International Journal of Engineering Inventions
e-ISSN: 2278-7461, p-ISSN: 2319-6491
Volume 5, Issue 4 [March 2016] PP: 22-26

Web Security: Countermeasures for Application level Attacks

Nataasha Raul [a] , Radha Shankarmani[b]
Research Scholar, Sardar Patel Institute of Technology, Mumbai, India[a]
Professor, Sardar Patel Institute of Technology, Mumbai, India[b]

ABSTRACT: - A session is a semi-permanent interactive information interchange, between two or more communicating devices, or between a computer and user (e.g. Login session). Web applications communicate using HTTP protocol. HTTP is stateless, which means there is no support at the protocol level to identify the state of a particular request. Sessions are tracked by developers primarily through the use of session identifiers (SIDs).[1] Once the user is successfully authenticated, session id is generated and managed by the web server. From there on, session IDs are used as an authentication token so that user does not have to re-enter the credential information with every request.[2] A session hijacking attack works when it compromises the token by either confiscating or guessing what an authentic token session will be, thus acquiring unauthorized access to the Web server.[2] Various mitigation methods to prevent session hijacking exist but none of them are able to prevent it completely. This paper gives a study of Session Hijacking Attack, its prevention method and future research scope.

Keywords: *Session. Attack, Web, Mitigation, CSRF*

I. INTRODUCTION

A session is a semi-permanent interactive information interchange, between two or more communicating devices, or between a computer and user (e.g. Login session). Web applications communicate using HTTP protocol.[1] HTTP is stateless, which means there is no support at the protocol level to identify the state of a particular request. For instance, let us say a client logged into his Facebook account by sending his credentials. Now if he wishes to see his messages, he has to send his credential information again, because the server doesn't normally know that he was already authenticated in the previous request.[1] This is something that developers have to do themselves. This is called 'Session Tracking'.Sessions are tracked by developers primarily through the use of session identifiers (SIDs).[2] Once the user is successfully authenticated, session id is generated and managed by the web server. From there on, session IDs are used as an authentication token so that user does not have to re-enter the credential information with every request.[2] Session hijacking occurs when a session token is sent to a user browser from the Web server following the successful authentication of a user logon.[2] A session hijacking attack works when it compromises the token by either confiscating or guessing what an authentic token session will be, thus acquiring unauthorized access to the Web server.[2]

Session hijacking deceives a server or a client into accepting the upstream host as the actual legitimate host. Instead the upstream host is an attacker's host that is manipulating the network so the attacker's host appears to be the desired destination.[3]

Session hijacking is a method of taking over a Web user session by surreptitiously obtaining the session ID and masquerading as the authorized user.[3] Once the user's session ID has been accessed (through session prediction), the attacker can masquerade as that user and do anything the user is authorized to do on the network.[3]

Session hijacking is the exploitation of a valid computer session to gain unauthorized access to information or services in a computer system. In particular, it is used to refer to the theft of a magic cookie used to authenticate a user to a remote server.[4] It has particular relevance to web developers, as the HTTP cookies used to maintain a session on many web sites can be easily stolen by an attacker using an intermediary computer or with access to the saved cookies on the victim's computer A popular method is using source-routed IP packets.[3] This allows a hacker at point A on the network to participate in a conversation between B and C by encouraging the IP packets to pass through its machine. If source-routing is turned off, the hacker can use "blind" hijacking, whereby it guesses the responses of the two machines.[3] Thus, the hacker can send a command, but can never see the response. However, a common command would be to set a password allowing access from somewhere else on the net. A hacker can also be "inline" between B and C using a sniffing program to watch the conversation. This is known as a "man-in-the-middle attack".[3]

II. TYPES OF SESSION HIJACKING ATTACK

Predictable token:[1]

1. After user logs in, attacker collects some valid session ID values (by using sniffing tool eg. wireshark)
2. analyse the generation pattern of the token
3. using brute force method predicts token to gain unauthorized access.

Session Fixation :-[3]

1. Attacker sends email to the user with predefined session id and has to convince user to click on that link.
2. user clicks on the login link with predefined session id.
3. web application recognizes attacker's predefined session id.
4. thus, attacker can impersonate as user with the help of predefined session id.

Session Side-jacking :-[3]

1. the attacker monitors the unencrypted traffic for session cookies that uniquely identify the client to the web server.
2. The attacker then steals these values and loads them into their own web browser

Cross-site Scripting:-[2]

1. Attackers send out links with the poisoned queries in them directed at a specific vulnerable site.
2. The malicious code within the query string is reflected back to the victim by the vulnerable server.
3. The victim's browser sees the code is coming from a trusted site, so it runs the code and/or gives it access to its cookies from that same domain.

Cross site Request Forgery:-[4]

1. After logging into a typical website, the website will issue your browser an authentication token within a cookie.
2. request to sends the cookie back to the site to let the site know that you are authorized
3. Suppose you visit a malicious website soon after visiting your bank website.
4. visiting a carefully crafted malicious website (perhaps you clicked on a spam link) could cause a form post to the previous website.
5. Your browser would send the authentication cookie back to that site and appear to be making a request on your behalf, even though you did not intend to do so.

Session Replay Attack:[2]

1. attacker steals user session id.
2. reuses session id to gain unauthorized access

III. ATTACK METHODS

There are four main methods used to perpetrate a session hijack. These are:

a) Session fixation attack is a class of Session Hijacking, which steals the established session between the client and the Web Server after the user logs in. The Session Fixation attack fixes an established session on the victim's browser, so the attack starts before the user logs in. The Fig. 1 explains a simple form, the process of the attack, and the expected results: (1)The attacker has to establish a legitimate connection with the web server which (2) issues a session ID or, the attacker can create a new session with the proposed session ID, then, (3) the attacker has to send a link with the established session ID to the victim, she has to click on the link sent from the attacker accessing the site, (4) the Web Server saw that session was already established and a new one need not to be created, (5) the victim provides his credentials to the Web Server, (6) knowing the session ID, the attacker can access the user's account. [6]

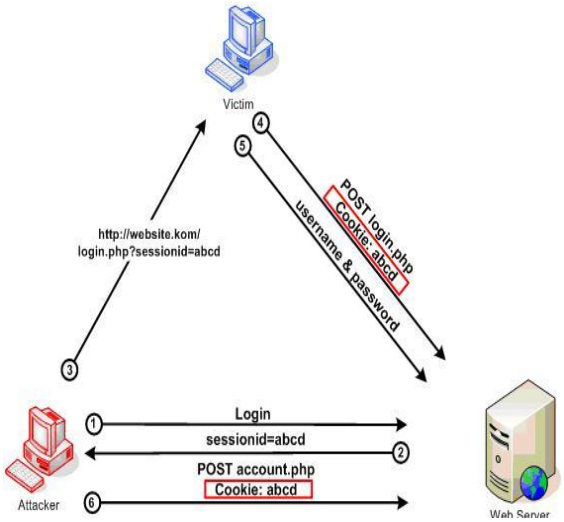

Fig. 1 Example of Session Fixation Attack [6]

b) Session sidejacking, where the attacker uses packet sniffing to read network traffic between two parties to steal the session cookie. Many web sites use SSL encryption for login pages to prevent attackers from seeing the password, but do not use encryption for the rest of the site once authenticated. This allows attackers that can read the network traffic to intercept all the data that is submitted to the server or web pages viewed by the client. Since this data includes the session cookie, it allows him to impersonate the victim, even if the password itself is not compromised. Unsecured Wi-Fi hotspots are particularly vulnerable, as anyone sharing the network will generally be able to read most of the web traffic between other nodes and the access point.[3]

c) Alternatively, an attacker with physical access can simply attempt to steal the session key by, for example, obtaining the file or memory contents of the appropriate part of either the user's computer or the server.[1]

d) Cross-site scripting, where the attacker tricks the user's computer into running code which is treated as trustworthy because it appears to belong to the server, allowing the attacker to obtain a copy of the cookie or perform other operations.[6]

IV. COUNTER MEASURES FOR SESSION HIJACKING

A. Basic Counter Measures

1. Use long and combination of alphanumeric and some special characters (entropy of 64 bit minimum).
2. regenerate session Id when going from secured to unsecured content and vice-versa
3. using session timeouts
4. Use of SSL and other encryption methods/signatures for session id
5. using VPN other than public networks
6. Use of Post method so that URL rewriting fails

B. Advanced Counter Measures For Session Hijacking

1. Session Shield: SessionShield is based on the idea that session identifiers are data that no legitimate client-side script will use and thus should not be available to the scripting languages running in the browser. It uses mechanism of proxy server to mitigate XSS attacks and CSRF attacks. [5]

2. One Time Cookies (OTC): OTC prevents session hijacking attacks by signing each user request with a session secret securely stored in the browser. OTC does not require expensive state synchronization in the web application, making it easily deployable in highly distributed systems. Each OTC token is tied to a particular request by using a Hash-based Message Authentication Code (HMAC); hence, an adversary cannot reuse OTC tokens to illicitly redirect a session.[11]

3. Browser Finger printing: Browser fingerprinting is a technique of identifying an individual user by the unique patterns of information visible whenever a computer visits a website. The information collected is quite comprehensive and often includes the browser type and version, operating system and version, screen resolution, supported fonts, plugins, time zone, language and font preferences, and even hardware configurations. These identifiers may seem generic and not at all personally identifying, yet typically only one

in several million people have exactly the same specifications as you. It uses browser specific tags so that XSS attack can be mitigated. [10]

4. CSRF Gaurd: A library that implements a variant of the synchronizer token pattern to mitigate the risk of Cross-Site Request Forgery (CSRF) attacks. [6]

5. Referrer Header: This is an optional header field allows the client to specify, for the server's benefit, the address of the document (or element within the document) from which the URI in the request was obtained. This allows a server to generate lists of back-links to documents, for interest, logging, etc. It allows bad links to be traced for maintenance. If a partial URI is given, then it should be parsed relative to the URI of the object of the request.[8]

6. Origin Header: The Origin header is added by the user agent to describe the security contexts that caused the user agent to initiate an HTTP request. HTTP servers can use the Origin header to mitigate against Cross-Site Request Forgery (CSRF) vulnerabilities. It is used to track the origin of the message or data sent over the network so that CSRF attacks can be mitigated. [9]

C. Mitigation Method

Methods to prevent session hijacking include:

a) The session key (encrypted data traffic) passed between the parties, though ideally all traffic for the entire session by using SSL/TLS. Web-based banks and other e-commerce services widely relied-upon this technique, because it completely prevents sniffing-style attacks. However, other kind of session hijack is still possible to perform.

b) Recommended to use long random number or string as the session key. This will reduce the risk that an attacker could simply guess a valid session key through trial and error or brute The Origin header is added by the user agent to describe the security contexts that caused the user agent to initiate an HTTP request. HTTP servers can use the Origin header to mitigate against Cross-Site Request Forgery (CSRF) vulnerabilitiesforce attacks.

c) The session id after a successful login can be regenerated. At this point the attacker does not know the session id of the user after s/he has logged in, thus preventing session fixation attack.

d) The identity of the user is given a secondary check through some services. For example, a web server could check with each request made that the IP address of the user matched the one last used during that session. This does not prevent attacks by somebody who shares the same IP address, however, and could be frustrating for users whose IP address is liable to change during a browsing session.

e) Some services change the value of the cookie with each and every request. Due to which the window in which an attacker operate and makes it easy to identify whether an attack has taken place get reduce, but can cause other technical problems (for example, two legitimate, closely timed requests from the same client can lead to a token check error on the server). [11]

f) Users may also wish to log out of websites whenever they are finished using them. However this will not protect against attacks such as Firesheep.

g) Virtual Private Networks endpoints can be made to configure authentication and integrity check.

V. CONCLUSION

As it is observed that none of the migitation method are able to prevent session hijacking attack completely, thus there are lots of future scope for research in these area. One of the research can be to propose a new methodology to mitigate more than one attack through single countermeasures.

REFERENCES

[1] Damon Reed "Applying the OSI Seven Layer Network Model To Information Security", By, © *SANS GIAC GSEC*, November 21 2003.

[2] Gurvinder Kaur Pannu, "A Survey on Web Application Attacks", (*IJCSIT) International Journal of Computer Science and Information Technologies,* Vol. 5 (3) , 2014 PP. 4162-4166

[3] "Security in Session Hijacking", http://itsecurity.telelink.com/session-hijacking

[4] Robert Auger, "Cross Site Request Forgery", http://www.cgisecurity.com/csrf-faq.html/, v1.62 (Last Modified: 28/4/10), [accessed: 10/09/2015].

[5] Nick Nikiforakis, Wannes Meert, Yves Younan, Martin Johns, Wouter Joose, "SessionShield: Lightweight Protection against Session Hijacking", *Chapter: Engineering Secure Software and Systems. Springer Berlin Heidelberg*, 2011 PP. 87-100.

[6] https://www.owasp.org/index.php/ XSS_ (Cross_Site_Scripting)_Prevention_Cheat_Sheet/

[7] Boyan Chen, Pavol Zavarsky, Ron Ruhl and Dale Lindskog, "A Study of the Effectiveness of CSRF Guard", *IEEE Inter national Conference on Privacy, Security, Risk, and Trust, and IEEE International Conference on Social Computing*, 2011 PP. 1269-1272.

[8] Mohd. Shadab Siddiqui and Deepanker Verma,"Cross Site Request Forgery:A common web application weakness", IEEE Conference and white paper, 2011

[9] Tandel, Nikunj, and Kalpesh Patel, "Mitigation of CSRF Attack.", International Journal of Science and Research (IJSR), ISSN (Online): 2319-7064

[10] Thomas Unger , Martin Mulazzani , Dominik Fr¨uhwirt , "SHPF: Enhancing HTTP (S) Session Security with Browser Fingerprinting", IEEE International Conference on Privacy, Security, Risk, and Trust, and IEEE International Conference on Social Computing, 2011

[11] Italo Dacosta, Saurabh Chakradeo, Mustaque Ahamad and Patrick Traynor, "One-Time Cookies: Preventing Session Hijacking Attacks with Stateless Authentication Tokens" *ACM Transactions on Internet Technology (TOIT)*, Volume 12 Issue 1, June 2012.

[12] Sentamilselvan. K Lakshmana Pandian. S Ramkumar. N, "Cross Site Request Forgery: Preventive Measures", International Journal of Computer Applications, Volume 106–No.11, November 2014 PP. 0975 – 8887.