ISSN 2661-2666 (Online) International Scientific Journal Monte (ISJM) DOI: 10.33807/monte.20232840 Volume 7, (No).2 (2023): April

Substantial security challenge to web applications, using modified OTC and OWASP update

Aleksander Biberaj, Igli Tafa, Kristi Ndoni , Islam Tahiraj, Andrea Muco
Faculty of information and technology
Polytechnic University Of Tirana
Tirana, Albania
abiberaj@fti.edu.al, itafaj@fti.edu.al, kristi.ndoni@fti.edu.al , andrea.muco@fti.edu.al , islam.tahiraj@fti.edu.al

Abstract
Internet security is studied by computer science and serves as a safe medium for exchanging data while minimizing the likelihood of online threats. Through use of advanced web-based software is growing because they provide the user with a lot of features. Web technologies have an important role in different industries, like schooling, retail, medical care, and payment systems. Session bugs are becoming more common throughout web applications whereas their value in community grows. Hackers try to profit of incorrectly designed websites so they take hold of victim's sessions and also of identities. As a result, session handling represents a substantial security challenge to web applications. Weak programming methods are among the causes for effective session acquisition. A further explanation is that the server as well as the customer verify themselves differently at first.

In the recent years most common strikes used amongst attackers is session hijacking. Based on latest recent OWASP update, session hijacking is indeed one of the second frequent assault that happens mostly. It is one important attack among others, which a hacker may use to connect directly to a customer's operating session. User Hijacking occurs when a hacker takes victim's session id, and uses it to obtain entry into the victim's actual session. This system will provide protection in case of this attack form once it has been successfully implemented.

Keywords—sessions, cookies, hijacking, attacks, hacker, cybersecurity, bug, vulnerability, one time cookies, HTTP, sql injections.

I. INTRODUCTION

People are seeing a realistic advancement of both the internet also of web technologies in current history. Most programs previously ran on desktop back in the day. However, tables have turned, and certain programs can be accessed via a browser. As a result, websites that were once static HTML websites have evolved into interactive Web apps, or full of content services available everywhere on Internet. Commercial transfers and mail sharing are only two of the utilities provided by web apps. The financial sector, schooling as well as related organisations, care and wellness facilities, various corporate groups, and state institutions are some of most influential industries in which web-based technologies are main tool of optimizing everyday operational efficiency or updating current system. Users may acquire a core resource, including Web server, as well as other resources, including database servers, via web apps. Users think these technologies are safe, however the fact is that certain web apps have serious security vulnerabilities that cause basic hacks to occur.

Session bugs are the most prevalent risk within web-based apps. All of that is attributed to web application's poor session control. Attackers take advantage of incorrectly designed websites and hijack user's sessions, which leads in identity theft. Session states stores precious data, so this sensitive data, e.g session state, provides an important objective for attackers. Whenever a person has to login to a protected website, user must fill information that verifies their legitimacy, including a username, password, or even a birthdate and contact information, which enables them to verify their identity also to unlock the information available. To reduce complexity of re-authentication, session control involves web-based application to build a session because then user does not have to go through this process any moment they want to do something. Session control means that a person connecting to cloud and viewing information is same user that logs during first time. As a result, unauthorized users also attackers aim sessions, which can be exploited to obtain access to the device with no need of authentication. User identifiers are the most popular method of session control.

A session begins whenever a user accesses or logs to another certain web site or application via their device, that concludes once consumer closes or logs out of the device, or closes website also the application. While connected, a session will briefly store data correlated also with actions of consumers. Primary purpose of session would be to keep track of user's authentication information, as well as the working session allows the user to enter the program. A session key, known

as SID, would be a name=value combination. Value seems to be a sequence of alphanumeric characters that corresponds to a web session. Every submitted query will have the SID inserted or added to it, the SID can be used as an identity provider inside the program. As a result, the SID must be created as well as to be stored safely. Data breach and Session Management attacks are third most significant Web application security concern, according to Open Web Application Security Project (OWASP). Session hijacking, session locking, or network-based spying attempts, are all popular techniques on session control.

In this paper, its presented an analysis on different session-related weaknesses as well as detection and prevention strategies, as well as different algorithms used in this branch.

LITERATURE REVIEW

A. Session hijacking

One frequent form of threat is session hijacking. Due to obvious ease with which attackers can connect directly to session, makes this form of attack dangerous[1]. Whenever an person is logging in, also about to log in to the system but has also formed a link also with server, intruder hijacks session while pretending to be original consumer. Once intruder gets control inside the server, he does not have to go to any efforts to break login key because he's been verified to get the connection. Client hijacking is where a hacker takes around a user's session key also has complete ownership of such machine although session also is running[2].

Active session hijacking occurs when an attacker assumes control of an operating data channel. The intruder will silence one computer, usually user desktop, as well as to take over user's position throughout contact interaction here between server as well as digital device, releasing server as well as the user computer's connection[3].

Passive session hijacking is similar to active session, but rather than removing consumer out of active session, hacker monitors communication within the server as well as desktop device. Inside a passive link, attacker tracks person's details also saves everything inside an personal database with attacking purposes. It's also recommended that an intruder begin by hijacking a passive session[4].

B. Related Works

With initial introduction throughout mid-90s, using cookies like session authentication tokens had already created privacy issues. Many other studies have shown that web authentication schemes have many flaws, like susceptibility of session hijacking threats[5],[6].

Browsers do have variety of bugs. Webpages have been shown to be susceptible to cyberattacks every ten minutes. That's why it is important to comprehend and address threat of such assault. Session-related bugs are well-known comparing to others. Several experts as well as academics already suggested numerous identification and avoidance approaches.That why security analysts have suggested improvements to strengthen authentication cookies' reliability. [7] proposed cookie systems which use possibly the best cryptographic approaches of having greater security as well as credibility assurances. Furthermore, some writers suggested that cookie expiry dates is being used to mitigate effects of session hijacking threats. Most systems, though, utilize longer termination periods to prevent compromising costumer experience, which reduces the feasibility of such strategy.

Cache cookies which are various types of permanent state within web (that is, client data, short term internet documents), have been found helpful to cookies in preserving client as well as session identities by [8]. Cache cookies, although immune towards phishing threats, anyway they require HTTPS security to avoid malicious activities. Linked threats, as illustrated by [9], present quite a modern category of cookie-stealing threats wherein cookies saved through one page could also be changed from an other if indeed both of domains possess a considerably longer prefix. To counteract such threats, results recommend core cookies, a simple addition to regular cookies with low operational cost.

Authors Felten and Schneider were ones to bring issue about intrusive cached data to notice[10], plus they were also the ones who created so called phrase "cache cookies." Researchers demonstrated why a host may identify the characteristics of a specific email attachment in such a web browser, allowing stored pictures being used as data tags.

Its methods, but at other hand, were recorded in the data studies and therefore are relatively hard to complete.

Researcher [11], on the other hand, revealed that store data depending on internet backgrounds might have been more readily exploited. A byproduct of Css (technology of displaying Webpages) allows a site to include software in a website which identifies if one client has one certain site inside its memory.

[12] investigate potential consequences of cached data or associated computer capabilities, which offer a comprehensive perspective of site to site domain tracing risks for consumers. Users often discover new aspects of cached data, like object identifiers, something we'll talk about later. [13] suggest using web applications that impose uniform security standards along a variety of cross-domain monitoring mechanisms.

Likewise, we focus upon on beneficial aspects using caching data inside this study. Here are suggested strategies that benefit from caching data whilst aggravating security issues.

II. SESSION HIJACKING VULNERABILITIES

Session monitoring keeps record to customer's behavior through several connections into webpages. Common application of monitoring is mostly sign in, because they are often included where customer expected sign in does not occur, like on certain b2b companies including networking platforms.

Most common method of doing this is assigning each individual a special code, like a connection identifier also either one connection token. Usually, several methods are used to enforce tokens:

- Cookie is used for token storing.
- Token is submitted throughout secret sectors generated from particular system type.
- Tokens get attached inside every connection that visitor taps after they have been generated throughout system.

Some tools are used in connection monitoring. Few programs, as starters, utilize HTTP verification. For submitting login information, search engine might utilize Http request instead of system's Website script. Nearly all this type of verification remains uncommon. Some apps utilize sessionless systems, which means apps wont utilize tokens and just transmit customer's information between every system contact. Typically, cryptographic algorithms are being utilized in accompanied by method[14].

The most serious vulnerabilities relating session hijacking are in the token generation and session monitoring strategies.

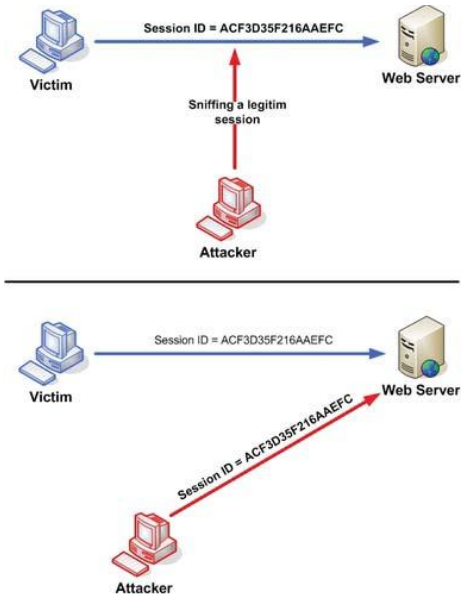

Fig. 1. Session hijacking hack is carried out by modifying a token session.

A. Token generation

Such type of vulnerability allows hackers to establish a token, so as a result they can use legitimate token. Tokens may get obtained when putting different elements of customer data, including username and perhaps e-mail address. Any hacker may decrypt token as well as generate a legitimate one when the methods are adjustable. Tokens may also form out of alphabetic character series components only with condition that every token generates irregularly[15]. Whenever token-generating algorithm implements one of three techniques, hackers have better chances of predicting tokens. Encrypted variations build tokens from encoding any standard numerical series. Infirm generation method seems to be third approach here. Since machines depend of deterministic processes, they don't seem to offer flexibility. Computers use arbitrary value producers to get around the problem (pseudorandom number generators). Distinct input devices, including sound panel performance as well as amount of button presses, get combined to produce PRNGs[16].

B. Sessions control mechanisms

Although tokens are produced correctly and volatility, hackers may be clever enough to tackle them. Hackers will do so by taking advantage of unsecured packets including vulnerabilities with in secret algorithms which websites needs in producing tokens. Tokens can also be intercepted through looking for in log data like window reports, proxy logs. Any hacker will retrieve token from logs when its included in a URL as variable. One further option is looking for tokens inside search engine as well as proxy buffer, that will save whole site also response headers. Utilizing weak token assignment systems, granting different tokens to similar consumer, also using fixed tokens of every customer are few of many methods. Furthermore, ineffective session closure strategies open up several cyber threats. Activity must be short also feasible to decrease contextual timeframe of threats. Since certain programs don't have a procedure specific to session's termination, hackers will seek several different properties prior to session ending. Whenever customer signs off, system deletes token from client's computer; however, because customer (either a hacker) has sent any formerly accessed token, system continues allowing it [17].

In plight scenarios, system gets zero demand during sign out also the connection is not invalidated. When any intruder acquires that key, he or she will be able to still employ session much like a person that has not signed out at all. Eventually, unless tokens are stored inside cookies, cookie variables can be vulnerable to many attacks. Because protected label is not placed inside cookies, it gets transmitted through unsecured packets [18]. Cross-site scripting threats will capture hackers

because HTTPOnly marker hasn't been placed. The reach of a cookie may be used by hackers. Some weaknesses are related to incorrect HTTPS utilization. For starters, certain programs recognize HTTPS secured sectors yet are using matching token outer of secured sectors. As a result, hackers will get token through eavesdropping HTTP traffic. Second, also inside secured sectors HTTPS can be utilized, several systems support HTTP protocol. For this reason , hackers may persuade customers to send HTTP calls so they snatch token. Spoofing emails, posters, and psychological manipulation are widely used in several threats. Ultimately, many apps utilize HTTP for viewing static resources such as photos, script as well as Css. Detecting such inquiries allows hackers catching tokens.

Hackers will pull out threats including cross-site request forgery, session sniffing, predicting, as well as session fixation, also HTTP reply separating through leveraging weaknesses already mentioned above. Here its defined the triggering weaknesses to every threat, that hackers should check prior to launching a thrust [19].

C. *Session sniffing*

Such threats include indirectly blocking any information gathered being shared in sessions.

a) HTTP packets sniffing.

HTTP streams are intercepted within that attempt. Hackers should find any analyzer on a computer inside this user's system or even Website software's institution's domain. Four supporting underlying problems exist. Initially, non-HTTPS areas of such site may be identified. Stable marker is also not fixed, for starters. Third, service provides HTTP calls to sites that are protected by HTTPS. Consequently, prior to authorization, program employs HTTP[20],[21].

b) Cache sniffing

Token can be obtained in every configuration including them if hacker gains entrance to proxy cache as well as search engine. Two triggering weaknesses are related to cache management of Webpage. HTTP response headers does not include instructions. Than CacheControl:private policy hardly allows buffer to be used on computer where customer is currently running. With distributed computers, such scenario poses danger (e.g. in Internet cafes)[22].

c) Sniffing logs.

Tokens are obtained through reviewing logs with in various structures engaged with server-client interaction throughout this threat. Two supporting weaknesses in this system exist. Primarily, token gets sent through URL variables, that also means token could end up with in logs. To continue, token gets sent with a secret sector, also GET calls are accepted rather than POST requests by system. The client-side scripting file may implement this same demand reversal. Token gets submitted as URL variable, resulting logs will contain it.

d) CSRF

Such threat induces that target performs acts inside an environment where they have been authenticated; a common tactic would be to deliver connection via email either via instant messages. This threat has the potential to damage customer details. Distinctly of several other threats CSRF also attempts to perform precise activities rather than gaining session access[23].

e) Fixation of the session

Even before one user's authorization, hacker addresses token. There are three stages towards hacker's threat:

1-The first step is to establish the session. Hacker establishes so called "bait session" into the computer also accepts either generates key. Under certain instances, intruder will submit demands during routine periods to maintain the connection active, so called session maintenance.

2-Fixation of the session. Token is inserted through user 's machine by the hacker.

3-The access into the connection. Hacker awaits around upon victim so they access session before hacker attempts getting connected himself. There are two types of connection control algorithms:

- Rigorous systems permits just established, predefined tokens, while tolerant systems welcomes recent tokens which launch brand fresh connection.
- Hacker offers specific tokens also utilizes them into tolerant schemes. In rigid schemes, hacker opens one connection which gets leave active during threat.

Conditioned by process of transmitting each token, various methods are used to session fixation[24]. Hacker will compel their target into tapping one connection built impromtu using URL variable. Depending on existence of protected sector, hacker will take advantage of an XSS weakness.

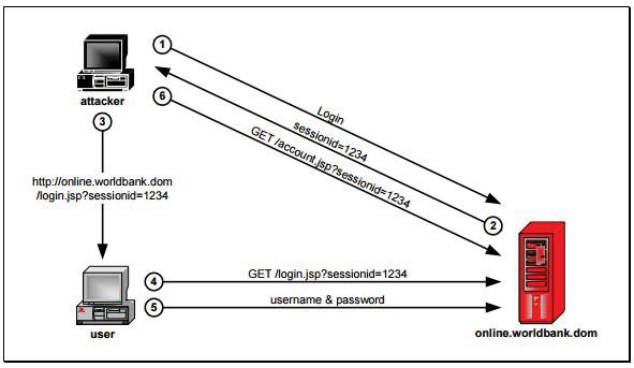

Fig. 2. Again from beginning of procedure until session use, this image describes mechanism of session fixation.

Bad configuration as well as deployment allows session fixation. As a result, through successful Software product

architecture, above mentioned approaches of such vulnerability are being removed[25].

III. PROPOSED ALGORITHMS

Here its propose a new approach for preventing hijacking threats in this segment, focusing in recent study to current work, also resolving their numerous implications.

A. One time cookies

That cookies gets substituted like connection authentication keys, its suggested an alternative method. One time cookies, this approach, offers a strong protection toward unauthorized access, making sure they still are meeting needs of dynamically dispense systems. OTC generally, keeps connection verification and further connection monitoring functions apart.

Attacker's target throughout this model would be to gain charge over connections that have been created through customers of a website. OTC considers all inactive yet aggressive polynomial time antagonists. Knowledge released betwixt client's browser as well as website is accessible toward silent adversaries. Data gets caught either of system (digital), and even by system log files. Inactive adversaries may attempt conducting or reusing authentication tokens depending towards knowledge in order to sabotage another customer's session connection. An aggressive attacker will have similar knowledge possession like inactive opponent, however attacker may also selectively manipulate enquiries as well as replies sent betwixt web page and system. Proactive attacker, may alter, construct, also prohibit messaging of entering its desired target. Any successful attacker may also carry out domain-level threats between the website also the app, such as session fixation, cross-site scripting, also cross-site tracing. Malicious software attempts of capturing OTC token or stealing OTC permanent data from customer's device are both options for a practical challenge. Here threats are not counted under which attacker assumes possession including its victim's account as well as operating system (for example, via leveraging cache overload either malicious programs) but rather threats mostly on software product architecture. Furthermore, OTC hasn't had security from dishonesty threats. Encrypting the configuration among one's authentication data during sign in process, OTC uses HTTPS. Then as result, OTC expects if HTTPS has been set up properly also securely. Developers don't accept threats which compromise HTTPS's security assurances throughout successful authentication. Attacker might even extract customer's password, which is much useful accreditation, whether alike threats were necessary.

Here are defined certain characteristics that must be present in order for providing a reliable as well as realistic solution for authorization cookie. Such characteristics were being used for OTC build: This possible framework ought to have strong user session authorization and be automatically protected towards session hijacking is called connection honesty. For demand authentication, this suggested algorithm doesn't need condition inside website. Namely , through consideration regarding system load condition, it really isn't distinct than authorization cookies. Such characteristic are important towards massively scalable websites. That theoretical system ought to be identical with cookies in terms of customer interface. There is no need for extra human engagement. Particularly, switching through authorization cookies and OTC doesn't really affect that customer functionality. This possible framework will provide authentication tokens that are completely classified as well as promises of honesty. Authorization token particularly, also shouldn't release data that undermines website confidentiality, being immune against cryptographic algorithms threats and being abscond.

As order, OTC generates another specific token. The connection key binds every token in specific demands; therefore, token could become reusable over several demands. Particularly , OTC tickets are stored in state details needed for evaluating token. Every other card becomes secured using a lengthy code that is exchanged across servers throughout website. As a result, data contained throughout card could become accessed by website servers. Details of the card are not visible towards recipient. Creds are keys that are saved inside application, while tokens are properties that are added into each requisition.

B. SSL Stripping-based hijacking attacks

This architectural design is split into two sectors: server as well as client side structures. This global selector collection gets downloaded via a single system by customer. Many of webpages inside this index have their safety ratings listed here. Just like mentioned herein, such collection gets displayed different levels to alert notifications towards target consumers. Customer submits he's responses by consumers to system on regular basis such that neural networks could use them for boosting safety levels for webpages.

Customer mechanism operates through transmitting HTTP POST calls received through websites as well as tracking them. Requisition isn't apprehended anyway when link becomes HTTPS. Moreover, if somehow demand isn't of POST kind, inspection doesn't really exist. Next its searched whether page contains HTTPS when person gives every POST call. Whether this is the case, we couldn't intervene. Unless website uses HTTP also has a passcode as well as sign in area, demand becomes intercepted. Than we search whether page is already inside any database after intercepting this message. Than we get protection ranking through database, when something exists within data set.

The server collects data about person's behavior through different customers also utilizes this for increasing safety status in webpages. Implementations on server gets divided to two categories. Initially, we measure 50 percent differentiation threshold regarding ours sample using websites within certain current databases. For determining the linear regression r, its used a excellently analytical methodology named divide validity checking. Throughout this method, its divided the customer information for also sites across mutually exclusive subsets as well as looking for correlations among them.

C. *Using modified OTC*

Below you can find indeed elements of presumed structures. Person, also known as a customer, will be one that makes the case. If customer needs buying anything, they can submit another message into server which includes customer's login details. Customer would be granted an OTC upon effective authorization, that OTC is used to verify customer about any requisition user creates. When any customer replies, it's also accompanied by an OTC.

Proxy seems being a machine which serves like middleman betwixt any external system and the internet. It's always shown on the customer's either on user's end. In other hand, rather than utilizing proxy service on customer end, they utilize RPS on server.

Like a result, RPS must process any call from customer. RPS's goal is taking assigning OTC, address of IP, connection ID, but also website signature, then RPS would search through connection ID, OTC, IP address, also website signature with every single arriving message. When either one of variables shift, RPS will switch into different tab.

This really is the system to whom user sends an message. That server verifies passwords, processes every user demand, also communicates with users.

Below it is shown how suggested methodology operates.

The client inserts login details. Call gets into RPS submitted, that collects each user's internet protocol address search engine signatures as well as passes it all into database. Database verifies passwords, executes call (i.e., tarns then delivers http call straight to user), yet not until passing via algorithm. OTC is generated by RPS , either connection ID, also sends those straight to user, along with reply. After that, customer saves OTC which is sent. Customer also can submit OTC through RPS within each call he/she makes. Between each fresh call submitted mostly from customer, RPS tests this as well as generates OTC anew. Because OTC, internet protocol address, connection ID, or application signature shift, RPS halts connection.

IV. FUTURE WORK

In this peper one time cookies are analysed and presented. One time cookies within websites are an essential starting point toward ensuring connection consistency across current operating systems. For supporting OTC, technologies like Java Servlets, and Ruby on Rails must be modified. Most good topic implementations would require union assistance, whether within protocols among various implementations.

Since HTTPS is hard even expensive to implement, multiple people also searched towards a connection honesty approach which could run across HTTP while also being protected toward inactive system hackers . Which seems to be another intriguing area of study which isn't addressed via one time cookies.

V. CONCLUSIONS

Session hijacking remains one significant problem which any webpage owner as well as company should prioritize when everything comes to protecting any online details. Here into this scientific report covers many of the weaknesses associated with accessing website, as well as attacks which an intruder could face into compromising sensitive data.Here study discussed different forms of connection intercepting attacks but also whether they impact web operations. Majority of web-based session hijacking are caused by spoofing hacks viruses, cross-site scripts, as well as SQL injection attacks , among other things. Strategies towards stopping connection hijacking as well as methods employed among attackers for committing web-based crimes are addressed. Prior research also shown that many loopholes already remain throughout online payments, necessitating an immediate implementation of such strong amount of protection in websites which secure personal data during assaults.

Through ages, certain dangers that come with using cookies as connection authorization tokens are being established.

Solutions of verification cookies are being suggested, although most haven't yet been implemented since them cannot fulfill standards in big scalable online systems. Most of suggested solutions, in particular, necessitate expensive operate coordination through website, which would be very major problem with scalable applications. Here OTC is proposed, a premise safe solution of verification cookies, throughout this analysis. OTC may be hardly immune towards session theft however that often keeps cookies convenience as well as consistency advantages.

VI. RERENCES

[1] (n.d.). Retrieved from XSS stealing cookies 101: http://jehiah.cz/a/xss-stealing-cookies-101

[2] (n.d.). Retrieved from Deadliest web attacks : Entertaining insights into web security:

https://deadliestwebattacks.com/2010/05/18/cross-site-tracing-xst-the-misunderstood-vulnerability/

[3] Andrew Bortz, A. B. (2011). Origin Cookies: Session Integrity for. Web 2.0 Security and Privacy Workshop .

[4] Annies Minu Sathiyaseelan, V. J. (2017). A Proposed System for Preventing Session Hijacking with Modified One-Time Cookies. International Conference On Big Data Analytics and computational Intelligence (ICBDACI).

[5] Anuj Kumar Baitha, P. S. (2018). Session Hijacking and Prevention Technique. International Journal of Engineering & Technology.

[6] Ari Juels, M. J. (2006). Cache cookies for browser authentication. Symposium on Security and Privacy. IEEE .

[7] C. Jackson, A. B. (2016). Web privacy attacks on a unified same-origin browser.

[8] Clover, A. (2002). Timing attacks on Web privacy (paper and specific issue). Retrieved from www.securiteam.com/securityreviews/5GP020A6LG.html

[9] E. W. Felten, M. A. (2018). Timing attacks on Web privacy. ACM Conference on Computer and Communications. http://www.cs.princeton.edu/sip/pub/webtiming.pdf.

[10] Enos LETSOALO, P. S. (2017). Session Hijacking Attacks in Wireless Networks: A Review of Existing Mitigation Techniques. IST-Africa. Africa: Paul Cunningham and Miriam Cunningham .

[11] Goodin, D. (2009). Newfangled cookie attack steals/poisons website creds. Retrieved from Theregister: http://www.theregister.co.uk/ 2009/11/04/website_cookie_stealing/print.html

[12] Israel O. Ogundele, A. O. (2020). Detection and Prevention of Session Hijacking in Web Application Management. International Journal of Advanced Research in Computer and Communication Engineering. Computer Technology Department, Yaba College of Technology, Yaba, Lagos, Nigeria.

[13] Italo Dacosta, S. C. (n.d.). One-Time Cookies: Preventing Session Hijacking Attacks with Stateless Authentication Tokens. Georgia Institute of Technology.

[14] Jasim Hasan, A. M. (2019). Evaluation of Web Application Session Security. Kingdom of Bahrain: University of Bahrain.

[15] Jeevitha. R, N. B. (2017). Malicious node detection in VANET Session Hijacking Attack. Coimbatore, India: Dr.G.R.Damodaran College of Science.

[16] Karis D'silva, V. J. (2017). An Effective Method for Preventing SQL Injection Attack and Session Hijacking. 2nd IEEE International Conference On Recent Trends in Electronics Information & Communication Technology (RTEICT). India: University, Manipal, India.

[17] Kevin Fu, E. S. (2001). Dos and Don'ts of Client Authentication on the Web. Security Symposium. USENIX.

[18] Mainduddin Ahmad Jonas, M. S. (2018). An Intelligent System for Preventing SSL Stripping-based Session Hijacking Attacks. Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Weisberg Division of Computer Science, Marshall University, Huntington, WV.

[19] Namitha P, K. P. (2018). A Survey on Session Management Vulnerabilities in Web Application. International Conference on Control, Power, Communication and Computing Technologies (ICCPCCT).

[20] Nidhi Thakkar, R. V. (2018). Secure Model for Session Hijacking using Hashing Algorithm. Computer Science & Engineering, NSIT, Jetalpur, Ahmedabad.

[21] Qiao Hu, G. P. (2018). A Session Hijacking Attack on Physical Layer Key Generation Agreement. University of Hong Kong.

[22] Renascence Tarafder Prapty, S. A. (2020). Preventing Session Hijacking using Encrypted One-Time-Cookies. Huntington, WV, USA: Weisberg Division of Computer Science, Marshall University.

[23] Rupinder Gill, J. S. (2006). Experiences in Passively Detecting Session Hijacking Attacks in IEEE 802.11 Networks. Australia: Queensland University of Technology.

[24] Stefano Calzavara, A. R. (2018). Sub-session hijacking on the web: Root causes and prevention. Italy: Dipartimento di Scienze Ambientali, Informatica e Statistica, Università Ca' Foscari Venezia.

[25] Visaggio, C. A. (2010). Session Management Vulnerabilities in Today's Web. IEEE Security and Privacy Magazine .