Root Cause Analysis of Broken Authentication and Session Management

Jigar Patel[1], Prof. Chandresh Parekh[2]
[1]*Student, M.Tech, School of Information Technology & Cyber Security, Raksha Shakti University*
[2]*Dean, School of Information Technology & Cyber Security, Raksha Shakti University*

Abstract- **While there are many ways to protect web applications as one of the most common ways to harness the power of the Internet, attackers almost daily come up with new attempts to exploit various vulnerabilities and undermine the information found on the net. One of the possible areas for finding sustainable solutions is to follow strategic approaches based on more detailed analysis and understanding of problems than with some common and practical approaches. The purpose of the paper is to derive the function of cause analysis (RCA) in session management and the weaknesses of the validity of how it is used and how it is developed with specific security features of web applications. Using RCA, we were able to identify the specific causes of uncontrolled session control and the specific causes of the authentication**

Index terms- **Cyber Security; Web Application Vulnerabilities; Exploitation Techniques; Broken Authentication; Session Management**

I.INTRODUCTION

Authentication and session management include all aspects of true user management and effective time management. This includes user authentication management mostly done by username and password and manage that session after authentication is verified. Error handling both of these items may result in stealing user or administrative accounts, undermining authorization and accountability, and resulting in breach of confidentiality. Due to accessibility and session management many attacks can be performed by the attacker. I will be describing given two types of attacks:[a]brute force attack.[b]session hijacking.[c]replay attack.[d]session fixation.[e]session timeout.

II. BROKEN AUTHENTICATION AND SESSION MANAGEMENT

Broken authentication is a type of web vulnerability that occurs due to random session management. After completing the authentication process, A session will be created that will be used to communicate data between the server and the specific user. If any teenager can gain access to the active status of any user that goes through the authentication process, then that situation is treated as a breach of Authentication broken application issue. The session request is suggested by the web app user via the login page where the user authentication is provided. When a given request is sent from the client side to the server side, the server launches a query to the test data to determine whether the given user authentication is matched to the data record. As soon as the verification process is successful, a session with a specific ID will be assigned to enable the user to contact the request. The user can access the program with the privileges granted to the system administrator for accessing the unique resources. A valid session is valid for a period specified by the system builder. Browsers store user information in the authentication cookie for the duration of the session when the session expires by sending authentication information to the server component. This process is done automatically after a user interface that will reduce the user's attempt to verify that they are duplicated. However, an adversary may seize and gain access to another applicable period of time through different applications, such as, cookie manager, ate my cookie, advanced cookie manager, etc.

III. EXPLOITATION TECHNIQUE

[a]. Brute Force Attack

Brute Force attacks are a very bad option for those with bad server access. The basis of this type of attack is to eventually have root access to serve any purpose the attacker may have. To do this, an attacker often uses software trying to guess a password with multiple failed login attempts. If there is no security in place, these failed login attempts can continue until they are properly guessed. Finding a website is at risk of a severe attack. We can easily check whether the site is at risk. It can be viewed manually. To check the site's vulnerability, simply go to the site's login page and also a specific username and try different passwords. If the site does not ask for additional information and allows you to try an additional password this is a result of the fact that this website is at risk of being attacked.

[b]. Session Hijacking
According to computer science, hijacking is the exploitation of a valid computer session is sometimes called session key to gain unauthorized access to information or services on the computer system. In particular, it is used to refer to stealing a magic cookie that is used to authenticate a user on a remote server. It directly affects web developers, since http cookies are used to save time on many websites that can easily be stolen by an attacker using a counselor computer or access cookies stored on the victim's computer.

[c]. Replay Attack
Duplicate attacks are a type of network attack where the valid data transfer is malicious or duplicated. This is done by the inventor or by the adversary who divides the information and returns it, perhaps as part of a secret attack by replacing a packet-like stream attack.

[d]. Session Fixation
The timing correction attacks are trying to exploit a system vulnerability that allows one person to set another person's time identifier. Most session correction attacks are web-based, and most rely on the time identifier accepted in URLs (query line) or postal data.

[e]. Session Timeout
The output structure specifies the timeout given in the application session object, in minutes. If a user does not refresh or request a page within the expiry time, the time expires.

IV. DATA COLLECTION PROCEDURE

A hands-on approach was used following the double blind strategy of conducting the Study. Small sample techniques were also selected to determine the sample size of this study. Various manipulation methods for authentication of the request where the session control problem was used in this study. To demonstrate the feasibility (i.e. broken authenticity and weaknesses of temporary control) the use of the public and private sectors, is searched at http://www.google.com/ using several operators. The most effective of our tests i.e
Inurl:apanel/admin/index.php,
inurl:news.php?id=,Inurl:gallery.php?id=,
inurl:article.php?id=, and
Inurl:event.php?id=.
After accessing our sensitive web applications, it will be exploited using the unique authentication method and session management exploitation described above and find that the vulnerability types are provided in this application. The level of access after abuse has been identified in this study.
Environment & ToolsHydra tool has been used for brute force attack to cracking the weak password. Mozilla Firefox version 54.0 has been used to stop java script re-direction. No redirection plugins version 1.3.2.13.1 and is switch 0.2.10.1 add-ons have also been used.

V. PREVENTION

Basic guidelines to manage the session are provided below for preventing the given types of exploitation. It is to be noted that all the solution examples are given in php code.

[a]. Session id life cycle
Session ids can be generated in two types i.e. permissive And strict. The default method initially accepts any amount of id time set by the user to create a new session. In contrast, the robust process forces the web application to accept the values of the time ids generated by the program. If web applications are unsure and filter incorrect id values before processing, then an attacker may be used to exploit other web vulnerabilities. The session id must be renewed or renewed even if the same user is upgrading / downgrading their users.

[b]. Session reset

A defined session is created during an authorized user session you are logged in. The system saves authentication cookies for user authentication during their active session. These Session cookies must be reset after the user exits the system to ensure privacy. After the user logs out, the module should complete the following types of code

```
.If (isset($_request['logout']))
{
Unset($_session[logout])
}
```

[c]. Session expiration

Sessions hijacking is one of the types of attack by which an attacker can exploit over an active session. Therefore, it is necessary for the developer of the web application to set expiration timeouts for every session to prevent sessions hijacking attacks. The developer should also ensure the mechanism to keep the session active as long as the valid user remains in work. Irregular session Expiration increases different types of session-based attacks as the attacker could reuse the valid session ids and also can hijack the active associated sessions. Example of cookie expiration is shown as below:

Set-cookie: id=; expires=friday,-15- july-19.

[d]. Cookies

Cookies based session id exchange mechanism ensures Numerous security properties in the form of cookie attributes which it can be used to safeguard the exchange of the session id.

[e]. Session attacks detection

When an attacker tries to guess/ brute force a valid session id or analyze the predictability of the session id using statistical analysis, multiple sequential requests against the target web application has to be launched using different session ids from a single or multiple ip addresses. Web application's firewall has to have the capability to detect the above scenario based on the number of attempts that the system observed from different session ids. Alert to the Administrator has to be ensured and block those offending ip addresses by analyzing the payload.

[f]. Client-side defenses for session management

Web application's session maintenance technique using Java-script validation for client site protection is a regular way to make it safe from general users. Although it is not enough for defending any skilled intruder, but it may generate another layer of security. Attacker can bypass this client site protection using some advance tool (e.g. burp suite and techniques. Therefore, the server side security needs to be address properly. The confidential pages must use the defined system session strictly for being secure from unauthorized access. Proper session maintenance is the main key point of reducing broken authentication vulnerability. Insecure sessions are generally compromised by the attackers for interrupting in General session mechanism. In this case, developers need to meet some initiatives which are described below for proper management of sessions.

I) predefined session period:

Session should be started with the proper validation of user's credentials i.e. username and password. The session cookie will be used to authenticate the user continually as long as the user stays active in the system. If the user found without any activity for a certain period, the session will be destroyed automatically by the system. Sample of the automatic session destroy code is given below:

```
If (isset($_session['activity']) && (time() -
$_session['activity'] > 1200)) {
// here previous request was 20 min ago
Session_unset(); //session_destroy(); // destroying
active sessions} $_session['activity'] = time(); // now
updating last activity
```

from the above code, it is observed that the system will destroyed the active session once it finds the user inactive for 1200 seconds.

II) destroy old sessions:

The system should not allow long duration session without proper authentication for ensuring users validity. The following types of code may help the developer to prevent session based attack.

```
If(!empty($_session['deleted_time'])&&$_session['de
leted_time'] < time() - 180) {Session_destroy(); //
delete the old sessions
```

III) set cache limitation as private:

The cache expiration is reset to the default value of 180 stored in the function of session.cache expire during request startup time. Thus, the developer should ensure to call the function, session_cache_expire() for every request to define Every cache limit as private. Example of sample solution code is given below:

```
/* set the cache limiter to 'private' */
Session_cache_limiter('private');
$cache_limiter = session_cache_limiter();
```

[g]. Generating an access token:

Use of access token for entering into any active session is now very popular for web applications. When a user requests for creating a new session after completing the authentication process, the system generates an access token randomly to Validate the user. Users have to enter the given token code with their credential to get access into their session. Since the token code are generated randomly for a limited time period, an attacker cannot hijack the user's sessions using brute-force Technique even if the attacker discovers the correct user credential.

VI. CONCLUSION

Virtually all web applications maintain a user profile separately to ensure quality and communication services to their user. Authentication and temporary management problem are one of the biggest obstacles to ensuring web app privacy. Therefore, the above weaknesses have been listed as the weakest web application since 2007 and are now ranked 2nd in the open web application security project (OWASP). It has been observed after conducting this experiment that the presence of a validation problem and a session management problem are found mostly in educational institutions and on the government's website. It is also revealed in this case that improper session-time attacks and fragmentation / guessing of weak passwords are the most effective ways to exploit the broken authenticity and risk of web application control over those domains. This study has identified five forms of exploitation and has been explored on Bangistani websites. Our perception that the risk of exploitation discussed will be reduced if the developer follows the security measures described in this paper. In the future, we intend to work on other forms of exploitation and investigate other websites in terms of gaining the authenticity of failures and session management.

ACKNOWLEDGMENT

I am Thankful To Mr. Chandresh Parekh For Guiding in My Research Work.

REFERENCES

[1] "world internet users statistics and 2017 world population stats", internetworldstats.com,2017.[online].available:http://www.internetworldstats.com/stats.htm. [accessed: 31- oct- 2017].

[2] "usage statistics and market share of server-side programming languages for websites, november 2017", w3techs.com, 2017. [online].available:https://w3techs.com/technologies/overview/programming_language/all. [accessed: 17- july-2017].

[3] J. Thome, l. K. Shar, d. Bianculli, and l. Briand, "security slicing for auditing common injection vulnerabilities," 2017, journal of systems and software.,to be published.

[4] I. Hydara, a. B. M. Sultan, h. Zulzalil, and n. Admodisastro, " current state of research on cross-site scripting (xss)–a systematic literature review, " 2015 information and software technology, pp. 170-186.

[5] A. Z. M. Saleh, n. A. Rozali, a. G. Buja, k. A. Jalil, f. H. M. Ali and t. F. A. Rahman, "a method for web application vulnerabilities detection by using boyer-moore string matching algorithm," 2015 procedia computer science, pp.112-121.8

[6] A. Begum, m. M. Hassan, t. Bhuiyan and m. H. Sharif, "rfi and sqli based local file inclusion vulnerabilities in web applications of bangladesh," 2016 international workshop on computational intelligence (iwci), dhaka, 2016, pp. 21-25.

[7] N. Nikiforakis, l. Invernizzi, a. Kapravelos, s. Van acker, w. Joosen,c. Kruegel, f. Piessens & g. Vigna, "you are what you include: largescale evaluation of remote javascript inclusions, " 2012 in proc. Of acm conf. On computer and communications security., pp. 736-747

[8] M. I. Ahmed, m. M. Hassan, t. Bhuyian, "local file disclosure vulnerability: a case study on the web applications of public sector, 10th international conference on computer and electrical engineering (iccee 2017) ", edmonton, canada, october 2017,11-13,

[9] Zone-h.org, 2017. [online]. Available: http://zone-h.org/?zh=1.[accessed: 11- aug-2017].

[10] P. V. Ami and s. C. Malavy, "top five dangerous security risks over web application" 2013 international journal of emerging trends & technology in computer science, 2(1), 41-43.

[11] T.petsios, v. P.kemerlis, m.polychronakis and a. D.keromytis, "dynaguard: armoring canary-based protections against brute-force attacks, " in proc. 31st annu. Computer security applications conference, 2015, pp. 351-360.

[12] N.kaaniche and m.laurent, "data security and privacy preservation in cloud storage environments based on cryptographic mechanisms," 2017 computer communications, pp.120-141.

[13] R. Johari and p. Sharma, "a survey on web application vulnerabilities (sqlia, xss) exploitation and security engine for sql injection," 2012 international conference on communication systems and network technologies, 2012, rajkot, pp. 453-458.h.

[14] A.torkamanatashzar, m. Bahrololum and m. H. Tadayon, "a survey on web application vulnerabilities and countermeasures,"6th international conf. On computer sciences and convergence information technology (iccit),seogwipo, 2011, pp. 647-652.

[15] G.deepa and p. S.thilagam, "securing web applications from injection and logic vulnerabilities: approaches and challenges, " 2016 information and software technology, 74, 160-180.

[16] B. Rexha, a. Halili, k. Rrmoku and d. Imeraj, "impact of secure programming on web application vulnerabilities," 2015 ieee international conference on computer graphics, vision and information security (cgvis), bhubaneswar, 2015, pp. 61-66.