Secure Session Management using Cookie for Mobile Agent

Aradhana[1], Samarendra Mohan Ghosh[2]
Department of Computer Science Engineering
Dr. C. V. Raman University Bilaspur, C.G

Abstract- Internet based mobile agent system often use cookies to maintain the authentication states between agent to cloud. Every subsequent request will generate a cookie that will automatically be allowed by internet based application on agent platform. The attackers easily identify the user details using cookies. Cross site scripting (XXS) is a common attack technique that theft cookies related to subsequent request from mobile agent or web browser's databases .In this paper, we proposed a new method for cookie rewriting that encrypts the cookies and protect from Cross Site Scripting attack. This method is implemented for mobile agents it will be automatically rewrite the cookies therefore it protect against XXS attack.

Keyword- Agent, XXS attack, Cookie, Offloading, Encryption, Decryption, Hash

I. INTRODUCTION

Mobile agents are the mobile code on cloud environment. The concept of mobile agent is the dynamic installation of code on a remote host. Complexity of execution demands resources. Agent system is simple approach to protect the computation environment. Agents are the dedicated lines for home platform normally it is more trusted environment. An agent platform may support multiple computation environments where agent can interact with cloud. Mobile Agent has some hidden security problem in these phases:-

- **Transferring Phase:** The agent transfer from host to another platform.
- **Running Phase:** The agent's code in another platform during computational offloading.

This approach of mobile agent is new and attractive interaction of computation, it arises significant challenges in terms of data and code security and their cookies security during the offloading. Executing data and code during offloading with secure session management requires understanding a few basic concept of cookies like their attributes value their fields, path, domain and name and also requires to developers to understand how to make it confidential, and how real world attackers are miss using he weak session management in real application today. A mobile agent contains:

- Code – It defines the agent's behaviour
- State – It stores the agent's internal state which enables it to resume its activities after moving to another host.
- Attributes - information describing the agent, its origin and owner, its movement history, resource requirements, authentication keys, for use by the infrastructure. Part of this may be accessible to the agent itself, but the agent must not be able to modify the attributes [2].

II. COOKIE REVIEW

Most web application frameworks use client-side cookies to index a state table on the server side. Session state is usually represented with a special-purpose object type, stored on the server, and could contain anything relevant to the application like user profile, user privileges, cached data from a back-end store, browsing history and page flow state.

Many times application was to be stateless in which server did not store any session state. These are the special security factors which develops make a stateless session management system. These issues depend on application's requirement and implementation. Consider these potential issues:

- The cookie's confidentiality is protected from attackers. They can read the session state. This could leak the confidential issues.
- The cookie's integrity is protected from attackers. They could change the information leakage issues.
- Cookies has limited size, decreasing the amount of storage available for session state.

III. STRUCTURE OF COOKIES

Cookies are simple text files that store the information about client's detail. It maintains the state of application. Cookies are used for authentication, storing website information / preferences, other browsing information and anything else that can help the Web browser while accessing Web servers [3]. Cookies are very sensitive in nature. More than one cookie will be a set upon subsequent request. It would be encoded and encrypted in an attempt to protect the information of cookie from attacks

For example, in the case of an online purchasing, user visits many items so they add multiple items to the shopping cart. Additionally, there will typically be a cookie for authentication (session token as indicated above) once the user logs in, and multiple other cookies used to identify the items the user wishes to purchase and their auxiliary information (i.e., price and quantity) in the online store type of application[4].

The main attributes name, domain, path and value .Other attributes of cookies are shown in figure 1.

- Name: It signifies the name of cookie.

- Domain: This attribute signifies the domain for which the cookie is valid and can be submitted with every request for this domain or its sub-domains
- Path: This attribute signifies the URL or path for which the cookie is valid.
- Value: It signifies the Id generated by network
- Secure: Cookies may be theft by sniffing. It will be solved by encrypting the cookie's detail before sending the network.
- HTTPOnly: It signifies the accessibility control of cookies means it will be allow to access or not.
- Expires: This attribute is used to determine the time and date when the browser will delete the cookie [5].

Fig.1: Attributes of cookies

IV. THREATS RELATED TO COOKIES

1) Sniffing Network Traffic for Cookies: In this attack the malicious user hacks the unencrypted data. There are many software available for hacking the cookies like wireshark, kismet, Cain and able, commView, Microsoft network monitor [6].
2) Cross-Site Scripting Attack: In this type of attack an attacker steals cookie information by making user clicking on a link that contains a malicious script. This script code reads cookie information and sends this information to the attacker by mail [7].
3) Cross-site request forgery (CSRF) Attack: In this type of attack an attacker forces a logged in user to perform an important action without his consent or knowledge. This attack can also be used to modify firewall settings, posting unauthorized data or even to conduct fraudulent financial transactions [8] [9].
4) Session Fixation Attack: Session fixation attacks exploit the vulnerability of a system which allows one person to find another person's session identifier [5].

V. PRIVACY CONSIDERATIONS IN AGENT

Cookies are often criticized for letting servers track users. For example most of web based company has used cookie to identifying the user detail to track their domain or sub domain across agent sessions can be shared with different hosts [1].

A. Third-Party Cookies

In cloud, agent uses more than one severs .The third party server track the user domain even if user never visit the server directly. Some mobile agent blocks third party cookies. These policies are sometimes very ineffective.

B. User Controls

Some agents provide users with the ability to approve individual writes to the cookie store. Agent provides many facilities to user to manage and disabling the cookies session.

C. Expiration Dates

For the user privacy agent can select reasonable cookie expiration periods based on the purpose of the cookie.

VI. PROPOSED METHODOLOGY

Proposed methodology carried out from existing technique where we create dynamic hash value of four fields which randomizes all four original values in to the dynamic hash value. The creation of four dynamic hash values will deviate the hackers who want to access the cookie value. Hacker will not be able to understand the original cookie name as well as its original value. In this technique all four fields along with their original values will be randomize before send to the client memory. Further at request time of the client the hash values will be taken and convert all hash values in the original form and send to the server.

If unauthorized person has accessed the fields from client memory and tries to identify the original value then he has to first identify the name of all the fields and then their respective value that would be so tedious and would be time consuming therefore the generation of hash values of four fields will be more secure as compare to single field hash generation. The process of cookies encryption and decryption is shown in figure 2.

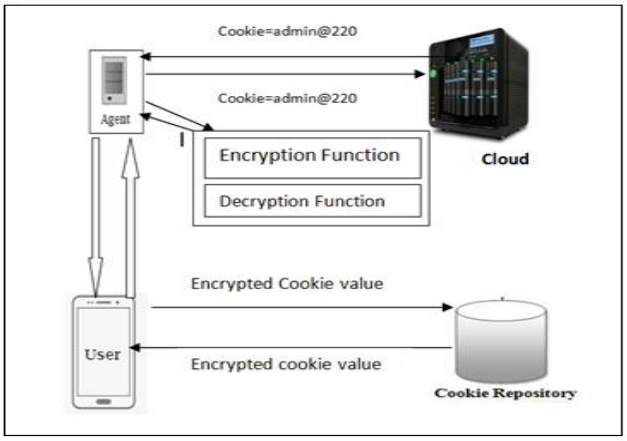

Fig.2: Agent sends to Hash value of cookie to User

Following step should be follow to encrypt the cookie:

1. Retrieve cookie values.
2. Extract Fields and respective values.
3. Apply random method to encrypt original value.

a. Get the original value
b. Use random method to randomize the asscii value of character.
c. Val=val*random(val)+counter++
d. Store the value in session object for future use
e. Rewrite the field

4. Apply hash function to change the position of field.

a. Retrieve the field position
b. Change the position using toggling bit value function
c. register = (register & ~bitmask) | value;
d. Rewrite position

5. Rewrite the encrypted field into cookies.

Here are the original values of cookie sent by server as shown in table 1:

Table 1: Original value of Cookie

Original Field	Name	Domain	Path	Value
Original Value	Cookie 111	www.paymentdesk.com	./Root	admin@220

Further the agent will convert the original value into hash value and send these values to the client machine. As the browser stores the hash value of cookies, so even the XSS attack can steal the cookies from browser‘s database, the cookies cannot be used later to hijack or take off the user‘s session. Cookies will be secured as shown in Table 2.

Table 2: Hash Value of Cookie

Hash Field	$%^^&	&^%^^&^	%^&%$&*	#$%**^^
Hash Value	$%^745	#$%@#3774$%	%^&1212	&&^#$%^%$

Following step should be follow to decrypt the cookie:

1. Retrieve cookie values.
2. Extract Fields and respective values.
3. Apply random method to decrypt original value.
4. Apply hash function to find the actual position of field.
5. Rewrite the decrypted field into cookies.

After randomizing the value of the cookie it is send to the browser’s memory. At the requesting time agent will access the hash value which is stored in the browser’s memory and convert it’s into original format and sends again to server

VII. RESULT ANALYSIS AND STRENGTH

Result of this approach will be improved the security of the cookies at agent. It improves the speed of the transportation of the cookies. If attacker found the cookie of the browser and tries to decrypt the hash values of four fields then he has to identify the sequence of the four fields as well as all corresponding values. This process will take time till then user will be finished his work and will be session out so hacker has not much time to decrypt the cookies value. In the hash generation we use four fields, which are as follows:

- Server sends new session cookie to the victim and agent will generate its random value using random function before sending to victim’s browser.
- After generating in hash format agent will send all random values to the victim’s browser.
- Random value will be stored by the computer Memory of the victim.

Strengths

- ✓ The proposed method described above does not affect the performance of agent & victim’s browser.
- ✓ This method is used by agent therefore it is too secure with respect to victim’s browser.
- ✓ Even if attacker will perform the XSS attack to steal the cookies from browser’s machine, the attacker will get the hash version of the cookies which are not appropriate to impersonate the users.

VIII. CONCLUSION

We have applied a cryptographic hash approach to encrypt cookies on agent that protects the user’s authentication information (encrypts four fields name, domain, path & value). Hash values of four fields will be more secure as compare to single field hash generation .It improves the cookie’s strength. Hacker will not be able to understand the original cookie name as well as its original value therefore we will manage secure session.

IX. REFERENCES

[1]. Singh R. and Kumar S. ,” A Study of Cookies and Threats to Cookies , International Journal of Advanced Research in Computer Science and Software Engineering Research, Volume 6, Issue 3, March 2016 ISSN: 2277 128X.
[2]. https://www.w3.org/Conferences/WWW4/Papers/150/
[3]. https://www.techopedia.com/definition/7624/cookie
[4]. https://www.owasp.org/index.php/Testing_for_cookies_attributes_(OTG-SESS-002)
[5]. https://tools.ietf.org/id/draft-ietf-httpstate-cookie-23.html#rfc.section.4.2
[6]. https://en.wikipedia.org/wiki/Packet_analyzer
[7]. http://www.paladion.net/cross-site-scripting-attacks/
[8]. https://www.tinfoilsecurity.com/blog/what-is-cross- site-request-forgery-csrf
[9]. http://searchsoftwarequality.techtarget.com/definition/cross-site-request-forgery
[10]. https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
[11]. http://www.acunetix.com/websitesecurity/xss/
[12]. http://excess-xss.com/
[13]. http://www.acunetix.com/blog/articles/preventing-xss-attacks/
[14]. http://resources.infosecinstitute.com/how-to-prevent-cross-site-scripting-attacks/
[15]. https://en.wikipedia.org/wiki/Cross-site_request_forgery