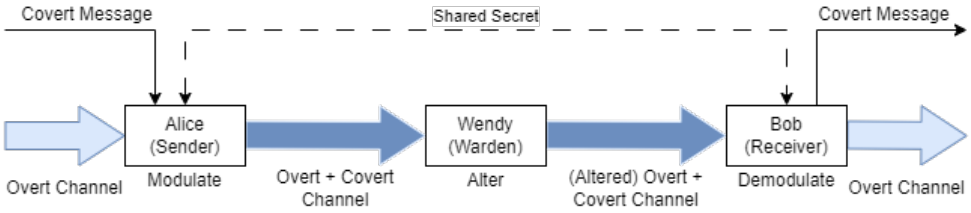


Figure 2: de-facto model for covert channel communication. (Simmons 1984a)

To reflect this to the covert channel definition by NIST (NIST 2020), the warden is in this sense the enforcer of the system's security policy and hiding the message in notes is the same as creating an unintended or unauthorized intra-system channel. Similarly, the prisoners are cooperating and they're given access authorization for communication, which they attempt to use to pass information and bypass the warden's inspection.

The prisoner's problem (Simmons 1984a) is originally used to describe subliminal channels but it is generally accepted to also describe covert channels since subliminal channels are currently seen as a subfield of covert channels (Petitcolas, Anderson, and Kuhn 1999). However, originally subliminal channels were categorised as a field of cryptography by its creator, due to how the hidden communication channel is always embedded into an cryptographic algorithm (Simmons 1984a). In this sense, while subliminal channels are a subfield of covert channels, they inherently require cryptographic algorithms to exist. This dependancy is highlighted in Figure 1 with a dotted line.

In relation to subliminal channels, Lubacz, Mazurczyk, and Szczypiorski (2014) believe that there is possibility to misunderstand cryptography and steganography, which is why it is important to clarify the distinction between the two. While both crypthography and steganography are part of communications security, what they are securing differs. Steganographic methods hide, and thus secure, information by making it difficult to notice by embedding it inside an information carrier, which is often called a *cover*. This is different to cryptography, where the message itself is not be hidden, but its information is secured by making it difficult to understand by transforming it with an algorithm into an unrecognizable form. (Lubacz, Mazurczyk, and Szczypiorski 2014). Fundamentally, cryptographic techniques at-

tempt to conceal the contents of a message, whereas steganography goes yet a bit further; it attempts to hide the very existence of communication (Katzenbeisser and Petitcolas 2016). In this sense a message could be first encrypted with cryptographic techniques, which is then hidden through steganographic techniques.

3.1 Network Covert Channel Features

For network covert channels, the terminology overlap surrounding 'modulation' is a challenge, since covert channel's and the traditional meaning of network communication modulation coexist in this environment. In network communication, 'modulation' traditionally refers to the process of modifying a signal or waveform to carry information or data. The purpose of this modulation is to allow the transmission of information over a communication channel effectively.

Conversely, in covert channel research, 'modulation' refers to the process of converting information into secret symbols, which are then injected into an existing overt channel. These secret symbols are embedded into an overt channel by modifying its properties, effectively hiding the communication in plain sight.

During this thesis, 'modulation' refers to the modification of the overt channel for the purpose of creation and use of secret symbols for covert communication.

The first application of covert channels into networked environment was done by Girling (1987), where he initially identified two storage channels and one timing channel, which he initially classified based on what is sent and when the data is sent respectively. Covert channels still remain a strong interest in networked environments, and is researched due to how they enable stealthy malware command and control for ransomware attacks, such as in Hammertoss APT (Mazurczyk and Wendzel 2017).

Simmons' prisoners' problem can be extended into networked environments, but it requires some changes due to information processing being done with machines. Warden is often likened to firewalls, since they're the enforcers of network traffic's security policies, while prisoners are seen as computers. Similarly, to keep track of communication flow, which is

distributed in networked environment, the sender and receiver computers are named Alice and Bob respectively, while the firewall or system's policy enforcer (warden) is named as Wendy. In this sense computer Alice is attempting to send a message with hidden message through firewall Wendy to computer Bob. Extending communication to networked environment also changes how information is transferred from Alice to Bob. Instead of messages on paper, computers communicate through various protocols. This leads to overt channel definition to be similarly extended as the transmission of information through a system mechanism. Similarly, covert channel's capacity's dependency to the overt channel in real-life scenario follows when brought into networked environment; covert channels embedded into protocols are limited by its type and length of transmission (Mazurczyk and Wendzel 2017).

In reality networked environments are made out of more than a connection of two computers and a firewall in-between, and it includes switches, routers, servers, and so forth. Network covert channel is thus inherently different from single-host covert channels, since the former is embedded into distributed systems, while the latter is embedded into stand-alone systems (Carrara and Adams 2016a). This leads to a situation where the communication channel includes more participants than simply Alice, Wendy and Bob, such as servers or links. Alice and Bob are not limited to function as overt sender and receiver, and might exist only partway through a longer chain of communication, acting as *middlemen*. To remain stealthy as middlemen, Alice and Bob may look to only have a covert channel partway through the overt channel by modulating and demodulating the covert data into and from the channel at their respective points. (Zander, Armitage, and Branch 2007a) Figure 3 illustrates the possible combinations of covert senders and receivers in a network communication channel. The application scenario of a covert channel often determines the locations of both the sender and receiver. In the context of data exfiltration, both the sender and receiver often act as middlemen. The sender could be compromised computer, while the receiver could be positioned on a router at the edge of the compromised network. When circumventing censorship or surveillance, the sender/receiver pairs are likely to intentionally function as both the covert and overt endpoints. (Zander, Armitage, and Branch 2007a).

Carrara and Adams (2016a) and Heda and Shah (2015) note that there is possibly some ambiguity as to the difference between steganography and covert channels, which is espe-

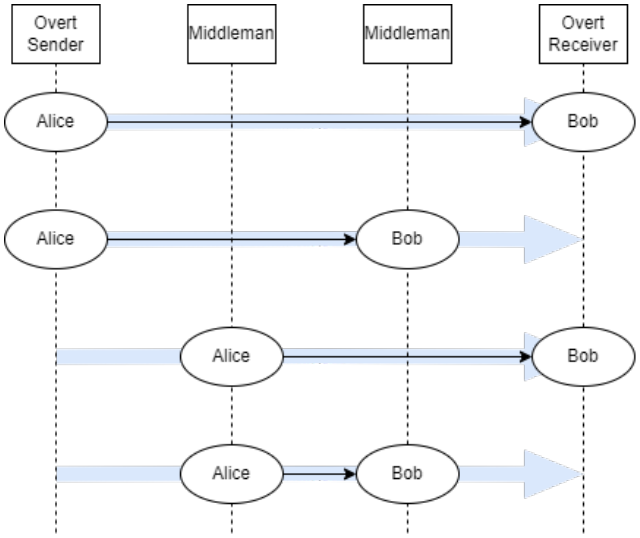


Figure 3: Possible combinations of covert senders/receivers. (Zander, Armitage, and Branch 2007a)

cially prominent when they are within networked environments (Lubacz, Mazurczyk, and Szczypiorski 2014). Both network covert channels and steganography require a cover for embedding their secret messages. In networked environments, network protocols serve as the cover for concealment. Wendzel et al. (2015) argue that the difference between the two is that in steganography the cover is generally interpreted by humans and the goal is to hide information from them, whereas in network covert channel the cover is interpreted by machines, and the data is hidden within protocols. Carrara and Adams (2016a) argue in their taxonomy that there is a possibility to have an open covert channel, which does not employ a steganographic cover. This type of covert channel could exist unnoticed and fit the definition of a covert channel, if the enforcer of system's security policy is insufficient, such as a firewall that is not configured effectively enough. However, they are unlikely in networked environments, where covert channels should and often do use encryption and steganography (Heda and Shah 2015).

Conversely, Mazurczyk et al. (2016) argue that terms covert channel and steganography inside a networked environment should be unified as *network steganography* due to how steganography is essentially required for creating an effective covert channel that is capable of bypassing networked system's security policies. Fundamentally, the main difference between the two is that steganographic techniques are used to hide communication in net-

worked environment and the channel that is formed through these techniques to bypass security policies is called a network covert channel. This is also seen in the capacity of a network covert channel, since it depends on the applied steganography technique in the context of a particular communication network (Lubacz, Mazurczyk, and Szczypiorski 2014). This essentially means that covert channel that employs steganography has its capacity limited by both the overt channel it is embedded into and the stenagographic technique it employs. As with subliminal channels and cryptography, overlap of network covert channels and steganography is highlighted in Figure 1 with a dotted line.

Networked environment is also unique since it enables covert channels to appear simultaneously alongside each other (Carrara and Adams 2016a). For example, network covert channels have a common cover in the form of protocols, that often use digital signatures to prevent impersonation in digital communication (Carrara and Adams 2016a). El-Gamal signature theme and Digital Signature Algorithm (DSA) are possible covers for subliminal channels, which enables them to occur concurrently with network covert channels (Simmons 1984b, 1993; Anderson et al. 1996). This leads to networked environment designers having to consider both subliminal channels and network covert channels when monitoring its system security policies.

3.2 Network Covert Channel Categorisation

Covert channels have been historically split between *covert timing channels* and *covert storage channels* (Latham 1986), although it is often noted that there is theoretically no fundamental distinction between them (Gligor 1994; Zander, Armitage, and Branch 2007a) or that they are too ambigious to be clearly separate in networked environments (Wendzel et al. 2022). In a similar manner, pattern based categorisation surveys for network covert channels have followed this initial split.

Categorisation for network covert channels is heavily based on the patterns of the techniques they use or what the techniques modify. Zander, Armitage, and Branch (2007a) performed the initial survey for network covert channel techniques, where they established that they could be grouped into general techniques that are independent of protocols. Building upon

that and similar works, Wendzel et al. (2015) published a more comprehensive and current catalog of the existing techniques. This established that covert channels could be grouped together into general patterns based on the general techniques used for their communication but also by the protocols they utilised. Mazurczyk, Wendzel, and Cabaj (2018) expanded upon this taxonomy, and note that it is possible for a covert channel to use both storage and timing based steganographic techniques simultaneously within a networked environment, which they define as a *hybrid covert channel*. Furthermore, this taxonomy is constantly extended through research on novel steganography techniques that create new types of covert channels that the old taxonomy did not include (Velinov et al. 2019; Mileva et al. 2021; Hartmann, Zillien, and Wendzel 2021).

Lastly, the authors of the initial 2015 pattern-based taxonomy suggested a taxonomy for the whole domain of steganography in 2021 (Wendzel et al. 2021). This taxonomy tackles the ambiguity of storage and timing channels by instead separating steganographic techniques into non-temporal and temporal patterns, which are only loosely related to previous distinction. While this taxonomy did provide solutions to the 2015 taxonomy for network steganography specifically, such as including embedding and extraction patterns, it still suffered from the artificial distinction of temporal and non-temporal methods that storage and timing channels suffered from (Wendzel et al. 2022). For example, when TCP packets' sequence is altered for a steganography technique, the real occurring sequence of the packet matters as much as their stored sequence number. MITM attacker could theoretically only alter the stored sequence number without altering the real sequence of the packets and vice-versa. To combat this, Wendzel et al. (2022) developed a taxonomy, where steganography techniques are categorised into two groups known as *core patterns*, which comprise of element's state/value modulation and element's occurrence modulation. *Element* in this context is, for example, a network packet, header field, packet size and time of occurrence property of a packet, whereas the *state/value* of an element could be the actual packet size in bytes, the actual TCP sequence number of a header field and the time of sending/arrival. The steganography techniques are performed by the covert sender and the covert receiver, and they may differ from one another.

While there are clear attempts to divert steganography taxonomy away from storage and tim-

ing terms, the authors driving this do not extend it to covert channel terminology (Zillien and Wendzel 2023). With this in mind, we will follow the method of Mazurczyk, Wendzel, and Cabaj (2018) and continue categorising covert channels as storage or timing channels based on the steganography techniques they employ in their information hiding as seen in Figure 4. However, the steganographic technique pattern categorisation will follow the updated 2022 taxonomy by Wendzel et al. (2022) that avoids ambiguity.

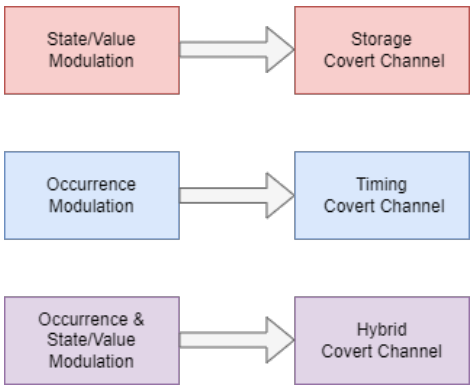


Figure 4: Introduction to covert channel types based on the steganography technique

3.2.1 Storage Based Network Covert Channel

Network Covert Storage Channels make use of hiding methods where information is injected or manipulated into network traffic's digital artifacts, such as the payload or the header of a packet. The updated taxonomy simplifies network covert storage channels as channels that use hiding methods that modulate a particular *state* or *value* of a network element, such as a header element, packet, payload field or sequence number, to embed a message. This core pattern is extended to create subpatterns based on the state or value that is modulated. Any covert channel made with a technique that is part of one of these patterns is considered a network covert storage channel.

Subpatterns of state/value modulation core pattern are modulations of element's *reserved/-unused*, *random*, *least significant bit* (LSB), *character* and *redundancy* state or value. *Reserved/unused* state/value modulation consists of overwriting the unused or reserved fields in different protocols, such as the IP identifier. *Random* state/value modulation takes place when a (pseudo-)random value or state of/in network data is replaced with a secret mes-

sage that appears pseudo-random, for instance the TCP initial sequence number (ISN), the dynamic host configuration protocol (DHCP) xid field or the Secure Shell (SSH) medium access control (MAC) field. *LSB* state/value modulation focuses on the modulation of network elements LSB, such as IPv4 time-to-live (TTL) field, IPv6 Hop Limit field or TCP timestamp option. *Character* state/value modulation is a pattern where the cases of textual characters are modulated, e.g. the modulation of characters in HTTP headers or in other protocols, such as Simple Mail Transfer Protocol (SMTP), Post Office Protocol version 3 (POP3), Network News Transfer Protocol (NNTP), or Internet message access protocol (IMAP). Lastly, *Redundancy* state/value modulation compresses the network element's content, which is usually followed by filling the gained space with covert with the use of another pattern. State/Value modulation steganography sub-patterns are summarised in Table 2, which is based on the updated taxonomy by Wendzel et al. (2022).

Table 2: Subpatterns of element's state/value modulation based on Wendzel et al. (2022)

Modulated Element's State/Value	**Description**	**Examples**
Reserved/Unused	The covert message is embedded by modulating a state or value that is reserved or unused.	Overwriting the unused or reserved fields in IPv4, IPv6, TCP, or DHCP, such as the IP identifier.
Random	A (pseudo-)random value or state of/in the network data is replaced with a secret message that appears pseudo-random.	Utilization of the pseudo-random IP Identifier field, the TCP ISN, the DHCP xid field, or the SSH MAC field random value fields.
LSB	The least significant bits (LSB) of network elements are modulated.	Modulation of the LSBs in the IPv6 Hop Limit field, IPv4 TTL field, modulation of the IP timestamp option's LSB, TCP timestamp option, and DHCP's LSB of the secs field.

Table 2: Subpatterns of element's state/value modulation based on Wendzel et al. (2022)

Character	The cases of textual characters are modulated.	Modulation of characters in HTTP headers or in other textual protocols, such as SMTP, IMAP, POP3, and NNTP.
Redundancy	The redundancy of a network element's content is modulated by means of compression, which is usually followed by a succeeding pattern that fills the gained space with covert data	Compression of existing payload which is filled with reserved/unused modulation.

3.2.2 Timing Based Network Covert Channel

Network covert timing channels are made with steganography techniques that manipulate the behaviour of flows and conversations, that are made up out of coherent sequences of packets in networked environment. This is done by modulating the occurrence of the network elements, such as the timing or the sequence of packets (Wendzel et al. 2021), which can also affect the overall number or the rate of the packets appearing in the flow (Wendzel et al. 2022). For example, this could be sending a packet at a specific time or influencing the time or order at which a packet is sent or received.

Network environments have strictly standardised communication flows, and the techniques to modulate element occurrence can vary greatly. This standardisation functions as a framework by which a developed covert channel must function. However, the strict adherence to the standards may be the tool by which the covert channel communicates. For example, the retransmission of a specific frame or packet in TCP due to their intentional corruption (Zillien and Wendzel 2018), performing a high number of frame transmissions so it affects the rate/thoroughput of an internet link (Wendzel et al. 2022), and dropping TCP segments to create artificial loss, which could be based on an even or uneven sequence number (Mazurczyk, Smolarczyk, and Szczypiorski 2011).

Similar to steganography techniques for state/value modulation, element *occurrence* modulation is a core pattern and it extends sub-patterns to element *ennumeration* and *positioning*. *Ennumeration* describes a technique where the quantity of network sub-elements is modulated or added to increase the overall size of the element, such as adding more sub-elements to the payload in packets. For example, fragmentation of a network packet to multiple packets, modulating the number of DHCP options, and encoding information through the number of IPv6 or IPv4 headers. *Positioning* is a subpattern that describes the embedding of a covert message by inserting or changing the temporal or spatial position of a network element. For example, a case of temporal positioning is where a specific packet is sent at some specified point in time in a flow, where as spatial positioning would be modulating the position of an existing TCP segment in a TCP stream. Occurrence modulation sub-patterns are summarised in Table 3, which is based on the updated taxonomy by Wendzel et al. (2022).

Reflecting to state/value modulation, the altering of the sequence of packets will lead to changes in the packets' stored sequence number, but it inherently separates it from the modulation of the stored sequence number. This leads to avoiding the previously problematic ambiquity, since while outcome might appear the same, one is done by modulating element's occurrence and the other by modulating the stored value of the element.

Table 3: Subpatterns of element occurrence modulation based on Wendzel et al. (2022)

Element Occurrence Subpattern	**Description**	**Examples**
Ennumeration	Quantity of network sub-elements is modulated or added to increase the overall size of the element	Fragmentation of a network packet to multiple packets, modulating the number of DHCP options, and encoding information through the number of IPv6 or IPv4 headers.
Positioning	Embedding of a covert message by inserting or changing the temporal or spatial position of a network element.	Specific packet is sent at some specified point in time in a flow, and modulating the position of an existing TCP segment in a TCP stream

3.2.3 Other Defined Properties

Covert channels are primarily described as storage or timing based on the hiding pattern they employ, but there are other descriptors for their unique properties. These descriptors are not as universally agreed upon and are a naming convention often based on new features that are found in a new channel or hiding method. For example, as covert channel research developed, there have been cases hiding methods that utilise both storage and timing patterns, creating hybrid covert channels (Mazurczyk, Wendzel, and Cabaj 2018).

Most common accepted properties for channels are noisy or noise-free (Cover 1999), capacity (Millen 1987), robustness (Maurice et al. 2017), stealth (Zander et al. 2010), passive or active (Cabuk 2006), indirect or direct (Zander, Armitage, and Branch 2007a), *distributed* or *undistributed* (Mazurczyk, Wendzel, and Cabaj 2018), and uni- or bi-directional (Hartmann, Zillien, and Wendzel 2021). There are taxonomy updates that also propose expanding the commonly used terminology, such as defining covert channels as *open* or *steganographic*, but their propositions do not necessarily gain traction for further use (Carrara and Adams 2016a). Listing all properties is outside the scope of this study as there is a lack of topical taxonomy for them, but mentioning the commonly used properties is important to understand how covert channels might be created and detected. A topical mapping of the prominent properties of covert channels would be a subject for further research.

Hybrid covert channels employ both core patterns (e.g. element occurrence and state/value modulation) to transfer covert data between covert sender and covert receiver (Mazurczyk, Wendzel, and Cabaj 2018; Wendzel et al. 2022). This inference is drawn from the previous definition of hybrid covert channels and the revisions to steganography taxonomy. In the past, hybrid covert channels were generated when a channel employed both storage and timing steganography techniques in the communication between the covert sender and receiver (Mazurczyk, Wendzel, and Cabaj 2018). Since covert channels are defined based on the steganography patterns and techniques, the revision to steganography techniques extends to the definition of hybrid covert channels (Wendzel et al. 2022).

In relation to hybrid covert channels, *hybrid patterns* are used to define the combination of concurrent use of the core steganography patterns, thus hybrid covert channels may be cre-

ated either in the form of a *hybrid pattern*, where covert sender explicitly uses both core patterns, or by using different core patterns in sending and receiving the covert data. However, *hybrid pattern* requires that the different core patterns are explicitly done by the covert sender or it is instead considered a *non-hybrid pattern*. For example, size modulation requires two patterns to increase the size of a network packet. In the case of IPv4, size of the packet is increased by increasing the count of header elements with **element enumeration**, which leads to the need of adjusting the actual value of the IPv4 Internet Header Length (IHL) header field with **state/value modulation**. This combination is a *hybrid pattern* when the patterns are done explicitly by the covert sender. However, if the IHL header field is instead adjusted automatically by the operating system kernel, it is considered a *non-hybrid pattern*. In this scenario the **state/value modulation** is a byproduct of the explicit **element enumeration**, which still allows the receiver to receive the covert data through both patterns. Therefore a *hybrid pattern* requires explicit use of both core patterns and will always produce a hybrid covert channel. *Non-hybrid pattern* may therefore be considered as either a non-hybrid state/value pattern or non-hybrid occurrence pattern based on which is explicitly performed, and will only lead to a hybrid covert channel if the receiver uses both or another core pattern. (Mazurczyk, Wendzel, and Cabaj 2018; Wendzel et al. 2022) Some of the combinations of patterns and the produced covert channels are summarised in Figure 5.

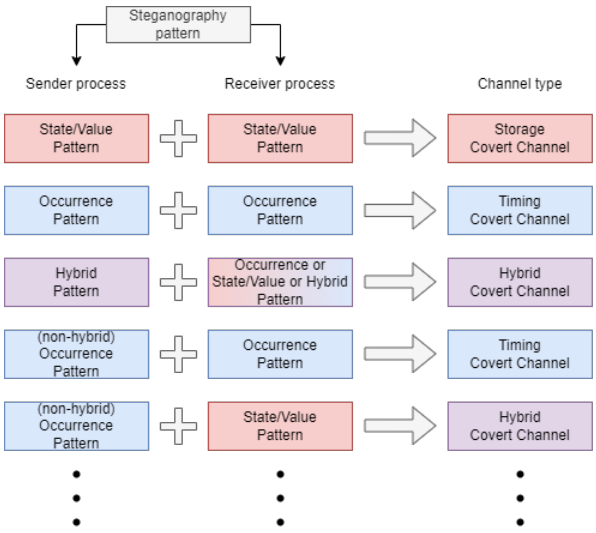


Figure 5: Combinations of steganography patterns and the produced covert channels.

Noise or *noise-free* are used to describe the error-rate of the communication channel. The

channel is defined as noise-free, if the channel has no channel errors. However, they are not realistic as most channels have some degree of noise, such as thermal noise or electromagnetic interference. In network channels this is the case, if the bits transmitted by the sender are always decoded correctly by the receiver. Noisy channel instead has channel errors, such as substitutions, erasures, deletions and insertions. Thus, increased noise increases the error rates in a communication channel. (Cover 1999) Regardless of the degree of noise contamination, it is theoretically possible to transmit information (bits) with no errors through some code, though possibly slow one with heavy redundancy. (Shannon 1948).

Capacity of a channel is its maximum possible error-free transmission rate (Shannon 1948). This is often described in bits per second but it can also be described as bits per overt packet for covert channels (Zander et al. 2010). Channel capacity is inversely correlated to channel noise, since the code that enables error-free communication in noisy communication channels reduces the transmission rate of the channel.

Robustness determines how easily a covert channel is eliminated or its capacity is limited by channel noise. Artificial noise can be additionally introduced by a warden to prevent covert channel function. Strong interference by a warden can prevent normal function of communication channels. (Zander et al. 2010)

Stealth determines how easily a covert channel can be detected. For example, by comparing the characteristics of traffic with covert channel and unmodified legitimate traffic, the traffic patterns of covert channel may be classified as anomalies (Zander et al. 2010). One way to increase stealth of a covert channel is for it to be *passive* instead of *active* (Zander, Armitage, and Branch 2007a).

Capacity, *robustness* and *stealth* have conflicting goals, and improving one weakens another. For example, increasing redundancy improves robustness and sending less data increases stealthiness, but both reduce channel capacity. Similarly, increasing signal amplitude increases robustness, but reduces stealth. Covert channels cannot therefore usually maximise all three features and have to choose what is best for the scenario. (Zander et al. 2010)

Passive or *active* describes covert channels based on how they transfer the covert data in network traffic. Passive covert channels (PCC) do not generate new traffic, but use the data

of the existing communication to modulate covert data into the network traffic (Tumoian and Anikeev 2005). Active covert channels on the other hand generate completely new traffic in which to modulate the covert data inside. Since passive covert channels modulate legitimate traffic, it is often more *stealthy* than active covert channels which generate new traffic that attempt to appear legitimate. (Wendzel et al. 2015)

Direct or *indirect* covert channel refers to the route and the participants that the covert data is conveyed with. *Direct* channels are formed when overt traffic containing the covert data flow directly from covert sender to covert receiver. The sender and receiver may function as middle-men as in Figure 3. (Zander et al. 2010) *Indirect* channels are instead established when an intermediate host is exploited to store, represent or unintentionally redirect covert data between sender and receiver. This creates two flows of overt traffic conveying the covert data. First is between sender and an intermediate host and the second is between the intermediate host and receiver. (Zander, Armitage, and Branch 2007a; Schmidbauer and Wendzel 2022)

Distributed covert channels are sophisticated network covert channels that transfer covert data through multiple flows, protocols or hosts (pattern variation), or uses multiple patterns simultaneously (pattern combination) or sequentially (pattern hopping) within the same flow or protocol for the covert data exchange. In contrast, *undistributed* covert channels are instead the typical network covert channels; they are either storage or timing channels that use a single flow or protocol, and use only one steganography pattern for embedding covert data into a cover. (Mazurczyk, Wendzel, and Cabaj 2018) The addition of patterns distributes the covert data into multiple "locations", which improves stealth of the channel due to increased monitoring and detection requirements for the warden. For example, it is not necessarily difficult to block a specific technique, but distributed covert channels use multiple modulation techniques in various combinations, which aids in bypassing defenses that focus on preventing only one technique (Wendzel et al. 2015).

Uni- and *bi-directional* covert channel describes the directions that the covert channel may be used in. *uni-directional* covert channels only allow one direction of communication, whereas *bi-directional* can be used in both ways, from Alice (via node) to Bob and vice versa. In latter both Alice and Bob can function as senders and receivers. (Hartmann, Zillien, and Wendzel

2021)

3.3 Network Covert Channel Countermeasures

The specificities of various covert channel embedding methods is an issue when creating cohesive general countermeasures for their detection and neutralization, and there is a notable lack of their systemic organizing or reviewing. To aid in this is, Zander, Armitage, and Branch (2007a) reviewed existing covert channel methods that is often used as the source for countermeasures as progressive steps. Additionally, there are recents attempts to highlight the issues around this section of covert channel research, in addition to efforts made towards developing a more structured framework to create new countermeasures that highlight the negative tradeoffs that the defenses may cause towards the system's normal function. (Caviglione 2021)

Countermeasures for network covert channels are seen as interconnected multi-step process as seen in Figure 6. In networked environment covert channe countermeasures are synonymous with network steganography countermeasures. The countermeasures comprise of identification, prevention, elimination, detection and auditing (Zander, Armitage, and Branch 2007a). This follows the traditional logic of NIST Cybersecurity Framework's five steps, which are comprised of identify, protect, detect, respond and recover (Pascoe, Quinn, and Scarfone 2024).

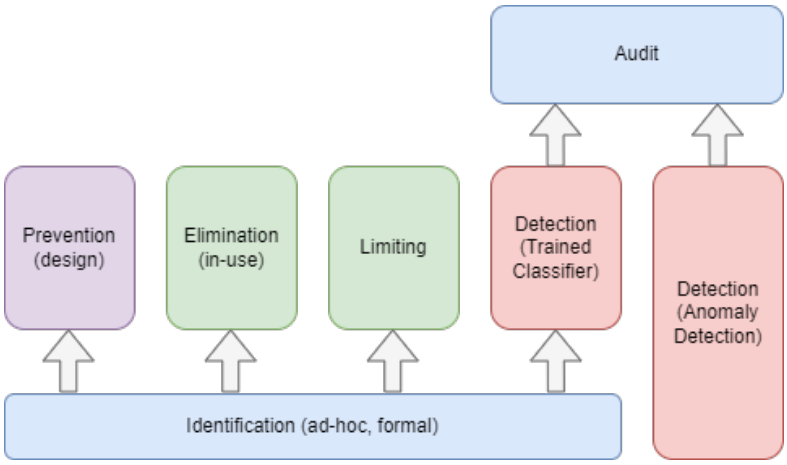


Figure 6: Covert channel countermeasures and their combinations (Mazurczyk et al. 2016).

Initial step is the *identification* of the channel, thus becoming aware of its existence. Without knowing the channel, even naive information hiding techniques are covert (Mazurczyk, Wendzel, and Cabaj 2018). The identification may occur during the design phase, or in a deployed system, and aims to expose a shared resource that may be used as a covert channel variable, such as an unused state/value. Knowing the mechanisms by which the covert channel functions during design phase allows the *prevention* of the channel's creation. However, malicious activity is impossible to completely prevent due to design oversights and inherent system design weaknesses, which is why there are additional countermeasures against active in-use covert channels (Zander, Armitage, and Branch 2007a).

Ideally the established covert channel is *eliminated*, but this may be too difficult, too expensive to perform, leads to a very inefficient system or is inherently impossible (Moskowitz and Kang 1994; Zander, Armitage, and Branch 2007a). However, other actions that give similar outcome, such as when the channel's bandwidth is *limited* severily enough, it will render the covert channel mostly useless in practice (Kang and Moskowitz 1993). This however comes at the cost of overt channel's bandwith and small messages may still be sent through these channels to command things such as malware (Moskowitz and Kang 1994).

If the covert channel is possible to *detect*, it enables the system to take actions against the channel participants, such as the covert senders and receivers. Detection is split further into *trained classifiers* or *anomaly detection* depending on, if the covert channel is searched based on its identified features or based on anomalies when compared to the system's normal behaviour. The process after the detection to gather information about the covert senders and receivers, and their used communication techniques and patterns, is called *auditing* (Cabuk 2006).

The countermeasure actions are not isolated instances and are often instead done in combination with one another or require others to enable them. For example, it is impossible to train classifiers or perform informed actions to prevent, eliminate or limit something without identifying it first. Similarly, it is impossible to audit something that is not detected. (Mazurczyk et al. 2016).

Existing countermeasures affect in one way or another how adversaries develop their covert

channels, since they aim to circumvent the system's protections. Therefore, how a system is protected affects how a covert channel is developed and embedded, which in turn will affect how the covert channel may be detected.

4 Inter-Arrival Time Covert Channel

Inter-arrival time (IAT) or Inter-packet time covert channels are covert timing channels that use element pattern occurrence modulation. The channel is formed by covert sender altering the timing interval between network packets to encode hidden data into overt traffic. (Wendzel et al. 2022) The IAT is the time elapsed between two succeeding IP packets, where each IAT represents a secret symbol which is used to transfer hidden data. For example, a simple IAT covert channel might have IAT of 50ms indicate a 0 bit while IAT of 100ms might indicate a 1 bit. By contrast, a more advanced IAT covert channel might instead have two or more IATs to correspond to two or more secret symbols. (Zillien and Wendzel 2023) Additionally, advanced IAT covert channels might implement machine learning methods in the selection of IAT delays based on the legitimate traffic's IAT patterns. This increases the stealthiness of the channel by obfuscating the difference between legitimate and malicious IATs, which increases the difficulty to detect the embedded covert channel. (Gianvecchio et al. 2008) However, it is important to note that an increase in stealth comes at the cost of robustness. This is primarily due to the uncertainty of creating the required precise delays in congested networks. This congestion also impacts the creation of secret symbols, since it may cause precise millisecond based IATs to become impossible to create.

While simple IAT covert channels might appear straightforward, in practice IP packets do not offer any guarantees for their delivery times and thus introduce natural unwanted IAT delays. This delay can be calcluated, but it would need to be done with machine learning methods and would lead to an advanced covert channel or require additional reconnaissance and manual calculations before injection of a simple covert channel. Additionally, sender and receiver need clock synchronization to ensure channel accuracy due to possible clock skew, where two clocks run at different frequencies. This leads to IAT covert channels requiring schemes to preserve synchronization and to resynchronize the channels as needed to ensure error-free communication. These schemes however come at the cost of channel capacity, since they introduce additional noise to the channel. (Cabuk 2006) Lastly, networks may contain natural or warden introduced delay jitter, which is more difficult to calculate than network delays and need to be taken into consideration when developing IAT covert

channels, since it reduces the channel accuracy. IAT covert channels are sometimes evaluated against emulated network jitter to determine the robustness of the channel to determine how applicable the channel is in practice (Zander, Armitage, and Branch 2011).

IAT covert channels are more difficult to create than more traditional storage based covert channels, but they have other benefits. Because IAT covert channels are not dependant on structure to the same extent as storage covert channels, they are applicable to most protocols and overt channels. Additionally, they may bypass next-gen firewall settings that analyze packets' content, since the information is not transferred inside the packet itself.

The changes in IAT occur through the manipulation of inter-packet departure times (IDT) at covert sender. In contrast to IAT, IDT represents the time difference between the departure of current and previously departed packets. The IDT is manipulated to create the wanted IAT at the receiver to transfer hidden information. Manipulation of the IDT is commonly done by delaying the transmission of the packet, while taking into account possible common delays and network jitter to ensure the accuracy of the hidden message (In short, IAT = IDT + common delays + artificial delays). (Iglesias et al. 2022) Possible common delays are processing, queuing, transmission and propagation delays.

While covert channels are often discussed in the context of malicious attackers, the research focuses heavily on the information hiding methods and communication detection rather than the practical execution of these techniques. Therefore the implementation mechanisms of IAT delay are rarely dicussed, as the research focus is on what to do with it, rather than how it may be done. However, these implementation mechanism are another footprint for covert channel detection and should not be completely ignored. Even in network based detection, if IAT delay is performed through retransmission and packet dropping, they leave footprints behind by creating excess packets compared to normal network traffic flow. Therefore, it would be beneficial for researchers to implement in practice in a real-world scenario without offline injection. This is because hardware and software limitations and restrictions create additional noise, which is a concern for the function of a covert timing channel.

4.1 Network Covert Channel Detection

The detection of covert channels is commonly split into statistical-based methods and machine learning methods (Wendzel et al. 2015). Machine learning methods are similarly categorised into sub-methods based on the detection strategy they employ. The three major machine learning detection strategies are pattern-based, anomaly-based and specification-based detection. (Liao et al. 2013)

Network covert channel detection is generally considered to occur either online or offline. The offline detection mechanism may be run as a batch process where collected network traffic data is analysed afterwards. In contrast, online methods focus on the detection algorithm's speed to implement real-time detection schemes. Because of this distinction, online detection is closely connected to NIDS systems and systems implementing them are sometimes referred to as 'wardens'. However, historically online detection systems have faced a trade-off: storing and analyzing large volumes of traffic data history can require non-negligble storage and computing power. (Cabuk 2006) For small or medium-sized actors, deploying a warden in a large-scale network or running sophisticated detection software that maintains satisfactory performance may be unfeasible (Caviglione 2021).

Pattern-based detection in machine learning employ detectors that are constructed as classifiers. These classifiers have a natural disadvantage of needing a training phase, which includes the need for training data. There is a distinct lack of training material, which is why some researchers have focused on developing software tools to inject covert channels into overt traffic. Therefore, detectors are prominently built to detect self-injected covert channels. (Iglesias et al. 2022) Additionally, the training phase may make the detector over-specialized to the chosen cover source, such as specific protocols or pattern. Thus, while a detector might be extremely accurate on the training source, it may not be applicable on sources of other protocols or patterns. (Katzenbeisser and Petitcolas 2016) Limitation of pattern-based learning is the need for predetermined knowledge of covert channel's existence, which means that the embedding method needs to be identified before it may be detected.

Anomaly-based detection in machine learning employ detectors that are trained to understand

the standard operations of the environment and identify any deviations or anomalies from this norm. The benefit of this in comparison to pattern based detection, is that anomaly detection is only limited by the warden's collected system logs. This means that anomaly-based detection solutions may detect covert channels that have not yet been identified. (Chourib 2019)

Specification-based detection in machine learning employ detectors that are trained based on the specifications of a protocol to verify any misuse or attacks. This type of solution is more capable in detecting covert storage channels, since they modulate the contents of protocol fields. (Chourib 2019) However, most sophisticated covert channels look to employ acceptable network traffic as their overt channel, which may not be caught as protocol's misuse.

Statistical-based methods involve the use of statistical techniques to analyze network traffic and identify irregularities that may indicate the presence of a covert channel. These methods focus on analyzing characteristics of network traffic, such as packet timing and size, protocol use, and applying statistical tests to identify anomalies or patterns. They are often divided into shape tests, which focuses on first-order statistics, such as mean, variance and distribution, and regularity tests, which is described by second or higher-order statistics, for example the correlations in the data (Gianvecchio and Wang 2007). From these statistical analysis tests, two are often especially highlighted: ε-similarity test and compressibility test (Schmidbauer 2023). Additionally, entropy based methods are sometimes highlighted as a form of shape tests, which may be used to detect covert channels (Gianvecchio and Wang 2010).

4.2 Inter-Arrival Time Covert Channel Detection Techniques

Ethical considerations emphasize the need to develop detection methods alongside new steganography techniques. According to Lubacz, Mazurczyk, and Szczypiorski (2014), the potential for malicious use underscores the importance of implementing detection countermeasures when publicising a steganography technique. Detection methods are therefore researched after a new technique is publicized or in the initial publication.

Initial simple IP IAT covert channel was developed by (Cabuk, Brodley, and Shields 2004), which requires the sender and receiver to negotiate a parameter, which is used as a fixed time interval. If the sender needs to send hidden information of ´1´, it sends a data packet within the negotiated time interval. If the sender needs to send ´0´, nothing is sent during this time interval. This choice is made and repeated for each time interval. Cabuk, Brodley, and Shields (2004) provided the detection method for these inter-arrival time based covert timing channels by implementing statistical analysis in the form of ε-Similarity between adjacent sorted IAT values. This is done by sorting IAT values and computing the relative difference between each pair of consecutive points. ε-Similarity is then performed by computing the percentage of relative differences that are less than ε. For covert channels, most differences between pairs in the sorted list of IAT values should be very small.

Extending upon previous work, Cabuk (2006) introduced compressibility score for covert timing channel detection. This detection is based on the Kolmogorov complexity of a string. Kolmogorov complexity provides the lower bound on the string representation and computes the maximum compression available for it, therefore producing the shortest possible compressed string. IAT values require conversion before they are computable as strings, which needs to be done as pre-processing step. Minimal noise simple covert timing channel produces higher compressibility than legitimate channel. Compressibility score may be utilised in smaller segments, such as in 2000 packet segments.

Entropy tests are a subset of statistical analysis (Archibald and Ghosal 2014). Entropy of a process, such as communication, is a measure of the uncertainty or information content. It may be used to measure complexity or regularity of a process. Originally used in covert timing channel capacity analysis, Gianvecchio and Wang (2010) extended this to covert timing channel detection. They observed that covert timing channels cannot be injected without it affecting the entropy of the communication channel. They argue that since entropy rate for finite sample cannot be measured, conditional entropy (shape test) and corrected conditional entropy (regularity test) of finite samples may be used to estimate the entropy rate. Low shape test score indicates covert channels, because the tested sample does not fit the appropriate distribution. Abnormally high or low regularity score indicate a possible covert channel. Therefore, covert chanels exhibit lower conditional entropy scores, and higher or

lower corrected conditional entropy scores than legitimate channels.

GAS is a recurrent neural network (RNN) based detection method, based on the standard principle of extracting and depicting features, such as the traffic timing behaviour of a short-term network traffic capture. It learns legitimate traffic patterns by extracting relevant information from IATs capture in network traffic, such as the discrete and variation-meaningful values, which explicitly reflect the patterns of variation of the IAT sequences. The trained patterns function as the training data for the anomaly-based machine learning model. This is utilised later in detection, where unknown incoming traffic capture is similarly computed. These extracted IAT features from unknown traffic sample are then used in sequential prediction to fit against the original IAT variation patterns, which provides a loss value as an output. GAS applies Long Short-Term Memory (LSTM) network in the sequential prediction, which are a form of RNN. A high loss value output from the GAS model suggests that the unknown network traffic sample set has significantly different variation patterns compared to the original trained legitimate traffic sample, which indicates the presence of a covert timing channel. (Li, Song, and Yang 2022)

Al-Eidi et al. (2020) developed SnapCatch, a machine learning method that utilises image classifier for covert timing channel detection. The method functions by converting a traffic capture into colored images, from which various image-based features are extracted from, such as mean grey value, center of mass, median value of pixels, and standard deviation of grey values. These features are then used to train several machine learning models using different machine learning algorithms, such as Support Vector Machine, Decision Tree, Naïve Bayes, and Artificial Neural Network. Their solution was capable of detecting small 8 bit-size covert messages, which are hard to find with statistical detection methods, such as common correlated effects.

While statistical analysis is the most common approach in covert timing channel detection, the machine learning approaches are sometimes more capable in generic covert timing channel detection. For example, GAS (Li, Song, and Yang 2022) and SnapCatch (Al-Eidi et al. 2020) are capable of accurately detecting simple covert timing channels, but their accuracy suffers when sophisticated covert timing channels are evaluated against them (Zillien and Wendzel 2023).

5 Design Science Research

In this study we approach Design Science Research (DSR) through A. R. Hevner et al. (2004) definition, as it is the most accepted and widely used approach in the context of information systems. DSR seeks to enhance human knowledge through innovative artifacts. Fundamentally, it is a problem-solving paradigm, with the end goal of producing an artifact to a previously identified targeted problem. The insights generated from this research guides us in improving an artifact, making it more effective and efficient in addressing the targeted problem than current solutions. Therefore, improving upon existing solutions or offering a first solution to a problem should be the goal for the built artifacts. (A. Hevner et al. 2010)

Primarily, the creation of DSR artifacts consists of design and evaluation phases, which create a build-and-evaluate loop. The design phase produces an innovative artifact through a sequence of expert activities. After the design phase, evaluation phase is performed for the artifact, which provides feedback and enhances the understanding of the problem. This is then used to improve both the design process and the the quality of the artifact. The build-and-evaluate loop is iterated until the final desired quality of the artifact is achieved. (A. R. Hevner et al. 2004) The researcher performing the loop must remain aware of the evolving of both the design process and the design artifact. As the artifact evolves throughout the iterations, it must remain purposeful, and be evaluated with respect to the utility it provides towards solve its targeted problem (March and Smith 1995).

The DSR artifact's problem space is defined through the environment it resides in (Simon 2019). A. R. Hevner et al. (2004) have designed a framework, which establishes the dependencies between business needs and applicable knowledge in information researc, as seen in Figure 7. For example, when designing a solution for detecting covert channels in network environments, it is important to review people, organizations, and technologies. Networked environments have established protocols and schemas for function, which must be considered when normal and acceptable behaviour is defined. Similarly, OT environments might require high system availability with minimal delays, which can create challenges when trying to integrate high-processing-power algorithms as a defense mechanism.

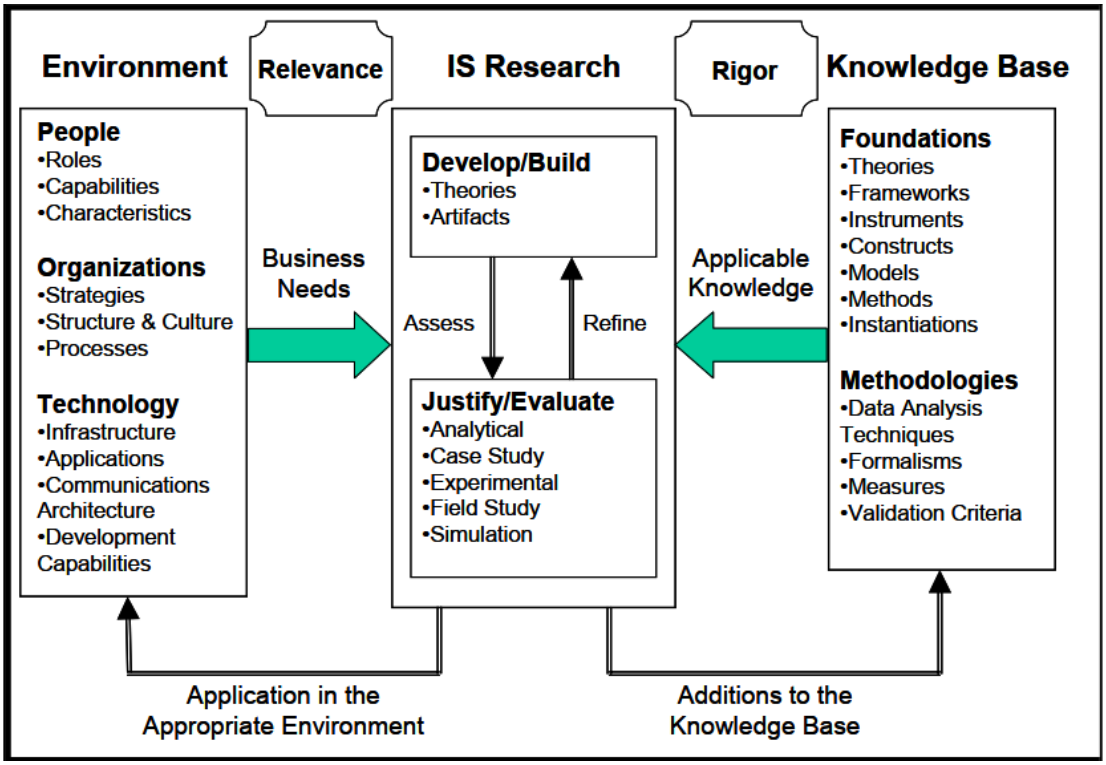


Figure 7: Information Systems Research Framework by A. R. Hevner et al. (2004)

The research questions for the study are the following:

1. Can the initial foothold's backdoor communication detection be improved?
2. Can the detection of defense evasion techniques employed by backdoors be improved through generic covert timing channel feature analysis?

The research question "Can the initial foothold's backdoor communication detection be improved?", is approached through the lens of covert channel detection. An attacker would have to evade normal or bypass defense mechanisms implemented in a well-protected environment, such as authentication systems and remote connection limitations. To bypass these, covert channels may be used to enable information transfer from outside the perimeter to inside the system and vice versa. However, the variance in covert channel communication techniques is a limiting factor for the scope of the study. Additionally, these techniques might have protocol or infrastructure dependence. To combat this, the scope is limited to covert timing channels and two simple IAT techniques, where one is used in design phase and one in evaluation phase. These chosen techniques can be integrated more universally

across different environments, which aids in avoiding the detection mechanism becoming environment dependant. This is formulated in the second research question, which aids in answering the first one: "Can the detection of defense evasion techniques employed by backdoors be improved through generic covert timing channel feature analysis?".

In this study, the problem space for DSR artifact is defined by the issue of generic detection methods. against simple IAT covert channels in network communication. Since network communication transfers significant amount of data, which is difficult for humans to analyse, the detected features would benefit from transferrability to machine learning methods. The framework chosen for the artifact development is human observation of covert channel's effects on packet IATs. Since previous knowledge base of covert channel detection suffers from a lack of effective generic detection methods, evaluation of covert channel techniques effects on channel IATs provides added value towards their development (Caviglione 2021). Therefore, the developed artifact provides information on the covert channel's features in networked environments, which may be used to develop generic covert channel detection methods with machine learning techniques that rely on pattern-based classifiers.

The developed artifact is based upon observable effects and features of a single simple IAT covert channel technique and validated against a second one. Future research could focus on developing the artifact further by improving the detection efficacy of sophisticated IAT covert channel techniques, which conforms its packet delays to the occurring overt network traffic or by increasing the detection capabilities against other simple IAT covert channel techniques.

6 Design Science Artifact Development

Datasets with covert channels are difficult to find, however there are tools developed in scientific papers for injecting covert channels into packet capture. However, these tools require a dataset of normal traffic. The IoT dataset developed by Alsaedi et al. (2020) is used as the overt channel into which various covert channels will be injected. (”The TON_IoT Datasets” 2020) This dataset contains various attack and normal IoT traffic recordings, from which "normal_1.pcap" is used to function as the overt channel traffic recording.

The tool used for covert channel injection in this thesis is CCgen (Iglesias et al. 2022), which has been further developed into CCgen.v2 to include various other injectable covert channel techniques after the original release (Meusburger 2023). The software is Python based and offers wide variety of techniques, which ensures higher variance when developing a generic detection artifact. The software provides two types of injection methods: online and offline. However, given that the dataset utilized in this study is in the .pcap format and is not being replayed, the offline injection method is employed.

CCgen.v2 offline injection embeds a text format covert message into a pre-existing packet capture. It offers filters to target specific source IP address, source port, destination IP address, destination port and IP protocol, thus allowing control on selecting receiver and sender for the covert channel. Additionally, there are injection specific variables, such as `Covert Message`, which is a string message that is converted into binary and injected into the overt channel via the chosen technique, and `Bits`, which determines the set size of bits per covert channel packet. These filters and variables are used to configure the injection of a covert channel between two computers. The general configuration used in all of this thesis' covert channel injections are showcased in Table 4.

Table 4: Covert channel sender and receiver information.

Variable	**Value**
Sender IP Address	192.168.1.190
Receiver IP Address	192.168.1.152

Table 4: Covert channel sender and receiver information.

Sender port	43539
Receiver port	1880
Layer	IP Layer
Bits	1
Covert Message	This is a text that is hidden

All of the injection techniques have their own specific configuration options, which affect how the covert channel performs or communicates. In this study, default configuration parameters were used to prevent the selected technique from being tailored to overt traffic and to target simple covert channels with general detection. Furthermore, this approach helps to eliminate subjective bias in configuration selection and prevents the introduction of personally selected patterns into the analysis. However, offline injection does not emulate covert channels, but rather simulate them. Therefore, designers must be aware in selecting reasonable parameter values, since the software embeds covert channels by overwriting existing IATs. The selected default parameters are showcased in Table 5.

Table 5: CCgen.v2 injection variables (Meusburger 2023).

Technique	**Variable**	**Value**	**Variable**	**Value**	**Variable**	**Value**
Jitterbug	pw2	0.1	prdx	3	pmask	3
Time-to-Live	pTb	0.1	pTinc	0.03	pmask	3

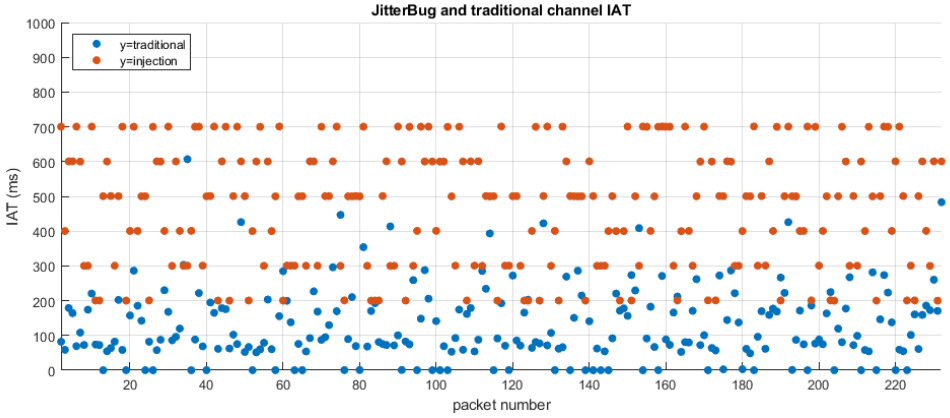
The initial dataset is duplicated and each of these is injected with a different IAT covert channel. This creates a learning dataset that is made out of a single IAT covert channel technique, from which a ruleset for detection is developed and then validated against another technique's features. The initial training technique is Jitterbug Covert Channel (Shah, Molina, Blaze, et al. 2006), and the detection features' generality are validated with a Time-To-Live (TTL) Covert Channel (Zander, Armitage, and Branch 2007b).

The analysis is performed by overlaying two identical overt channel traffic IATs, where one is injected with a covert channel technique and other one remains original. This aids in highlighting the changes that the covert channel technique has on IATs against the control

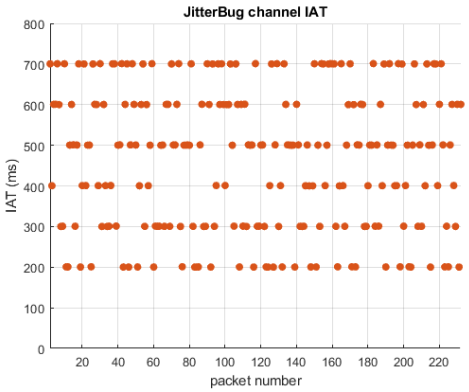
group's IATs. The analysis is primarily performed with three graphs of the IATs that are generated via MATLAB: plot, scatter and sorted plot. Plot graph highlights the changes between adjacent IAT values, which is a possible target in IAT covert channels. Scatter in turn highlights the values of the IATs, which aids in recognising IAT value patterns. Lastly, sorted plot highlights the overall change in value variance when compared to original overt channel. These graphs are analysed by overlaying the exact sections where the covert channel is active, however there is also a reason to consider covert channel's features from defenders' perspective, since defenders do not know exactly where a covert channel resides in network communication. For example, in Zillien and Wendzel (2023) the detection is performed in 2000 packet segments, which is why a separate fourth graph is created with sorted plot to highlight covert channel's features inside a 2000 packet segment when larger quantity of IATs are estimated.

6.1 Artifact Development

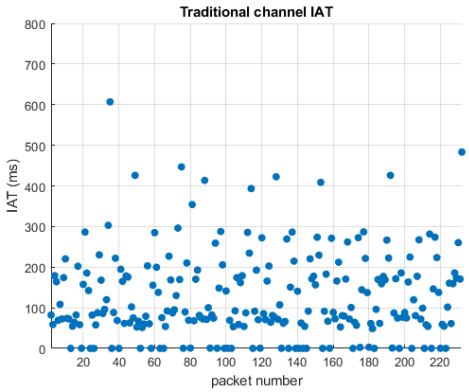
The training dataset's technique is known as JitterBug, which was initially developed by Shah, Molina, Blaze, et al. (2006), and is intended to disrupt legitimate communications across a variety of device types. The original implementation of the technique was in keyboards, where typed secrets could be captured and exfiltrated during the operation of interactive network applications. This technique operates by setting a base sample interval and then introducing a delay to the IDTs. The interpretation of a covert symbol, either "1" or "0", is based on whether a specific IAT is divisible by the entire interval or only half of it. In CCGen.v2 implementation there are two primary variables defining the performance of the covert channel. Variable 'pw2' is used to specify the duration of a half-interval, measured in seconds. On the other hand, variable 'prdx' is an integer that sets a limit on the number of times that a duration of two 'pw2' intervals (full-intervals) can occur between two successive packets. The larger the value of 'prdx', the greater the randomness in the IDTs. In addition to these, a third parameter 'pmask' is required in offline injection. The role of 'pmask' is to define an exact number of decimal places below one second. This precision is important because it allows the function to maintain the original value while simulating a residual transmission delay. The chosen default values are highlighted in Table 5.



(a) JitterBug and traditional IAT



(b) JitterBug IAT



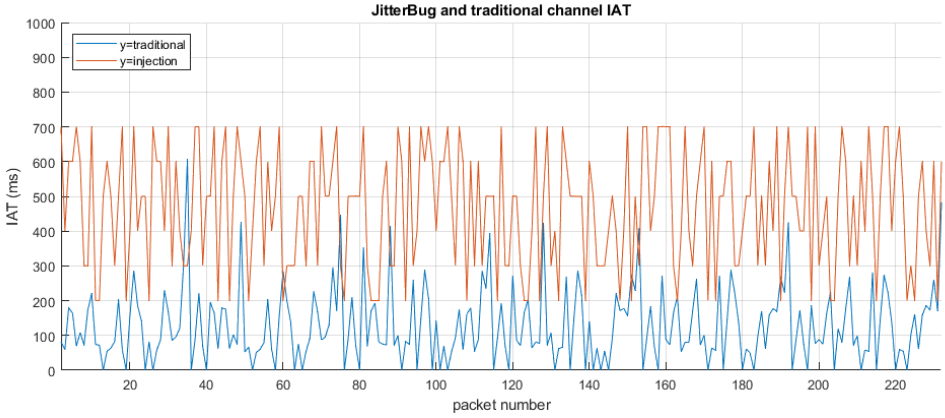
(c) Traditional channel IAT

Figure 8: JitterBug covert channel scatter graphs.

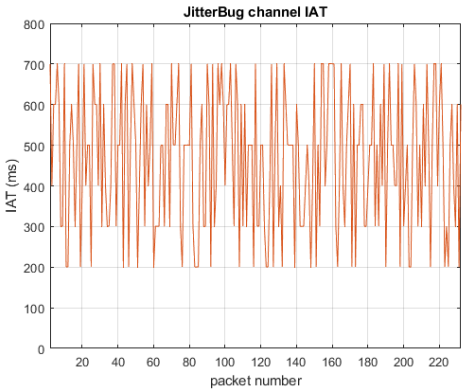
Figure 8 highlights value related features that JitterBug covert channel creates with the selected default injection variables. For example, it raises the floor of packet IATs by delaying all packets to minimum 200ms. This ensures that minimum delay for IAT may land in the required value with a pre-determined delay, e.g., 175ms IAT cannot be delayed to become 150ms. Legitimate channel contains IATs occurring between 0 and 600ms with average IAT of 127ms, whereas JitterBug injection increases this communication to occur between 200ms and 700ms with average of 465ms. Therefore the total range of IATs for legitimate channel is 600ms, in comparison to covert channel's 500ms. Additionally, in legitimate channel weights the communication towards the minimum IATs, rather than the average between minimum and maximum IATs of covert channel. The legitimate channel has 79,3% (184 out of 232) IATs under lower third of the total range. On the other hand, Jitterbug has 31,9% (74

out of 232) under lower third of the total range.

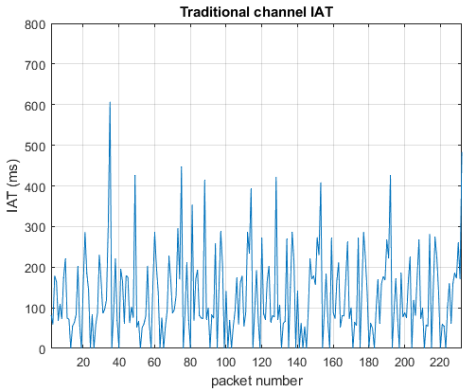
Another feature of JitterBug is that the communication is extremely symmetrical, and the values are predictable. This is because the values occur at 100ms intervals and do not vary until the microsecondth. Additionally JitterBug IAT values range from 200ms to 700ms, with 46 near identical IATs at the maximum value. Conversely, legitimate communication contains less high values near the maximum IAT, where maximum IAT is 607ms and seven other high points occur between 500ms and 400ms. However, where JitterBug and legitimate communication overlap is in the quantity of communication near the minimum IATs. In this area, JitterBug has 34 IATs, while legitimate communication 40 IATs. Additionally, both channels contain similar IAT values near these points, each within a millisecond from their respective minimum IATs. However, since they have different minimum IATs, this further emphasizes JitterBug channel's lack of IATs under 200ms.



(a) JitterBug and traditional channel IAT

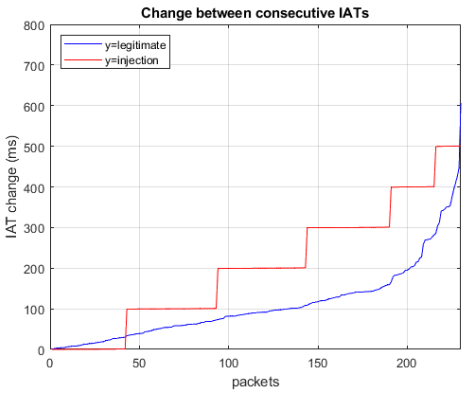


(b) JitterBug IAT

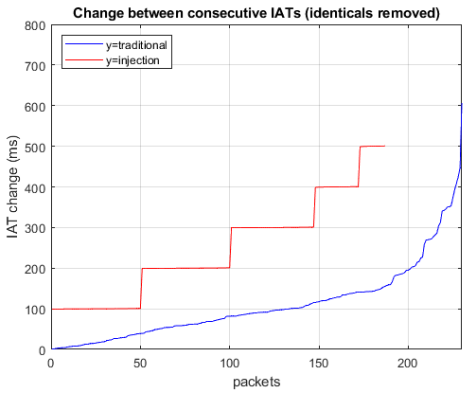


(c) Traditional channel IAT

Figure 9: JitterBug covert channel plot graphs.



(a) Consecutive identicals included



(b) Consecutive identicals removed

Figure 10: JitterBug covert channel consecutive IAT millisecond change.

Figure 9 highlights that the pattern of occurence of IATs between legitimate and JitterBug channel differs. This is further showcased in Figure 10, which depicts the absolute change between consecutive IAT values. Patterns are approached through the millisecond change that occurs between consecutive IATs, be it negative or positive. Therefore, *change* in this describes means the absolute millisecond difference between consecutive IATs. The dataset is manipulated to remove the initial 0ms packet timing for consecutive IAT change analysis. This initial capture functions as the transition from legitimate channel to covert channel, which is not the focus of the study. The interest is in covert timing channel's pattern of IAT delays, which excludes the IATs before the start of covert communication. Since the covert channel may use identical interval to transfer consecutive secret symbols, it reduces the range of variance when minimal changes between consecutive IATs occur. This is seen by the fact that out of 232 IATs, 43 remained within a millisecond of the previously occurred IAT. Additionally, when there is meaningful change in consecutive IATs, it occurs in increments of 100ms, ranging from 100ms to 500ms change. The average change in consecutive IATs of JitterBug was 203ms. This average becomes larger when the negligible changes are removed from the dataset, leading to average change to become 249ms, which is depicted in Figure 10 (b).

Another point to consider is the relative amount of change occurring in respect to the total range of the channel's IAT values. For this, we consider the total range of 500ms for JitterBug and 600ms for legitimate channel. We are interested in the lower third of this range, for JitterBug's this is 166ms, and for legitimate channel this is 200ms. For JitterBug, only 40,4% (93 out of 230) of the IAT changes are under 166ms. This percentage drops further to 26,7% (50 out of 187) when near-identical abnormal changes are excluded from the dataset. As seen in Figure 10, while JitterBug channel has higher rate of occurence for lower changes than higher ones, the rate of change increase when compared to legitimate channel is apparent. Especially, when considered in the context of lower third of the total range, the occurring changes in JitterBug are more likely to be large than small.

Legitimate communication on the other hand has constant varying oscillation of lower IAT changes, since 87,3% (201 out of 230) of IAT changes occur in the lower third of the range (200ms), with average change of 110ms. Therefore, legitimate channel's IAT changes will

very likely occur at lower increments. However, there are irregular spikes that push the IAT change over 200ms from time to time. When compared, legitimate communication's average IAT change of 110ms is much smaller than JitterBug's 203ms or 248ms, leading to it being a prominent disparity. Additionally, when average IAT change is compared to the total IAT range, the legitimate channel's average IAT change constitutes 18.1%, while the covert channel's is either 40.6% or 49.6%, dependant upon the inclusion or exclusion of near-identical consecutive IATs.

Analysis of Figure 9 and Figure 10 highlight four JitterBug features when compared to legitimate traffic: There is large amount of near-identical consecutive IATs leading to negligible IAT changes, the changes occur as set increments, the average IAT change is large relative to total IAT range, and there is abnormally balanced spread of IAT changes across the total IAT range.

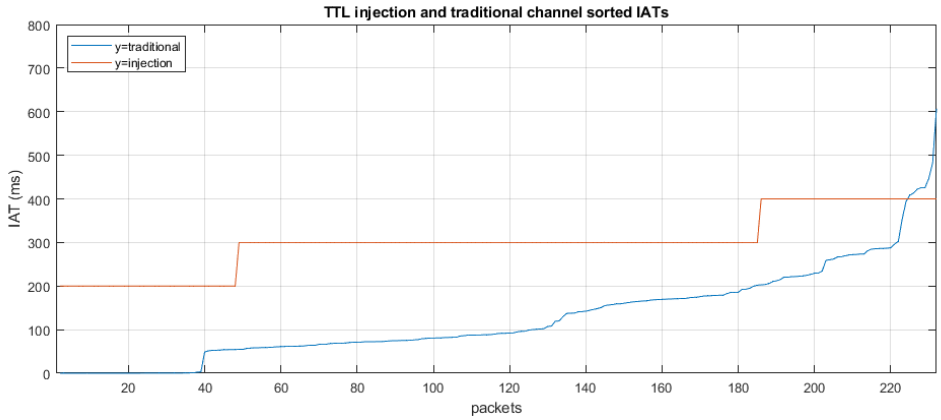


Figure 11: JitterBug covert channel sorted plot graph.

Figure 15 illustrates the relationship between IAT milliseconds and packets, sorted by the IATs in ascending order. Focus of this graph is the established curve of traffic IATs. There are various distinctive features of the JitterBug channel's IATs when sorted and compared to the IATs of the legitimate channel. These include significantly higher IAT values, an absence of traffic near 0ms and below 200ms, a greater and more systematic increase due to 100ms increments, and a high number of IATs at the maximum of the IAT range. Additionally, since the increase is systemic and occur predominantly at specific values, this leads to the lack of IATs between specific ranges.

On the other hand, the legitimate channel has only 10 packets with over 300ms IAT, which leads to the curve spiking heavily towards the end in comparison to otherwise gradual increase in the curve. Similarly, the legitimate channel has 39 packets below 3ms, which are abnormal compared to the otherwise gradual increase. Therefore, legitimate traffic centres the IATs towards specific range, such as 50ms to 300ms, where vast majority of the communication IATs exist. It may still have erratic IATs at the extremes, but these exist as rare cases, rather than systemic, as seen in the sudden increase at the end of the curve. While the legitimate channel may have other sudden increases in smaller dataset analysis, such as in this case the segment between 119ms and 129ms, the increments of increase are not as extreme as 100ms, and are rare when compared to the otherwise gradual incease of the curve.

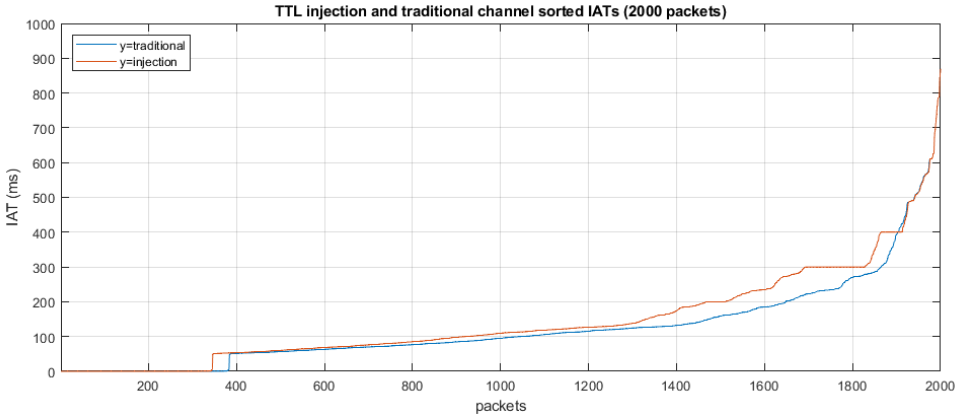


Figure 12: JitterBug covert channel sorted plot graph of 2000 packet segment.

Last graph to analyse is a 12 containing 2000 packets, which replicates a case where covert channel's possible features are searched when it occurs in a larger segment of communication. This is due to the fact that defender will not be able to split communication into pre-existing segments where only the covert channel communication occurs. The covert channel communication occurs over 232 packets, which is 11.6% of the overall communication in a 2000 packet segment. JitterBug, like all covert channels, functions by introducing delays to the predicted packet IATs. This means that the presence of a greater IAT is contingent on the absence of a lower IAT. In other words, for a larger IAT to be possible, a smaller IAT must be deliberately avoided.

When an overt channel containing a small section of JitterBug covert channel communication

is examined against legitimate channel's communication in the context of a larger dataset of IATs, there must be less lower IAT values due to the requirement to delay the IATs. This characteristic is clearly demonstrated in the 12. In order to accommodate the necessary IATs, the overt channel that has been injected with the JitterBug exhibits a gradual, yet greater, increase in IATs when compared to the legitimate overt channel. Furthermore, the previously identified systemic incremental increases in IATs are also evident in the larger dataset. Similarly, a number of previously low IATs at the beginning of the curve have been delayed. The delayed IATs create a shift in the IAT distribution, further highlighting the operational characteristics of JitterBug and the disparity between the two channels. This disparity is apparent in smaller dataset with only the covert channel and translates to the larger dataset in a significant way.

Statistical analysis is possible for the curves, but it may not provide consistent general features for covert channel detection without additional false-positives. For example, the area between the JitterBug and the overt channel's curves is $75907ms^2$. This functions as a statistical measurement for the delay that the JitterBug inflicts to the overt channel. However, this area does not function consistently across different techniques. The discrepancy arises from the fact that the area changes in relation to the delay required for the intended covert symbol. In other words, a larger delay leads to a greater difference between the covert and overt channel curves, which in turn results in a larger area between the curves. Even a simple covert channel technique may produce only a small discrepancy between covert and overt channel curves, if the covert channel does not require large delays for the transfer of covert symbols. To make area between the two curves consistent for detection with reduced false-positives, a large training dataset must be used to determine the maximum area in which overt channel communication may occur, which could function as a threshold. Anything above this area would be considered an anomaly. However, this feature alone would not be enough, since there is a possibility that the covert channel targets a low area section in communication, and introduces low enough delays to remain within the acceptable area threshold.

The features found from graphs and their statistical analysis are collected in Table 6, together with their respective sources. The feature thresholds are left abstract, since the focus is in validity of the feature, rather than optimisation of the threshold of a known feature for au-

tomation. To enable the estimation of the threshold, the features must be validated first with various covert channel techniques, and there needs to be large enough dataset to optimise a threshold that would avoid unnecessary false-positives while detecting the injected covert channels from normal traffic.

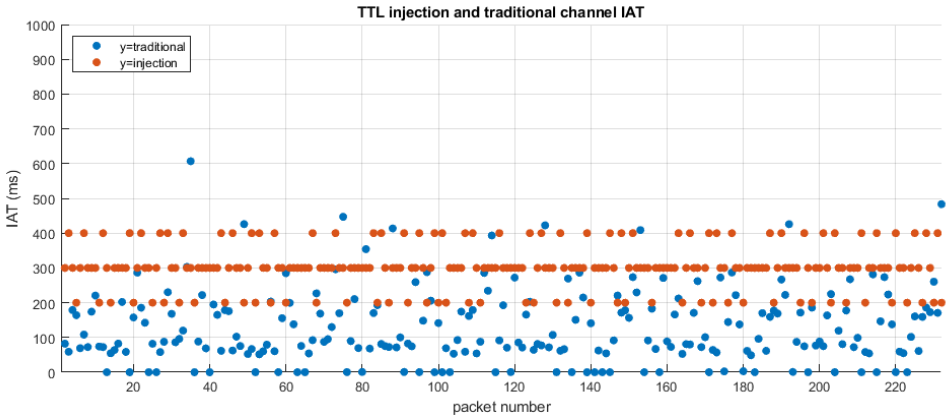
Table 6: Covert Channel Features

Feature	**Source graph**
1. Increase in IAT minimum floor	scatter, plot, sorted plot
2. Increase in IAT maximum ceiling	scatter, plot, sorted plot
3. Increase in IAT average	scatter, sorted plot
4. High repetition of near identical IATs	scatter, plot, sorted plot
5. High repetition at IAT maximum ceiling	scatter, plot, sorted plot
6. Communication is not concentrated near IAT floor	scatter, plot, sorted plot
7. Systemic IAT values	scatter, plot, sorted plot
8. Lack of IATs between set IAT points	scatter, plot, sorted plot
9. Identical or near identical consecutive IATs	consecutive IAT plot
10. Increase in average consecutive IAT during changes	consecutive IAT plot
11. Increase in large consecutive IAT during changes	consecutive IAT plot
12. Increase in IAT area	sorted plot 2000

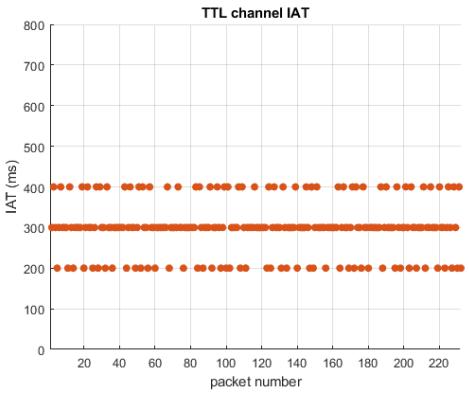
6.2 Artifact Validation

The proposed general covert timing channel features presented in Table 6 are validated step by step, with evaluation for each feature based on respective graphs or statistics formulated from the graph data. The technique employed here during validation is inspired by the modulation of the TTL field, a method proposed by Zander, Armitage, and Branch (2007b). However, in this context, the technique is applied to IATs instead (Meusburger 2023). The technique uses two parameters to control its function: tb (the base IDT at which packets are sent) and tinc (the time that is added or subtracted from tb). In CCgen.v2 these are variables pTb and pTinc. When the covert symbol to be transmitted is 0, the IDT is set to 'tb'. However, if the covert symbol to be transmitted is 1, the IDT is adjusted to 'tb' plus

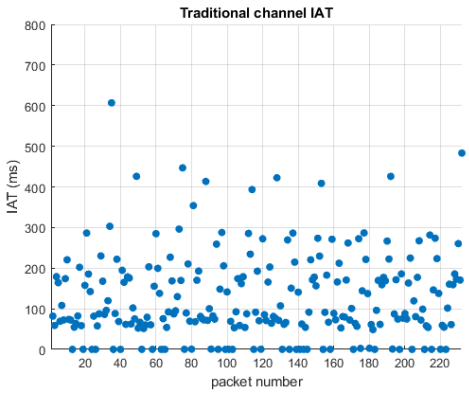
or minus ‘tinc’. For consecutive occurrences of the symbol 1, the operations of addition and subtraction are alternated. Similarly to training injection, the injection function requires a third parameter when it is used in offline mode, which is referred to as ‘pmask’. The chosen default values are highlighted in Table 5.



(a) TTL and traditional channel IAT

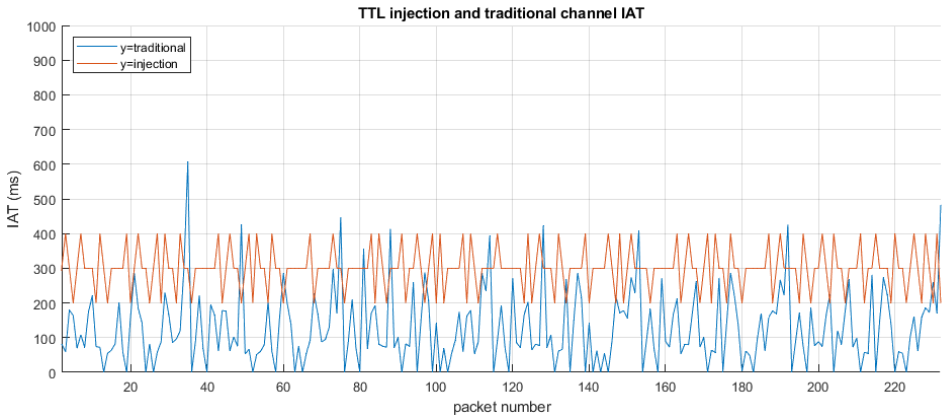


(b) TTL IAT

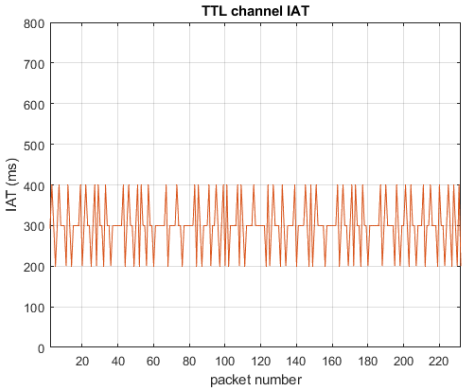


(c) Traditional channel IAT

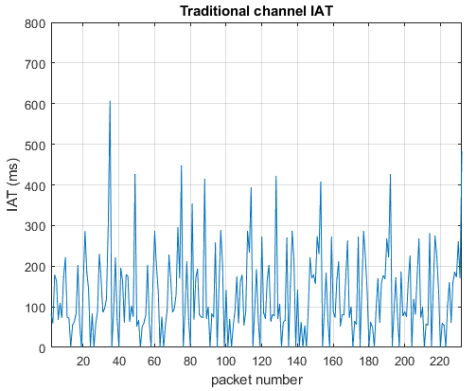
Figure 13: TTL covert channel scatter graphs.



(a) TTL and traditional channel IAT



(b) TTL IAT



(c) Traditional channel IAT

Figure 14: TTL covert channel plot graphs.

Feature 1. is validated through the inspection of the minimum IAT of the dataset, in addition to inspection of Figure 13 and Figure 14. In all, the IAT floor is increased from 0ms to 200ms, confirming the suggested feature. *Feature 2.* is validated through the inspection of the maximum IAT of the dataset, in addition to Figure 13 and Figure 14. In all, the maximum for TTL injection technique is reduced to 400ms, from the previous 607ms, contradicting the suggested feature. *Feature 3.* is validated statistically. The average IAT of the legitimate channel remains 127ms, whereas the TTL injection increases the average IAT to 299ms, confirming the suggested feature.

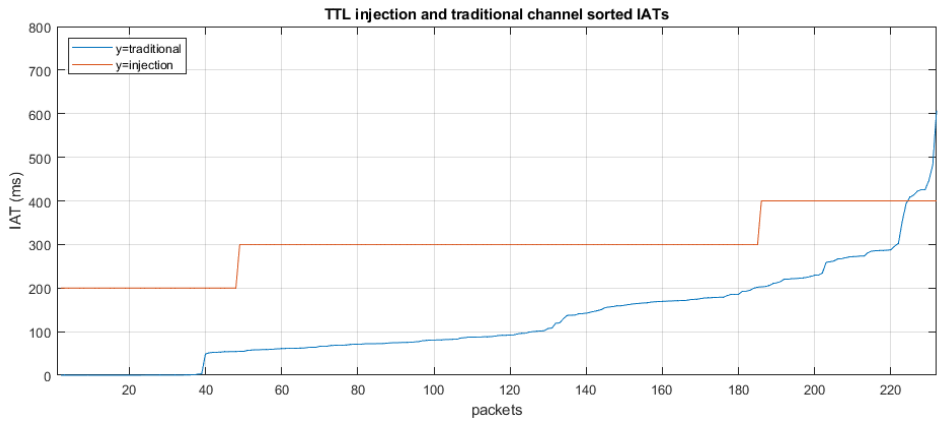
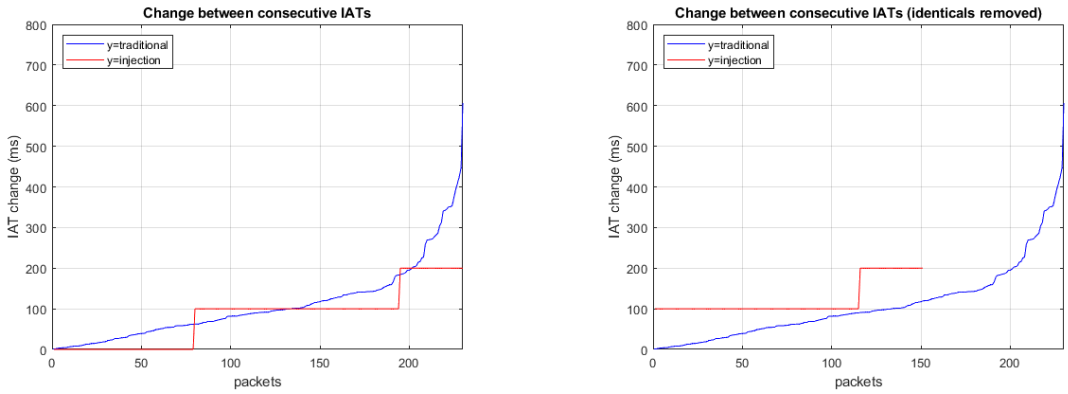


Figure 15: TTL covert channel sorted plot graph.

Feature 4. is confirmed through visual graph analysis of Figure 15. This showcases TTL covert channel using only three IATs, confirming the suggested feature. *Feature 5.* is similarly confirmed through visual graph analysis of Figure 15. TTL covert channel has 46 IAT occurrences at the IAT maximum ceiling, confirming the suggested feature. *Feature 6.* is evaluated by considering the lower third of the total IAT range as the range for "near IAT floor". For TTL covert channel, this is 200ms to 266ms. Only 20,6% (48 out of 232) of the TTL covert channel communication occurs at the lower third of the range, compared to 79,3% of the legitimate channel, confirming the suggested feature. *Feature 7.* is already validated partially by confirming features 4. and 5., but this is similarly seen in Figures 13 and Figures 14. TTL covert channel exhibits only 3 unique IATs over 232 packets, which alternate, thus confirming the suggested feature. *Feature 8.* is partially confirmed by feature 1. in removing IATs occurring in the range between 0ms and 200ms, in addition to feature 4. and feature 7. Closer inspection of Figure 13, Figure 14, and Figure 15, showcase that since the IATs only occur as 200ms, 300ms and 400ms, there are no IATs between these points, which confirms the covert channel feature.



(a) Consecutive identicals included

(b) Consecutive identicals removed

Figure 16: TTL covert channel consecutive IAT millisecond change.

Feature 9. is validated with Figure 16 and statistical analysis of the change occurring between two consecutive IATs. The graph highlights abnormal amount of no change between consecutive IATs, which is confirmed with analysis as 79 consecutive IATs with no change, which confirms the suggest feature. *Feature 10.* is approached through statistical analysis to determine the possible discrepancies between the consecutive IAT change averages. The average change between consecutive IATs for legitimate channel is 110ms, whereas for TTL covert channel it is 81ms. This provides with absolute values an average difference of 29ms and they are within 35% of each other, leading to a plausible difference when considered in an environment where variation for IATs may occur. However, if the abnormal consecutive identical IATs are removed, the TTL channel average change becomes 124ms, leading to only 15% difference between the two channels. Since the feature wording focuses on the increase in the average consecutive IATs on the premise that consecutive are not identical, the 15% increase is selected. This confirms the suggested feature, but optimisation of acceptable threshold would be required due to environment induced variation. *Feature 11.* is validated by considering the range of both channels and the frequency at which consecutive IAT changes exceed one-third of the range. Since the total range of IAT is 200ms, one third of this range is 66ms For the legitimate channel, this condition is met in 12.6% of cases (29 out of 230), while for the TTL channel, it is met in 65,6% of cases (151 out of 230). If instances where no change occurs between consecutive IATs are excluded from the dataset,

depicted in Figure 16 (b), the percentage of TTL channel changes exceeding one-third of the channel IAT range increases to 100%. Since the feature focuses on cases where there are changes between consecutive IAT, this feature is confirmed.

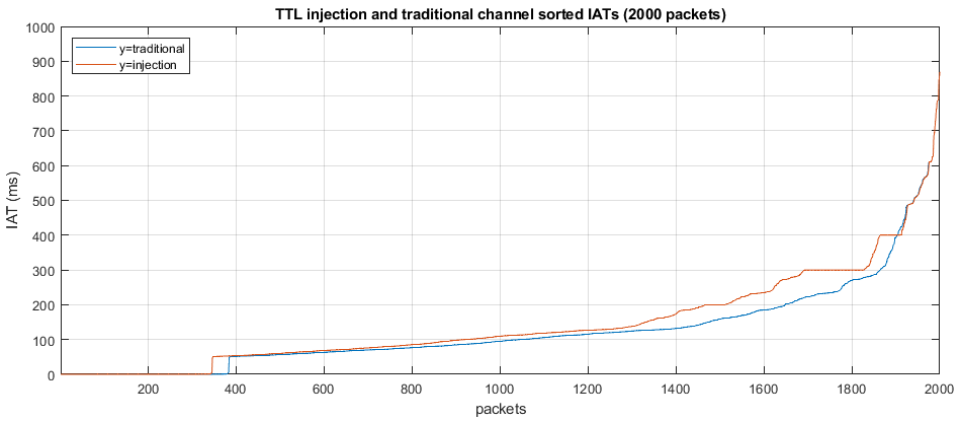


Figure 17: TTL covert channel sorted plot graph of 2000 packet segment.

Feature 12. is validated through Figure 17, which depicts 2000 IAT segment, and by calculating the area between the two points where the curves intersect. The selected intersection points on x-axis are 68 and 1904. The area between the two curves is $37032ms^2$, which is only 48,8% of the increase in area found during analysis of JitterBug channel, which functioned as the basis for the feature. The TTL channel increases the IAT area, which confirms the feature, but further optimisation is required to determine acceptable threshold. Additionally, since the offline injection reduced certain pre-existing IATs, there are sections where TTL covert channel curve is below the legitimate channel curve in Figure 17. Since covert channels function by delaying pre-existing communication, rather than accelerating it, this is an error in offline injection when compared to real life environment. This would be avoided if the pre-existing parameters were selected through analysis of the legitimate channels IATs, but this would lead to a more sophisticated covert channel, since it is fitted to existing communication. Therefore the area between the TTL channel and legitimate channel would be greater in reality than in this instance.

The general covert timing channel features, sources and validation results are summarized in Table 7. The results are binary true or false depending on the feature, with additional metric of partial to indicate possible ambiguity in validation threshold. Ambiguity is accepted as

the framework for development is based upon human observations and will contain natural subjective estimates for the acceptable threshold, which is acceptable in systems that are built based on human expertise. These thresholds would improve in accuracy through iterative process and by considering environmental properties in a larger dataset.

Table 7: Validated Artifact

Feature	**Source graph**	**Result**
1. Increase in IAT minimum floor	scatter, plot, sorted plot	True
2. Increase in IAT maximum ceiling	scatter, plot, sorted plot	False (partial)
3. Increase in IAT average	scatter, sorted plot	True
4. High repetition of near identical IATs	scatter, plot, sorted plot	True
5. High repetition at IAT maximum ceiling	scatter, plot, sorted plot	True
6. Communication is not concentrated near IAT floor	scatter, plot, sorted plot	True
7. Systemic IAT values	scatter, plot, sorted plot	True
8. Lack of IATs between set IAT points	scatter, plot, sorted plot	True
9. Identical or near identical consecutive IATs	consecutive IAT plot	True
10. Increase in average consecutive IAT during changes	consecutive IAT plot	True (partial)
11. Increase in large consecutive IAT during changes	consecutive IAT plot	True (partial)
12. Increase in IAT area	sorted plot 2000	True

7 Results and Discussion

Of the twelve features tested, eleven were validated as true, while one was determined to be false. Additionally, there were two instances (features 10 and 11) where they were accurate for the chosen descriptor, but threshold optimisation was deemed as especially necessary to avoid possible false positives in possible expert systems. The initial validated generic covert channel features are applicable to JitterBug and TTL covert timing channels. Two new generic detection properties were found from the feature analysis: the absolute change between consecutive IAT values and the change distribution relative to the total IAT range. These data type provides information on the pattern of occurrence for IAT, which is manipulated during covert timing channel injection through IAT delays. These properties could be a focal point in future statistical analysis detection methods, since they describes how IAT values occur in relation to each other. Notably, previous research has primarily concentrated on the sequence of IAT values or the relative differences between IAT values, rather than the absolute change occurring between consecutive IAT in the sequence. The validated generic simple covert timing channel features provide properties that may be used in the development of future expert systems and cybersecurity automation.

There were two cases in the artifact validation (features 2 and 12) that were directly affected by offline injection operation logic. The invalid general feature "Increase in IAT maximum ceiling", could be argued to exist in TTL covert timing channel due to fundamental operation logic of an accurate covert channel. In offline injection of CCgen.v2, the .pcap file is manipulated according to the technique and selected parameters, while not taking into consideration the original IAT values. However, a pre-existing IAT may not be accelerated past hardware and software limitations that exist in real life. In the performed offline injection this illogical acceleration is considered as an acceptable loss in message accuracy for a tradeoff in channel stealth, which places TTL covert timing channel IAT maximum below the normal channel's maximum. Similarly, the increase in IAT area for TTL covert timing channel was reduced due to some sections of channel's curve existing below normal channel, since some IATs in covert channel were accelerated due to offline injection. Potential issues could have been circumvented by conducting an analysis of the overt channel's IAT patterns prior to injection

and selecting parameters that align with the overt channel's IAT ranges. However, these steps would have introduced increased the channel complexity, which contradicts our research objective. Moreover, this scenario would necessitate the malicious attacker to possess specific data related to the system's communication, rather than merely hardcoding default values into the deployed malware. This distinction is crucial, as our research objective is to detect initial backdoor communication via covert channels, a goal that would be undermined by the use of parameters that were based on overt channel IAT analysis.

Given the reduced IAT observed with offline injection, a possible future research topic in covert channel development is the acceleration of estimated IATs. In some cases, it's theoretically possible to reduce the estimated IAT by subdividing the packets scheduled for transmission into smaller units, thereby reducing the transmission delay. However, this is protocol and system dependant. This would not work in cases where the system requires the sent packets to be in set size. Another possible acceleration is through the use of Quality of Service (QoS) settings and queuing rules, since they determine how data is treated as it travels over a network. By assigning a higher priority to packets involved in covert channels, its possible to influence the timing of packet delivery. This could potentially lead to a reduction in the IAT, compared to original estimations. Future research could investigate these topics and validate the discussed methods and theories. This would also validate the use of offline injections as covert channel simulators in academic research.

Common use of offline injections in covert channel detection literature highlights the possible gap in covert channel development and detection research. It would be beneficial to develop covert timing channels in practice with real world test scenarios. Moreover, the practical implementation of covert timing channel techniques offers added value, particularly in terms of the methods employed to generate delays. For example, delays in network traffic could be in the form of processing, queuing, transmission or propagation delay, which may leave fingerprints on the host machine when done by malware. When a specific delay method is used, it would leave additional fingerprints and logs into the system, which supports holistic approach towards covert channel detection within a vulnerable environment. Similarly, when covert timing channels are tested in real world test scenarios, the environmental effects on channel are documented and taken into account. For example, crossing oceans can have

profound impact on covert timing channel transmission speeds and require specific techniques with a corresponding robustness to combat uncertain channel delays (Berk, Giani, and Cybenko 2005). Therefore when a theorised covert timing channel technique is developed, it would be beneficial to analyse and test how an implementation of said technique would work in real world environment.

The literature review established need for generic detection methods against covert channels due to large amount of protocol and system dependant patterns. Additionally, the developed detection methods are often very dataset dependent and may fail to identify specific patterns when the parameters and properties of these patterns themselves are manipulated (Zillien and Wendzel 2023). Furthermore, the history of information hiding taxonomy, and the ambiguity between storage and timing covert channels are identified issues in the research literature. These issues have only recently been addressed through the development of a revised steganography taxonomy, which aims to avoid overlaps in terminology and definitions. With the implementation of an improved taxonomy, the development of generic detection methods could potentially be more accurate. This is due to the fact that the techniques associated with covert channels would not overlap with various patterns, thereby enhancing the precision of both the development and validation processes. While the detection and development of covert channels are well-researched topics in academia, the development of prevention and elimination countermeasures against them is not as prominent. This is particularly relevant when covert channels are used during the initial backdoor's network communication for C2 actions. Since the command related communication does not require high channel capacity, the amount of covert communication for detection techniques to find is low. The findings of the review suggest that the prevention of such communication is not prioritised in academic research compared to detection.

This research is constrained by its partial adherence to the design science methodology, particularly in the implementation of the feedback loop between the design and evaluation stages. The choice to limit the number of iterations was influenced by time constraints and the understanding that an exhaustive analysis and validation of all possible generic covert channel features would necessitate extensive data collection. This difficulty is further intensified by the need to develop covert channel injection methods with the required configura-

bility to facilitate the required data generalization.

Additional research should be performed to enhance the emerged generic simple covert timing channel features further to reach a definite conclusion about the applicability of the features in other covert timing channel techniques. Furthermore, covert timing channels function by various parameters, which may lead to different features when the channel is injected into a pre-existing overt channel. Therefore, a continued iterative process to develop and and validate generic covert channel features further is recommended. This process includes elements such as, adjustment of the pmask parameter for covert timing channels that prioritise high stealth over robustness, introduction of additional simple and complex covert timing channels with different injection parameters, testing of these channels across a variety of datasets from different network environments, and testing in online and offline injection settings.

8 Conclusion

This research resulted in a design science artifact. The artifact aims to support expert system development through human observation based analysis by providing features that function as fingerprints for covert channel injection in overt channel traffic. The literature review highlighted issues with historical covert channel definitions and pattern-based taxonomies. It also pointed out that detection methods are often overspecialized or overfitted for specific datasets or techniques.

As a part of analysis of generic covert channel features, twelve features were identified. Eleven of the twelve generic features were considered valid in terms of accuracy. The last generic feature did not meet the required criteria. However, its validation was considered inconclusive due to the operation mechanism of offline injection. The identified generic features were based on JitterBug covert timing channel, which were validated with TTL covert timing channel. During the feature analysis, two new types of covert channel features were found: the absolute change between consecutive IAT values and the change distribution relative to the total IAT range. These data type provides information on the pattern of occurrence for IAT, which is manipulated during covert timing channel injection through IAT delays. Previous research focuses on the IAT sequences or the relative change between consecutive IAT value pairs.

The research questions of the study revolved around improving initial backdoor detection, specifically, (1) can the initial foothold's backdoor communication detection be improved? This was approached through covert channel detection, since they may be employed by backdoors as a defense evasion technique to avoid detection. This formulated the second question (2) can the detection of defense evasion techniques employed by backdoors be improved through generic covert timing channel feature analysis. Analysis of generic covert timing channel features provides concrete data from which to develop expert systems, which may be implemented by SOC operators or in NIDS devices. Lack of generic detection techniques for covert channels is a crucial and identified issue in previous research, as the vast amount of protocols and systems make it impossible to create specialised detection methods for every covert timing channel in every unique environment. Simple covert timing channels

may be detected through generic detection methods, and they are more likely to function as the initial backdoor communication defense evasion technique when previous reconnaissance on the system's network communication isn't performed. By identifying and validating twelve generic covert channel features, the backdoor remote communication detection is enhanced. These features provide a framework for developing covert channel detection techniques, which are defense evasion techniques employed by malicious entities.

A key limitation of the study was the limited number of iterations for development and validation of generic covert channel feature artifact, in addition to using offline injection due to the lack of available training datasets. Therefore, the artifact would benefit from future iterations to enhance and validate generality of the features further. This could include topics such as, increase in variety of the training dataset's network environment, adjusting robustness and stealth of the covert timing channels, introducing additional simple and sophisticated covert channels. Additionally, several other future research topics were highlighted during artifact validation and literature review. The covert channel research in academia relies heavily on offline injections and would benefit from real world online injection, which would enable more holistic covert channel detection approaches in networked environments. Similarly, researching the possibility of reducing estimated IATs through packet subdivision and quality of service setting and queuing rule manipulation could enable more sophisticated covert channels, and provide insight towards their detection. With the addition of such scientific research, the use of covert channels by backdoors for command and control and data exfiltration would be more detectable, and therefore increase the security in critical environments.

Bibliography

Alexander, Otis, Misha Belisle, and Jacob Steele. 2020. "MITRE ATT&CK for industrial control systems: Design and philosophy". *The MITRE Corporation: Bedford, MA, USA* 29.

Alsaedi, Abdullah, Nour Moustafa, Zahir Tari, Abdun Mahmood, and Adnan Anwar. 2020. "TON_IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems". *Ieee Access* 8:165130–165150.

Anderson, Ross, Serge Vaudenay, Bart Preneel, and Kaisa Nyberg. 1996. "The newton channel". In *International Workshop on Information Hiding,* 151–156. Springer.

Archibald, Rennie, and Dipak Ghosal. 2014. "A comparative analysis of detection metrics for covert timing channels". *Computers & security* 45:284–292.

Assante, Michael J, and Robert M Lee. 2015. "The industrial control system cyber kill chain". *SANS Institute InfoSec Reading Room* 1:24.

Bahrami, Pooneh Nikkhah, Ali Dehghantanha, Tooska Dargahi, Reza M Parizi, Kim-Kwang Raymond Choo, and Hamid HS Javadi. 2019. "Cyber kill chain-based taxonomy of advanced persistent threat actors: Analogy of tactics, techniques, and procedures". *Journal of information processing systems* 15 (4): 865–889.

Berk, Vincent, Annarita Giani, and George Cybenko. 2005. "Detection of covert channel encoding in network packet delays".

Bristow, Mark. 2021. *A sans 2021 survey: Ot/ics cybersecurity.* Technical report. SANS Institute.

Cabuk, Serdar. 2006. "Network covert channels: Design, analysis, detection, and elimination". PhD thesis, Purdue University.

Cabuk, Serdar, Carla E Brodley, and Clay Shields. 2004. "IP covert timing channels: design and detection". In *Proceedings of the 11th ACM conference on Computer and communications security,* 178–187.

Carrara, Brent, and Carlisle Adams. 2016a. ”A survey and taxonomy aimed at the detection and measurement of covert channels”. In *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security,* 115–126.

———. 2016b. ”Out-of-band covert channels—A survey”. *ACM Computing Surveys (CSUR)* 49 (2): 1–36.

Caviglione, Luca. 2021. ”Trends and challenges in network covert channels countermeasures”. *Applied Sciences* 11 (4): 1641.

Chourib, Mehdi. 2019. ”Detecting selected network covert channels using machine learning”. In *2019 International Conference on High Performance Computing & Simulation (HPCS),* 582–588. IEEE.

”Cost of a data breach 2022”. 2022. Visited on February 17, 2023. https://www.ibm.com/reports/data-breach.

Cover, Thomas M. 1999. *Elements of information theory.* John Wiley & Sons.

”Cybercrime To Cost The World $10.5 Trillion Annually By 2025”. 2020. Visited on February 17, 2023. https://www.ibm.com/reports/data-breach.

Dhirani, Lubna Luxmi, Eddie Armstrong, and Thomas Newe. 2021. ”Industrial IoT, cyber threats, and standards landscape: Evaluation and roadmap”. *Sensors* 21 (11): 3901.

Eckhardt, Philipp, and Anastasia Kotovskaia. 2023. ”The EU’s cybersecurity framework: the interplay between the Cyber Resilience Act and the NIS 2 Directive”. *International Cybersecurity Law Review* 4 (2): 147–164.

Al-Eidi, Shorouq, Omar Darwish, Yuanzhu Chen, and Ghaith Husari. 2020. ”SnapCatch: automatic detection of covert timing channels using image processing and machine learning”. *IEEE Access* 9:177–191.

”Finland in the Digital Economy and Society Index”. 2022. Visited on February 20, 2023. https://ec.europa.eu/newsroom/dae/redirection/document/88700.

Gianvecchio, Steven, and Haining Wang. 2007. ”Detecting covert timing channels: an entropy-based approach”. In *Proceedings of the 14th ACM conference on Computer and communications security,* 307–316.

———. 2010. ”An entropy-based approach to detecting covert timing channels”. *IEEE Transactions on Dependable and Secure Computing* 8 (6): 785–797.

Gianvecchio, Steven, Haining Wang, Duminda Wijesekera, and Sushil Jajodia. 2008. ”Model-based covert timing channels: Automated modeling and evasion”. In *Recent Advances in Intrusion Detection: 11th International Symposium, RAID 2008, Cambridge, MA, USA, September 15-17, 2008. Proceedings 11,* 211–230. Springer.

Girling, C. Gray. 1987. ”Covert Channels in LAN’s”. *IEEE Transactions on software engineering* 13 (2): 292.

Gligor, Virgil D. 1994. *A guide to understanding covert channel analysis of trusted systems.* Volume 30. National Computer Security Center.

Hartmann, Laura, Sebastian Zillien, and Steffen Wendzel. 2021. ”Reset-and Reconnection-based Covert Channels in CoAP”. In *European Interdisciplinary Cybersecurity Conference,* 66–71.

Haseeb, Junaid, Masood Mansoori, and Ian Welch. 2020. ”A measurement study of iot-based attacks using iot kill chain”. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom),* 557–567. IEEE.

Heda, Yogesh, and Rinku Shah. 2015. ”Covert channel design and detection techniques: a survey”. In *2015 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT),* 1–6. IEEE.

Hevner, Alan, Samir Chatterjee, Alan Hevner, and Samir Chatterjee. 2010. ”Design science research in information systems”. *Design research in information systems: theory and practice,* 9–22.

Hevner, Alan R, Salvatore T March, Jinsoo Park, and Sudha Ram. 2004. ”Design science in information systems research”. *MIS quarterly,* 75–105.

Hu, Vincent C, Rick Kuhn, Dylan Yaga, et al. 2017. ”Verification and test methods for access control policies/models”. *NIST Special Publication* 800:192.

Hutchins, Eric M, Michael J Cloppert, Rohan M Amin, et al. 2011. ”Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains”. *Leading Issues in Information Warfare & Security Research* 1 (1): 80.

Iglesias, Félix, Fares Meghdouri, Robert Annessi, Tanja Zseby, et al. 2022. ”CCgen: injecting covert channels into network traffic”. *Security and Communication Networks* 2022.

Jain, Jitendra, and RP Parashu. 2017. ”A recent study over cyber security and its elements”. *International Journal of Advanced Research in Computer Science* 8 (3): 791–793.

Kang, Myong H, and Ira S Moskowitz. 1993. ”A pump for rapid, reliable, secure communication”. In *Proceedings of the 1st ACM Conference on Computer and Communications Security,* 119–129.

Katzenbeisser, Stefan, and Fabien Petitcolas. 2016. *Information hiding.* Artech house.

Kemmerer, Richard A. 1983. ”Shared resource matrix methodology: An approach to identifying storage and timing channels”. *ACM Transactions on Computer Systems (TOCS)* 1 (3): 256–277.

Knerler, Kathryn, Ingrid Parker, and Carson Zimmerman. 2022. *11 Strategies of a World-Class Cybersecurity Operations Center.* MITRE.

Lampson, Butler W. 1973. ”A note on the confinement problem”. *Communications of the ACM* 16 (10): 613–615.

Latham, Donald C. 1986. ”Department of defense trusted computer system evaluation criteria”. *Department of Defense* 198.

Lehto, Martti. 2022. ”APT cyber-attack modelling: Building a general model”. In *International Conference on Cyber Warfare and Security,* 17:121–129. 1. Academic Conferences International Limited.

Lemay, Antoine, Joan Calvet, François Menet, and José M Fernandez. 2018. ”Survey of publicly available reports on advanced persistent threat actors”. *Computers & Security* 72:26–59.

Li, Haozhi, Tian Song, and Yating Yang. 2022. ”Generic and Sensitive Anomaly Detection of Network Covert Timing Channels”. *IEEE Transactions on Dependable and Secure Computing*.

Li, Yiming, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. 2022. ”Backdoor learning: A survey”. *IEEE Transactions on Neural Networks and Learning Systems.*

Liao, Hung-Jen, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung. 2013. ”Intrusion detection system: A comprehensive review”. *Journal of Network and Computer Applications* 36 (1): 16–24.

Liu, Liu, Olivier De Vel, Qing-Long Han, Jun Zhang, and Yang Xiang. 2018. ”Detecting and preventing cyber insider threats: A survey”. *IEEE Communications Surveys & Tutorials* 20 (2): 1397–1417.

Lubacz, Józef, Wojciech Mazurczyk, and Krzysztof Szczypiorski. 2014. ”Principles and overview of network steganography”. *IEEE Communications Magazine* 52 (5): 225–229.

March, Salvatore T, and Gerald F Smith. 1995. ”Design and natural science research on information technology”. *Decision support systems* 15 (4): 251–266.

Maurice, Clémentine, Manuel Weber, Michael Schwarz, Lukas Giner, Daniel Gruss, Carlo Alberto Boano, Stefan Mangard, and Kay Römer. 2017. ”Hello from the Other Side: SSH over Robust Cache Covert Channels in the Cloud.” In *NDSS,* 17:8–11.

Mazurczyk, Wojciech, Miłosz Smolarczyk, and Krzysztof Szczypiorski. 2011. ”Retransmission steganography and its detection”. *Soft Computing* 15:505–515.

Mazurczyk, Wojciech, and Steffen Wendzel. 2017. ”Information hiding: challenges for forensic experts”. *Communications of the ACM* 61 (1): 86–94.

Mazurczyk, Wojciech, Steffen Wendzel, and Krzysztof Cabaj. 2018. ”Towards deriving insights into data hiding methods using pattern-based approach”. In *Proceedings of the 13th International Conference on Availability, Reliability and Security,* 1–10.

Mazurczyk, Wojciech, Steffen Wendzel, Sebastian Zander, Amir Houmansadr, and Krzysztof Szczypiorski. 2016. *Information hiding in communication networks: fundamentals, mechanisms, applications, and countermeasures.* John Wiley & Sons.

Meusburger, Kaspar. 2023. ”On the Statistical Detectability of Covert Timing Channels”. PhD thesis, Technische Universität Wien.

Mileva, Aleksandra, Aleksandar Velinov, Laura Hartmann, Steffen Wendzel, and Wojciech Mazurczyk. 2021. ”Comprehensive analysis of MQTT 5.0 susceptibility to network covert channels”. *Computers & security* 104:102207.

Millen, Jonathan K. 1987. ”Covert channel capacity”. In *1987 IEEE Symposium on Security and Privacy,* 60–60. IEEE.

Moskowitz, Ira S, and Myong H Kang. 1994. ”Covert channels-here to stay?” In *Proceedings of COMPASS’94-1994 IEEE 9th Annual Conference on Computer Assurance,* 235–243. IEEE.

NIST, SP. 2020. *800-53 Rev. 5: Security and Privacy Controls for Information Systems and Organizations.*

Papernot, Nicolas, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016. ”The limitations of deep learning in adversarial settings”. In *2016 IEEE European symposium on security and privacy (EuroS&P),* 372–387. IEEE.

Pascoe, Cherilyn, Stephen Quinn, and Karen Scarfone. 2024. ”The NIST Cybersecurity Framework (CSF) 2.0”.

Peterson, Erik. 2023. ”Achieving Visibility and Control in OT Systems: Remote Maintenance, Securing Remote Access, and the Zero-Trust Approach”.

Petitcolas, Fabien AP, Ross J Anderson, and Markus G Kuhn. 1999. ”Information hiding-a survey”. *Proceedings of the IEEE* 87 (7): 1062–1078.

Pols, Paul, and Jan van den Berg. 2017. ”The unified kill chain”. *CSA Thesis, Hague,* 1–104.

Schmidbauer, Tobias. 2023. ”Novel Sophisticated Network-Level Covert Channels”. PhD thesis, University of Hagen, Germany.

Schmidbauer, Tobias, and Steffen Wendzel. 2022. ”Sok: A survey of indirect network-level covert channels”. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security,* 546–560.

Shah, Gaurav, Andres Molina, Matt Blaze, et al. 2006. ”Keyboards and Covert Channels.” In *USENIX Security Symposium,* 15:64.

Shannon, Claude Elwood. 1948. ”A mathematical theory of communication”. *The Bell system technical journal* 27 (3): 379–423.

Shao, Zhihui, Mohammad A Islam, and Shaolei Ren. 2020. ”Your noise, my signal: Exploiting switching noise for stealthy data exfiltration from desktop computers”. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 4 (1): 1–39.

Simmons, Gustavus J. 1984a. ”The prisoners’ problem and the subliminal channel”. In *Advances in Cryptology: Proceedings of Crypto 83,* 51–67. Springer.

———. 1984b. ”The subliminal channel and digital signatures”. In *Workshop on the Theory and Application of of Cryptographic Techniques,* 364–378. Springer.

———. 1993. ”Subliminal communication is easy using the DSA”. In *Workshop on the Theory and Application of of Cryptographic Techniques,* 218–232. Springer.

Simon, Herbert A. 2019. *The Sciences of the Artificial, reissue of the third edition with a new introduction by John Laird.* MIT press.

Strom, Blake E, Andy Applebaum, Doug P Miller, Kathryn C Nickels, Adam G Pennington, and Cody B Thomas. 2018. ”Mitre att&ck: Design and philosophy”. In *Technical report.* The MITRE Corporation.

”The TON_IoT Datasets”. 2020. Visited on September 14, 2023. https://research.unsw.edu.au/projects/toniot-datasets.

Thomas, Sam L, and Aurélien Francillon. 2018. ”Backdoors: Definition, deniability and detection”. In *Research in Attacks, Intrusions, and Defenses: 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10-12, 2018, Proceedings 21,* 92–113. Springer.

Tumoian, Eugene, and Maxim Anikeev. 2005. ”Network based detection of passive covert channels in TCP/IP”. In *The IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05) l,* 802–809. IEEE.

Velinov, Aleksandar, Aleksandra Mileva, Steffen Wendzel, and Wojciech Mazurczyk. 2019. ”Covert channels in the MQTT-based Internet of Things”. *IEEE Access* 7:161899–161915.

Vielberth, Manfred, Fabian Böhm, Ines Fichtinger, and Günther Pernul. 2020. ”Security operations center: A systematic study and open challenges”. *IEEE Access* 8:227756–227779.

Villalón-Huerta, Antonio, Hector Marco Gisbert, and Ismael Ripoll-Ripoll. 2022. ”SOC critical path: A defensive kill chain model”. *Ieee Access* 10:13570–13581.

Wendzel, Steffen, Luca Caviglione, Wojciech Mazurczyk, Aleksandra Mileva, Jana Dittmann, Christian Krätzer, Kevin Lamshöft, Claus Vielhauer, Laura Hartmann, Jörg Keller, et al. 2021. ”A revised taxonomy of steganography embedding patterns”. In *Proceedings of the 16th International Conference on Availability, Reliability and Security,* 1–12.

———. 2022. ”A generic taxonomy for steganography methods”.

Wendzel, Steffen, Sebastian Zander, Bernhard Fechner, and Christian Herdin. 2015. ”Pattern-based survey and categorization of network covert channel techniques”. *ACM Computing Surveys (CSUR)* 47 (3): 1–26.

”Worldwide information security services spending from 2017 to 2023”. 2022. Visited on February 17, 2023. https://www.ibm.com/reports/data-breach.

Xiang, Zhen, David J Miller, and George Kesidis. 2019. ”A benchmark study of backdoor data poisoning defenses for deep neural network classifiers and a novel defense”. In *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP),* 1–6. IEEE.

Yadav, Tarun, and Arvind Mallari Rao. 2015. ”Technical aspects of cyber kill chain”. In *Security in Computing and Communications: Third International Symposium, SSCC 2015, Kochi, India, August 10-13, 2015. Proceedings 3,* 438–452. Springer.

Younis, Awad A, Zachary Daher, Brendan Martin, and Connor Morgan. 2022. ”Mapping zero-click attack behavior into MITRE ATT&CK mobile: A systematic process”. In *2022 International Conference on Computational Science and Computational Intelligence (CSCI),* 890–896. IEEE.

Zander, Sebastian, et al. 2010. ”Performance of selected noisy covert channels and their countermeasures in IP networks”. *Centre for Advanced Internet Architectures Faculty of Information and Communication Technologies.*

Zander, Sebastian, Grenville Armitage, and Philip Branch. 2007a. ”A survey of covert channels and countermeasures in computer network protocols”. *IEEE Communications Surveys & Tutorials* 9 (3): 44–57.

———. 2007b. ”An empirical evaluation of IP Time To Live covert channels”. In *2007 15th IEEE International Conference on Networks,* 42–47. IEEE.

———. 2011. ”Stealthier inter-packet timing covert channels”. In *NETWORKING 2011: 10th International IFIP TC 6 Networking Conference, Valencia, Spain, May 9-13, 2011, Proceedings, Part I 10,* 458–470. Springer.

Zillien, Sebastian, and Steffen Wendzel. 2018. ”Detection of covert channels in TCP retransmissions”. In *Secure IT Systems: 23rd Nordic Conference, NordSec 2018, Oslo, Norway, November 28-30, 2018, Proceedings,* 203–218. Springer.

———. 2023. ”Weaknesses of popular and recent covert channel detection methods and a remedy”. *IEEE Transactions on Dependable and Secure Computing.*