Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 2, Issue. 9, September 2013, pg.302 – 309

REVIEW ARTICLE

A Review of the Vulnerabilities of Web Applications

Ravneet Kaur Sidhu

Punjabi University, Patiala, Chandigarh

Abstract: Insecure software is undermining our financial, healthcare, defense, energy, and other infrastructure. As our digital infrastructure gets increasingly complex and interconnected the difficulty of achieving application security increases exponentially. An attacker can target and compromise a database server in a number of ways by exploiting a variety of configurations and application level vulnerabilities. In this paper we discuss some attacks and weaknesses that can result in the compromise of a web application or its user.

Keywords: Misconfiguration; Indexing; Anti-automation; Validation; Expiration; Authentication

Introduction: Weakness is a mistake which may occur during implementation, design, or other phases of the software development life cycle. It is a mistake in the software which could contribute towards introducing vulnerabilities. The some identified weaknesses have been discussed below:

I. INSUFFICIENT AUTHENTICATION

This can occur when a web site unknowingly permits an attacker the right to use sensitive content or functionality without properly authenticating him/her. The web-based administration tools can be stated as a good example of web sites giving access to sensitive functionality. Such

web applications should not be directly accessible, especially without requiring the proper authentication and verification of the user's identity.

To make the application more secure, sometimes no link to the location is given into the web site. However, even if a resource is unknown to an enemy, it still has an accessible URL. The explicit URL could be discovered through brute force. Hence, all resources of content or functionality should be appropriately protected

II. INSUFFICIENT AUTHORIZATION

This happens when an adequate number of authorization checks are not performed. The user may perform a function or access data in a manner which is not in sync with the security policy. Procedures of authorization should enforce only that which is permitted. Even an authenticated user should not necessarily have complete access to all content and functionality.

INSUFFICIENT FUNCTION AUTHORIZATION: Usually different functionality is granted to different users. A news site may allow its users to view stories, but not necessarily publish them. An accounting system may have different permissions for different employee operators. Inadequate function permission happens when users are not prevented from accessing functionality which is not on the lines of the security policy.

INSUFFICIENT DATA AUTHORIZATION: Programmers normally have complete knowledge of functionality of an application, but not always have a entire mapping of all the data that an application may access. Relying on third party systems such as database to perform data authorization results in insufficient data authorization.

III. INSUFFICIENT TRANSPORT LAYER PROTECTION

The communication may be exposed to un-trusted third-parties, hence, providing the attacker an opportunity to steal insightful information. Websites typically use protocol layers to provide encryption. Still, unless the website is configured to use layer protocols correctly, the vulnerability remains.

LACK OF TRANSPORT LAYER ENCRYPTION: Unencrypted communication between client and the websites leaves it open to man-in-the-middle attack. An attacker passively intercepts the communication which gives them access to any data that is being transmitted. An attacker can actively inject or remove content from the communication. An attacker can also redirect the communication so that the website and client may no longer communicate with each other. Instead they unknowingly communicate with the attacker.

WEAK CIPHER SUPPORT: In the past, high level cryptography was restricted from extensive use. Hence, websites supported weak cryptography. Weak ciphers are more vulnerable to attack. This is because of the relative ease of breaking them.

Today, much stronger encryption techniques and ciphers are used, but some websites still support weak ciphers. Just because of this, the attacker can manage to break the weak encryption. For this very reason, the server should accept only strong ciphers.

IV. INFORMATION LEAKAGE

An application may sometimes unknowingly reveal secretive data. Any kind of leakage of sensitive data should be prevented whenever possible. The results of unchecked information leakage could be huge.

Information leakage in context of sensitive data deals with exposure of data that is confidential. Credit card numbers are examples of user data that needs to be protected from leakage.

DEVELOPERS COMMENTS LEFT IN PAGE RESPONSES: Failure to remove comments can result in the leak of information such as internal network information or SQL query structure. Comments are left within code to help the debugging. There is no apparent harm in allowing inclusion of inline comments but they should be removed before public release.

IMPROPER APPLICATION OR SRVER CONFIGURATIONS: This would be the response to an invalid SQL query. Error messages and version numbers are improper to use because these are useful to an attacker. The information leaked by an error message can provide detailed information on how to construct valid SQL query for the backend database. The default configurations of a server provide software version numbers and error messages. Changes to disable these features can prevent the display of such information.

DIFFERENCES IN PAGE RESPONSE BEHAVIOURS: "User friendly" application usually have a Forgot Password feature due to which an attacker can find valid email accounts. Once the email address is confirmed before sending a mail, information leakage occurs.

V. SERVER MISCONFIGURATION

Exploitation of the configuration weaknesses of web application servers is very common. Servers usually come with default files. They may have some unwanted services; debugging or administrative functions which may be accessible to any user. Such features assist an hacker to gain sensitive information. Inability to completely lock down the server may set improper directory permissions. The error messages occasionally result in data leakage. This leaked information could be used to enhance the accuracy of the attack.

VI. APPLICATION MISCONFIGURATION

Weakness exists in configuration of web applications. Many applications come with debug and quality analysis features which are enabled by default. These features help to gain access to sensitive information. Similarly, default installations include usernames and passwords and special access mechanisms. Application-based configuration files may reveal the settings in configuration files. All this may lead to unauthorized access to private data.

VII. DIECTORY INDEXING

A web server function lists the files of a requested directory if the normal base file is not present. When the request of the main page of a site is requested to the web server, the root directory is searched for the default file name. The requested page is then sent to the client. However, if the requested page is not present; the server dynamically issues a listing of the directories and sends the output to the client. This could contain information not intended for public. Although it is potentially undamaging but it could allow leakage of information.

VIII. IMPROPER FILE PERMISSIONS

The confidentiality and integrity of a web application is under constant threat. When file system has incorrect permissions, it is then that the problems arise. An attacker can be able to access restricted directories and also be able to modify their contents. For example, an anonymous account which has write permission to a file can modify the contents of the web application in an unpleasant manner.

The resources offered by an underlying file system are accessed by a web server by using the operating system account which has permissions to access the source code and execute the server side scripts. When a browser requests for a file, the pre-defined security settings and the file type influence the web server in deciding how to load a file. The web server may either serve the request depending on the permissions assigned or return a permission-denied error.

IX. IMPROPER INPUT HANDLEING

One of the most common weaknesses across applications today is poor input handling. It is a primary cause of critical vulnerabilities that exist in various web applications.

The term ‘input handling’ is generally used to describe validation and filtering of input data. Input is received from various sources. For web applications, input can be handled in various formats and obtained via numerous locations. Application variables and configuration files, etc. are the source of non-web application input. All input should initially be considered untrusted applications which process such input may become vulnerable to attacks

IMPROPER INPUT VALIDATION: Validating the input is one of the most important aspects of input handling. It is vital to identify the form and type of data that has to be acceptable for the application. Accurately defining restriction requires defining the expected format and the usage of each instance.

Checks for correct syntax, type safety, length of string and it's character set, upper and lower bound of numeric values must be validated. Validation should be performed during concatenation of input from multiple sources.

CLIENT SIDE VS SERVER SIDE VALIDATION: Most developers commit a common mistake of including only JavaScript function routines for client side of an application. This is beneficial on the client side but do not provide a security feature. This is so because all data on the client side is modifiable by an attacker. The risk from certain features and plug-ins becomes noteworthy when the developer starts rely on it as the only means of input validation. This is not a safe practice. It gives a false sense of security. However, it must be noted that while client side validation is good for user interfaces, it cannot be a substitute for server side validations. Performing the latter validation ensures the integrity of controls. Additionally, the server-side validation is always effective.

ATTACKS DUE TO IMPROPER INPUT HANDELING:

BUFFER OVERFLOW: If the length of the variable in the input is not validated before copying to the destination, then the weakness may be exploited by exceeding the size of input as compared to the size of the destination. This will cause overflow of the destination's address in memory. An SQL query may return all the rows from a database, irrespective of whether the user name is real or the password is a valid or not. This may be caused by appending the OR statement to the WHERE clause. This elevates all rows in the table to true .If an attacker user this method then he shall be logged in as the first or last user from the users table.

OS COMMANDING: This attack technique is used for executing the commands of an operating system in an unauthorized manner. Improper input handling is one of the common weaknesses that can be used to run unauthorized commands. Due to non-existent input handling the change in the parameter value of the template by an attacker can trick the web application.

X. IMPROPER OUTPUT HANDLING

This refers generation of outgoing data. Improper output handling mat lead to vulnerability. Those actions which were never intended by the application developer may become performable. Such involuntary interpretation is also application vulnerability.

Whenever any data leaves the boundary of an application, it may be subjected to improper output handling. Such boundaries exist where data traverses contexts such as application passing data to other applications. It also includes passing data within layers of the application architecture

This may take various forms within an application. Protocol, application and data consumer related errors can be the various forms into which these can be categorized into. The protocol errors include improper output escaping and incorrect passing of output data. If in any case, the application is unable to avoid the previously known vulnerabilities, it may cause improper data handling resulting in consumer data's abuse.

Proper handling of output data prevents the unintended data interpretation. To achieve this, developers must understand the complete working of the application in context to its data model. Any application cannot be completely secure unless it is protected against un-intended interpretations, the core requirement for which is to handle output operations in a secure fashion.

COMMON DATA OUTPUT LOCATIONS: The most common data output locations are:

INSIDE HTTP HEADERS: HTTP headers exist in HTTP requests and responses. They define client and resource characteristics. Attacking the HTTP headers involve changing the HTTP message structure. Doing this enables the attacker to abuse the clients and servers.

INSIDE HTML TAGS: The browser treats the text between <tag></tag> as text to be displayed. If data is somehow unintentionally included in this text, then it may lead to vulnerabilities.

INSIDE HTML ATTRIBUTES: The tag attributes contain data about web applications. Some special attributes have specific meaning and hence require extreme care in usage avoid introducing vulnerabilities.

INSIDE CLIENT-SIDE SCRIPT: The data inside <script>tag more often than not deserves special care. If not correctly coded there is a risk of advancing risk which may invite various attacks.

INSIDE XML MESSAGES: XML can be found at almost every layer of web applications. Apparently, even if properly encoded, some types of XML messages can contain certain attributes which may have special meaning. The interpretation may be such so as to cause vulnerability.

INSIDE SQL QUERIES: Behind almost every web applications is a relational database. Applications must insure that SQL queries based on user influenced data do not allow the data to be interpreted as instructions to the database.

XI. INSUFFICIENT PROCESS VALIDATION

Ideally, there should be no way of working around manipulating the intended flow of logic of a business application. In a real world scenario, insufficient process validation has most often resulted in monetary loss due to ineffective access controls.

Validation is required for two types of processes: Flow control and Business logic.

Flow control is the multi-step process that needs to be followed in a specific order by the user. When an attacker performs the steps in a jumbled order certain to access controls may be bypassed. This may result in an application integrity error. Some examples of multi-step processes are account signup and password recovery etc.

Business logic is the context in which a process may execute as governed by the requirements of a particular business. Exploiting such weakness requires knowledge of business. Due to this, the typical security measures like scans and code review are not able to uncover such a weakness.

XII. INSUFFICIENT SESSION EXPIRATION

This occurs when an attacker is able to reuse old session ID for authorization. Insufficient session expiration increases a web site's exposure to attacks that steal user's previously used session identifiers.

HTTP is a stateless protocol. Cookies are used to store a session ID's which may uniquely differentiate one user request from other. Therefore, confidentiality of each session ID is a must. It may be able to prevent multiple users from accessing the same account.

There are two types of session expiration time-outs, one which is absolute and another which happens due to inactivity. The former is defined by total amount of time a session can be valid without re-authentication, where as the latter occurs after the expiry of idle time allowed before the session is invalidated. The absence of proper session expiration may amplify the likelihood of attacks. The chance of successfully guessing a valid session ID increases for a long expiration time. The longer the expiration time, the more number of concurrent sessions can exist at a given time. And, the larger the pool of sessions, more likely it is for the attacker to guess one at random. Even a short session inactivity timeout does not help. This is the case, if a token is immediately used. The short timeout helps to insure that the token is relatively harder to capture while it is still valid.

To keep the lifespan of a session ID as short as possible, a web application should invalidate a session after completion of a predefined amount of idle time. The user should be provided with the means to7 invalidate their own session. Self initiated logout helps and is indispensable in a shared computing environment. Either the logout function could be notably visible to the user or explicitly invalidated for a user's session to disallow reuse of the session token.

Conclusion: From the literature survey, it has been concluded that the aforementioned weaknesses cannot be completely avoided. Some newer methods to combat them need to be devised.

References:

[1] "Authentication Attacks"
http://pic.dhe.ibm.com/infocenter/sprotect/v2r8m0/index.jsp?topic=%2Fcom.ibm.ips.doc%2Fconcepts%2Fwap_authentication.htm

[2] "Guide To Authorization"
https://www.owasp.org/index.php/Guide_to_Authorization

[3] "Misconfigured, Open DNS Servers Used In Record-Breaking DDoS Attack" By Kelly Jackson Higgins
*https://www.projects.web**apps**ec.org/w/page/.../**Application**%20**Misconfiguration***
[4] "Directory Indexing Attacks"
http://pic.dhe.ibm.com/infocenter/sprotect/v2r8m0/index.jsp?topic=%2Fcom.ibm.ips.doc%2Fconcepts%2Fwap_directory_indexing.htm
[5] "WASC Threat Classification 2.0 - WASC-17 - Improper Filesystem Permissions"
http://capec.mitre.org/data/definitions/350.html

[6] "WASC Threat Classification 2.0 - WASC-22 - Improper Output Handling"
http://capec.mitre.org/data/definitions/355.html

[7] "Insufficient anti-automation" by Ofer Shezaf
http://www.xiom.com/taxonomy/term/27

[8] "Insufficient Process Validation"
http://blog.whitehatsec.com/insufficient-process-validation-2/#.Ujb-Ej87XIU

[9] "Insufficient Session Expiration"
http://projects.webappsec.org/w/page/13246944/Insufficient%20Session%20Expiration

[10] "Insecure Indexing"
http://projects.webappsec.org/w/page/13246937/Insecure%20Indexing

[11] "Insufficient Password Recovery"
http://projects.webappsec.org/w/page/13246942/Insufficient%20Password%20Recovery

