Juha Jussila

HTTP COOKIE WEAKNESSES, ATTACK METHODS AND DEFENSE MECHANISMS: A SYSTEMATIC LITERATURE REVIEW

UNIVERSITY OF JYVÄSKYLÄ
FACULTY OF INFORMATION TECHNOLOGY
2018

ABSTRACT

Jussila, Juha
HTTP Cookie Weaknesses, Attack Methods and Defense Mechanisms: A Systematic Literature Review
Jyväskylä: University of Jyväskylä, 2018, 61 pp.
Computer science, master's thesis
Supervisor(s): Siponen, Mikko & Semenov, Alexander

HTTP cookie has been a commonly used technique in the world wide web. Several widescale data breaches have shown that cookies can be compromised with multiple attack types. It was inevitable to identify the weaknesses of cookies. Researchers in the ICT field have emphasized several vulnerabilities and weak points in cookies. Cookie protocol has been based on a draft that was signed over two decades ago. By means of systematic literature review the weaknesses of cookies, the attack methods that exploit the weaknesses, and defense methods to mitigate the attacks were disclosed in this research. Literature addressing cookie specification, attack methods, and defense methods, was examined and evaluated. Based on the literature the research indicated that cookies and the transmitting protocols contain weaknesses and vulnerabilities that can be exploited by attackers. The research addressed that cookies lack confidentiality and integrity. Cookie protocol should be updated to a new level of security. In the current form, cookies are exposed to poisoning, hijacking, manipulation, cross-site scripting, cross-site request forgery, TCP/IP hijacking, and session fixation. Several defense methods should be applied to mitigate the attacks.

Keywords: HTTP cookie, vulnerability, cross-site scripting, cross-site request forgery, TCP/IP hijacking, session fixation

TIIVISTELMÄ

Jussila, Juha
HTTP Cookie Weaknesses, Attack Methods and Defense Mechanisms: A Systematic Literature Review
Jyväskylä: Jyväskylän yliopisto, 2018, 61 s.
Tietojenkäsittelytiede, pro gradu -tutkielma
Ohjaajat: Siponen, Mikko & Semenov, Alexander

HTTP eväste on ollut yleisesti käytetty tekniikka maailmanlaajuisissa tietoverkoissa. Useat laajamittaiset tietomurrot ovat osoittaneet, että evästeitä voidaan murtaa useilla erilaisilla hyökkäystyypeillä. On väistämätöntä tunnistaa evästeiden heikkouksia. ICT-alan tutkijat ovat osoittaneet lukuisia evästeiden haavoittuvuuksia ja heikkouksia. Eväste-protokolla on perustunut yli kaksi vuosikymmentä sitten laadittuun luonnokseen. Tässä tutkimuksessa tutkittiin systemaattisen kirjallisuuskatsauksen metodein evästeiden heikkouksia, heikkouksia hyödyntäviä hyökkäysmetodeja ja puolustusmekanismeja hyökkäyksien ehkäisemiseksi. Tutkimuksessa analysoitiin evästeiden määrittelyä, hyökkäysmetodeja ja puolustusmekanismeja tutkivaa kirjallisuutta. Käytetty kirjallisuus arvioitiin tutkimusmenetelmän metodien mukaisesti. Kirjallisuuteen perustuen tutkimus osoitti evästeiden ja siirtoprotokollien heikkouksia ja haavoittuvuuksia, joita hyökkäyksissä voidaan käyttää hyväksi. Tuloksissa havaittiin puutteita evästeiden eheydessä. Evästeiden luottamuksellisuus todettiin heikoksi. Tulokset osoittivat, että eväste-protokolla tulisi päivittää uudelle turvallisuustasolle. Tutkimus osoitti, että nykyisessä muodossaan evästeet altistuvat hyökkäyksille, joissa evästeitä kaapataan ja manipuloidaan. Lisäksi tulokset osoittivat, että evästeet altistuvat XSS ja CSRF -tyyppisille hyökkäksille. Useita puolustusmekanismeja tulisi asettaa evästeisiin hyökkäysten ehkäisemiksi.

Asiasanat: HTTP eväste, haavoittuvuus, cross-site scripting, cross-site request forgery, TCP/IP hijacking, session fixation

FIGURES

TABLES

TABLE OF CONTENTS

1 PREFACE

Most of the services in the Internet include advertisements. Advertisements are used to fund the services. For an example, according to the PwC Advisory Services advertising revenues were 59,6 billion dollars in total in 2015 in the United States. The total revenues of advertising increases rapidly year by year. The term "online advertising" consists of advertisements that are paid. The advertisements enable web sites and web applications to be used for free. Advertising that is custom-made based on the users' interests is called Interest-based advertising (IBA). This allows companies to promote products and services straightforward to users according to their needs. Advertisers need to be able to profile and track the users to customize advertisements. (Hassan & Hijazi, 2017, p. 9.)

Cyber criminality and large-scale organizations, which are capable to operate cyber-attacks, are also interested in user information. For an example, 15 large-scale attacks were discovered in the EU in 2016, in which tens of millions of users lost their personal information, like email, passwords and credit card information, to the attackers (Fraunholz, Krohmer, Anton & Schotten, 2017). Could HTTP cookies be used for cyber attacks or collecting information about the adversary? Do cookies expose critical information about our defense? Only one incautious user of the defense is needed to expose the whole defense. According to Green (2015) a victim web site could send back a cookie with unique identifying code to the cyber attacker that would enable the recognition of the attacker on subsequent activity. Green says that the method works only for web protocols and the attacker should accept cookies.

Cookies are exposed to multiple attack techniques. To secure cookies, several security guards need to be adapted. For an example, Yahoo revealed that hackers have gained access to 32 million user accounts in 2017 by forging cookies to log in without password (CNet, 2017). Cookies are effective yet fragile. They need to be safeguarded in suitable and efficient means. Several studies indicate that cookies do not provide high enough security level. Cookies are used by the variety of agents that provide services across the world wide web as a technique for understanding the users and their needs.

There exist two types of transaction protocols on the Internet: stateful and stateless protocols. Stateful protocols are used for maintaining connection status, running processes and running process statuses. After closing the connection, the state information needs to be wiped. Stateless protocols do not store completed transaction information and processes. No data needs to be wiped. (Bangia, 2005.) Cookies are sent via HTTP that is a stateless protocol. Cookies are required to be able to remember stateful information. With cookies, it is possible to for an example tell if two requests come from the same browser. (Mozilla, 2018.)

Cookies are normally data that a web site stores to a user's computer whenever the user visits the web site. Cookies are a small text file that identifies user. The user is identified as a unique visitor to the web site. Cookies include records about performed actions or preferences that are set during the user's visit on the web site. Whenever the user returns to the same web site, the web site reads the stored cookie to display appropriate contents. (F-Secure.)

There are three purposes for using cookies: session management, personalization, and tracking. For session management, cookies are used for an example to manage logins and shopping carts. Cookies are also used for storing user preferences and other settings to personalize web site experiences. Tracking is also important function of cookies to record and analyze user behavior. (Mozilla, 2018.)

For tracking purposes, there exist a special type of cookies known as tracking cookies. Tracking cookies can be shared between multiple web sites. Tracking cookies can be used for tracking users's browsing behavior and to present customized content to the users. (F-Secure.)

To track online users, cookies are used to record user entries. The recorded information is sent back to server. Cookies that used for tracking purposes send a log of user's online activities. The activities are bind to the user's IP address. The log is sent to be analyzed in a remote database. Whenever a cookie is recognized on a user's browser while loading a web page or advertisement, the visit is recorded and sent to logs. (Tom's Guide, 2013.)

Third-party advertising services keep track of the advertisements a user has seen on a web site by using cookies. This feature enables displaying relevant material to the user. When a web site including third-party banners is visited by a user, information about the displayed banners is saved on the system of the user in a form of cookie. Whenever the user visits some other web site that has the same advertising service, the cookie is red by the other web site. The other web site displays new banners. The service logs the user's visit to both web sites. Tracking cookies can be blocked with many antispyware and antivirus products. (F-Secure).

The significantly increasing number of products purchased online forces organizations to change their marketing strategy. Traditional marketing cannot reach the modern customers. The customers themselves are the best marketers. (Zhang, Wang & Xia, 2010.) The customers' information and preferences need to guide the marketing targeted on the customers. Cookies are a potential

means to guide the way the information and preferences can be tracked. However, cookies in the form of tracking concern the agents who discuss the privacy aspects. However, privacy has been absent since a user performs her first search in a search engine on the Internet.

There are multiple technical solutions for actors of the Internet to collect information from users. The solutions are called web analytics which are tools for collecting user information (Clifton, 2012). Web analytics solutions allow organizations to measure how many users visit the monitored site, how they end up to the site and what actions do they perform on the site (Clifton, 2012). An example of web analytics solution is Google Analytics.

Online stores collect user information to examine users' recent browsing history. Online store providers are interested on users' browsing history, to understand what the users are interested in, to be able to target marketing and perform targeted recommendations. This is called personalized recommendation system. Various types of service providers, such as online music service providers, use personalized recommendation system to target products for the users. (Zhu, 2016.)

Cookies have been researched widely. Yet, there exists not much studies combining cookie weaknesses, attack methods, and defence methods in satisfying spectrum. The exponential expansion of the usage of cookies challenges security and privacy of the various agents of the world wide web. Meanwhile, cookies are used by the most of the services of the Internet to gain information about the users of the services. Web servers encase extensive databases that include considerable amount of information about the variety of the users.

This study combines cookie specification, weaknesses of the specification, attack methods that exploit the weaknesses, and defense mechanisms to mitigate the impacts of the attack methods. The research indicates cookie specification based on the evidence found in RFC 6265 document. Multiple studies are examined to address the weaknesses of the specification. Various studies were examined that discuss the attack methods. Also, multiple studies indicate defense mechanims against the attack methods.

There are studies that combine cookie specification, weaknesses of the specification, attack methods that exploit the weaknesses, and defense mechanisms to mitigate the impacts of the attack methods. None of the found studies combined the cookie specification, attack methods and defense mechanims on adequate level to be able to formulate the magnitude of the phenomenon. Consequently, this study amalgamates the found evidence to generate an integrated comprehension of the variables that impact the phenomenon from a multifaceted perspective. The current evidence do not satisfy the necessity of research on the phenomenon in adequate spectrum. The phenomenon is examined in constricted magnitude in this study as the previous research concentrates in specific variables impacting the phenomenon.

The purpose of this research is to examine the weaknesses of cookies, the attack methods that cookies face and the defense mechanisms to secure cookies

against the attacks. The research examines the different aspects that affect cookies as a commonly used mechanism.

1.1 Objectives

The objective of this research is to study the essense of cookies to understand why cookies are vulnerable and what weaknesses cookies include. When we understand how cookies function and how the features of cookies affect the security of cookies, we gain knowledge about the weaknesses of cookies. It is inevitable to generate a clear understanding of the cookie mechanism that has an effect on most of the individuals in the world. We need to understand what makes cookies vulnerable, and which are the components of cookies that are vulnerable to create new mechanisms or enforce the existing mechanisms to protect the cookies.

The objective of this research is to indicate results that disclose weaknesses, attack types, and defence mechanisms of cookies. Consequently, disclosing an elaborate explanation of the security of cookies. This research aims to answer the research questions by the means of systematic literature review by examining cookie specification, the attack methods that exploit weaknesses found in the specification, and the safeguards to protect cookies. The research will indicate the weaknesses, attack methods, and defense mechanisms of cookies.

1.2 Motivation

Most of the population of the world use Internet services. The population of the world is approximately 7,6 billion. About 4,1 billion individuals used the services of the Internet in December 2017. The penetration rate was 54,4 %. (Internet World Stats, 2018.) Mirroring the total amount of Internet users, HTTP cookies impact the most individuals in the world. Consequently, HTTP cookies should be one of the major concerns in the modern digital world.

According to the recital 30 of the GDPR (IT Governance, 2017) that will apply from May 25, 2018, whenever IP addresses or cookie and other identification techniques are able to identify individuals through user's device, the idenfified information is considered as personal data. GDPR states that these identifiers may leave a trace. The trace combined with the idenfiers may be used for creating profiles of the individuals to identify the individuals. The majority of cookies are used for identifying users. Therefore, the majority of cookies will be subjects to the becoming GDPR.

GDPR addresses the rights of individuals by controlling the collection and processing of personal information. GDPR sets responsibilities on organizations to protect data more efficiently. (IT Governance, 2018.) GDPR guides the official agents of the Internet in the right direction to protect privacy rights in the Eu-

ropean Union. However, multiple agents operating in the Internet might not be willing to follow the guidelines set in the GDPR. User information is valuable. Therefore, it is an inevitable fact that cookies in the form of user identifiers should be protected more efficiently. It is necessary to research the potential risks and vulnerabilities of cookies.

Analysing the essence of cookies and cookie attributes allows us to understand the weaknesses and vulnerabilities that cookies encase. It is essential to research the technical aspects that have an impact on the cookie's operation. Consequently, the achieved knowledge of the cookie operation enables to increase the online security. None of the former developed security solutions has produced firm answers for the problem.

1.3 Research Question

Thia systematic literature review will answer the following questions: ”What are the weaknesses of HTTP cookie?” Secondary research questions are: ”What types of attacks exploit the weaknesses of HTTP cookie?” and ”What defence methods can be applied to mitigate the attacks?”

No previous studies combine specification of cookies with attack methods exploiting the weaknesses of cookies and the defense techniques to protect the cookies in this scale. There has been studies that examined the vulnerabilities and weaknesses of cookies and a specific attack method to exploit the vulnerabilities and weaknesses.

The main research question is selected to result in advantageous knowledge what specifications of cookies should be reconsidered. The first secondary research question supports the main research question by adding a viewpoint of how the weaknesses of cookies can affect the security of cookies. The third research question seeks solutions to improve the security of cookies.

The research addresses cookies from the point of view of the weaknesses of cookies, attack methods exploiting the weaknesses, and defense methods to mitigate the attacks. The strengths of cookies are omited and are not included in this research. This research does not discuss why cookies should be used nor why cookies should not be used. Cookies are the most commonly used method to identify online users and sessions. Consequently, there is no necessity to point out whether cookies should or should not be used.

1.4 Research Formula

This research followed the guidelines of systematic literature review research method. Previous studies were examined to develop knowledge of the essence of cookies. The essence of cookies help to understand the weaknesses that cookie specification and functionality includes. Evidence relating the research ques-

tions was located, evaluated and synthesized. The research formula provided informative evidence-based answers for the research questions.

The research problem was defined based on the personal interest of the resaercher and by examining the previous studies. The available evidence was gathered from reliable resources of documents addressing cookie specification, attack methods used to exploit the weaknesses of cookies, and defensive techniques to protect cookies against the attacks. Based on the created include and exclude criteria, the documents were selected to be used in the research.

The scope of the research and the research objectives were placed to understand the phenomenon from a specific angle and to answer the research question correctly. Relevant studies according to the research question were selected and observed in detail to verify their quality.

The research addressed multiple weaknesses in the cookie specification that can be exploited by attack methods. The findings showed weaknesses of cookies. The premier findings were that cookies lack isolation by port, by scheme, and by path on the server side. The findings indicated also that cookies do not quarantee integrity for sibling domains and their subdomains on the server side. The cookie attributes do not provide integrity. Secure, HttpOnly and Path attributes protect only the confidentiality of cookies and not the intergrity of cookies.

The research addressed that attack methods are able to exploit the weaknesses of cookies. The attack methods in this research include poisoning, hijacking/stealing, manipulation of cookies, cross-site scripting, cross-site request forgery, and TCP/IP hijacking. The findings addressed that several defense methods should be applied to cookie.

The findings have an important magnitude for understanding the security of cookies. The findings should be considered when considering the usage of cookies. The found weaknesses should be further examined to improve cookie security.

The structure of the thesis is as follows: Section 2 discusses the related literature. The discussed themes are privacy, specification of cookie, types of cookies, cookie vulnerabilities and attack methods, and the prevention of the attack methods. Section 3 discusses the methodology of the thesis. Section 4 presents the findings of the research. Section 5 deliberates the findings and their significancy. The reliability and usability of the findings is discussed. Section 6 concludes the research.

2 LITERATURE REVIEW

This chapter reviews the literature that discusses the topic. There are particular concepts concerning the topic. First, digital privacy needs to be discussed in terms of online user tracking that cookies are used for. European Union has legislated a directive concerning the storing of user information. Next, we discuss the essence of HTTP cookies. HTTP cookies are an effective solution for user tracking. There exist different types of cookies.

The research focuses mainly on literature that defines the specifications of cookies, literature that examines attack methods that have an effect on cookies, and literature that examines the defense methods to mitigate the attacks. The review structure is as follows: privacy aspects of cookies, specification of cookies, types of cookies, vulnerabilities of cookies, cookie attack methods, and defense methods to mitigate the attacks.

2.1 Digital Privacy

Digital privacy encompasses protecting personal information in the Internet. Whenever personal or business communications are conducted through public networks, information originates. Identifying the originated information is the essence of digital privacy. Since Edward Snowden revealed the documents addressing the programs of mass surveillance, a debate between making surveillance activities legal and protecting personal information has existed. Whenever a user searches online using Google, the user's keywords, search date and time, and IP address can be tracked. Browsing activities and Internet habits are being logged to create formulated profiles. (Hassan & Hijazi, 2017, p. 6.)

Information collected from online activities can be divided in two types. First type is personally identifiable information (PII) or sensitive personal information (SPI). Second type is anonymous information. Personally identifiable information includes information such as name, biometric records, social security number, gender, and passport number. Anonymous information includes

information such as browser type, browser version, current location, school, country, and connected device type. (Hassan & Hijazi, 2017, p. 7.)

The European Parliament and the Council of the European Union (2009) direct the storing of information in Member States. According to the directive, Member States need to ensure that a subscriber or user has been provided with clear and comprehensive information about the purposes of the processing of their information. Storing of information, or gaining access to information already stored, in the subscriber's or user's terminal equipment is allowed only if the subscriber or user has given her concession. The directive does not restrict the type of information stored nor how the information is stored.

Cookies allow web servers to track users. Consequently, cookies are subjects for criticism. Companies use web analytics to follow the trails users navigate in and between sites. Cookies are persistent across sessions of user agents and cookies can be shared between hosts within the same domain. The sharing between hosts within the same domain may take place if a user agent treats a missing Domain attribute as present and containing the current host name. (Barth, 2011.)

2.2 HTTP Protocol

HTTP (Hypertext Transfer Protocol) a protocol on application-level. HTTP is designed for distributed, collaborative, and hypermedia systems. HTTP is used for data communication on the Internet. HTTP is based on TCP/IP. (Tutorials Point.) TCP/IP consists of TCP (Transmission Control Protocol) and IP (Internet Protocol). These two protocols are distinct network protocols. Protocol is a set of procedures and rules. Protocols allow computers to understand each other and exchanging data. Web browsers and web servers communicate with TCP/IP. (Lifewire, 2018.)

Messages or files that are transmitted over Internet, are divided into packets by TCP. The packets are reassembled when the destination is reached. IP takes care that the packets are sent to desired destination. TCP/IP functions on four layers: Datalink layer, Networking layer, Transport layer, and Application layer. Datalink layer includes on a link operating methods and protocols. The link is a network component to interconnect hosts. Networking layer connects network boundaries and independent networks to transport data packets. Communication between hosts is handled by Transport layer. Flow control, reliability, and multiplexing are Transport layer's responsibilities. Applications' data exchange is standardized on Application layer. (Lifewire, 2018.)

Construction and transmission of clients' data is specified in HTTP specification. Also, the way servers respond to clients' requests is specified in HTTP specification. HTTP consists of three features. First, HTTP request is initiated by HTTP client (for an example browser) to a server. The client is disconnected from the server after making the request. The client waits for server's response. The request is processed by the server and the connection between the server

and the client is re-established and a response is sent. The disconnection after initiating a request means that HTTP is connectionless. Next, all types of data can be sent by HTTP. Although, it is required that the client and the server can handle the content of the data. This means that HTTP is media independent. HTTP being connectionless results in HTTP being stateless. Awareness between server and client is present during current requests. (Tutorials Point.)

HTTP is based on request/response. HTTP is based on client/server architecture. Requests are sent to servers by clients over TCP/IP connection. The requests are sent in a form of a request method, URI, and protocol version. This information is followed by message that includes request modifiers, client information, and possible body content. The request is responded by the server. The response includes protocol version of the message, a success or error code, server information message, entity meta information, and possibly content of entity-body. (Tutorials Point.)

HTTP identifies resources and establish connections by using URI (Uniform Resource Identifier). HTTP messages are passed after establishing the connection. The messages are client's requests to server and server's responses to client. URIs are strings that include name, location, or other resource identifiers. (Tutorials Point.)

2.3 HTTP Cookie

When a user visits a web site for the first time, the user is unknown to the web server in which the web site is running. The web server will expect the user to return to the web site again and uploads a unique identifier called cookie to the user's browser. The cookie comprehends a list of *name=value* information. The cookie is attached to the user by using the Set-Cookie or Set-Cookie2 HTTP response (extension) headers. Cookies can include a unique identification number, that is generated by the web server, for tracking purposes. (Gourley & Totty, 2002, p. 264.) The unique identification number identifies the user. When the user returns to the web site, the returning cookie is verified to identify the user.

HTTP protocol is stateless. Consequently, all HTTP requests that are sent by user agents are independent from all the other requests. Web developers bypass the fact that HTTP is a stateless protocol by pinning session identifiers to web site visitors. To identify a user agent, the session identifier is sent with every HTTP request. In some cases, session identifier is saved into a cookie. Every request includes the complete cookie that is sent to the web site the cookie is generated from. Cookies are sometimes used for user tracking purposes by implementing the cookies in displayed advertisements on different web sites to be able to analyze user behavior. Cookies can be used for handling authentication by adding a session identifier. Cookies are overlooked because cookies are not expected to get manipulated. (Ballmann, 2012, p. 90.) Cookies should not be used for authentication to prevent unauthorized access using cookies. However, if cookies are used for authentication, cookies should be properly safeguarded.

2.3.1 Cookie Header

Cookie headers are used to send cookies. Cookie headers include key/value pairs. A web server requests a user agent to save a cookie by using a Set-Cookie header. Cookies can be valid for one session or a specific time limit. If a cookie data includes "secure" property, the cookie should be sent over a secure transmission protocol, such as HTTPS. (Ballmann, 2012, p. 91.) If cookies that contain the secure property are sent via insecure connections, the secure property loses its purpose.

There are different types of cookies. Some cookies are non-persistent and some cookies are persistent. Mozilla (2017) defines Cookie header syntax as shown in table 1.

TABLE 1 Cookie Header

Cookie: <cookie-list>
Cookie: name=value
Cookie: name=value; name2=value2

A HTTP server can pass name/value pairs and affiliated metadata (cookies) to user agent by using the Set-Cookie header field. The user agent uses the cookies and additional information, when making consecutive requests to the HTTP server, to decide whether to return the name/value pairs in the Cookie header. (Barth, 2011.) Web servers should be configured to discover if the name/value pair is compromised.

Cookies seem simple yet they are rather complex. A HTTP server designates a scope for every cookie when the cookie is sent to the user agent. The user agent should return the cookie and the URI schemes, which the cookie is applicable, to the HTTP server within the maximum time limit, that the scope indicates. (Barth, 2011.)

Cookies also include numerous security and privacy distresses. The Secure attribute of Cookie header does not provide integrity in the presence of an active network attacker, yet a HTTP server can designate that the provided cookie is purposed for secure connections. (Barth, 2011.) Cookie security needs specific observation to secure the user information from malicious operators.

2.3.2 Set-Cookie Header

The web server where the cookie is downloaded from includes a Set-Cookie header to HTTP response to store state. In the following requests, the user agent returns the web server a Cookie request header. If the user agent has received any cookies in previous Set-Cookie headers, the cookies are contained in the Cookie header. The web server can ignore the Cookie header or use the Cookie header's contents. Web servers may send the Set-Cookie response header with

any responses. User agents must process any Set-Cookie headers excluding Set-Cookie headers contained in responses with 100-level status codes. (Barth, 2011.)

Multiple Set-Cookie header fields can be included in a single response by web servers. If a Cookie or a Set-Cookie header field exists, the header fields do not prevent a HTTP cache to store and reuse a response. (Barth, 2011.) Mozilla (2017) defines Set-Cookie header syntax as shown in table 2.

TABLE 2 Set-Cookie header

Set-Cookie: <cookie-name>=<cookie-value>
Set-Cookie: <cookie-name>=<cookie-value>; Expires=<date>
Set-Cookie: <cookie-name>=<cookie-value>; Max-Age=<non-zero-digit>
Set-Cookie: <cookie-name>=<cookie-value>; Domain=<domain-value>
Set-Cookie: <cookie-name>=<cookie-value>; Path=<path-value>
Set-Cookie: <cookie-name>=<cookie-value>; Secure=<true> or <false>
Set-Cookie: <cookie-name>=<cookie-value>; HttpOnly=<true> or <false>
Set-Cookie: <cookie-name>=<cookie-value>; SameSite=Strict
Set-Cookie: <cookie-name>=<cookie-value>; SameSite=Lax

User agent saves a cookie and the cookie's attributes whenever the user agent receives a Set-Cookie header. The user agent includes all applicable and non-expired cookies in the Cookie header whenever the user agent performs a HTTP request. If a new cookie that the user agent receives has the same cookie-name, domain-name, and path-value as a previously stored cookie, as a result, the previous cookie is ejected and replaced with the fresh cookie. (Barth, 2011.)

2.3.3 Attributes

The content of a cookie consists of given attributes. The cookie attributes include the following: Expires, Max-Age, Domain, Path, Secure, and HttpOnly. Cookie's maximum lifetime is defined in the Expires attribute. The lifetime is expressed as the date and time when the cookie will be expired. Similarly to the Expires attribute, the Max-Age attribute points out the cookie's maximum lifetime. Unlike the Expires attribute, Max-Age attribute expresses the maximum lifetime of a cookie as the number of seconds to the cookie expiration. A cookie can include both of the above attributes. In this case, the Max-Age attribute has priority over the Expires attribute and dominates the expiration time of the cookie. In some cases, a cookie might not have neither of these attributes. In these cases, the user agent stores the cookie until the ongoing session is closed. (Barth, 2011.) It is essential to use expiration time for cookies to prevent them being manipulated and misused.

Cookie is sent to hosts that are specified in the Domain attribute. The user agent returns cookies to origin web server only if the web server elides the Domain attribute. Otherwise, the user agent includes the cookies in the Cookie header with HTTP requests to the hosts working under the domain. A missing

Domain attribute is handled as present and like the Domain attribute would include the contemporary host name by some of the existing user agents. As a result, if a host returns a Set-Cookie header excluding the Domain attribute, the user agents explained above send the cookie falsely to other hosts in the domain. Cookies are rejected by the user agent if the Domain attribute does not specify the cookie's scope that includes the origin web server. Several user agents are configured to exclude the Domain attributes that are equal to public suffixes. The configuration is produced for security reasons. (Barth, 2011.)

The Path attribute controls the set of paths that limits the scope of cookies. User agents use the directory in the request-URI's path component by default whenever a web server elides the Path attribute. The cookies are incorporated by the user agents only when the request-URI's path segment is equal to the cookies' Path attribute. (Barth, 2011.) Cookies might need a mechanism to verify the routing the cookies are sent and returned.

The scope of a cookie is limited to secure channels in the Secure attribute. The cookies are included in HTTP requests by the user agent if the request will be transmitted over a channel that is secure, if the cookies contain the Secure attribute. Typically, a secure channel is HTTP over TLS (Transport Layer Security). (Barth, 2011.) A secure channel should be used for all the cookies that contain the secure property and cookies that include sensitive information, such as login credentials.

The scope of a cookie is limited to HTTP requests in the HttpOnly attribute. If cookies are provided an access through application programming interfaces (APIs) that are not using HTTP, the cookies are elided by the user agent according to the instructions set in the HttpOnly attribute. If a web browser exposes cookies to scripts, the web browser is considered as a API that does not include HTTP. Cookies can include both the Secure attribute and the HttpOnly attribute. (Barth, 2011.)

A web server can send a user agent a short string within a HTTP response. The user agent returns the string to the web server in future HTTP requests. The HTTP requests need to be within the cookie's scope. Web server sends the string with the Set-Cookie header. The string can define a session identifier with a defined value. The user agent returns the session identifier in following requests. (Barth, 2011.) What if this short string gets modified by an attacker? What functions could be done by the attacker through this property? Session identifier should not be included in cookies.

Cookies have a default scope. The default scope can be modified by web servers by using the Path and Domain attributes. Web servers are able to give instructions to user agents. For an example, web servers can tell the user agents to return cookies to every path and every subdomain of a specific domain. Web servers can save several cookies to the user agent. By returning multiple Set-Cookie header fields, web servers are able to save multiple parameters. For an example, web servers can save a session identifier and the preferred language of the user agent. Web servers should use the Secure and HttpOnly attributes to provide security protections for the more sensitive parameters. (Barth, 2011.)

Web servers can make the user agents to persist cookies over several sessions. This is done by specifying an expiration time in the Expires attribute. This feature is essential in situations like, for an example, if the user agent restarts. If the user agent's cookie store exceeds its contingent, the user agent may delete the cookie prior to the expiration time. The user agent may also delete the web server's cookie manually. In the end, web server will return a Set-Cookie header including an expiration date in the past to remove a cookie. The removal will succeed only if the Set-Cookie header's Path and Domain attributes match the values of the original cookie values. (Barth, 2011.) The user agent can delete most cookies but some types of cookies are resistant and are not deleted along the general cookies.

SameSite attribute is designed to limit the scope of cookies by attaching the SameSite attribute to requests that are "same-site". There are two possible values to be set on SameSite attribute: "Strict" and "Lax". Whenever the value is set to strict, cookies are sent only with "same-site" requests. If the value of SameSite attribute is set to lax, the cookie is sent with same-site requests and cross-site requests. A request being "same-site" means that the registrable domain of the origin's target URI matches the site for cookies value of the initiator of the request. (West & Goodwin, 2016.)

2.3.4 Syntax

Set-Cookie response header comprehends header name "Set-Cookie". The header name is followed by ":" and a cookie. Cookies commence with name/value-pair. The name/value-pair is followed by none or more attribute-value pairs (Barth, 2011.) RFC 6265 (Barth, 2011.) defines the cookie grammar as shown in table 3. If a Set-Cookie header fails to conform the grammar, web server should not send the header. However, the cookie syntax was renewed by adding the SameSite attribute as shown above in table 2.

TABLE 3 Cookie syntax

```
set-cookie-header = "Set-Cookie:" SP set-cookie-string
set-cookie-string = cookie-pair *( ";" SP cookie-av )
cookie-pair = cookie-name "=" cookie-value
cookie-name = token
cookie-value = *cookie-octet / ( DQUOTE *cookie-octet DQUOTE )
cookie-octet = %x21 / %x23-2B / %x2D-3A / %x3C-5B / %x5D-7E
token = <token>
cookie-av = expires-av / max-age-av / domain-av /
            path-av / secure-av / httponly-av /
            extension-av
expires-av = "Expires=" sane-cookie-date
sane-cookie-date = <rfc1123-date>
max-age-av = "Max-Age=" non-zero-digit *DIGIT
```

```
non-zero-digit = %x31-39
domain-av = “Domain=” domain-value
domain-value = <subdomain>
path-av = “Path=” path-value
path-value = <any CHAR except CTLs or “;”>
secure-av = “Secure”
httponly-av = “HttpOnly”
extension-av = <any CHAR except CTLs or “;”>
```

2.4 HTTP Cookie Types

According to Dubrawsky (2010) cookies can be classified in three main types. The types of cookies are session cookies, persistent cookies, and tracking cookies.

2.4.1 Session Cookie

Web sites use session cookies to save information. Session cookies are also known as temporary cookies or non-persistent cookies. Whenever a user discontinues a web browser session, the cookie is removed. Session cookies can include information, that is related to authentication, about a user’s session. The information can be, for an example, display preferences or session identifiers or user identifiers. (Dubrawsky, 2010, p. 37.) Session cookies keep track of settings and preferences whenever a user is navigating a web site (Gourley et al, 2002, p. 264).

2.4.2 Persistent Cookie

Web sites use persistent cookies to save user connection information. Typically, persistent cookies are used for saving insensitive user preferences about web sites. As a result, there exists less concern with the persistent cookie being persistently saved on hard drive of a user. Compared to session cookies, persistent cookies are not removed whenever a user discontinues a web browser session. Persistent cookies carry a timeout value that is set by the web site. The cookies are set in the user’s web browser and deleted when the timeout value expires. (Dubrawsky, 2010, p. 37.) The timeout value should be accurate to prevent persistent cookies existing excessively extended lifetime.

2.4.3 Tracking Cookie

The purpose of tracking cookies is to record web activity of users. Whenever a user connects a web site that uses tracking cookies, tracking cookie is down-

loaded. According to Dubrawsky (2010), web sites use often tracking cookies of the same form. As a result, if multiple web sites use the same tracking cookie, the web sites can read the cookie and write to the contents of the cookie.

Tracking cookies are also known as third-party cookies. User agents might request resources from other servers, such as advertising networks, whenever user agents render a HTML document. Third-party cookies can be used for tracking users. Even if a user does not visit a third-party server directly, the third-party server might use cookies for tracking the user. Third-parties are able to track users between two web sites. This occurs when the users visit a web site containing third-party's content and then another web site containing the same third-party's content. (Barth, 2011.) Targeted content is enabled with tracking cookies. This type of cookies is widely used in the modern world wide web.

User agents can limit the way third-party cookies behave. The Cookie header can be refused by user agents to be sent in third-party requests. Some user agents abstain processing the Set-Cookie header when responding the third-party requests. Third-party cookie policies vary between different user agents. Nevertheless, blocking policies of third-party cookies frequently prove to be powerless when trying to achieve privacy goals. This is a result from third-party servers attempting to bypass user tracking restrictions of the third-party cookie blocking policies. Two collaborating servers can inject identifying information into dynamic URLs to track users without using cookies. (Barth, 2011.) Modern tracking techniques are tricky to block.

2.4.4 Flash Cookie

Flash cookies are collections of cookie-like data that can be placed on users' hard drive by a website running Adobe Flash. Flash cookies comprehend information about user visiting the site and potentially tracking and settings information. Flash is able to install cookies on a computer without user's permission by default. (Hassan & Hijazi, 2017, p. 18.) Flash cookies, that are also known as evercookies or supercookies, are efficient techniques to bypass blocking and restrictions.

Flash cookies are also known as local shared objects. Generally, Flash cookies are used for storing user preferences and settings. (Steward, 2014.) Flash cookies can be also used for saving game progress and user tracking. Flash cookies regenerate "cross-platform" by being able to be accessed or created in one browser and can be accessed by entirely different browsers. (Sarris, 2014, p. 302.) Adobe Flash configuration settings limit or restrict the use of Flash cookies through specific options. However, default settings are reset after each update of Flash. If a browser operates in privacy or incognito mode, Flash cookies are not stored by the most recent Flash versions. (Steward, 2014.)

Flash cookies are typically stored in multiple hard drive locations. Single Flash cookie might occupy 100,000 bytes of storage. Regular cookies that are deleted or blocked by a user, can be reinstated by using Flash cookies. This reinstating process is known as respawning. In respawning, the unique identifier

of the deleted cookie is assigned back to a new cookie. The data stored in a Flash cookie is used as a backup. (Ciampa, 2012, p. 93.)

In contrast to common cookies, flash cookies add complexity in storing data. Flash cookies are typically used for improving authentication experience in, for and example, online banking services. Flash cookies can improve security by adding a level of security. Flash cookies enable understanding clients better and creating risk score evaluation profiles. Flash cookies are used for risk-based authentication to enable those features. Flash cookies are controlled via Flash security settings. (Barrett, Weiss & Hausman, 2015, p. 256.)

2.5 HTTP Cookie Vulnerabilities

Cookies are an efficient technique for collecting user information and identifying the user agent but how vulnerable are they exactly? According to Dubrawsky (2010) web sites utilize cookies as files to store data for processing. Potential security risks exist whenever data inputs are performed. Cookies present wide security concern and face many attack types.

In the security point of view, cookies have numerous security exposures. Cookies can become vulnerable to attacks because developers might rely on ambient authority for authentication. Cookies can encourage developers to use the ambient authority. As a result, cookies can become vulnerable to, for an example, cross-site request forgery (CSRF) attacks. Developers might also store session identifiers in cookies. This procedure might generate session fixation vulnerabilities. (Barth, 2011.) Using cookies for authentication in their common form expose cookies for attacks.

Transport layer encryption (for an example the one employed in HTTPS) is an insufficient safeguard againts attacks because the cookie protocol is weak with various vulnerabilities, such as weak confidentiality and weak integrity. The attackers can exploit the vulnerabilities and obtain or alter cookies of a victim. Confidentiality and integrity from attackers are not provided by default in cookies. Even if cookies are used in combination with HTTPS, confidentiality and integrity of cookies are not secured. (Barth, 2011.)

Some user agents allow remote parties to distribute HTTP requests from the user agent. Therefore, if a web server applies cookies to user authentication, the web server might suffer security vulnerabilities. The user agent might allow a remote party to gain authority at an uncautious web server by attaching cookies to the HTTP requests. The remote party might not even know the contents of the cookies. This security concern can be called for an example a cross-site request forgery or confused deputy. The issue emerges if cookies are used as a form of ambient authority. Web server operators are encouraged by cookies to segregate designation from authorization. An attacker might designate a resource that can be supplied an authorization by the user agent. As a result, web server or web server's clients might undertake actions that the attacker has de-

signated. The actions are undertaken in a belief that the actions are authorized by the user. (Barth, 2011.)

If cookies are sent over an insecure channel, the information included in the Cookie header and Set-Cookie header will be transmitted in a clear text. As a result, the sensitive information of the headers can be predisposed to eavesdroppers. The headers are disposed to malicious intermediates while being transmitted. This threat might result in unpredictable consequences. The Cookie header might also be modified by a malicious client before transmitting it. This might also result in upredictable consequences. (Barth, 2011.) All cookies should contain the secure property and be sent via secure channels.

In case a web server stores session identifier in cookies and is accessed over HTTPS, but does not set the Secure attribute on cookies, the ongoing requests are exposed to an attacker intercepting user agent's any outbound HTTP requests. The attacker can be able to redirect the requests to the web server over HTTP. The user agent includes the cookies in the request, even if the web server is not even listening the HTTP connections. The attacker can examine the contents of the user's email by intercepting the cookies and replaying the cookies against the web server. If the web server includes the Secure attribute on cookies, the cookies will not be included in the clear-text requests by the user agent. (Barth, 2011.)

Session identifier usage is not riskless. Web servers should avoid vulnerabilities caused by session fixation. An attack focused on session fixation (session fixation attack) consists of three steps. In the first step, session indentifiers are transplanted from the attacker's user agent to the victim's user agent. In the next step, the session identifier is used by the victim for interaction with the web server. As a result, the session identifier might be filled with the victim's user credentials or the victim's confidential information by the attacker. In the third step, the session identifier is used by the attacker for interaction with the web server directly. As a result, the attacker might achieve the authority or confidential information of the victim. (Barth, 2011.)

In case a cookie can be read by a service that runs on a port, a service running on another port of the same web server is also able to read the cookie. This happens because cookies are not isolated by port. Web servers should not be running mutually distrusting services on the same host's various ports and storing security-sensitive information in cookies. Cookies are not providing isolation by port, by scheme, nor by path. Cookies are the most common solution used with HTTP and HTTPS schemes. However, cookies for given hosts might be in the reach of other schemes like FTP. The lacking isolation by scheme is present in cookie processing requirements. Cookies stored for one path are not sent to another path by the network-level protocol. However, cookies are exposed by some user agents through APIs that do not use HTTP. The received resources from different paths are not isolated by some user agents. As a result, resources from various paths might be able to access cookies that are stored for another path. (Barth, 2011.)

Guarantees of integrity for sibling domains and the subdomains of the sibling domains is not provided by cookies. As a result, for an example web server foo.website.com is able to set cookies with Domain attribute value website.com, that can overwrite an existing Domain attribute value website.com that could be set by bar.website.com. Hence, the cookie is included by a user agent in HTTP requests to bar.website.com. Bar.website.com might be unable to separate the cookie from the cookie bar.website.com set by itself. In this case, this ability could be leveraged by foo.website.com to mount attacks against bar.website.com. (Barth, 2011.)

Set-Cookie header supports the Path attribute. However, no integrity protection is provided by the Path attribute. An arbitrary Path attribute in a Set-Cookie header is accepted by a user agent. Cookies can also be injected into the Cookie header sent to https://website.com/ by an attacker. This is performed by imitating a response from http://website.com/. The website.com's HTTPS server is incapable to separate these cookies from cookies in an HTTPS response set by itself. The ability may be leveraged by an attacker to attack website.com. Even if website.com would use HTTPS solely. The attacks can be partly attenuated by using encryption and signing the cookie contents. However, cryptography does not attenuate the problem entirely. An attacker is able to replay cookies received from the authentic website.com web server in victim's session. This can have unpredictable results. (Barth, 2011.)

An attacker might be able to store a large number of cookies to oblige a victim's user agent to delete cookies. The victim's user agent will be obliged to eject some cookies when the storage limit of the user agent reaches its limit. User agents might not be able to conserve cookies. Therefore, web servers should not trust in user agents ability to conserve the cookies. For security, cookies trust the Domain Name System (DNS). Cookies protocol might not succeed providing the security properties that the applications require, if the DNS is compromised. (Barth, 2011.)

More than one Set-Cookie header field should not be included by web server in the very response with the very cookie-name. If several responses to a user agent contain Set-Cookie headers sent by a web server, the several responses cause a rivalry between the responses. This kind of event might cause unpredictable behavior. Technically, the several responses occur if a web server is, for an example, communicating over multiple sockets with the user agent. (Barth, 2011.)

There exists multiple security concerns concerning the attributes of the cookies. For an example, if the Domain attribute is missing, some user agents may misuse the missing Domain attribute and handle the Domain attribute as a present and a current host name containing attribute. The behavior results, for an example, in sending cookies to other hosts in the domain. For security, the Path attribute is not reliable even though the Path attribute seems worthwhile for secluding a cookie between various paths that exist in a given host. The Secure attribute seems worthwhile when protecting cookies from network attackers. However, the Secure attribute can protect only the confidentiality of a cook-

ie and attackers are able to overwrite secure cookies in insecure channels. The attackers can disrupt the integrity of the cookies. (Barth, 2011.)

2.5.1 Poisoning

Cookies are supposed to be stored to a user agent and sent back to the web server unchanged. An attacker may change the value of cookies and send the cookies back to the web server. Hence of this process cookie poisoning is performed. The attacker sends an invalid cookie to the web server by possibly changing the values of a valid cookie received from the web server. (Jacobs, 2016, p. 348.) When the web server uses the modified cookie, the values set by the attacker are processed by the web server. This may allow the attacker to gain access to the web site's or a user's sensitive information. The attacker may also impersonate the session of the user. (Dubrawsky, 2009, p. 105.)

Cookies are frequently used for transmitting sensitive credentials. Cookies can be relatively easily modified. This enables escalating access or assuming users' identities. HTTP protocol is stateless and cookies are used for maintaining a session state. The intention of sessions is to be uniquely anchored to the users accessing the web application. An attacker might modify the users' online experience by for an example injecting malicious content. Hence, the attacker might obtain unauthorized information. (EC-Council, 2017, p. 74.)

Session-specific data, that cookies can contain, includes information such as user identifier, password, account number, shopping cart content, supplied private information, and session identifier. An attacker might modify the cookie data to gain escalated access or affect maliciously the sessions of the users. Cookies are often encoded with easily cracked encoding methods like Base64 or ROT13. Whenever the attacker is provided user credentials by compromising cookies and sessions, the attacker can assume users' identities. The easiest example of cookie poisoning is to use the cookie directly for authentication. Another cookie poisoning method is to use a proxy to rewrite session data. In this method the attacker displays the data of a cookie or specifies a new user identifier or other identifiers of the session in the cookie. (EC-Council, 2017, p. 74.)

2.5.2 Hijacking/Stealing

Cookies can be hijacked or stolen by sniffing network traffic and capturing cookies that are downloaded from a web site to a web browser. An attacker may also gain access to a user's machine and view cookies that are stored on a local hard drive. Cookie hijacking enables an attacker to start another session to the very web site. Then the attacker submits the cookie to bypass authentication to execute malicious actions within the user's account. (Dubrawsky, 2009, p. 105.)

Cookie stealing attack is executed by applying a simple JavaScript code and supporting it with a PHP program. The code is sent to a victim. As the victim runs the code, the victim's cookie is sent to the attacker. The attacker may

use the cookie to access the victim's account. (EC-Council, 2010.) Ristic (2014) alleges that it is a common mistake for programmers to forget to encrypt the cookies to secure them. An attacker can use cookie stealing technique to obtain session tokens. According to Ristic, the attacker observes the victim's complete internet traffic. In other words, the attacker is an active MITM (Man-in-the-Middle).

If the traffic to the target web site is encrypted, the attacker cannot attack the traffic. Still, the attacker can expect for a victim to send an unencrypted HTTP request to some other web site. If the victim sends an unencrypted request, the attacker can hijack the unsecured connection. Then the attacker responds to victim's plaintext HTTP request by redirecting the victim's web browser to the target web site on port 80. The browser will redirect because web sites can issue a redirection to other sites. This process results in a plaintext connection to the target web site. The target web site carries unsecure cookies in the browser's possession. The attacker gains the session identifiers of the victim and may proceed to hijack the session. (Ristic, 2014, p. 115.)

2.5.3 Manipulation

Ristic (2014) discusses cookie manipulation attacks and claims that cookie manipulation can be divided into three types: cookie eviction, direct cookie injection, and cookie injection from related hostnames. Cookie eviction attack targets the store of the browser. Cookie stores have limited cookie size and the number of cookies that can be stored in a domain name. An attacker may try to exploit the cookie store facts mentioned above if the attacker dislikes the current cookies in the browser's store. The attacker submits multiple dummy cookies and as a result the browser cleanses all the actual cookies. This process leaves only the dummy cookies to the browser's store.

If a web site uses secure cookies, an attacker might perform direct cookie injection. The attacker would have to break the encryption of the secure cookies to proceed. Instead of breaking the encryption, the attacker might create new cookies or overwrite the present cookies. Insecure and secure cookies are located in the same namespace. Direct cookie injection attack can exploit the fact that the insecure and secure cookies locate in the same namespace. The attacker can catch a plaintext HTTP transaction that is launched by a victim to coerce a plaintext HTTP request to a target web site. The attacker catches the HTTP request. Then the attacker can reply the request with HTTP response including arbitrary cookies. The arbitrary cookie must match the original cookie's name, domain, and path. Also, metadata values used by the target web site need to be observed and replicated. (Ristic, 2014, p. 119.)

Cookies are shared with related hostnames. The attacker could exploit this fact, if it is impossible to impersonate the target web site. As a result, the attacker can use a technique called "cookies injection from related hostnames" to compromise some other website that is located in related hostname. The attacker might inject a cookie from the related hostname. In the attack, a victim sub-

mits a HTTP request to a vulnerable web site. The arbitrary cookies are set on the web site. (Ristic, 2014, p. 119.)

2.5.4 Cross-Site Scripting

Cross-site scripting (XSS) can be used for stealing sensitive information, hijacking user sessions, and compromising browsers and system integrity. (Grossman, Hansen, Petkov, Rager & Fogie, 2007, p. 11.) XSS attack is possible whenever a software does not neutralize user-controllable inputs or neutralizes user-controllable inputs incorrectly before the user input is plazed in an output used as a web page served to users (CWE, 2018).

In XSS attack an attacker injects HTML tags or scripts into a web site (Flanagan, 2006, p. 267). The injected malicious code is downloaded and executed by a user from the web site (Dubrawsky, 2010, p. 39). The attacker exploits a vulnerable application to send malicious code to the application's user. Usually the used coding language for XSS attack is JavaScript. (Faircloth, Beale, Temmingh, Meer, Walt & Moore, 2005, p. 207.) XSS attack process realises when the injected code is part of the original code. The focus of XSS is on attacking the client, not the web server. The goal of the XSS attack is to have the client executing the malicious script to perform an action. (Engebretson, 2011, p. 121.) In XSS, the victim is used for performing the malicious script's running to perform an action the attacker desires.

Cookies can be exploited for attack initiation with persistent XSS. According to Oriyano and Shimonski (2012) cookies are used for storing user information, users' sessions with web sites, and other information depending on the web site. The information that cookies contain can attract attackers. The cookies' information can be maliciously accessed with XSS. An attacker can create or alter a page on a web site with an embedded script designed for extracting cookies' information to access the cookie. The page is sent to a victim's browser whenever the victim visits the page. The victim's browser renders the page and executes the script. This process will allow the attacker to steal the victim's information.

XSS attack consists of three components. First, a victim enters a web site that has been infected or controlled by an attacker. Next, a series of scripts are used by the web server to interact with the victim. The scripts allow the attacker to gain access or control. Eventually, the gained data is used by the attacker to attack the victim. Cookies are used by most web sites. Cookies are used for advertising, user tracking, and storing information. A security professional needs to accept and learn to deal appropriately with the fact that cookies are created and managed inherently in every web browser. (Oriyano & Shimonski, 2012, p. 37.)

XSS is able to read links, scan web pages, and read any web page on the same hostname. Cross-site request forgery attack can be executed on the web page whenever there exists XSS on the web page because nonces can be read. Malicious JavaScript is able to interact with a web page like a user. Once an at-

tacker finds a XSS vulnerability on a web page, the attacker owns the web page and is able to spawn requests on the host's other pages. (Grossman et al, 2007, p. 94.)

According to Wu and Zhao (2015) There are multiple types of XSS attacks. The types are reflected XSS, stored XSS, and DOM-based XSS. In reflected XSS, an attacker tricks a user to click malicious links to be able to perform the attack. The user's data input is reflected onto the browser. Reflected XSS can be also named as nonpersistent XSS. Data is read by the web server from the HTTP request. The web server reflects the data back in the HTTP response. An attacker causes a victim to deliver malicious content to a vulnerable web site. The malicious content is reflected back to the victim. The victim's web browser executes the malicious content. Malicious content can be delivered by including the malicious content in an URL as a parameter. The URL is sent to the victim publicly or by email. This manner of constructing URLs allows multiple phishing schemes. An attacker can convince a victim to visit a URL referring to a web site that is vulnerable. Victim's browser executes the malicious content after the web site reflects the malicious content back to the victim. (CWE, 2018.)

Stored XSS has a strong stability. The attack sends user data stored on the target web server. An attacker may write a blog that contains malicious JavaScript code. As a result, a browser that has access to the blog executes the malicious JavaScript. This attack can be called persistent XSS. The effect of the attack is relatively long. (Wu & Zhao, 2015, p. 47.) Malicious data is stored in a database by a web site. Afterwards, the malicious data is read back into the web site. The data is included in dynamic content of the web site. The displayed area of a web site that is visible for multiple users or particularly interesting users is an optimal place to inject the malicious content. Elevated privileges in web sites or interacting with sensitive data make a user interesting from the attacker's perspective. The attacker may perform privileged operations on the behalf of the user or gain access to the users' sensitive data whenever users that were mentioned above execute the malicious content. (CWE, 2018.)

DOM-based XSS differs from the other types of XSS. In DOM-based XSS, instead of the web server performing the injection, the client performs the injection into the web site. DOM-based XSS uses a trusted script that is controlled by the web server and sent to the client. The trusted script can be for an example a JavaScript that checks the sanity of a form before a user submits the form. DOM-based XSS is possible if the script that is supplied by the web server processes data supplied by a user and injects it back into the web site. (CWE, 2018.)

After the success of the XSS attack, the attacker can implant a malicious script that will control the victim's browser (Wu & Zhao, 2015, p. 49). Untrusted data entry into a web site that originates typically from a web request can lead into a XSS vulnerability. XSS vulnerabilities may also occur in a web page containing untrusted data that is dynamically generated by a web site. Whenever a web page is generated, the web site might not prevent data from including content that can be executed by a web browser. The content could be for an example JavaScript, HTML tags, HTML attributes, mouse events, Flash, or ActiveX.

XSS vulnerabilities can be manifested whenever the generated web page is visited through a web browser by a victim whose web browser includes malicious script after using untrusted data. The malicious script is executed in the context of the domain of the web server because the script comes from a web page that was sent by the web server. Web browsers support the same-origin policy, which is effectively violated in the vulnerability. The same-origin policy state that resources of another domain should not be able to be accessed by scripts in another domain. Neither should one domain be able to run code in another domain. (CWE, 2018.)

When the attacker manages to inject the malicious script, the attacker is able to perform a variety of malicious activities. Private information could be transferred from the victim's machine to the attacker. The private information could be for an example, cookies that may include information about the victim's session. Also, malicious requests could be sent to a web site on the behalf of the victim by the attacker. If the victim has privileges of an administrator to manage a web site, this action could be especially dangerous to the web site. Another means could be the use of phishing attacks to emulate trusted web sites. The victim can be tricked to enter a password. This would allow the attacker to compromise the account of the victim on the web site. If the web browser can allow the attacker to exploit the vulnerability to capture the machine of the victim. This technique is also known as drive-by hacking. The victim is rarely aware of the launched attack. There exists a variety of methods that can be used by attackers to encode the malicious portion of the attack. Techniques for this kind of action include URL encoding or Unicode. The request looks less suspicious when the attacker uses these methods. (CWE, 2018.)

2.5.5 Cross-Site Request Forgery

The idea of cross-site request forgery (CSRF or XSRF) is to exploit the fact that web sites trust the HTTP requests of users (Alcorn, Frichot & Orrù, 2014, p. 440). According to Halton, Weaver, Ansari, Kotipalli and Imran (2016) CSRF attack follows a flaw similar to XSS. Unlike in XSS, in CSFR an attacker attempts to cause a victim to perform an action without need for using a script. The attacker attempts to take over the victim's identity. The attacker attempts to perform actions on the behalf the victim. The target of the attack is a web site in which the victim is authenticated currently. According to Stuttard and Pinto (2011) web sites may be vulnerable to CSRF attacks if the web site solely relies on cookies as session identifier transmitting method.

CSRF is possible if a web site is not able to adequately verify if a well-formed, valid, and consistent request is provided intentionally by the user that submits the request. If a web site has no mechanism for verifying that requests are intentionally sent, from clients, an attacker might be able to trick a client to perform unintentional requests to the web site. The unintentional request is treated as an authentic request. Methods for doing this trick are through a URL, image load, or XMLHttpRequest. As a result, data can be exposed or code can

be executed unintentionally. The attacker might gain privileges or assume identity. The attacker might be able to bypass protection mechanisms, read application data, or modify application data. Another possible technical impact is denial-of-service (DoS) in the form of crash, exit, or restart. (CWE, 2018.)

The nature of the functionality that is vulnerable to CSRF effects on the consequences. In CSRF attack, an attacker might be able to efficiently perform operations as the victim. The attacker might be able to obtain complete control over the web site if the victim is an administrator or a user with privileges. As a result, the attacker might be able to delete or steal data, uninstall products, or use the products to attack all of the users of the products. The privileges of the victim are the only limiting factor of the scope of CSRF. The attacker has the identity of the victim. (CWE, 2018.) If an attacker succeeds in CSRF attack, the entire web site is in danger.

In CSRF, a web site is maliciously exploited. The target web site trusts the user that transmits the unauthorized commands to the web site. Other terms of which CSRF is also known are one-click attack, and session riding. In contrast to XSS, CSRF requires no scripting. As a result, turning all scripting off does not prevent a user from becoming a victim. Scripting may though be used for creating a sophisticated CSRF attack. XSS attack exploits the user's trust for a web site. CSRF attack exploits web site's trust for a user's browser. (Timm & Perez, 2010, p. 35.) CSRF attack is challenging to be identified because the target web site trusts the client that send the requests.

There exists persistent and nonpersistent methods of CSRF attacks. In the persistent method, the code is stored on the server. In contrast to persistent method, in the nonpersistent method of CSRF attack, the code is not stored on the server. In case of persistent CSRF vulnerability, an attacker uses a web site itself to supply a malicious exploit link to a victim. The attacker can also supply any other types of content that can direct the browser of the victim back into the web site. As a result, actions controlled by the attack are executed by the victim. This type of attack contains a risk that the attack leaves a trail that could potentially point back to the attacker. However, persistent CSRF attacks are more presumable to succeed than nonpersistent CSRF attacks. In nonpersistent CSRF vulnerability, the victim is exposed to the exploit link or content by the attacker using an application outside the system. (Timm & Perez, 2010, p. 35.)

If an attacker manages to perform a successful CSRF attack, end user data can be compromised. Whenever the targeted user has administrative privileges, the entire web server can be compromised. Normally the targets of CSRF attacks are web server's state changing functions, however, CSRF attacks can target sensitive data in case of targeting an account with administrator privileges. Web servers are commonly not able to separate malicious requests from legitimate requests due to the fact that browsers commonly incorporate credentials concerning a particular web site with requests. The credentials are automatically incorporated with the requests. The credentials may include session cookies and basic authentication credentials. Whenever a user is authenticated to the web site, the web site is unable to separate malicious and legitimate requests.

CSRF attack can also be stored on a vulnerable web site itself. Whenever, a field in a web site accepts HTML, IMG or IFRAME tags, they can be used to store CSRF attack. (Barrett et al., 2015, p. 300.)

2.5.6 TCP/IP Hijacking

TCP/IP hijacking problem has existed in the most of applications that are TCP/IP-based. TCP/IP hijacking is also known as session hijacking. An attacker needs to be able to intercept the data of a legitimate user in order to hijack TCP/IP connection. Then the attacker inserts herself into that session. In web-based application's session hijacking involves hijacking a user's cookie. The cookie can be used for storing sensitive information such as login credentials. The attacker may use the cookie for accessing the session of the user. The user is probably not aware what happens and receives a "session expired" or "login failed" message. If session timeouts are incorrectly configured in web server application, an attacker may perform session hijacking. Typically, timeouts are configured to happen after a set period of inactivity in user's session. An attacker may potentially use a hijacked cookie or predict session identifier numbers to hijack a session of a user if the time frame of timeouts is too extensive. (Dubrawsky, 2007, p. 44.) The connection should be closed in a concise timeframe to prevent session hijacking.

An attacker trying to hijack user credentials and session information to gain an unauthorized access to a server can easily identify browser traffic. Browsers use predefined port (for HTTP port 80 and for HTTPS port 443) to access web servers' resources. In the case where data traffic is encrypted when transferred between endpoints with SSL, an attacker can use a web proxy with SSL. This web proxy may enable the attacker to allow a victim to connect the web proxy. Whenever a victim connects the web proxy, a secured link between the web proxy and the web server's resource the victim intended is created. The attacker captures data transport in plaintext on the web proxy. The victim will receive responses reporting that the connection is secured. (Barrett et al., 2015, p. 251.)

In addition to TCP/IP hijacking and session hijacking, TCP/IP hijacking is also known as active sniffing. The attacker gains access to a host and logically disconnects the host from the network. Then another machine with the same IP address is inserted by the attacker. The attacker gains access to the session. The attacker also gains access to all information on the original system. The web server trusts the client and will not discern the hijack. (Dulaney, 2009, p. 76.) TCP/IP hijacking is hazardous attack technique because the attacker might gain access to the original system.

Web sites are generally generated to encrypt the login process to protect users' accounts. What takes place after the login process is infrequently encrypted. The fact that web sites rarely encrypt the functions after the login process, make the cookies and the users vulnerable. (Anto, 2012, p. 193.) Session identifiers can be retrieved from network traffic stream that is unencrypted.

Performing TCP/IP hijacking attack is exceedingly easy in wireless or local network. The attacker observes the unencrypted network traffic and extracts the session identifiers from the traffic. (Ristic, 2014, p. 113.)

2.5.7 HTTPS Protocol

HTTPS (Hypertext Transfer Protocol Secure) functions the same as HTTP. The difference between HTTP and HTTPS is that HTTPS traffic is transported tunneled over Secure Sockets Layer (SSL). (Stuttard & Pinto, 2011.) HTTPS is intented to protect data privacy and data integrity over networks. Lobo and Lakshman (2014) stated that HTTPS use SSL and TLS (Transport Layer Security) for providing authentication and encrypting data. Whenever HTTPS requests are initiated, TCP connection is established to designated port on a web server. The port is 443 by default. URIs or URLs use HTTPS URI schemes that are used for identifying the resources to be accessed by HTTPS. (Stuttard & Pinto, 2011.)

HTTPS includes a digital certificate that is designated in the web server. Clients connect to the web server's HTTPS port and are authenticated to the web server by the certificate. Next, the connection's security protocols are negotiated between client and web server. Session keys are generated to encrypt and decrypt the exhange. Clients should not be able to establish encrypted secure session and the negotiation of the security protocol should not proceed if the authentication would fail. (Stuttard & Pinto, 2011.)

Is HTTPS actually a safe protocol? There exists multiple side-channel attack methods against HTTPS. The thought of securing secret data by the protection of TLS is revoked. HTTPS leaks information, such as, timing and ciphertext's length. Attackers can leverage the leaked information. The leaked information can be used to recover secret data. The information leakage in TLS compression can be exploited by CRIME and BREACH attack methods. Browsing privacy can be recovered also by an attack method using traffic analysis with statistical algorithm. (Chen, Duan, Zheng, Jiang and Chen, 2018)

If the Secure attribute of a cookie is set to 'true', the cookie is transmitted only over HTTPS connection. The HttpOnly attribute should prevent the reading of cookies by client side scripts. (Chen et al., 2018) However, the HttpOnly flag is ignored by a browser, if the browser is not capable to support the HttpOnly. As the HttpOnly attribute is set to 'true' and the cookie is attempted to set by the browser, the HttpOnly flag is ignored. Therefore, the procedure creates a cookie that could be accessed by scripts. (OWASP, 2017.)

Separation between different schemes (like schemes in HTTP and HTTPS) lacks in cookies. Integrity lacks are present in cookies. If an active MITM attacker can manipulate network traffic, the attacker may inject into a browser of a victim arbitrary cookies. The cookies are attached to subsequent HTTPS requests. For this to realise, the cookies' Domain and Path attributes need to match the request-URL. This feature works as a leverage for a MITM attacker to be able to inject cookies from HTTP sessions. The attacker is able to observe HTTPS requests' size. Whenever the Path attribute of a cookie correspond

HTTPS request-URL path, HTTPS request size inreases. Consequently, side-channel attacker may be able to deduce request-URI's sensitive information from the traffic that is encrypted. (Chen et al., 2018)

2.6 HTTP Cookie Attack Prevention

Next, we discuss the prevention methods for cookie attacks. There are several methods for preventing attacks that are targeted at cookies or use cookies as an attack vector. According to Barth (2011) web server operators could treat URLs as capabilities to entangle designation and authorization. This should be done instead of using cookies as authorization resources. Hence, secrets would be stored in URLs instead of cookies and a remote entity should supply the secret itself. More robust security could be achieved through a reasonable solution of the mentioned principles. However, no solution is absolutely bullet proof in case of cyber security.

The contents of cookies should be encrypted and signed by web servers when web servers transmit cookies to user agents. However, an attacker can transplant a cookie from a user agent to another or replay the cookie later. Encryption and singing does not prevent this. The Cookie and Set-Cookie headers should be transmitted only over a secure channel by servers requiring a higher security level. The Secure attribute should be set for every cookie when transmitting cookies over a secure channel. Without using the Secure attribute, the protection supplied by the secure channel is controversial. (Barth, 2011.) All cookies should use the secure attribute and be sent via a secure channel.

Web servers generally store a nonce or session identifier in a cookie. This is done instead of storing session information directly in cookies. Web servers use the nonce as a key when searching state information associated with the cookie. This is done after receiving a HTTP request with a nonce. The damage an attacker can cause when learning the cookie's contents is limited when using sessions identifier cookies. The nonce is useful only when interacting with web server. An attacker is prevented from mounting cookie content from two interactions with the web server by using a single nonce. (Barth, 2011.)

2.6.1 TCP/IP Hijacking and HTTP Cookie Theft Prevention

Instead of cookies, session identifiers should be transmitted and managed with some other standard. The standard could be new request/response header. There should be a mechanism in the standard to attach the session identifier to SSL session. This should be done on both the browser and the web server. (OWASP, 2013.)

Session identifiers should be transmitted only over encrypted protocols. If the session identifier is transmitted insecurely, the session should be terminated or regenerated. Sensitive cookies must be enforced to add Secure and HttpOnly

flags. This could be established by using Web Application Firewall. (OWASP, 2013.)

A framework for protecting against TCP/IP hijacking and cookie theft should include certain features. Session token leakage should be prevented. Direct access to cookies should be prevented by a centralized API for management. Secure and HttpOnly flags should be applied in cookies. Domain and Path attributes should be correctly set. (OWASP, 2013.)

Users should be instructed to enable cookies. Browser should be upgraded to support the cookie features that are required. Session identifier should be passed as a POST paramater on all forms to support cookieless sessions. Commonly supported cookie data size by browsers or the amount of cookies might exceed. This should be automatically validated and signaled. Web site code should not overwrite or manipulate the session cookie. Web site could be blocked from accessing session identifiers. Session identifiers should not be included in URLs. Neither should they be accepted as URL parameters. (OWASP, 2013.)

The user should be alerted and the oldest sessions deauthorized if web server detects multiple simultaneous logins. The default is that multiple simultaneous logins are forbidden. However, the simultaneous logins may be enabled through modifying the configuration. Whenever User-Agent string or other fingerprinting of clients change, session should be terminated and security SNMP trap should be sended. Session identifiers should be attached to SSL session. Configurable options should be taken whenever session identifier is transmitted through new SSL sessions. Session should be pinned to originating IP address when user logins. (OWASP, 2013.)

2.6.2 Cross-Site Scripting Prevention

There exists very few studies that try to resolve the XSS problem. OWASP (2018) tries to create a model for mitigating the problem. XSS prevention model could follow certain rules. If data is not trusted, the data should not be inserted in allowed locations. Untrusted data should not be accepted into a HTML document. The most important aspect of this is not to accept JavaScript codes from sources that are not trusted.

XSS flaws prevention is hard. HttpOnly flag should be set on sessions cookies and custom cookies. In most coding languages the HttpOnly flag needs to be set manually. (OWASP, 2018.)

2.6.3 Cross-Site Request Forgery Prevention

According to OWASP (2018), two separate checks should be added to standard CSRF defense. Verifying that the request is from same origin should be done by checking standard headers. CSRF token should be checked also. To verify that the request is from same origin, the source origin of the request should be determined. Also the target origin should be determined. Examining the HTTP

request header value is relied by these checks. Whenever CSRF attack is running, spoofing headers from a browser with JavaScript is impossible. Situation where an XSS vulnerability exists in the web site that is attacked with CSRF. The checks rely on Origin, Referer, and Host headers.

Source origin should be idenfied by using Origin Header or Referer Header. These headers are included in most of requests. Target origin value should match the Origin header value. Altough, there are some cases where the Origin header is not present. The hostname in the Referer header should verified that it matches the target origin whenever Origin header is not present. In both of these checks, the target origin check should be strong. Changes in the path of origin should not be accepted. In a case where both of the headers are missing, the requests should be blocked. (OWASP, 2018.)

Identifying the target origin is rather difficult. Target origin may locate behind multiple proxies and the original URL differs from the received URL. In case where the web server can be accessed directly by users, the URL's origin is the same. Whenever the web server locates behind proxies, the web site could be configured to know the target origin. Alternatively, Host header value could be used. Another alternative is to use the value of X-Forwarded-Host header. (OWASP, 2018.)

When the request is verified as same origin request, a another check that may include custom defence mechanisms that use tokens that are CSRF specific. The tokens may be created and verified by the web site. Another option is to rely on HTTP headers being present. To specifically defend against CSRF, there exists multiple methods in addition to the checks. The methods are Synchronizer Tokens, Double Cookie Defence, Enrypted Token Pattern, and Custom Header. (OWASP, 2018.) This research focuses on cookies. Therefore, Double Cookie Defence is examined next.

Double Cookie Defence is known as Double Submit Cookie. A random value is sent in a cookie and as request parameter. The web server verifies if the value in cookie and in request match. Web site should generate cryptographically strong value that is pseudorandom whenever a user authenticates to a web site. The value should be set only on the machine of the user as a cookie. The cookie should be separated from the session identifier. Every request in the transaction should be required to include this value in a hidden form (hidden form value). An attacker is unable to change this value. (OWASP, 2018.)

2.6.4 Session Protection – HTTP Cookie Attributes

Secure attribute of the cookie gives a browser instruction to transmit the cookie only over ecrypted HTTPS connection. This is a protection mechanism provided to be able to prevent the session identifiers to be disclosed through Man-in-the-Middle attacks. Consequently, an attacker should not be able to capture session identifiers from web traffic of the browser. (OWASP, 2017.)

HttpOnly attribute tells browsers to disallow scripting to access cookies. The access would be through DOM document.cookie object. HttpOnly attribute prevents session identifiers to be stolen via XSS attacks. (OWASP, 2017.)

Web server can define a cookie attribute. The ability is brought by SameSite attribute. The ability disallows browser to send the cookie with cross-site requests. Cross-origin leakage is mitigated and protection against CSRF attacks is increased. (OWASP, 2017.)

Cookie should be sent only to specified domain and all subdomains according to the instructions set in Domain attribute. The cookie is sent only to the origin server by default if the attribute is not set. Cookie is sent to specified directory or subdirectories of web site according to the Path attribute. The cookie is sent only to directory the resource requested whenever the attribute is not set. (OWASP, 2017.)

The scope of Domain and Path attributes should be restricted or as narrow as possible. When the Domain attribute is not set, the cookie is restricted to the origin server. Path attribute should set highly restrictive to path of web site that uses the session identifier. If Domain attribute value is highly permissive, an attacker may launch attacks against session identifiers between various hosts and web sites that work in the same domain. If www.website.com is vulnerable, the attacker is allowed to gain access to session identifiers from foo.website.com. (OWASP, 2017.)

Cookie-based session management mechanisms may use two different types of cookies: session cookies and persistent cookies. If Max-Age is set in cookie, the cookie is considered a persistent cookie. Browser stores the cookie until expiration time. Session cookies are used to track users after authentication. Whenever the browser is closed, the session disappears. Session management should be implemented with session cookies. Consequently, web client cache will not preserve the session identifier for a long time. (OWASP, 2017.)

3 METHODOLOGY

This research reviews the literature that discusses cookie vulnerabilities, attack vectors, and defense mechanisms to investigate the weaknesses of cookies. The chosen research method is systematic literature review. Systematic literature review helps to understand the phenomenon via previously developed knowledge. The purpose of this study is to locate, appraise and synthesize the evidence that relates to the research questions. The study will provide informative evidence-based answers for the research questions.

The objective of systematic literature review is to deliver a clear and targeted answer to the selected research question. To minimise bias, systematic literature review aims to allow for replication in a way that minimises the bias. Systematic literature review outcomes results that are searched from variety of studies that relate to the topic. The strenght of the results are evaluated and evidence for practice is summarized. Systematic literature review is instituted with specific review question, identifying all relevant studies, evaluating the quality of the studies, and summarising the results of the studies with a scientific methodology. (O'Brien & Guckin, 2016.)

There are several advantages for approaching the research with systematic literature review. To assure that the maximum scope of relevant research has been examined, systematic literature review exploits extracting search strategies. The examined studies are evaluated methodologically and synthesized. All research on the topic is located, evaluated and synthesized. Systematic literature review inludes a procedural quality that is designed answer the research questions. To reduce the risk of distortion systematic literature review provides a transparent methodology and follows a strict and reproducible protocol of procedures. Systematic literature review is able to obtain information about the phenomenon being robust and portable. Systematic literature review is capable of identify research areas with less or no relevant research and research areas with a need of further research. (O'Brien & Guckin, 2016.)

Becides the advantages, there exists disadvantages in systematic literature review. The bias reduction of systematic literature review does not eradicate the risk of distortion completely. If criteria for inclusion and exlusion to extracting

data for critical evaluation is not implemented correctly, the inclusion and exlusion criteria are predisposed to create distortion. There exists no broadly accepted method to assess the validity of studies. Consequently, reviewers may dissent with information conduction and analysis. (O'Brien & Guckin, 2016.)

To decide if systematic literature review is relevant, several aspects should be considered. First, it is necessary to investigate whether related research exists. Next, the methodological quality of the chosen studies should be assessed. Then the distortion should by identified and minimised. (O'Brien & Guckin, 2016.)

3.1 Research Process

The first stage of systematic literature review is identifying a research question. The main research question of this research was defined based on the interest of the researcher. To identify deficiency in the research of cookies, literature and online sources were examined. An existing gap was found. Based on the interest of the researcher and the found gap, the purpose of this study was defined. To verify that there exists no identical studies, two supporting research questions were added. The aim of the study was to identify weaknesses of cookies combined with cookie attack methods and defense mechanisms.

After defining the research questions, different databases were searched systematically. The keywords used for finding relevant literature were:

- HTTP cookie (7214 search results)
- HTTP cookie specification (2244 search results)
- HTTP cookie vulnerability (896 search results)
- HTTP cookie weakness (332 search results)
- HTTP cookie attack (1259 search results)
- HTTP cookie protection (1991 search results)
- HTTP cookie safeguard (239 search results)

The keywords were specified as shown above. The keywords were intentionally centralized cookie specific. If the term ”HTTP” would not be specified in the keywords used for the searches, the searches would give unrelevant results. The search process followed the stream shown in figure 1.

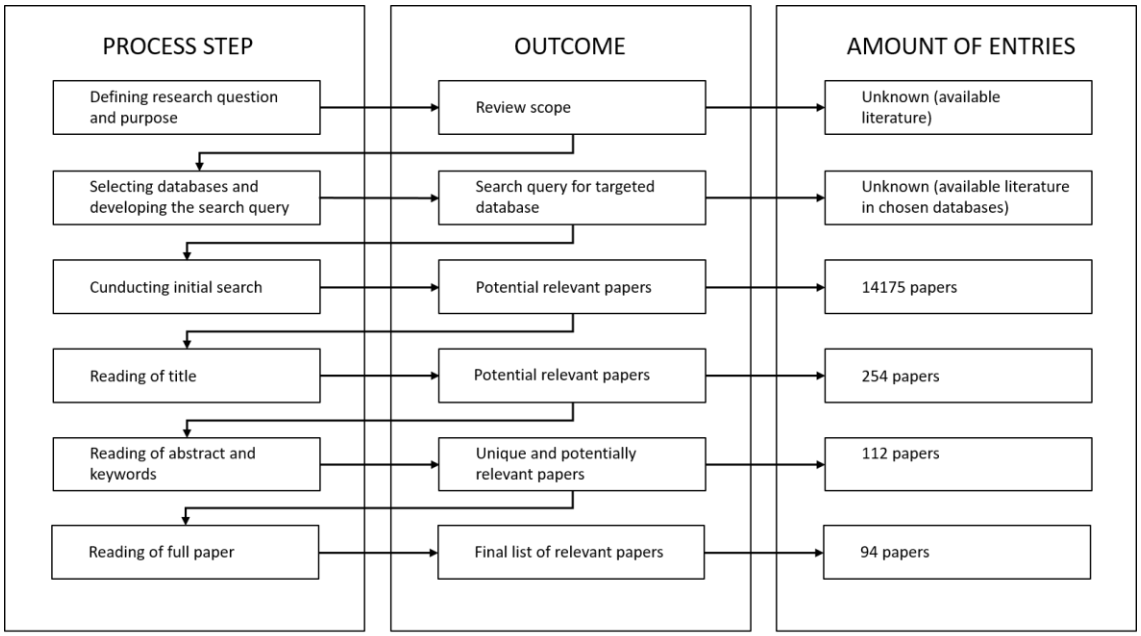

FIGURE 1 Search strategy

The search strategy consisted of 6 steps. First, research question and purpose was defined. Next, databases databases were selected for the searches and the search queries were developed. The search queries were feed to targeted databases. The databases were Google Books, Google Scholar, and JYKDOC. Also Google search engine was used to supplement the evidence found in the literature.

The initial search was conducted to find potential relevant papers. The searches disclosed 14175 papers. The title of the papers were read to exclude papers that had unrelevant title. The total amount of papers was reduced to 232 papers. Next, the abstracts and keywords were examined to find unique and potentially relevant papers. The amount of papers was reduced to 112 papers. The last step of the search strategy was reading of full papers. The reading resulted in 94 papers, which became the final list of relevant papers.

Next, inclusion and exlusion criteria was set. The scope of the study and objectives were placed to answer the research question correctly. The scope of the study is limited on the weaknesses, attack methods and defense mechanisms found in relevant literature. Relevant studies according to the research question were selected and observed in detail to verify their quality. Inclusion and exclusion criteria was used to capture and encompass the research question ”what are the weaknesses of HTTP cookie?” that this thesis seeks to answer. The inclusion and exclusion criteria is described in table 4.

TABLE 4 Inclusion and exclusion criteria

Criteria	Inclusion criteria	Exclusion criteria
Criterion 1	**Must address the technical mechanisms of**	**No technical aspects**

	cookies	
Criterion 2	**Must discuss cookies in comprehensive environment**	**Does not address the comprehensive environment of cookies**
Criterion 3	**Must indicate weaknesses of cookies**	**Does not provide relevant and up-to-date information**
Criterion 4	**Discusses the various components that impact the function of cookies**	**Does not discuss the elements that have significant affect on cookies**

Literature was selected for inclusion based on the predefined criteria. Based on title and abstract, literature that comprehended information about cookies, cookie attributes, cookie vulnerabilities, cookie attack methods, and cookie defense methods were chosen for deeper examining. The adequacy of the results influenced the selection of precise results. As the search results were mirrored againts the inclusion and exclusion criteria, 31 books, 6 research papers, and 22 online references were selected for the literature review.

The premier references from the perspective of the research problem and the most informative references included RFC document 6265 (Barth, 2011), OWASP online security community, CVE cyber security vulnerability and exposure dictionary. The papers from Dubrawsky (2007, 2009 & 2010), Ristic (2014) and EC-Council (2010, 2017) proved beneficial to the research. This research discussed principally the contents of the RFC 6265 document and supported and challenged the specification of the RFC 6265 by examining the related papers.

RFC 6265 document was chosen for defining the specification of cookies and to understand the weaknesses of the specification. The specification was combined with supporting literature to establish a comprehensive understanding the essense of cookies, how cookie protocol functions, what are the mechanisms that participate the cookie function, and how the surrounding mechanisms impact the function of cookies.

The searches were manually transfered into categories and documented. The categories were:

- Specification
- Weakness or vulnerability
- Attack method
- Defense mechanism

The searches containing similar information were meta-analyzed for similarities and differences. The similarities were examined to import variety to the aspect cookies. The searches were documented to avoid duplicates.

Next, the quality of the studies was assessed. The quality was assessed by observing the design, conduction and reporting of the studies. If the design,

conduction and reporting of the literature in a way the study could be considered reliable and robust, the study was retained. The studies that supported providing a meaningful answer to the research question were selected. The studies mirrored with the inclusion criteria. Irrelevant studies were removed.

In the last step, the search results were detailed and summarized in the literature review section. The search findings were analysed and written in a comprehensible form. The findings were summarized for further examining. The summarized findings were discussed.

3.2 Reliability and Validity

The results of this research are conducted mainly from documents that describe the technical specification of cookies, attack methods against cookies, and defense methods to secure cookies from the attacks. The collected evidence is mainly tested in cyber environment by the researches to proof the quality of the outcome in the documents. It is possible to generate different outcomes by adding or modifying the variables of the tests.

Repeating the research in the same methods with the same literature will generate similar results with high probability. If there were technical testing environment included in the research, it might be possible to outcome in different results. Whenever the results would be different, some specification have been modified. However, the functionality of cookies is mainly specified. Modifying attack methods and the use of different attack methods may result in different results. However, cookie specification and functionality is specified to function in a certain manner. Consequently, the functionality of cookies and attacks should be transmuted to gain different results.

This research resulted in answering the research questions. The objective of this research was to examine the weaknesses of cookies, the attack methods that can be used to exploit the weaknesses, and the defense methods to protect cookies against the attacks. The results introduced clear understanding of the aspects that affect cookies by allowing cookies to be vulnerable and the aspects that create weaknesses in cookies. The results also introduced attack methods that can be directed to cookies and exploit the weaknesses of cookies. The results also introduced defence methods that can be applied to cookies or to the mechanisms in the enviroment cookies function.

4 FINDINGS

Cookies involve multiple weaknesses. The problematic aspect of information security is that attack methods evolve rapidly and defence mechanism evolve slowly. Cookies are an important component in making user experience superior in web services. There are multiple complexities in the use of cookies. As a technique, cookies are vulnerable and bias as incomplete component that has pontential, yet the vulnerabilities and weaknesses of cookies can lead into serious arbitrary impacts through severe attack methods.

In this section the results reached in the research process are addressed. First the results found in the viewpoint of weaknesses of cookies are presented. Next, the results found in the viewpoint of attack and defence methods are presented.

4.1 Weaknesses of HTTP Cookie

Cookies can encourage developers to use ambient authority. Using ambient authority may lead into remote parties to be able to distribute HTTP requests from users agents. This is enabled whenever a web server applies cookies to user authentication. If cookies are applied to user authentication, security vulnerabilities are exposed. Hence, an attacker may perform CSRF attack. This weakness may also cause confused debuty problem.

Cookies also encourage web server operators to segregate designation from authorization. Hence, an attacker might designate a resource that can be supplied an authorization by a user agent. This may lead into situation where actions that an attacker has designated might be undertaken by a web server or the clients of the web server. The malicious actions will be addressed as authorized in belief that the actions are authorized by the authenticated user.

There are always risks in data inputs. User inputs should always be validated in web services. User input should never be trusted. There are potential security risks with data inputs in web services.

Storing session identifiers in cookies discloses a severe problem. If session identifiers are stored in cookies, an attacker may transplant the session identifiers from the attacker's user agent to victim's user agent. This may result in victim using the session identifier for interaction with a web server. The transplanted session identifier is filled with the user credentials or confidential information of the victim.

Even if the web server is equipped with HTTPS protocol, the not set Secure attribute in cookies exposes the ongoing requests to an attacker that intercepts any outbouding requests of the user agent. This may realise in case in which the session identifier is stored in the cookie. By intercepting the requests, the attacker may redirect the requests to a web server over HTTP. Even if the HTTP connections are not listened by the web server, the user agent includes the cookies in the request.

However, transmitting cookies over secure channel (HTTPS) provides superior protection to cookies when compared to transmitting cookies over insecure channel (HTTP). Whenever cookies are transmitted over insecure channel, cookies are transmitted in clear text. Consequently, sensitive information of the Cookie header and Set-Cookie header may be predisposed to eavesdropping. During the transmission over insecure channel the headers may be disposed to malicious intermediaries. It is possible for a malicious client to modify the Cookie header before the transmission.

A missing Domain attribute may be misused by user agents and handled as present and current host name containing. As a result, the cookies may be sent to other hosts in the domain.

Due to the fact that user agents' cookie storage is limited to specific amount of data, user agents may be forced to delete cookies. An attacker may store a large number of cookies to victim's user agent. The user agent is forced to drop cookies whenever the storage limit is full.

4.2 Attack Types and Defense Methods

It is inevitable to pay attention to the fact that no defense method can quarantee full security against continuously developing attack methods. Defense methods against cookie attacks should be developed from the point of view that they can and will be compromised. Developers should imitate the mindset of attackers and contemplate how the function being developed could be exploited or compromised. There is no bulletproof systems and no bulletproof safeguards. The job of attackers should be made more challenging.

Cookie weaknesses, attack methods exploiting the weaknesses, and defense methods to mitigate the attacks are shown in table 5.

TABLE 5 Cookie weaknesses, attack methods, and defense mechanisms

Weakness	**Attack method**	**Definition**	**Defense mecha-**

			nism
Identity-based cookies (cookies in a form of ambient authority)	CSRF, cookie poisoning	The origin of a request is impossible to reliably authenticate.	Source origin and target origin should be checked with separate checks. SameSite attribute should be set to strict mode. Web application firewall should be used.
Separating designation (URLs) from authorization (cookies)	CSRF	A resource designated by an attacker might be supplied authorization by a legitimate user.	Cookies should not be used for authorization. Instead URLs could be treated as capabilities. SameSite attribute should be set to strict mode.
Untrusted data inputs	XSS	An attacker can access cookies by injecting script that is designed for extracting information in cookies to a web site.	Do not allow any untrusted data inputs. Set the HttpOnly flag.
Storing session identifiers in cookies	Session fixation	An attacker transplants session identifier from the attacker's user agent to victim's user agent to interact with with a web server.	Do not store session identifiers in cookies.
Unsecure transmission protocol (clear text)	Eavesdropping, cookie hijacking/stealing	1) The sensitive contents of a cookie are exposed to an attacker. 2) An attacker modifies cookie headers.	Encrypt and sign the contents of cookies. Set the Secure flag.
Lacking isolation by port		Cookies are readable by services run-	Do not run resiprocally distrus-

		ning on different ports of a server.	ting services on the same host's various ports and store sensitive information in cookies.
Lacking isolation by scheme		Cookies are available to different schemes (for an example FTP) instead of isolating availability to HTTP and HTTPS schemes.	
Lacking isolation by path		User agent do not isolate resources that it receives from various paths of the same host. Cookies that are stored for different path can be accessed by the retrieved resources.	
No integrity for sibling domains and their subdomains	Cookie injection from related hostnames	Sibling domains (for an example foo.website.com) are able to set cookies with Domain attribute value of another domain (for an example website.com) and overwrite an existing cookie set by another domain (for an example bar.website.com).	
Reliance upon DNS		Cookies do not provide secure properties that applications require whenever the DNS is compromised.	
Limited cookie storage in browser	Cookie eviction	An attacker sends multiple cookies to	Changes in cookie storage

		a victim to force the victim's browser to cleanse the actual cookies and to fill the victim's cookie storage with arbitrary cookies.	should be monitored.
Secure transmission protocol	Direct cookie injection	Secure and insecure cookies may locate in the same namespace. Whenever an attacker captures plaintext transaction from victim's target web site, the attacker replies the http response with arbitrary cookies.	Locate secure and insecure cookies in different namespace.
Storing sensitive information, such as login credentials, in cookies	TCP/IP hijacking	An attacker uses a user's cookie to access the user's session. Session timeouts are incorrectly configured.	Do not store sensitive information in cookies and configure timeouts correctly.

4.2.1 HTTP Cookie Attacks and Defense Methods

As shown in the table 5, cookies face multiple attack methods. The attack methods include cookie poisoning, cookie manipulation, and cookie hijacking/stealing. Other attack methods that impact on cookies include XSS, CSRF, TCP/IP hijacking, and session fixation.

In cookie poisoning attack, an attacker modifies the values of a cookie that is stored in victim's browser. The cookie is sent back to the web server by the attacker. The web server processes the values set by the attacker. The attacker may gain access to sensitive information of the victim or a web site running on the web server. In worst case, the attacker is able to use the cookie directly for authentication and becomes authorized on the web site.

In cookie manipulation attack, according to the literature, there are three different attack methods. The methods are cookie eviction, direct cookie injection, and cookie injection from related hostnames. In cookie eviction, an attacker fills the cookie store of a victim's browser with dummy cookies. This causes the browser to cleanse actual cookies and the actual cookies are replaced with the attacker's cookies.

If a web site uses secure cookies, in other words, the Secure flag is set on cookies, an attacker may perform direct cookie injection. The attacker does not

break the encryption. Instead, the attacker creates new cookies or overwrite the existing cookies. Direct cookie injection exploits insecure cookies and secure cookies locating in the same namespace. The attacker listens the victim's traffic and captures plaintext transaction from victim's target web site. HTTP response is replied with included arbitrary cookies by the attacker.

In cookie injection from related hostnames method, an attacker injects a cookie from a related hostname to a web site locating in the same hostname. A HTTP request is submitted by the victim to the vulnerable web site. In the vulnerable web site the arbitrary cookies are set.

It is inevitable to notice that literature offers no solutions for preventing cookie manipulation and poisoning attacks. Some web sites suggest how to prevent these types of attacks. Portswigger (2018) suggests that cookies should not be dynamically written using data that originates from untrusted sources. According to CWE (2018), cookie data should be avoided in security-related decisions. There should be input validation for cookie data if cookie data is used for security-related decisions. To detect tampering, integrity checks should be added.

Radware (2018) states that web application firewall (WAF) protects web sites against cookie poisoning. The WAF detects "set" commands sent by web servers in cookies. WAF intercepts all HTTP requests and compares the requests to the information in received cookie.

Cookie hijacking/stealing attack is executed by sniffing a victim's network traffic and capturing cookies downloaded to victim's web browser from a web site. The attacker may gain access to the victim's machine to view the stored cookies. This allows the attacker to create new session to the web site and to submit the cookie to bypass authentication.

To prevent cookie hijacking/stealing, cookies should be encrypted. Although, encryption does not provide full security for cookies. An attacker may listen to victim's traffic and expect the victim to send an unecrypted HTTP request to some other web site. The attacker hijacks the unsecured connection and responds to the unsecured request and redirects the victim's browser to the target web site on port 80. This attack method is related to TCP/IP hijacking. To prevent this type of attack, session identifiers should not be transmitted and managed by cookies.

4.2.2 HTTP Cookie Related Attacks and Defense Methods

Cookie related attack methods include XSS, CSRF, TCP/IP hijacking, and session fixation. In XSS an attacker may inject HTML tags or scripts into a web site. A user may download and execute the injected code. With persistent XSS cookies may be exploited for attack initiation. Cookies contain information that can attract the attacker. XSS is used to maliciously access the information. A page on the web site may be created or altered by the attacker with an embedded script. The script may be designed for extracting information in cookies and the attacker accesses the cookie.

To prevent XSS, data inputs that are not trusted should not be accepted. HttpOnly flag should be set on session's cookies. The absence of studies concerning the prevention of XSS is prominent.

In CSRF attack, an attacker tries to cause a victim to perform a malicious action. The attack can be performed with no use of scripts. CSRF attack aims to take over the identity of the victim. Consequently, the attacker performs actions on behalf of the victim. If a web site solely rely on cookies as session identifier transmitting method, the web site is vulnerable to CSRF attack.

To prevent CSRF attack, two separate checks should be performed to verify that the request is from the same origin. Source origin and target origin of the request should be resoluted. Target origin value should match the origin header header value. If there exists changes in the path of origin, the changes should not be accepted. SameSite attribute value should be set to "strict".

In TCP/IP hijacking attack, an attacker intercepts a legimative user's data. The user's TCP/IP connection is tried to be hijacked by the attacker. The attacker inserts herself into the user's session. The user's cookies are aimed to be hijacked. Whenever the cookie includes sensitive information, such as login credentials, the attacker may access the user's session by using the cookie. If session timeouts are incorrectly configured, TCP/IP hijacking may be performed.

To prevent TCP/IP hijacking attack, session identifiers should not be transmitted and managed by cookies. Session identifiers should be sent over encrypted protocol by attaching the session identifiers to SSL session. If cookies include sensitive information, Secure flag and HttpOnly flag should be set. A centralized management API should be used to prevent direct access to cookies. Cookies' Domain and Path attributes should be set correctly.

In session fixation attack, a session identifier from an attacker's user agent is transplanted by the attacker to a victim's user agent. The session identifier is used by the victim in interaction with a web server. The session identifier may be filled with the credentials or sensitive information of the victim. The attacker interacts with the web server directly by using the session identifier. The attacker may achieve the authority or confidential information of the victim.

To protect sessions, Secure flag should be set in cookies. This way the cookies are sent only over encrypted HTTPS connection. When HttpOnly flag is set, the browser should not allow scripting to access the cookie. This should prevent session identifiers to be stolen via XSS attacks. Cross-site requests are prevented by setting the SameSite attribute on cookies. Cookies are instructed to be sent to specified domain and subdomains by setting the Domain attribute. Cookies are sent to specified directories and subdirectories by setting the Path attribute. The scope of Domain and Path attributes should be restricted.

4.3 HTTP Cookie Confidentiality

Confidentiality includes measures that ensure confidentiality to prevent sensitive information to be reached by malicious operators and to ensure the sensitive information is reachable by the appropriate operators. Sensitive information should by accessed only by the appropriate authorized operators. A common method to ensure confidentiality is encryption.

The study indicated that cookies have weak confidentiality. The findings address that isolation deficiency is exposed in isolation by port, by scheme, and by path on the server-side. The lacking isolation by port exposes cookies to other services running on various ports of the same server. Services running on various ports of the same server, may read a cookie that is readable by a service that runs on one port of the server. The services that run on different ports of the server, may be able to write the cookie that is writable by the service that runs on one port of the server. Distrusted services should not be running mutually on various ports of a host. Neither should security-sensitive information be stored in cookies by hosts.

Another variable that impacts the confidentiality of cookies is the fact that cookies do not provide isolation by scheme. In general, cookies are used with HTTP and HTTPS schemes. The lack of isolation results in situations where cookies could be reached by other schemes like FTP. Due to the fact that cookies lack isolation by scheme, the lack of isolation is present in requirements of cookie processing.

The research indicated that cookies do not provide isolation by path. Though, cookies that are stored for a path and are not sent to another paths by network level protocols, cookies may be exposed by user agents via APIs not using HTTP. Some user agents do not isolate received resources from different paths. Therefore, cookies stored for one path may be accessed by resources from various paths.

4.4 HTTP Cookie Integrity

Integrity comprehends maintaining the consistency, accuracy, and trustworthines of data. Data should not be modified during transmission and should not be transmuted by operators that are unauthorized.

As a result of cookies being used with HTTP and HTTPS schemes, Cookies do not quarantee integrity for sibling domains and their subdomains on the server-side. A specific web server may set cookies with a specific Domain attribute value that does not correspond absolutely the value of the sibling domain (for an example, web server foo.website.com could set a Domain attribute value website.com). The Domain attribute value may overwrite an existing Domain attribute value that is set by subdomain of the web server. In this case, user agent includes cookies in HTTP requests to the subdomain. The subdomain

may not be able to separate the cookie from a cookie set by itself. This ability allows a subdomain to mount attacks against other domains on the host.

The Path attribute of cookie provides no integrity. User agents accept arbitrary Path attributes in a Set-Cookie header. Cookies can be injected into Cookie header by an attacker. The attacker may imitate a response from a web server. The HTTPS server in the web server is not able to separate the cookies that are injected by the attacker from the cookies in an HTTPS response. Therefore, HTTPS protocol do not provide security for cookies and can be exploited by the attacker even if the web server sees HTTPS only.

An attacker may use the Cookie header that is sent to https://website.com/ and inject cookies to the header and impersonate a response from http://website.com/ and inject the Set-Cookie header. The HTTPS server at website.com is not able to separate the cookies the attacker injected from the cookies set by the server itself in the HTTPS response. Even though the server uses HTTPS exclusively, the attacker may leverage the ability to mount attacks. Encryption and cookie contents signing may partially mitigate these types of attacks. However, the attacker may replay a cookie received from the authentic website.com server in a user's session. Therefore, cryptography does not prevent the attacks completely.

RFC document 6265 do not specify mechanisms to provide confidentiality and integrity quarantees. Therefore, browsers do not invariably authenticate the domain that sets the cookie.

5 DISCUSSION

The main research question of this master's thesis was: "What are the weaknesses of HTTP cookie?". Secondary research questions were: "What types of attacks exploit the weaknesses of HTTP cookie?" and "What defence methods can be applied to mitigate the attacks?".

Cookies have multiple weaknesses. Cookies may encourage developers to use cookies in a form of ambient authority for authentication. When cookies are used as "ambient authority" for authentication, the system exposes vulnerabilities and may be attacked with CSRF. If session identifiers are stored in cookies, the system may become vulnerable for session fixation. The examined literature mostly agree the fact that cookies should not be used for storing session identifiers.

Cookies are weak in user authentication. Attackers may gain authority to web sites via cookies. Session identifiers are not safe in cookies. Multiple documents indicate that if session identifiers are stored cookies, cookies become interesting for attackers. If session identifiers are exposed to attackers, the results may be severe. Multiple widescale attacks that have been targeted in cookies indicate that cookie exposure may cause severe damage to users and web servers.

Cookies should not be used for authorization. Instead, URLs could be treated as capabilities to entangle designation and authorization. Consequently, secrets would be stored in URLs and not in cookies. This strenghtens the security of an application. Remote entities would be required to supply the secret.

If developers do not set the Secure flag in cookies and transmit the cookies over a secure comminucations channel, such as HTTPS, the Secure attribute misses its purpose. It is inevitable to set the Secure flag when transmitting cookies over encrypted protocol. A missing Secure attribute allows cookies to expose the ongoing requests to attacks. However, the Secure attribute do not protect the integrity of cookies. Secure attribute protects only the confidentiality of cookies. Similarly, HttpOnly attribute protects only the confidentiality of cookies and not the integrity of cookies. The fact that the integrity of cookies is not protected is remarkable.

Cookies are not isolated by port, by scheme, nor by path on the server-side. Integrity for sibling domains and their subdomain is not quaranteed. The Path attribute do not provide integrity. Cookies have overall weak security model and weak confidentiality. This is mostly possible because of the fact that cookies are mostly based on a draft from 1994. The model of cookies should be reconsidered and think about the mechanisms of cookies to examine all the components that expose cookies to attacks.

There exists multiple attack types that can exploit the weaknesses of cookies. Cookie poisoning allows an attacker to gain access to sensitive information of web site or user. Cookie manipulation allows an attacker to create new cookies or overwrite the existing cookies, or place arbitrary cookies. XSS allows an attacker to inject a web page with an embedded script to extract cookie's information to access the cookie. CSRF allows an attacker to gain identity of a victim to perform malicious actions. TCP/IP hijacking attack allows an attacker to access victim's session.

The attack methods that cookies face are widely discussed in the literature. Literature states that these types of attacks are powerful. If cookies remain in the same form as they are, certain prevention of the attacks is not prospective. To mitigate the attacks specific defense methods should be applied. The Secure attribute should be set whenever transmitting the cookies over encrypted protocol. HttpOnly attribute should be set to mitigate arbitrary scripts accessing the cookies. Setting the HttpOnly attribute may prevent session identifiers to be stolen by XSS attack. The SameSite attribute should be applied to prevent cookies to be sent with cross-site requests. The Domain attribute should be specified to instruct the cookie to be sent only to specific domain and subdomains. The Path attribute should be applied to instruct the cookie to be sent only to specific directory or subdirectories.

The findings of this research are devastating when considering that cookies are a functionality that is commonly used to track users, to store session identifiers, and to store login credentials. Overall trust on cookies is threatened. More research should be done on the functionality and vulnerabilities of cookies to cover the weaknesses on a wider scale to understand what activities should take place to protect cookies.

The findings of this research have scientific significance. The research indicates that cookies have not been examined sufficiently to generate comprehensive answers for the problem. Science should examine the weaknesses of cookies and endeavour to assist sofware development to develop improving solutions for the problem of cookies. The findings of this research could be used for trapping the weaknesses of cookies.

It is remarkable that literature does not address defensive solution models for several attacks that cookies face. It is inevitable to examine the protection methods of cookies because cookies have an effect on most of the individuals in the world. The findings of this research do not address new defense methods for cookie protection. The introduced defense methods are based on the litera-

ture and standards. More extensive research should be conducted to examine the vulnerabilities and weaknesses of cookies more profoundly.

The findings are mainly based on reliable resources. The research focused largely on the RFC 6265 document. This document in itself imports credibility to the findings of this research and the research itself. Several studies apply the document as a source of information that addresses cookies.

There are some research topics exposed in this research that should be examined deeper. The fact that cookies are not isolated by port, scheme, or path, should be investigated to find solutions for the problem. Another research problem that can be emphasized from the science of cookies is that how the attributes of cookies could be strenghtened and modified to improve cookies security.

6 CONCLUSION

Cookies are a commonly used technique in the services on the Internet. The purpose of this study was to research cookies to find out what weaknesses exist in cookies. Weaknesses lead into activity that attempts to exploit the weaknesses. To protect the functionality of cookies, defense methods should be conducted. This research examined what are the weaknesses cookies have, what are the attack methods that can exploit the weaknesses, and what are the defense methods to mitigate the attacks. The objective of the research was to examine the function and features of cookies to gain understanding what are the weaknesses of cookies.

The research was conducted by the means of systematic literature review to be able to understand the phenomenon through previous studies. The related studies were examined carefully and their quality was considered. The research followed the process of the systematic literature review.

The findings of this research indicated that cookies and the environment where cookies function contain severe weaknesses and vulnerabilities that can be exploited by attack methods. The findings are important to take into count in future research on the topic. The research indicated distinct weaknesses that exist in cookies.

The findings showed that the weaknesses that cookies include is the lacking isolation by port, by scheme, and by path. Also the fact that cookies do not quarantee integrity for sibling domains and their subdomains is weakening cookies. The attributes of cookies neither provide integrity. Secure, HttpOnly and Path attributes protect only the confidentiality of cookies. The lack of integrity expose cookies to attacks. Cookies have also weak overall security model.

The weaknesses of cookies can be exploited in multiple attack methods. The weaknesses can be exploited by cookie poisoning, cookie hijacking/stealing, cookie manipulation, XSS, CSRF, and TCP/IP hijacking. These attack methods follow a specific design that leads into exploiting the weaknesses shown in the findings. To verify the validity of the findings, the attack methods should be tested by adding the defense methods shown in the findings.

The findings indicated that several defense methods should be applied to cookies to mitigate the attacks. Secure attribute should be set when sending the cookie over secure channel. HttpOnly attribute should be set for mitigating arbitrary scripts. SameSite attribute should be set to prevent cross-site requests with cookies. Domain and Path attributes should be set to ensure that cookies are sent to determinated domain and subdomains and determinated directory or subdirectories.

The findings of this research should lead into further research on the topic. The findings should be tested and if the weaknesses remain the same, detailed investingating should be conducted to find solutions to patch the weaknesses. There are several studies that propose concern on cookie weaknesses. The concern should be taken into count.

Cookie isolation should be examined in the future research. Future research should propose how to isolate cookies by port, by scheme, and by path. Another option is to examine the functionality of cookies attributes. Cookies attributes should be able to protect the integrity of cookies. Future research should examine how the cookie attributes should be modified to be able to protect the integrity.

REFERENCES

Alcorn, W., Frichot, C. & Orrù, M. (2014). *The Browser Hacker's Handbook.* Indianapolis : John Wiley & Sons, Inc.

Ansari, J. (2015). *Web Penetration Testing with Kali Linux.* Birmingham : Packt Publishing Ltd.

Anto, Y. (2012). *The Art of Hacking.* Saarbrücken : LAP LAMBERT Academic Publishing GmbH & Co.

Bangia, B. (2005). *Internet and Web Design.* New Delhi : Firewall Media.

Barrett, D., Weiss, M. & Hausman, K. (2015). *CompTIA Security+ SYO 401 Exam Cram.* Indianapolis : Pearson Education, Inc.

Boland, A., Cherry, G. & Dickson, R. (2017). *Doing a Systematic Review : A Student's Guide.* London : SAGE Publications Ltd.

Chen, F., Duan, H., Zheng, X., Jiang, J. & Chen, J. (2018). Path Leaks of HTTPS Side-Channel by Cookie Injection. *Constructive Side-Channel Analysis and Secure Design.*

Ciampa, M. (2012). *Security+ Guide to Network Security Fundamentals.* (Fourth edition).

Clifton, B. (2012). *Advanced Web Metrics With Google Analytics.* (Third edition). Indianapolis : John Wiley & Sons, Inc.

Dubrawsky, I. (2009). *CompTIA Security+ Certification Study Guide.* Burlington : Syngress Publishing, Inc.

Dubrawsky, I. (2010). *Eleventh Hour Security+.* Burlington : Elsevier Inc.

Dubrawsky, I. (2007). *How to Cheat at Securing Your Network.* Burlington : Elsevier Inc.

Dulaney, E. (2009). *Comptia Security+ Study Guide.* (7th edition). Indianapolis : John Wiley & Sons, Inc.

EC-Council. (2010). *Ethical Hacking & Countermeasures : Threats and Defence Mechanisms.* (2nd edition).

EC-Council. (2017). *Ethical Hacking & Countermeasures : Web Applications and Data Servers.*

Engebretson, P. (2011). *The Basics of Hacking and Penetration Testing : Ethical Hacking and Penetration Testing Made Easy.* (2nd edition). Waltham : Elsevier, Inc.

European Union. (2009). Directive 2009/136/EC of the European Parliament and of the Council of 25 November 2009. *Official Journal of the European Union.*

Faircloth, J., Beale, J., Temmingh, R., Meer, H., Walt, C. & Moore, H. (2005). *Penetration Tester's Open Source Toolkit.*

Flanagan, D. (2006). *JavaScript : The Definitive Guide.* (5th edition). Sebastopol : O'Reilly Media, Inc.

Fraunholz, D., Krohmer, D., Anton, S. & Schotten, H. (2017). Investigation of Cyber Crime Conducted be Abusing Weak and Default Passwords with a Medium Interaction Honeypot. *International Conference on Cyber Security and Protection of Digital Services (Cyber Security).*

Gourley, D. & Totty, B. (2002). *HTTP : The Definitive Guide.* Sebastopol : O'Reilly Media, Inc.

Green, J. (2015). *Cyber Warfare : A Multidisciplinary Analysis.* Abingdon : Routledge.

Grossman, J., Hansen, R., Petkov, P., Rager, A. & Fogie, S. (2007). *XSS Attacks : Cross Site Scripting Exploits and Defense.* Burlington : Syngress Publishing, Inc.

Halton, W., Weaver, B., Ansari, J., Kotipalli, S. & Imran, M. (2016). *Penetration Testing : A Survival Guide.* Birmingham : Packt Publishing Ltd.

Hassan, N. & Hijazi, R. (2017). *Digital Privacy and Security Using Windows : A Practical Guide.*

Jacobs, S. (2016). *Engineering Information Security : The Application of Systems Engineering Concepts to Achieve Information Assurance.* (2nd edition). New Jersey : John Wiley & Sons, Inc.

Lobo, L. & Lakshman, U. (2014). *CCIE Security v4.0 Quick Reference.* (3th edition). Indianapolis : Cisco Press.

O'Brien, A. & Guckin, C. (2016). The Systematic Literature Review Method.

Oriyano, S. & Shimonski, R. (2012). *Client-Side Attacks and Defence.* Waltham : Elsevier Inc.

Ristic, I. (2014). *Bulletproof SSL and TLS : Understanding and Deploying SSL/TLS and PKI to Secure Servers and Web Applications.* London : Feisty Duck Limited.

Sarris, S. (2014). *HTML5 Unleashed.* United States of America : Pearson Education, Inc.

Steward, J. (2014). *CompTIA Security+ Review Guide.* (4th edition). Indianapolis : John Wiley & Sons, Inc.

Stuttard, D. & Pinto, M. (2011). *The Web Application Hacker's Handbook : Finding and Exploiting Security Flaws.* Indianapolis : John Wiley & Sons, Inc.

Timm, C. & Perez, R. (2010). *Seven Deadliest Social Network Attacks.* Burlington : Elsevier Inc.

Wu, H. & Zhao, L. (2015). *Web Security : A Whitehat Perspective.* CRC Press.

Zhang, Y., Wang, Z. & Xia, C. (2010). Identifying Key Users for Targeted Marketing by Mining Online Social Network. *IEEE 24th International Conference on Advanced Information Networking and Applications Workshops,* 644-649.

Zhu, Y. (2016). A Book Recommendation Algorithm Based on Collaborative Filtering. *5th International Conference on Computer Science and Network Technology,* 286-289.

ONLINE REFERENCES

Barth, A. (2011, April). HTTP State Management Mechanism. Retrieved from https://tools.ietf.org/html/rfc6265.

CNet. (2017, February 15). Yahoo tells users they were hit with cookie attack. Retrieved from https://www.cnet.com/news/yahoo-tells-more-users-they-were-hit-with-cookie-attack/.

CWE. (2018, March 29). CWE-565 : Reliance on Cookies without Validation and Integrity Checking. Retrieved from https://cwe.mitre.org/data/definitions/565.html

CWE. (2018, March 29). CWE-79 : Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting'). Retrieved from https://cwe.mitre.org/data/definitions/79.html.

F-Secure. Tracking Cookie. Retrieved From https://www.f-secure.com/sw-desc/tracking_cookie.shtml

Google. (2018). Google Data Protection. Retrieved from https://privacy.google.com/your-data.html

Internet World Stats. (2017, December 31). Internet usage statistics. Retrieved from https://www.internetworldstats.com/stats.htm

IT Governance. (2017). The EU General Data Protection Regulation (GDPR). Retrieved from https://www.itgovernance.eu/blog/en/how-the-gdpr-affects-cookie-policies

Lifewire. (2018, January 5). Understanding Transmission Control Protocol/Internet Protocol (TCP/IP). Retrieved from https://www.lifewire.com/transmission-control-protocol-and-internet-protocol-816255

Mozilla. (2017, April 4). Cookie header syntax. Retrieved from https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cookie

Mozilla. (2017, October 9). Set-Cookie header syntax. Retrieved from https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie

Mozilla. (2018, May 21). HTTP cookies. Retrieved from https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies

OWASP. (2013). Owasp Periodic Table of Vulnerabilities – Cookie Theft/Session Hijacking. Retrieved from https://www.owasp.org/index.php/OWASP_Periodic_Table_of_Vulnerabilities_-_Cookie_Theft/Session_Hijacking

OWASP. (2017, August 24). HttpOnly. Retrieved from https://www.owasp.org/index.php/HttpOnly

OWASP. (2017, September 11). Session Management Cheat Sheet. Retrieved from https://www.owasp.org/index.php/Session_Management_Cheat_Sheet

OWASP. (2018, February 3). Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet. Retrieved from https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet

OWASP. (2018, May 9). XSS (Cross Site Scripting) Prevention Sheet. Retrieved from https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

Portswigger. (2018). Cookie Manipulation (stored DOM-based). Retrieved from https://portswigger.net/kb/issues/00500b02_cookie-manipulation-stored-dom-based

Radware. (2018). DdoS Attack Definitions – DdoSPedia. https://security.radware.com/ddos-knowledge-center/ddospedia/cookie-poisoning/

Tom's Guide. (2013, September 16). Tracking Cookies : What They Are, and How They Threaten Your Privacy. Retrieved from https://www.tomsguide.com/us/-tracking-cookie-definition,news-17506.html

Tutorials Point. Learn HTTP. Retrieved from https://www.tutorialspoint.com/http/index.htm

West, M. & Goodwin, M. (2016, April 6). Same-site Cookies. Retrieved from https://tools.ietf.org/html/draft-west-first-party-cookies-07#section-2.1