2020 International Conference on Information Technology Systems and Innovation (ICITSI)
Bandung - Padang, October 19 - 23, 2020
ISBN: 978-1-7281-8196-7

HTTPS Contribution in Web Application Security: A Systematic Literature Review

Fajar Wijitrisnanto
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
23220001@std.stei.itb.ac.id

Suhardi
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
suhardi@stei.itb.ac.id

Purnomo Yustianto
Dinas Komunikasi dan Informatika
Jawa Barat
Bandung, Indonesia
yustianto@jabarprov.go.id

***Abstract*—A Web application is one of the most used technology nowadays due to its flexibility in delivering services to society. It also plays a good portion in enhancing our daily life since it could provide almost any kind of services through an application served from the internet. Thus, many users' private information runs the risk of being exposed to an unauthorized party. Standard browser connection uses HTTPS protocol, while both TLS over HTTP and Web application are known for several of vulnerabilities. This paper presents the results of an SLR study on web application security of HTTPS implementation. The study selects 45 qualified papers related to the topic and analyzed 24 of the documents. The findings are categorized into three labels: threats, threats impact, and defense mechanisms. This work also classifies the attack and threats based on the impact produced. In this study, the lack of understanding about security-related mechanism in TLS, session management, and web application still become the culprit of most attack and vulnerability. Based on this work, a researcher could better prioritize and prepare security mechanism to overcome the threats.**

Keywords—web application, https, tls, cookie, hsts, hpkp, csp, security, systematic literature review

I. INTRODUCTION

Web Application becomes a critical subject nowadays due to its massive usage on society, government, and business. It also plays a good portion in enhancing our daily life since it could deliver almost any kind of services through an application served from the internet. Thus, many user's private information circulating around the internet becomes an easy target for prying eyes. To mitigate this threat, researchers had built a lot of security mechanisms and protocols to secure the information mentioned. One of them is a protocol named SSL/TLS designed to encrypt information in transit between peers. In the context of Web Application, TLS task is to encrypt communication between client and server in combination with HTTP protocol, which produces another protocol named HTTPS. HTTPS protocol since then becoming *de facto* standard in the scope of Web Application security.

On the contrary, TLS is not as secure as it said. TLS offers much flexibility in practice so that security depends on how developers implement TLS [1]. On the other hand, Web Application itself suffers a lot from another massive attack vector, mostly caused by malicious user input [2]. This lousy condition keeps improving over time, thanks to the majority of hacker's contribution in inventing new attack techniques without publicly reporting their research results. This lack of documentation is also supported by a lack of continuous study describing the overall security condition in Web Application in relation to HTTPS itself. This collection of security conditions is crucial because Web Application and HTTPS implementation weaknesses mentioned before could go untraced and make further research challenging to conduct. Because of this concern, this study will construct a Systematic Literature Review (SLR) on the current security state of HTTPS implementation in supporting Web Application Security Mechanism.

SLR is a formal, consecutive method to identify, evaluate, and interpret the research sources regarding a topic or question of interests [3]. With SLR as a basic approach, the wide-ranged interconnection of HTTPS and Web Application security topics could be systematically formulated. In general, this study will focus on researching current good quality publications related to the topic mentioned earlier. This process will follow an SLR based approach that maintains the quality of publication by using techniques such as search strategy and quality criteria.

This work will be done by reviewing publications from three points of view according to HTTPS contribution in the security of Web Application: (1) the type of available attacks or threats against the security mechanism, (2) the impact range of those attacks, and (3) available defense mechanism to overcome the threats. This effort will consider reviewing research publications from several popular literature databases such as ScienceDirect, IEEE Xplore, ACM Digital Library, SpringerLink, and Scopus that published since 2014 until 2020. In the end, the ultimate goal is to produce a broad picture of recent conditions in Web Application security mechanism, together with its defense strategy. This is done so that future researchers could understand current state and become easier to fill the gaps between attack vectors, defense mechanisms, and effective solutions.

The remainder of this paper is organized as follows. Section II describes the SLR methodology used in this work. Section III explores the sources of data used for analysis and its classification process. The result of this study is described in Section IV. Finally, Section V concludes the paper with its limitation and the addition of possible research suggestions.

II. METHODOLOGY

This paper will conduct an SLR approach based on the methodology arranged by Kitchenham et al.[4]. This approach has been chosen in the study because of its initial purpose in guiding SLR research in the field of Software Engineering. It is different from the usually used PRISM methodology, whose purpose is to do systematic review and meta-analysis in medical field [5].

A. Preparation Stage

The preparation stage describes the plan and protocol about how the SLR will be conducted. It consists of these six minimum requirements:

a) Research Motivation (A.a)

- To expose recent Web Application security state related to the usage of HTTPS protocol.
- To open new research topics that could fill the gaps of security holes found in this study.

b) Research Questions (A.b)

- RQ1.What type of security threats exist?
- RQ2.What impact of threats exist?
- RQ3.What defense mechanisms purposed against the threats?

c) Search Strategy (A.c)

Search strategy is done using search string contained these keywords: (1)Https, (2)TLS, (3)Security, (4)Web, and (5)Cookie/Header with Boolean operator AND between each word.

d) Primary Studies Identification (A.d)

This phase looks for apropriate publications based on the following inclusion criteria:

- Publications topic relate to the Research Questions.
- Publications have clear methodology.
- Publications published between 2014 – 2020.
- Use English language.

In focussing on the identification process, this study also eliminates out-of-scope publications using exclusion criteria as follows:

- Research from white paper, technical report, magazine, online article, review article, book review.
- Duplicate content.
- Related publications that took security topics in Wi-Fi protocol, cloud, DNS, IoT, blockchain, and tor.

e) Quality Assessment Criteria (A.e)

- Research was published in journals that fulfill quartile criteria of Q1, Q2, or Q3.
- Research was published in conferences or proceedings that have SJR rate above 0.3.
- Publications that massively discuss the security of HTTPS and Web Application.

f) Data Extraction (A.f).

Classify data based on all RQs asked, and extract the information from each related publication main Section. This phase eliminates any publication that matches the following exclusion criteria:

- Discuss about unproven or theoretical threats.
- Describes defense mechanisms that is outdated by newer version.

B. Execution Stage

The execution stage is the practical strategy for executing the plan derived from the preparation stage before. It is designed with these repeatable steps:

a) Using (A.c) to identify relevant publication (B.a);

b) Using (A.d) to identify primary studies (B.b);

c) Using (A.e) to evaluate publication's quality (B.c);

d) Using (A.f) technique to extract research data (B.d);

e) Data Analysis based on (B.d), and presenting data (B.e).

C. Documentation Stage

Data analysis and review results, together with the overall process in doing Preparation and Execution stage then documented in a systematic way using these two processes:

a) Dissemnation Strategy (C.a);

b) Format SLR Report (C.b).

III. DATA SOURCES

This section describes the result of Execution Stage explained before, from *(B.a)* until *(B.d)*. Data resources are being gathered from five diverse publication databases online, including ScienceDirect, IEEE Xplore, ACM Digital Library, SpringerLink, and Scopus. These five had been chosen for their well-known publicity number and good reputation. Besides, these databases have their own citation exporting feature that benefits this study. The publications itself must be published in 2014 until 2020 to fulfill the concept of novelty.

A. Identify Relevent Publication (B.a)

After conducting Search Strategy *(A.c)* as explained in Section II, the output gives a total of 411 publications found. It is the sum of 40 publications taken from ScienceDirect, a single research paper from IEEE Xplore, 103 publications from Springer, 111 results confirmed by Scopus, and 156 research articles found by ACM Digital Library. Table I show these findings and their details.

From this point, it is clear that there are diverse results obtained from different databases, even using the same keywords, Boolean operator, publication year restriction, and publication category. ACM Digital Library dominates the finding, and on the contrary, IEEE Xplore hardly finds any publication. It is due to each database characteristic in the way they process those keywords that made the gap exists, even with nearly the same database capacity.

TABLE I. SEARCH STRATEGY RESULTS

Search String	Publication Databases					Total
	Science Direct	*IEEE Xplore*	*Springer*	*Scopus*	*ACM*	
Https AND TLS AND Security AND Web AND (Cookie/ Header)	40	1	103	111	156	411

B. Identify Primary Studies (B.b)

Based on Section II's inclusion and exclusion criteria, this stage will take results from the search strategy and process it for further classification. The first step is to find any duplications inside the cluster. The study takes the Mendeley Citation Manager as a tool to find duplicates. This tool was chosen following a recommendation from Kwon et.al. that found Mendeley has minimal results of false positive and false negative [6]. We acknowledge 140 duplicates in this phase, and reducing our records results into 271 publications.

The next step is to review them by its title and keywords to find matching related to RQ and mismatch towards exclusion criteria. This approach successfully eliminates seven research articles and updates the sum into 264 publications.

C. Publication Quality Evaluation (B.c)

This work evaluates each journal, conference, or proceedings article based on Quality Assessment Criteria *(A,e)*. Around 198 publications beyond the criteria's boundary were stripped out, resulting in around 66 available publications. This stage describes the unique condition of research in this topic that the attack and defense mechanisms usually originated from informal and non-high-rated journals and conferences. They are usually developed by forums, white papers, technical reports, challenge writeups, RFCs, CVE, exploits, certifications, and many other free practical sources. This is a challenge that needs to be faced to successfully determine the security conditions of today's digital world and future development.

D. Data Research Extraction (B.d)

After reviewing each publication's abstracts, a sum of 21 publications was sorted out from our lists following our exclusion criteria explained before in Section II.A.f. A total of 45 publications were then classified into different topics based on RQ defined early in this paper. In general, there are 24 main papers that fully analyzed, and the rest of it will be analyzed by its main section discussion.

Topics discussed in the review-ready publications are varied into three main objects: HTTPS security, Web Application security, and the security of HTTPS implementation in the application security mechanism. Each of the three objects discusses different specific subtopics from one another. Some papers discuss more than one subtopic. The details of the main topics distribution from each paper described in Table II.

TABLE II. TOPIC DISTRIBUTION

Topic Discussed Related to Security								
Main	**HTTPS**				**Web Application**		**HTTPS in App**	
Sub-topic	TLS	TLS 1.3	MITM	Cookie/ Header	Auth Mech	App Vuln	Android	Crypto
Sum	12	7	3	10	5	3	2	3

IV. RESULT AND DISCUSSION

Data is extracted and analyzed according to the purpose of this study in answering each RQs. In brief, all paper discusses various security mechanisms on two main targets: HTTPS and Web Application. TLS over HTTP security is being questioned in the majority of the publications [7], [8], [9], [10]. Meanwhile, there is also a rising suspicion against TLS 1.3 security [11], [12], [13]. Some researcher also plays a good portion analyzing the cryptographic element of TLS [14], [15], [16], [17]. Man in the Middle (MITM) attacks are also discussed as the mindstream method bombarding HTTPS [18], [19]. Web application research inquires more about classical web vulnerability [20], [21] and it's authentication mechanism which is related to HTTPS [19] or unrelated to it [22]. In relation to both areas of HTTPS and Web App, the session security mechanism using cookies and security headers also exposed [23], [24], [25], [26].

These results are presented as in the following structure.

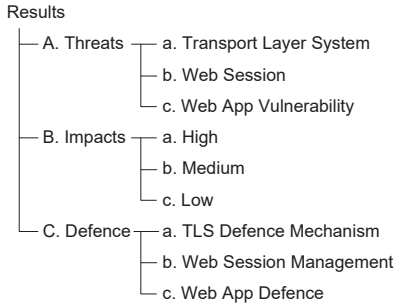

Fig1. Results Presentation Structure

A. Threats on HTTP implementation in Web Applications

There are three different threats categories found in the SLR results conducted in this study. They are potential attack methods on TLS, web session management, and web application vulnerability.

a) TLS

In a TLS-based attack, we categorize based on its attack methods into five areas, including MITM, BCP, cryptography, security headers, TLS 1.3 protocol analysis, and attack on android. All of this attack has a different attack vector and exploit unique weaknesses. Some of this even already have its

countermeasures and proven ineffective due to many implementation reasons.

MITM attack has been a legacy attack method for TLS. The main cause of this state is due to a weak server and client authentication during TLS Handshake [7]. An attacker could still claim a valid X.509 certificate using its own malicious domain so that it will be trusted by the victim. When user already trusted this certificate by verifying with the attacker's server public key, then the next step is trivial for attacker to connect towards real server as the user. In 2014 there are two attacks appeared named MITM Script-In-The-Browser attack (MITM-SITB) and MITM CCS (Change Cipher Spec) attack. MITM-SITB uses a technique to bypass Channel ID-based defenses [18]. Meanwhile, CCS attack conduct MITM by exploiting a vulnerability in OpenSSL to run Change Cipher Spec protocol early so the server and client would use zero-length master key [10]. CCS attack is then empowered by FREAK attack proposed in 2015 that allowed the attacker to downgrade the client's RSA export.

Private key leakage attack on RSA proposed in 2016 also make MITM possible. Private RSA keys made with a lack of entropy can lead to servers sharing primes in keys, so an attacker could factorize it using great common divisor (GCD) computation [10]. In the same year, there are also certificate validation bugs resulting in invalid certificates to impacts software and hardware that intercepts TLS connection. This kind of invalid certificate attack still occurs now. Another MITM attack proposed in 2016 called transcript collision attacks also try to find chosen prefix collision in hash functions [10]. DROWN attack that also came in 2016 become a threat since it makes servers vulnerable to decryption oracle because of the same RSA key usage. This identical key usage is due to the same certificate usage between peers [10]. In 2017, an attack named Diffie-Hellman key establishment attack or Subgroup attack also promoted an MITM scenario. It uses The Logjam attack (2015) to forces the server to choose a small 512-bit DH group. Lastly, an attack named ROBOT proposed in 2018 try to enhance former DROWN attack using parallel computing.

Besides MITM, another similar attack tries to mimic this approach using caching weakness. These named Browser Cache Poisoning (BCP) attack using one time MITM on user's HTTP session then substitute cached resources with another malicious one [18]. Around 90% of all browsers in 2015 fallen to this method.

In the realm of cryptography, the usage of AES-GCM with CBC-mode operation becomes an issue. It was detected in 2018 that 10% of TLS connection still uses this type of encryption and found vulnerable towards Lucky13 attack using caching timing technique [8]. Another method is targeting the implementation of a 64-bit encryption algorithm in TLS like 3DES and Blowfish. It is found that their implementation is insecure against short block collision attacks, so that an attacker could get the user's cookie or even HTTP BasicAuth for OpenVPN connection [15]. This threat is serious in practical because many connections still use this algorithm. It is known in 2018 that 31M unique IP still encourages the usage of 3DES and DES40 in TLS connection worldwide [16].

While the TLS DHE version in the TLS handshake phase is still common due to its proven security architecture [17], a misconfiguration issue found is also a risk. Many TLS implementation takes security shortcuts to reduce the cost of cryptographic computation and networking. A nine-week study found widespread usage of DHE and ECDHE private value reuse, TLS session resumption, and TLS session tickets. Also, the practice of TLS secrets and session state being shared across domains are also found [14].

Some mechanism has been developed to ensure the usage of HTTPS client-server connection. From the server point of view, the usage of security headers like HTTP Strict Transport Security (HSTS), Content Security Policy (CSP), and HTTP Public Key Pinning (HPKP) helps to ensure the usage of secured connection. With HSTS, no connection except HTTPS one could be established. CSP function is similar but with a different focus on the individual content to be secured. HPKP tends to pin the public key of a certificate so that it will be easy to verify certificate status.

The integrity of HTTPS connection is still in danger even with this concept. HSTS uses a preload list on the browser to identify which domain to connect with TLS connection. This practice is proven to be insecure [27] because of the preload list limitation in the browser storage capacity and in the initial connection could fail to enforce HTTPS connection based on a study on HSTS security taxonomy [28]. HPKP in the other hand suffers from the hardship of implementation, which makes it hard to support trusted HTTPS connection. These implementation problems also occur in HSTS [29]. Besides, HSTS also suffers from bootstrapping attack that exploit initial connection conditions that usually ignore TLS usage [30] and suffer from HSTS incapacitation

attacks [26]. Also, after a client fails to make a secure TLS connection, HPKP and HSTS are ignored by browsers eventually [24]. HPKP, as a feature with higher complexity, once again are proofed deployed scarcely and often incorrectly [9].

CSP in the other hand is rarely used. Its primary purpose to secure connection via upgrade-insecure-request headers is well known but not implemented correctly. Even with its support for fighting XSS attacks, CSP usage is still considered low. An analysis was also conducted to infer the settings that websites used in implementing CSP. This analysis shows that many sites with CSP do not have the right policies to prevent XSS [25]. There is also a known attack proposed in 2020 called Region Confusion Attack that exploits HTTPS security inconsistencies related to URLs and IPs diversity among regions. In this attack, two or more identical HTTPS clients, located at different geographic locations (regions), make an HTTPS request to the same domain simultaneously and proofed receive different HTTPS security guarantees in response. Even with HSTS and CSP header's usage, this inconsistency still appears in some cases [31].

On the protocol development process of TLS 1.3, some study has been conducted. One of that is the finding of a possible backward-compatibility attack against TLS 1.3 draft-18, like what has been done before in TLS 1.2 [11]. Another approach successfully proves that TLS 1.3 potentially weak against a scenario in which an adversary can successfully impersonate a client during a PSK-resumption handshake [32]. In terms of standardization for TLS 1.3, a framework analysis found that TLS standards lack details about what standardizing documents must provide. They must at least provide a partial specification of a protocol that admits a compliant collection, in fully realized implementations, in which the TLS 1.3 standard doesn't fulfill enough [33]. A symbolic analysis on TLS 1.3 draft-21 also reveals an unexpected behavior on some handshake models that will restrain strong authentication guarantees in some implementations of the protocol [13].

The vast usage number of HTTPS connection also rely on the massive adaptation of android application. Meanwhile, the usage of HTTPS connection in android is not as fair as in the browser. Many Android developers found not using HTTPS or violate rules which protect user data from man-in-the-middle attacks. The reasons vary from improper developer practices, server misconfiguration, lack of documentation, flaws in libraries, the fundamentally complex TLS-PKI system, and unbelievably lack of consumer understanding of the importance of HTTPS [34]. In contrast, the protection improvement rate against general SSL/TLS attack on the Android environment has been in good form. A study conducted in 2018 concluded that only 4% of 200 top android apps vulnerable to generic attacks like SSL stripping and SSL sniffing attack [35].

b) Web Session Management

Securing user web application session is one of the most targeted attack vectors due to its significant benefit in hijacking user sessions. Once done, the attacker could do whatever a real user able to do. The defense against this form of attack relies on cookies and token usage as a session information handler. This client-based heuristic is considered inadequate and also proven to be fragile. An assessment based on a ground truth of 2,464 cookies collected from 215 popular websites of the Alexa ranking proved that several pitfalls in the heuristics adopted are found [36]. Another analysis also found that using multiple cookie values for authentication efforts could easily confused the development process resulting in session hijacking vulnerability[37]. Even when some solution has been proposed on this session management security, the lack of protection, usability, compatibility, and ease of deployment make this solution ineffective [38].

The pitfalls may happen because of the lack of implementation of cookie security mechanisms in real-world scenarios. The standard itself provides security support via the usage of HttpOnly and Secure flag for cookies. The former flag is to make sure that no other option except HTTP connection could access the cookie. The later flag also ensures that cookies are only able to be transferred via secure TLS connection. Another mechanism is provided by scoping cookie usage via 'domain' and 'path' attribute. The well-known is 'samesite' attribute that only allows cookies to be accessed from trusted sources. Even these mechanisms proved to be not enough to protect cookies. A recent study conceals that the weaknesses in AES-GCM usage over the TLS handshake phase against forbidden attack impact an adversary's ability to replace security flags so that the mechanisms provided by the flags are useless [39]. This attack is adopted from a previous likely-identical method named cookie-cutter attack targeting HTTP header truncation weakness even in HTTPS connection [30].

Besides the mentioned findings, the cookie doesn't just affects the loss of session or authentication aspects. The confidentiality aspect also becomes a victim via the compromise of Personally Identifiable Information (PII) through cookie usage. By doing

cookie hijacking, an attacker could find various PII such as emails, username, contact lists, search history, shopping history, and address through various well-known platforms [27]. Most of these successful hijacking is caused by a lack of understanding from ordinary users upon rogue Wi-Fi impacts found in many public areas. An attacker could play a MITM rule across these networks.

c) Web Application Vulnerability

It is publicly known that web application security is commonly neglected due to its flexibility, vast scope, and development technology change. The most popular web vulnerability known is the injection-based attack that exploits unfiltered user input weaknesses on application code. With that in mind, many mechanisms have been proposed to withstand these attacks with lack of analysis so that injections attack still appear even today. Analysis results against a set of 41 previously proposed defenses based on their accuracy, performance, deployment, security, and availability characteristics concluded that the majority of these defenses suffer. This happened due to the poor manner of testing. Some approaches can also be bypassed by attackers with knowledge of how the mechanisms work [20].

Another attack vector appears from Uniform Resource Locator (URL) that reveals privacy. URL reveals a significant amount of metadata about the user's action widely. Even the usage of protection efforts like secure transport protocol and caching strategy, this endeavor still become a threat concerning privacy expectation [40].

Overall, the attacks mentioned above could be classified based on their proposed year. This classification is not strictly related to the number of studies instead, it will be related to the unique type of attack itself. Fig2 describes this relationship between the initiated year and the number of attacks that exist.

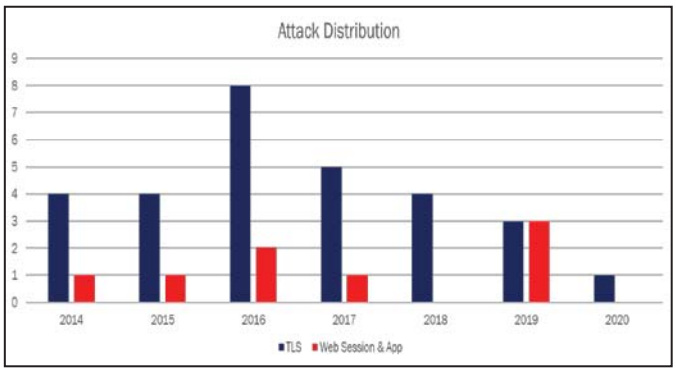

Fig2. Attack Distribution per Year

In 2014 there are 5 total proposed attacks, including MITM-SITB, MITM CCS attack, HSTS bootstrapping, HSTS incapacitation attack, and Cookie cutter attack. The same amount occurs in 2015 contains the Logjam attack, FREAK attack, BCP attack, cookie heuristic pitfalls, HPKP attack. The number increase in the following year reaching 10 attacks such as: DROWN attack, invalid certificate, transcript collision attack, DHE & ECDHE private value reuse, DES – Blowfish usage, HSTS preload list attack, CSP header misconfiguration, TLS 1.3 impersonation, multiple cookie value usage, and HTTP cookie hijacking targeting PII. In 2017, the number of attacks decreased to 6, including DH Key establishment attack, invalid certificate, HPKP bad implementation, strong unclear authentication in TLS 1.3, MITM in android, and cookie secure attribute misconfiguration. There is no web type attack in 2018 and leave it just for 4 TLS type, including ROBOT attack, invalid certificate, HSTS and CSP misconfiguration resulting in DoS and domain-based attack, and minimum standard specification in TLS 1.3. Some collection of attacks like invalid certificate, backward compatibility of TLS 1.3, Cookie theft by removing cookie flag for AES-GCM based TLS, Web app flawed defense mechanism, DES-based cipher usage worldwide, and URL privacy violation occured in 2019. Lastly, there is Region Confusion Attack for HSTS and CSP implementation proposed in 2020. All of these attacks make a total of 26 distinct attacks.

B. Attacks Impact

The impact of threats is vary depending on the usage of methods, target, medium, and magnitude in executing the attack. Hence, it is needed to address the attack implication to determine the risk category of threats described in the study. With that, it will be easier to determine defense mechanism priority to overcome the threats. Based on the paper's information, we could classify the attack impact based on three risk level which is high, medium, and low. We categorize each attack impact described before using the OWASP Risk Rating Methodology. In this method, there are two possible impacts to focus on: Technical and business impact. The technical impact can be valued based on information security aspects such as confidentiality, integrity, and availability. Meanwhile, the business impact is different according to each organization's condition. Because of this limit, we examine the technical impacts of related attacks. The number of attacks per risk level both related to TLS and web application is shown in Fig3. It describes that high-risk attacks contain 10 attacks, medium with 11, and low risk with 15 attacks consistent with 26 total distinct attacks mentioned before.

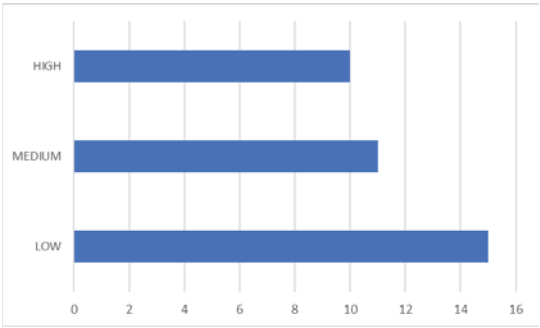

Fig3. Attack Impact Category per Risk Level

a) High Level threats

Attacks with high loss of confidentiality, integrity, and availability will be categorized into this level. In terms of confidentiality, TLS attacks will be of great concern. Meanwhile, the session management area will be the main focus of authentication loss. Web applications could be categorized into both.

MITM and BCP are likely similar methods amplifying the possibility of miss authentication between client and server. An attacker could fully impersonate an entity or change source credibility in order to use this attack. This initial condition will then cause a loss of confidentiality. In terms of integrity, information integrity, and connection integrity both are violated under this MITM and BCP position. An attacker could modify any massage without arising suspicion either on the client or server side.

Some cryptographic related attack has fallen into this category. The usage of past 3DES versions and AES with a weak mode of operation implementation in TLS greatly impacts confidentiality. The reason is that an attacker could read the content of information protected by obfuscate symmetric cipher and thus, also endangering its integrity. In addition, session cookie hijacking is also considered to be dangerous. A full entity authentication mechanism is obsoleted by obtaining a legal session cookie possible for malicious usage.

b) Medium Level threats

Another cryptographic weakness besides the high-level impact category also falls into this medium category. The developer's attempt to not enforcing strong encryption for example, is endangering user. Nevertheless, this not fully enforced encryption is usually supported by the addition of a session mechanism security pattern. So, the possible attacker could not easily defeat such a mixed mechanism.

The lousy implementation of cookie that leads to cookie misconfiguration also falls into these criteria. When an attacker succeeds in circumventing the secure and HttpOnly flag mechanism, it is still hard to find a way to steal the cookie. This stage is also endangered user's PII like email and address information by looking at search history. So the confidentiality is somewhat violated.

c) Low Level threats

This category describes attack with little impact either on usage, loss of private information, or simple considerable data changes. One of them is the miss implementation of security headers like HSTS, CSP, and HPKP headers. For example, HSTS and CSP's bad implementation doesn't impact highly in confidentiality except for a special MITM case that is already categorized in the high-level impact. HPKP is likely to be misconfigured so that attacker can forge user or client certificate and violate authenticity or integrity.

Impracticality that occurs in TLS 1.3-based attacks also makes this type fall into the low category. The attack proposed has not yet been conducted on a real-world basis. Besides, URL privacy violation is considered a low level because of its limited impact on confidentiality.

C. Defense Mechanism Proposed

To mitigate available threats, some of the paper analyzed in this study propose new security mechanisms including Web authentication mechanism [19], [18], [41] Web vulnerability security enhancement [20], [21], [22] Cookie-related authentication hardening [37], [42], TLS protocol hardening [43], and Security enhancement on Android TLS [34]. We will discuss these proposals from the same perspective as our attack explanation mentioned in Section IV.C. In other words, the solutions will be categorized into three main parts: TLS, web session management, and web application defense mechanism. In brief, Fig 4. shows the distribution of these mechanism based on each category.

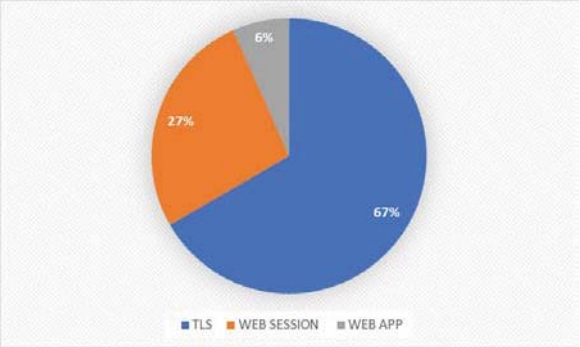

Fig4. Defense Mechanism Distribution Percentage

a) TLS Defense Mechanism

MITM penetrate the client and server authentication validity through the failure to identify certificate legality. To overcome MITM, an efficient TLS-based authentication mechanism which is resistant against MITM in web applications proposed. The TLS-based authentication mechanism is based on the SISCA mechanism and relies on Channel ID-based authentication and server invariance. This new approach then surpasses SISCA implementation by reducing the communication overhead by 50 % [19].

Different defense mechanism is proposed by using a soft-token-based approach for user authentication on top of the SSL/TLS's security features. It is proved that the solution is secure, efficient, and user friendly in comparison to other similar token-like approaches [7]. Recently, a new mechanism named QoS3 was created to overcome HTTPS limitation in blocking intermediate

services like caching proxy. This solution allows the client to introduce in-network caching proxies using fine-grained trust delegation without compromising the HTTPS content's integrity and modifying the format of Transport Layer Security (TLS) [44]. Besides the previous solution, there are some legacy protection ensures TLS usage like HSTS and certificate pinning feature using HPKP.

BCP defense is likely purpose the same method. User faults and browser implementation errors are the main reasons for BCP attacks. To defend this weakness, the proposed mechanism is done by setting a reverse proxy at the server-side and enabling the webserver to provide cross-origin resources with the URLs domain. Since the browser cache poisoning attacker cannot predict the sub -resources with random URLs, and the ones with regular URLs are not poisoned, this approach mitigates cross-origin BCP attacks [18].

A solution to overcome TLS's weaknesses related to encouraging TLS protocol implementation including cryptographic element is the usage of miTLS. The miTLS intends to solve general apparent contradiction between published proofs and real-world attacks, which reveals a gap between TLS theory and practice. It is enhanced with a cryptographic security proof basis that supports low-level protocol [43]. Another cryptographic security proofing method was also proposed in 2017 to supplement Bellare–Rogaway and Canetti–Krawczyk's classic security proofing method. It titled Authenticated Confidential Channel Establishment (ACCE) that also has found many further applications in assessing the security of TLS-DHE [17].

In the realm of TLS 1.3 development, some approaches to ensure its security has been proposed. A robust key confirmation mechanism was proposed in 2016 to strengthen the secrecy and authentication aspect of TLS 1.3 [45]. Another enhancement was targeting the record layer of TLS by building a security model against an adversary that controls the TLS sub-protocols [12]. A multi-stage security model, covering all known kinds of compositional interactions (e.g., cipher suite and protocol) and reasonably strong security notions also constructed to face backward compatibility attack by Jager et al. [11].

Related to TLS and certificate enforcement security headers like HSTS, HPKP, and CSP, the best remedy is for the developer to implement the headers in precise specifications mentioned in the RFC. Meanwhile, in android usage, the application of the TLS protocol is showing good improvement to fight against MITM-type attack [35].

b) Web Session Management Defense Mechanism

A holistic solution needs to be created to overcome many cookie implementation flaws for the session management function. A framework that could extend the guarantees provided by TLS to help maintain session management will be a good start. With enhancing TLS roles, it is possible to remove authentication, mutual authentication, continuous authentication, and session management from the application developments life-cycle. In other words, this defense mechanism effectively eliminates the need for session cookies, session tokens, and similar methods for securing session. In design, this method uses the registration phase to provide mutual authentication using a public key algorithm, the authentication phase to provide validity in authentication using a keyed hash function, and the session key phase[42].

Besides a holistic approach, a real-time solution for cookie-based authentication management also needs to be delivered. In answering this challenge, a solution for hardening against session cookie combination vulnerabilities is proposed. Nowadays, cookie combinations become usual to provide easy access to web sub-services. This is not the best practice for using cookies due to possibilities of wrong domain or false path implementation. To overcome the issue, a tool named Newton that can discover all auth-cookie combinations that permit a user to access a particular part or sub-service is created. This tool is mainly built for developers and penetration testers. In prior research, it could identify 65 sites of 149 popular websites that revealed security vulnerabilities in cookie combinations that also could be exploited by relatively weak attackers [37].

A different method is proposed to provide a password-based authentication mechanism using channel binding. This is done by combining a secure channel protocol, such as TLS, with password authentication or password-authenticated key exchange protocol. The two protocols are bound together using the secure channel's handshake transcript, the server's certificate, or the server's domain name. This named password-authenticated and confidential channel establishment (PACCE) mechanism results in a secure authentication protocol [41]. On the other approach, it is recommended to implement formal protocol verification to ensure the security of an authentication protocol. Formal verification language like ProVerif is one example that could be used to test protocols [46].

c) Web Application Defense Mechanism

An injection-based attack is just one method of many ways of attacking web applications. Related to this, Open Web Application Security Project (OWASP) has been known for its role in providing worldwide security-

based information related to the vulnerability of developing and testing web applicatios. The efforts in implementing the guide given by OWASP is vast and needs much retrying to get the best approach for every unique user. In order to simplify that, research on Security Qualitative Metrics against OWASP Compliance is conducted and produced 230 qualitative security metric, under six categories[21]. These security qualitative metrics are beneficial for security analysts and other parties such as designers, developers, and testers so that overall vulnerability level of the web applications would diminish significantly.

V. Conclusion and Limitation

The web application is one of the most used technology recently due to its flexibility in delivering services to society. Browser connection commonly employs existing HTTPS protocol. On the other hand, both TLS over HTTP and Web application itself have been known for many vulnerabilities. An SLR study on the security of HTTPS implementation on Web Application conducted in this paper is done to give a clear overview in the security state of this subject. By using SLR methodology initiated by Kitchenham et al.[4], the study assessed 45 qualified papers related to the topic and analyzed 24 primary papers. From this research, some findings about type of threats, threats impact, and defense mechanisms become clear.

TLS-based attack and vulnerability research has been conducted in higher number than web-based attack in the last 6 years. 2016 become the year with most attack or vulnerability proposed reaching a total of 10 types of attack. This period blooms because of the constant research rate in the previous years from 2014 till 2015, with an average of 5 research each year. The following period from 2017 till 2019 still reveals decent consistency of research number on this topic. It is also known that the main problem for most vulnerability and attack found is due to misconfiguration and miss implementation of protocols, security support, and cipher suite. These reasons bring mindstream conclusion that the lack of understanding about security-related mechanisms in TLS, session management, and web application still become the culprit of most attacks.

To overcome this bad practice, some defense mechanisms were also proposed. This defense mainly focused on the low-level mechanism and somewhat hard to implement in the right setting. This limitation in the defense mechanism needs to be fixed with other fast solutions that still uncovered in this research. A solution such as web application security testing is one possible answer. This method should be conducted to mitigate such configuration error and outdated software version usage. With that in mind, the future researcher could better prioritize, examine, and evaluate discussed solutions such as web application and protocol security testing to overcome this misconfiguration and misunderstanding issue in the realm of web application and HTTPS protocol security. As a limitation of this study, the discussion is focused on the security posture of HTTPS implementation on Web application and subjects limited to selected papers studied in this SLR.

References

[1] K. Bhargavan, A. Delignat-Lavaud, C. Fournet, A. Pironti, and P.-Y. P. Y. Strub, “Triple handshakes and cookie cutters: Breaking and fixing authentication over TLS,” Proc. - IEEE Symp. Secur. Priv., pp. 98–113, 2014, doi: 10.1109/SP.2014.14.

[2] D. Stuttard and M. Pinto, The Web Application Hacker’s Handbook, Second. Indianapolis, IN 46256: Wiley Publishing, Inc., 2011.

[3] J. C. Carver, E. Hassler, E. Hernandes, and N. A. Kraft, “Identifying barriers to the systematic literature review process,” Int. Symp. Empir. Softw. Eng. Meas., pp. 203–213, 2013, doi: 10.1109/ESEM.2013.28.

[4] B. Kitchenham et al., “Systematic literature reviews in software engineering – A tertiary study,” Inf. Softw. Technol., vol. 52, no. 8, pp. 792–805, 2010, doi: https://doi.org/10.1016/j.infsof.2010.03.006.

[5] A. Liberati et al., “The PRISMA statement for reporting systematic reviews and meta-analyses of studies that evaluate health care interventions: Explanation and elaboration,” PLoS Medicine, vol. 6, no. 7. 2009, doi: 10.1371/journal.pmed.1000100.

[6] Y. Kwon, M. Lemieux, J. McTavish, and N. Wathen, “Identifying and removing duplicate records from systematic review searches,” J. Med. Libr. Assoc., vol. 103, no. 4, pp. 184–188, 2015, doi: 10.3163/1536-5050.103.4.004.

[7] M. L. Das and N. Samdaria, “On the security of SSL/TLS-enabled applications,” Appl. Comput. Informatics, vol. 10, no. 1–2, pp. 68–81, 2014, doi: 10.1016/j.aci.2014.02.001.

[8] E. Ronen, K. G. Paterson, and A. Shamir, “Pseudo Constant Time Implementations of TLS Are Only Pseudo Secure,” in Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, 2018, pp. 1397–1414, doi: 10.1145/3243734.3243775.

[9] J. Amann, O. Gasser, Q. Scheitle, L. Brent, G. Carle, and R. Holz, “Mission Accomplished? HTTPS Security after Diginotar,” in Proceedings of the 2017 Internet Measurement Conference, 2017, pp. 325–340, doi: 10.1145/3131365.3131401.

[10] S. Calzavara, R. Focardi, M. Nemec, A. Rabitti, and M. Squarcina, “Postcards from the Post-HTTP World: Amplification of HTTPS Vulnerabilities in the Web Ecosystem,” in Proceedings - IEEE Symposium on Security and Privacy, 2019, vol. 2019-May, pp. 281–298, doi: 10.1109/SP.2019.00053.

[11] X. Lan, J. Xu, Z.-F. Zhang, and W.-T. Zhu, “Investigating the Multi-Ciphersuite and Backwards-Compatibility Security of the Upcoming TLS 1.3,” IEEE Trans. Dependable Secur. Comput., vol. 16, no. 2, pp. 272–286, 2019, doi: 10.1109/TDSC.2017.2685382.

[12] A. Delignat-Lavaud et al., “Implementing and Proving the TLS 1.3 Record Layer,” in Proceedings - IEEE Symposium on Security and Privacy, 2017, pp. 463–482, doi: 10.1109/SP.2017.58.

[13] C. Cremers, M. Horvat, J. Hoyland, S. Scott, and T. Van Der Merwe, “A comprehensive symbolic analysis of TLS 1.3,” in Proceedings of the ACM Conference on Computer and Communications Security, 2017, pp. 1773–1788, doi: 10.1145/3133956.3134063.

[14] D. Springall, Z. Durumeric, and J. A. Halderman, “Measuring the Security Harm of TLS Crypto Shortcuts,” in Proceedings of the 2016 Internet Measurement Conference, 2016, pp. 33–47, doi: 10.1145/2987443.2987480.

[15] K. Bhargavan and G. Leurent, “On the practical (in-)security of 64-bit block ciphers: Collision attacks on HTTP over TLS and OpenVPN,” in Proceedings of the ACM Conference on Computer and Communications Security, 2016, vol. 24-28-Octo, pp. 456–467, doi: 10.1145/2976749.2978423.

[16] V. Frost, D. (Jing) Tian, C. Ruales, V. Prakash, P. Traynor, and K. R. B. Butler, “Examining DES-Based Cipher Suite Support within the TLS Ecosystem,” in Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, 2019, pp. 539–546, doi: 10.1145/3321705.3329858.

[17] T. Jager, F. Kohlar, S. Schäge, and J. Schwenk, “Authenticated Confidential Channel Establishment and the Security of TLS-DHE,” J. Cryptol., vol. 30, no. 4, pp. 1276–1324, 2017, doi: 10.1007/s00145-016-9248-2.

[18] Y. Jia, Y. Chen, X. Dong, P. Saxena, J. Mao, and Z. Liang, “Man-in-the-browser-cache: Persisting HTTPS attacks via browser cache poisoning,”

Comput. Secur., vol. 55, pp. 62–80, 2015, doi: https://doi.org/10.1016/j.cose.2015.07.004.

[19] A. Esfahani et al., "An Efficient Web Authentication Mechanism Preventing Man-In-The-Middle Attacks in Industry 4.0 Supply Chain," IEEE Access, vol. 7, pp. 58981–58989, 2019, doi: 10.1109/ACCESS.2019.2914454.

[20] D. Mitropoulos, P. Louridas, M. Polychronakis, and A. D. Keromytis, "Defending Against Web Application Attacks: Approaches, Challenges and Implications," IEEE Trans. Dependable Secur. Comput., vol. 16, no. 2, pp. 188–203, 2019, doi: 10.1109/TDSC.2017.2665620.

[21] F. Ö. Sönmez, "Security Qualitative Metrics for Open Web Application Security Project Compliance," Procedia Comput. Sci., vol. 151, pp. 998–1003, 2019, doi: https://doi.org/10.1016/j.procs.2019.04.140.

[22] S. Calzavara, R. Focardi, M. Squarcina, and M. Tempesta, "Surviving the Web: A Journey into Web Session Security," Web Conf. 2018 - Companion World Wide Web Conf. WWW 2018, vol. 50, no. 1, pp. 451–455, 2018, doi: 10.1145/3184558.3186232.

[23] A. Lavrenovs and F. J. R. Melón, "HTTP security headers analysis of top one million websites," in 2018 10th International Conference on Cyber Conflict (CyCon), 2018, pp. 345–370, doi: 10.23919/CYCON.2018.8405025.

[24] W. J. Buchanan, S. Helme, and A. Woodward, "Analysis of the adoption of security headers in HTTP," IET Inf. Secur., vol. 12, no. 2, pp. 118–126, 2018, doi: 10.1049/iet-ifs.2016.0621.

[25] M. Ying and S. Q. Li, "CSP adoption: current status and future prospects," Secur. Commun. Networks, vol. 9, no. 17, pp. 4557–4573, 2016, doi: 10.1002/sec.1649.

[26] H. Kwon, H. Nam, S. Lee, C. Hahn, and J. Hur, "(In-)Security of Cookies in HTTPS: Cookie Theft by Removing Cookie Flags," IEEE Trans. Inf. Forensics Secur., vol. 15, pp. 1204–1215, 2020, doi: 10.1109/TIFS.2019.2938416.

[27] S. Sivakorn, I. Polakis, and A. D. Keromytis, "The Cracked Cookie Jar: HTTP Cookie Hijacking and the Exposure of Private Information," in Proceedings - 2016 IEEE Symposium on Security and Privacy, SP 2016, 2016, pp. 724–742, doi: 10.1109/SP.2016.49.

[28] S. Sivakorn, A. D. Keromytis, and J. Polakis, "That's the Way the Cookie Crumbles: Evaluating HTTPS Enforcing Mechanisms," in Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society, 2016, pp. 71–81, doi: 10.1145/2994620.2994638.

[29] S. D. los Santos and J. Torres, "Analysing HSTS and HPKP implementation in both browsers and servers," IET Inf. Secur., vol. 12, no. 4, pp. 275–284, 2018, doi: 10.1049/iet-ifs.2017.0030.

[30] K. Bhargavan, A. Delignat-Lavaud, C. Fournet, A. Pironti, and P.-Y. Strub, "Triple handshakes and cookie cutters: Breaking and fixing authentication over TLS," in Proceedings - IEEE Symposium on Security and Privacy, 2014, pp. 98–113, doi: 10.1109/SP.2014.14.

[31] E. S. Alashwali, P. Szalachowski, and A. Martin, "Exploring HTTPS security inconsistencies: A cross-regional perspective," Comput. Secur., vol. 97, p. 101975, 2020, doi: https://doi.org/10.1016/j.cose.2020.101975.

[32] C. Cremers, M. Horvat, S. Scott, and T. V. D. Merwe, "Automated Analysis and Verification of TLS 1.3: 0-RTT, Resumption and Delayed Authentication," in Proceedings - 2016 IEEE Symposium on Security and Privacy, SP 2016, 2016, pp. 470–485, doi: 10.1109/SP.2016.35.

[33] C. Patton and T. Shrimpton, "Partially specified channels the TLS 1.3 record layer without Elision," in Proceedings of the ACM Conference on Computer and Communications Security, 2018, pp. 1415–1428, doi: 10.1145/3243734.3243789.

[34] X. Wei and M. Wolf, "A Survey on HTTPS Implementation by Android Apps: Issues and Countermeasures," Appl. Comput. Informatics, vol. 13, no. 2, pp. 101–117, 2017, doi: 10.1016/j.aci.2016.10.001.

[35] D. Shin and J. Sun, "An Empirical Study of SSL Usage in Android Apps," in Proceedings - International Carnahan Conference on Security Technology, 2018, vol. 2018-Octob, doi: 10.1109/CCST.2018.8585431.

[36] S. Calzavara, G. Tolomei, A. Casini, M. Bugliesi, and S. Orlando, "A Supervised Learning Approach to Protect Client Authentication on the Web," ACM Trans. Web, vol. 9, no. 3, Jun. 2015, doi: 10.1145/2754933.

[37] Y. Mundada, N. Feamster, and B. Krishnamurthy, "Half-Baked Cookies: Hardening Cookie-Based Authentication for the Modern Web," in Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, 2016, pp. 675–685, doi: 10.1145/2897845.2897889.

[38] S. Calzavara, R. Focardi, M. Squarcina, and M. Tempesta, "Surviving the web: A journey into web session security," ACM Comput. Surv., vol. 50, no. 1, 2017, doi: 10.1145/3038923.

[39] H. Kwon, H. Nam, S. Lee, C. Hahn, and J. Hur, "(In-)Security of Cookies in HTTPS: Cookie Theft by Removing Cookie Flags," IEEE Trans. Inf. Forensics Secur., vol. 15, pp. 1–12, 2019, doi: 10.1109/TIFS.2019.2938416.

[40] R. Ferreira and R. L. Aguiar, "Repositioning privacy concerns: Web servers controlling URL metadata," J. Inf. Secur. Appl., vol. 46, pp. 121–137, 2019, doi: https://doi.org/10.1016/j.jisa.2019.03.010.

[41] M. Manulis, D. Stebila, F. Kiefer, and N. Denham, "Secure modular password authentication for the web using channel bindings," Int. J. Inf. Secur., vol. 15, no. 6, pp. 597–620, 2016, doi: 10.1007/s10207-016-0348-7.

[42] Z. A. Alizai, H. Tahir, M. H. Murtaza, S. Tahir, and K. McDonald-Maier, "Key-Based Cookie-Less Session Management Framework for Application Layer Security," IEEE Access, vol. 7, pp. 128544–128554, 2019, doi: 10.1109/ACCESS.2019.2940331.

[43] K. Bhargavan, C. Fournet, and M. Kohlweiss, "miTLS: Verifying protocol implementations against real-world attacks," IEEE Secur. Priv., vol. 14, no. 6, pp. 18–25, 2016, doi: 10.1109/MSP.2016.123.

[44] A. Al-Dailami, C. Ruan, Z. Bao, and T. Zhang, "QoS3: Secure Caching in HTTPS Based on Fine-Grained Trust Delegation," Secur. Commun. Networks, vol. 2019, 2019, doi: 10.1155/2019/3107543.

[45] M. Fischlin, F. Gunther, B. Schmidt, and B. Warinschi, "Key Confirmation in Key Exchange: A Formal Treatment and Implications for TLS 1.3," in Proceedings - 2016 IEEE Symposium on Security and Privacy, SP 2016, 2016, pp. 452–469, doi: 10.1109/SP.2016.34.

[46] I. B. Guirat and H. Halpin, "Formal verification of the w3c web authentication protocol," 2018, doi: 10.1145/3190619.3190640.