ISSN No. 0976-5697

Volume 5, No. 3, March-April 2014

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Implement Secure Authentication Mechanisms in Web Applications

Dr. Anil Kumar
Professor
Department of Computer Science
Manipal University
Jaipur, India

Mr. Krishna Reddy
M.Tech Scholar, Information Security
Department of Computer Science
Manipal University
Jaipur, India

Abstract:With the advent of World Wide Web, information sharing through internet increased drastically. So web applications security is today's most significant battlefield between attackers and resources of web service. It is likely to remain so for the foreseeable future. By considering recent attacks it has been found that major attacks in Web Applications have been carried out even system having authentication mechanisms. Malicious users getting access into systems, reasons may be anything but getting third party access into systems shell violet organization policies. Authenticating an object means confirming its provenance to service, whereas authenticating a person often consists of verifying their identity. Depends on application authentication scheme will implement one or more authentication factors. In computer security, authentication is the process of attempting to verify the digital identity of user to server for getting service, in this process server don't knows who requesting service, irrespective of identification if server provides service then possibility to getting access by unauthorized users. Mainly these vulnerable authentication applications lead to security risks.

Keywords:Security; authentication; service; identification; vulnerability;

I. INTRODUCTION

Constructing secure application is very difficult [1], in terms of complexity. More over there is no measures for security, providing security means to keep avoiding attacking patterns. When industry moving towards electronic communication, web service palace major role for information interchange. Mainly web applications serves public information, up to some extend attack patterns having less impact in web service, when service starts to store and transfer confidential in information through internet, attack patterns are involving in between normal communication, such activities spoils user or service present states. In initial stages security handles with antivirus, then in network level, now security threats more in application level, due to lack of secure code. So vulnerability is defined as weakness in system or future of system that males easy to exploit. Vulnerability might be exists at the host, network or application levels.

Many application especially web based application faces risks and those applications will cause to violate policies that are maintained in application. Web applications works in the principle "web server accepting user request and process it, again gives proper acknowledgments" this process enough to implement communication channel, and in case of vulnerable applications possible to compromise applications or service. Until there is no attack patterns application serves communication, whenever application moving to words business application deployment authentication schemes comes up and attack patterns will active. In this stage in some vulnerable web applications can possible to access by malicious user due to lack of proper authentication mechanisms. When organization moves to automate business with web service via internet attack surfaces comes in front, due to lack of security auditing. Web application authentication process is describes as follows.

Application level attacks place interesting role in web applications, which causes to financial lose, and creates serious reputation on service. Majorly application level attacks happen because of authentication leaks [2], digital information that requesting by user in application surface, service have to configure to process user requests in proper way, to keep out attack patterns. In application level vulnerability identification is absolutely difficult; strong authentication mechanism is the only way to keep application secure from application attacks.

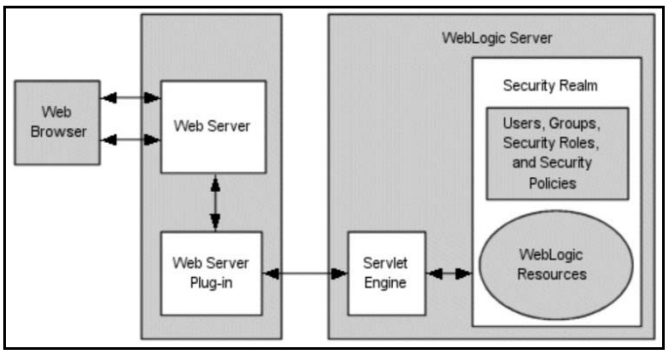

Figure 1: Typical Web Application Authentication process

More over avoiding authentication flaws after deployment is very difficult. Authentication vulnerabilities basically belong to application level attack, through this possibility to happen other attacks like denial of service. In web applications major vulnerabilities like Brute force, Authentication bypassing, Cache, and multiple factors authentication are commonly found in complex business applications, unfortunately commercial applications also not so far from application level attacks. Also keep on securing application, it will cause to arises complexity issues, and possibly launch new vulnerabilities.

II. AUTHENTICATION VULNERABILITIES IN WEB APPLICATIONS

In web applications Authentication is the process of proving them self they are authorized users to access service. So authentication is the process of sending user

request via web browser and process user request by server, if server getting true values then service will allocate session to user, then user can able to enter into private state on service. In this process user session is secure until no one can able to enter into private state, but there is possible to authentication flaws in web application, those describes as follows.

A. Information Transprt over Encrypted Channel:

When application sending sensitive information, web applications should takes the appropriate security measures by using a protocol like HTTPS, because HTTPS protocol is built on TLS/SSL mechanism. Data passing through HTTP protocol means it is non-secure way because it works on application layer, whereas using HTTPS is secure because it works on transport layer [3]. Else data can able to sniff while data is being transmitted. Suppose,

GET http://www.example.com:443/login.php

Here credentials pass through get method and HTTP which means whole information sends in URL, possible to sniff information while being transmitted.

POST https://www.example.com:443/login.php

In this request protocol used here is https and method is post, which means not that much easy to sniff.

B. Guessable User Accounts and Passwords:

Some users, gives user accounts for guessable accounts and passwords which are easily known by others. Guessable might be dictionary words, easily known to others. Some applications leak information as to the validity of usernames during either authentication attempts, like password not match then user name is correct. Issues describes as follows,

Username: admin

Password: admin

In such case if application having authentication like guessable strings then it is easy to enter into private state.

C. Bruteforce:

Brute forcing consists of systematically enumerating all possible combinations to enter into system. In case password checking is automated then it will causes to crash application because of DOS attacks, which leads to loss of availability. If a dictionary type attack fails, then possible to attempt use brute force methods to gain authentication. Brute force can easily crack with numeric and alphabet combinations [4]. Loss for attack patterns are extend to, crack directories, session id cookies, usernames and passwords, table length, number of accounts.

It mainly depends on error messages, in application request if service display as follows, wrong password, it means username is existed, then attackers have to find password, it is possible to check all possible combinations.

D. Bypassing Authentication Schema:

In web applications it is essential to require authentication for gaining access to private state or to get write permissions. In case of poorly configured mechanism it is possible to access private state. An authentication bypass attack targets files that are in use by the protected application [5]. Where attacker looks to the unprotected files for information about system and formulates a strategy to bypass the authentication. Where mainly possibility to getting root privileges, and attempt to access administrative environment. In some cases root folder also contains database connection scripts or may having sensitive information. Attempt to bypass the authentication schema able to access these resources without authentication describes as follows,

www.example.com/login.php

This is environment to authenticate user details, where poorly configured applications possible to skipping this page and gives URL like this,

example.com/admin/org/account/upload.php

Such case directly bypass for authentication mechanism.

E. Remember & Reset Password:

Browsers have capability to save user passwords, in such cases if user closes all their sessions even user passwords being saved into local machines. This will cause to enter into user private states without proper authentication mechanism. Application has to maintain at least following to prevent remember password mechanism,

<INPUT TYPE="password" AUTOCOMPLETE="off">

Reset password: user requesting to reset password, service can able to provide mechanism with following cases,

a. Set new password in same browser
b. Authenticate old password to giving new password
c. One time password mechanism
d. Sending reset link to personal mail address
e. Authenticate with third party

Among these cases choose better service, depends on service.

F. Logout and Browser CACHE Management:

Implement proper log out mechanisms to close use sessions otherwise it is difficult to maintain application private state. Basically user log information, user details are saved in user machine with help of cache management. Depends on application service have to kill the cookies, with particular time span.

G. Multiple Factor Authentication:

It is critical task to implement in all applications; mainly transaction oriented like banking and financial applications uses this service. It means there is more than one authentication scheme to enter into user private state.

This will prevents application attacks from Phishing, Brute Forcing, Trojan, Malwares, Password reuses session reading and Session fixation. These services have more complexity to implement, so it is not necessary in informational websites. Authentication page originally sets a cookie in the following way,

Set-Cookie: SessionID=sjdhqwoy938eh1q; expires=Sun, 09-Feb-2014 12:20:00 GMT; path=/; domain=example.com

H. Captcha:

Aim to implement to make clarification between automated services and humans. And sometimes poorly configured CAPTCHA services leads to compromise web applications. In case of easily broken captcha it is easy to trace image, leads to violate security policies. Some times captcha vulnerabilities causes to access private state.

I. Race Conditions:

Application produces unexpected result when timing of actions impact other actions. When working with shared data, whether in the form of files, databases, network connections, shared memory, or inter process

communication, there are a number of possibilities to made mistakes that can compromise security. For example if there is one request to write file then there is no conflict, if situation like multiple request to write same file then possibility to arise race condition [6].

Race conditions may occur when a process is critically or unexpectedly dependent on the sequence or timings of other events. In a web application environment, where multiple requests can be processed at a given time, developers may leave concurrency to be handled by the framework, server, or programming language.

III. IMPACT OF AUTHENTICATION VULNERABILITIES

Until applications use for constructive purpose there is problems occurs from third party, when application using for transaction oriented, it will attract attacking patterns, reasons maybe anything but application gets loss because of attack patterns. Impact of authentication flaws will describes as follows,

- ***a. Encrypted Channel:***Usually data that passing from web browser, depends on service information is encrypt with help of browser. In case of poorly configured service it is possible to sniff sending information which causes to big leak of confidential information. In case of sensitive information like passwords, card numbers such details, its leak of transactional information, if it is happen then there is no words for security, because with corresponding leak information other person easily access private state [7].
- ***b. Guessable User Accounts and Passwords:***If system authentication having default credentials and passwords being commonly used one and simple dictionary words then those accounts will easily leak by others, then it is easy to access by other system users.
- ***c. Brute Forcing:***Through this mechanism, keep on attempt all possible combinations sometimes application will crashes due to overflow of maximum number of requests. Nowadays cracking alphanumeric combinations of passwords are becoming easy, which is having less than 16 characters; it may be manual checking or automated brute forcing.
- ***d. Bypassing Authentication Schema:*** This vulnerability shell causes to access restricted directories, by simply requesting required path. So for others, it is easily to enter into private accounts [8].
- ***e. Remember and Reset Password:*** if application having this poorly configured remember password then possible to access application even after logged out. Also if session ID is stolen then those who have that session ID they can easily access original accounts.

In case of reset password, if reset mechanism implement in same page then sometimes it will misuse by others.

- ***f. Logout and Browser Cache Management:***After logged out if others requesting previous page then, it will opens previous account. This vulnerability will never kill previous state.
- ***g. Multiple Factor Authentication:***if second factor authentication done from same machine then application will easily compromise, in case of poorly configured mechanism in multiple factor authentications.
- ***h. Captcha:***Mainly captha vulnerabilities lead to violate authentication policies. Some poorly configured captcha leads tolaunch new vulnerabilities like enumeration attacks and cross site request forgery attacks. And some cases captcha will generates automated traffic, and sending anonymous mails to target id. Moreover, traffic will generates denial of service attacks, which leads to loss of availability or possibility to crash application.
- ***i. Race Condition:*** Poorly configured race conditions, machine doesn't know which condition will execute first. This will extend up to application crashes, because of maximum number of requests to corresponding applications. Also this will causes to create denial of service [6]. However identifying race conditions is very difficult. Any system that supports multitasking with shared resources is susceptible to race conditions. This can be avoided if appropriate synchronization primitives are used.

Majorly authentication flaws will causes to access private states, like restricted paths and directories, which cause to leak sensitive information.

IV. ISSUE REMEDIATIONS

Security is nonfunctional issues for service, when security threat arises in system; survivability is difficult in public service. More over finding authentication flaws is difficult before deployment of corresponding application. And it is better to follow security policies in application development itself; if not attack patterns are more active after deployment stage. Patterns to avoid authentication flaws are discussed as follows,

- ***a. Encrypted Channel:*** Even there is encrypted channel in application, it is better to provide strong encryption channel for sending sensitive information, because base level applications will easily decrypts in some cases.
- ***b. Guessable User Accounts and Passwords:***It is most common in digital world, whenever use wants to install new mechanism, corresponding technicians will access system with default usernames and passwords, which will know by others. So to avoid such risks, it is better to change default passwords to difficult credentials. Also reset passwords frequently [9].
- ***c. Brute Forcing:*** Best solution is only to use manageable switches, to keep avoiding system crashes. Also keep mechanism to block account after particular failed login attempts. Use strong passwords which having at least one special character, it is better to use password policy to create passwords. In case of administrative authentications, use tokens and certificates, it essential to exchange client and server side certificates. Restrict logins with multiple usernames from the same IP address, also logins for a single account coming from many different IP addresses. Avoid excessive usage and bandwidth consumption from a single user. Keep restrictions on logins with suspicious passwords which may effects the service, to avoid such suspicious activities properly configure IDS and IPS [10].

d. Bypassing Authentication Schema:To this vulnerability keep on directory level restrictions on each directory. Keep accesses restrictions on system files, and maintain authorization privileges on all directories to avoid authentication bypassing.

e. Remember and Reset Password: Active mechanism to kill session ID after successful logged out. Allocate new sessions to every new requests. Also use secure random number generator to create sessions. In application input fields that are accepting sensitive information, always those fields keep auto complete off.

In reset mechanism always use old password authentication or reset mechanism authenticate with one time passwords or sending reset links to personal mail address.

f. Logout and Browser Cache Management:Kill cookies after closing sessions, and store cookie information with strong cipher techniques.

g. Multiple Factor Authentication:Second authentication factor should be done with other machine. In application configure secondary authentication in proper way, means those configurations must be in encrypted channel. In highly secure mechanism prefer hardware authentication.

h. Captcha:Generate random image instead of sequence or predefined images. In case of high security service, it is better to generate own captcha, instead of using predefined services. In time of sending captcha, prefer encrypted mechanism. And always send captcha by POSTmethods.Generatecaptcha with special characters, which is difficult to crack in mean time.

i. Race Condition:Provide updated processors, and maintain maximum physical memory, to prevent unaccepted system crashes.Use appropriate synchronization primitives. Also better to implement following at the time of application development, Lock variables, Named pipes, Semaphores.

V. ANALYSIS

Authentication is process of changing user public state to private state, depends on authentication type providing file permissions to user. In secure applications it is essential to provide application accessing mechanism, which possible only with proper authentication strategies. From the considerations of basic application security principles, building secure applications is possible only to keep avoiding attack patterns. So to compromise application, authentication flaws are enough no need to concentrate on higher level vulnerabilities.

Web applications are mainly deals with request and response to process and manage information with webserver. In terms of business point of view, there must be place to security, if not possible to change private state turn to public. Result to loose integrity, confidentiality, availability. For this web server have to make sense before it process, which digital information that server receives from user requests. Use secure coding strategies from application designing phase onwards. Methods and functions used to host pages in web server causes to security threats.

In order to complete application, developers basically follows software development life cycle, which analyze whole functional architecture of application except security risks, this SDLC mainly manipulate by the security unaware team. To overcome these risk patterns implement secure software development life cycle (SSDLC), which consider only secure methods and functions.

Strong Authentications $\not\subset$ web server $\Rightarrow$ vulnerable web applications

Strong Authentications $\subset$web server $\Rightarrow$ secure web applications

Mainly security threats will occur because of functions and methods used in application, so major risks will active in the development stage itself, it is essential to aware about secure methods before application development stage. It is difficult to change methods and functionality of application when attack happens, so prefer only secure application development strategies. And consider security as functional issue in each development and deployment stages, to provide secure communication.

VI. DEPLOYMENT STRATEGIES

For business mobility web applications plays major key role. Any application it may be information or transaction oriented application active with authentication only. If there is no secure software development life cycle implementation, it is difficult to find security risks before deployment. It is better way to provide security is to avoid attack patterns. In case of administrative access keep on monitor network status, and keep on avoid multiple requests on single user.

Before deployment check applications from internal attacks, if so solve them. And come to sensitive information sharing always prefer strong encoding strategies, and store sensitive information with strong hash creation [8, 11]. And in case of transaction oriented services, always use multiple factor authentications. In hardware devices periodically reset default information, better to prefer manageable devices and secure plugins. Manage file permissions to control external user activities. In case of transactional oriented services always prefer manageable hardware and prefer maximum physical memory, and these strategies will prevents loss of integrity, confidentiality and availability.

VII. CONCLUSION

In this paper, we present secure authentication schemes, which change security implementation in organization level, and we focus on authentication vulnerabilities and preventing attack patterns in application entry level. This evolution of work will describes precautions to maintain in software pre and post deployment stages, which prevents internal and external authentication flaws.

Future evaluation of work shall focus on evaluating the secure web application development strategies, to provide reliable and secure communication, having lesser complexities and more reliable services, which prevent internal and external attack patterns on a system.

VIII. REFERENCES

[1] Gary McGraw and John Viega, “Building Secure Software: How to Avoid Security Problems the Right Way”, Addison-Wesley Pub Co, ISBN 020172152X.

[2] Dafydd Stuttard, Marcus Pinto, “The Web Application’s Handbook - Discovering and Exploiting Security Flaws”, 2008, Wiley, ISBN 978-0-470-17077-9.

[3] Joel Scambray, Mike Shema, Caleb Sima, "Hacking Exposed Web Applications", Second Edition, McGraw-Hill, 2006 - ISBN 0-07-226229-0.

[4] Sverre Huseby, "Innocent Code: A Security Wake-Up Call for Web Programmers", John Wiley & Sons, ISBN 0470857447.

[5] Hassan A, "Xiaowen Zhang, Bypassing web-based wireless authentication systems", Systems, Applications and Technology Conference LISAT, 2011 IEEE

[6] Jinpeng Wei ; Pu, C; "Multiprocessors May Reduce System Dependability under File-Based Race Condition Attacks", Dependable Systems and Networks, 2007, IEEE/IFIP, pp 358-367

[7] Saxena, P. ; Akhawe, D. ; Hanna, S. ; Feng Mao ; McCamant, S. ; Song, D., "A Symbolic Execution Framework for JavaScript", Security and Privacy (SP), 2010 IEEE, Page(s): 513-528.

[8] Mike Howard and David LeBlanc, "Writing Secure Code", Microsoft Press, ISBN 0735617228.

[9] Gary McGraw and Greg Hoglund, "Exploiting Software: How to Break Code", Addison-Wesley Pub Co, ISBN 0201786958.

[10] Atashzar, H. ; Torkaman, A. ; Bahrololum, M. ; Tadayon, M.H., "A survey on web application vulnerabilities and countermeasures", Computer Sciences and Convergence Information Technology (ICCIT), 2011, Page(s): 647-652.

[11] James S. Tiller, "The Ethical Hack: A Framework for Business Value Penetration Testing", Auerbach, ISBN 084931609X.