Available online at http://www.journalcra.com

International Journal of Current Research
Vol. 10, Issue, 08, pp.72621-72624, August, 2018

INTERNATIONAL JOURNAL OF CURRENT RESEARCH

ISSN: 0975-833X

RESEARCH ARTICLE

THREATS OF WEB APPLICATION WITH THEIR SOLUTIONS IN INFORMATION SECURITY

*Dr. Mahdi Abdullah Mohammed AL-Sebaeai

Assistant Professor- Information Technology, Dept. of Computer Science and Information Technology, Al-Razi University, Sana'a, Republic of Yemen

ARTICLE INFO

Article History:
Received 15th May, 2018
Received in revised form
20th June, 2018
Accepted 07th July, 2018
Published online 30th August, 2018

Key Words:
Web application,
Web Vulnerabilities,
Web Security.

ABSTRACT

This paper explain threats web applications, Web Vulnerabilities. The web vulnerabilities are SQL Injection, OS command Injection, HTTP Header Injection, Mail Header Injection, Lack of Authentication and Authorization, Improper Session Management. This paper explain solutions problems to vulnerabilities in web application.

Citation: ***Dr. Mahdi Abdullah Mohammed AL-Sebaeai*****, 2018.** "Threats of web application with their solutions in information security", *International Journal of Current Research*, 10, (08), 72621-72624.

INTRODUCTION

The increased usage of the Internet and network technology has changed the focus in assessing computer environments. The traditional approach considered the location of hardware and equipment first and then the data stored on the hardware. In addition, these assessments were primarily at specific points in time and primarily compliance-based reviews. With network-based technology, the primary concern is on the network and the contents of information or objects. Moreover, an assessment of network technology is focused on the implementation and management of real-time controls to meet the business needs and to provide continuous monitoring [http://tools.ietf.org/html/rfc1123),May 1996.]. Security is not a one-time event. It is insufficient to secure your code just once. A secure coding initiative must deal with all stages of a program's lifecycle. Secure web applications are only possible when a secure SDLC is used. Secure programs are secure by design, during development, and by default.

Web Vulnerabilities: We can no longer afford to tolerate relatively simple security problems like those presented below.

***Corresponding author:** Dr. Mahdi Abdullah Mohammed AL-Sebaeai
Assistant Professor- Information Technology, Dept. of Computer Science and Information Technology, Al-Razi University, Sana'a, Republic of Yemen
DOI: https://doi.org/10.24941/ijcr.31545.08.2018

The vulnerabilities [http://blog.gjl-network.net/blog/ index.php?/archives /78-Knorr.de-SQL-Injection -and -XSS-Valnerabilities.html), 01.12.2007] explained in this paper are: SQL injection ,OS command Injection, HTTP Header Injection, Mail Header Injection, Lack of Authentication and Authorization, Improper Session Management.

SQL injection: Most of web applications that use a database build an SQL statement (a command to operate the database) based on user input. That means if the SQL statement-building process is not securely guarded, attacking and manipulating the database would become possible. This issue is called "SQL Injection vulnerability" and the attacking method exploiting this vulnerability (1) is called "SQL Injection attack".

Possible Threats to SQL in Web Application:

This vulnerability could allow malicious attackers to:

- View sensitive data stored in the database
- e.g Disclosure of personal information.
- Falsify and/or delete data stored in the database
- e.g. Falsification of web pages, password change, system shutdown.
- Bypass login authentication. All the operations permitted under the privileges of a login account become unauthorizedly possible.
- Execute OS commands using stored procedures

- e.g. System hijacking, making the target PC a bot (launching point) to attack others.

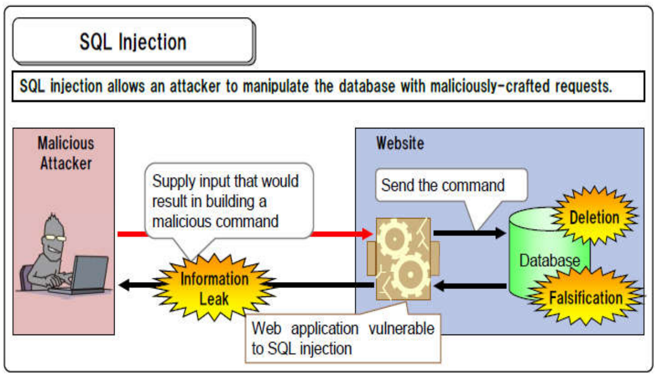
SQL Injection
SQL injection allows an attacker to manipulate the database with maliciously-crafted requests.
Malicious Attacker
Supply input that would result in building a malicious command
Website
Send the command
Deletion
Database
Information Leak
Falsification
Web application vulnerable to SQL injection

Figure 1. Web application vulnerable to SQL injection

Solutions Problems to SQL in Web Application

- Build all SQL statements using placeholders.
- When building an SQL statement through concatenation, use a special API offered by the database engine to perform escaping and make up the literals in the SQL statement correctly.
- Do not write SQL statement directly in the parameter in order to pass to the web application.
- Limit information to display in error message on the web browser.
- Grant minimum privileges to database accounts.

Operating System Command Injection: Web applications can be vulnerable in such a way that they allow a remote attacker to execute OS level commands via those applications. This issue also called "OS Command Injection vulnerability" and the attacking method exploiting this vulnerability called as "OS Command Injection attack".

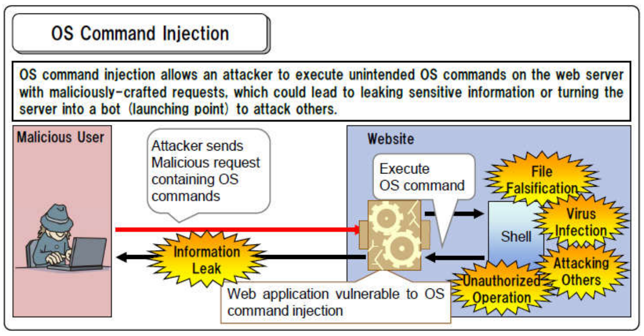
OS Command Injection
OS command injection allows an attacker to execute unintended OS commands on the web server with maliciously-crafted requests, which could lead to leaking sensitive information or turning the server into a bot (launching point) to attack others.
Malicious User
Attacker sends Malicious request containing OS commands
Website
Execute OS command
File Falsification
Virus Infection
Shell
Attacking Others
Information Leak
Unauthorized Operation
Web application vulnerable to OS command injection

Figure 2. Web application vulnerable to OS command injection

Possible Threats to OS Command in Web Application:

This vulnerability could allow malicious attackers to:

- View, falsify and delete files stored in the server
- e.g. Disclosure of sensitive information, falsification of configuration files.
- Maliciously manipulate the system
- e.g. Unintended OS shutdown, adding/deleting user accounts.
- Download and execute malicious programs
- e.g. Virus, worm and bot infection, backdoor implementation.
- Make the system a launching point to attack others
- e.g. Denial of Service attack, reconnaissance and spamming.

Solutions Problems Operating System Command in Web Application

- Avoid using functions that could call shell commands.
- When using functions that could call shell commands, check all variables that make up the shell parameters and make sure to execute only those that are granted to be executed.

HTTP Header Injection: Some web applications dynamically set the value of the HTTP response header fields based on the value passed by the external parameters. For example, HTTP redirection implemented by setting a redirected-to URL specified in the parameter to the Location header field, or a web application may set the names entered in a bulletin board to the Set-Cookie header filed. If the process of building an HTTP response header in such web applications has vulnerabilities, an attacker could add header fields, manipulate the response body and have the web application generate multiple responses. This issue is called "HTTP Header Injection vulnerability" and the attack method exploiting this vulnerability is called "HTTP Header Injection attack". In particular, the attack that leads the web application to produce multiple responses is called "HTTP Response Splitting attack".

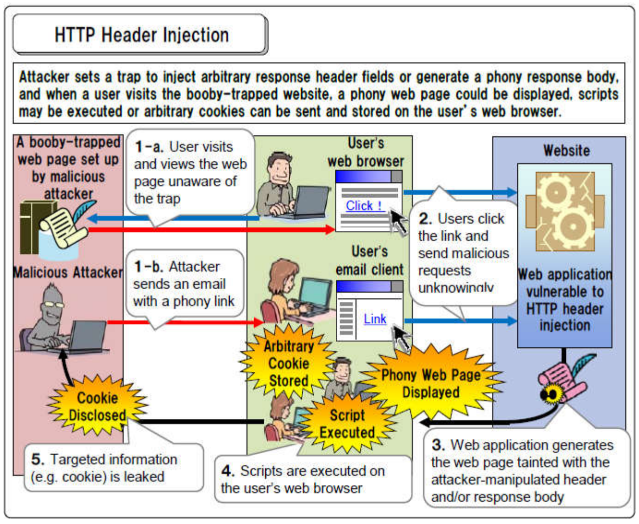
HTTP Header Injection
Attacker sets a trap to inject arbitrary response header fields or generate a phony response body, and when a user visits the booby-trapped website, a phony web page could be displayed, scripts may be executed or arbitrary cookies can be sent and stored on the user's web browser.
A booby-trapped web page set up by malicious attacker
1-a. User visits and views the web page unaware of the trap
User's web browser
Click !
Website
2. Users click the link and send malicious requests unknowingly
Malicious Attacker
1-b. Attacker sends an email with a phony link
User's email client
Link
Web application vulnerable to HTTP header injection
Arbitrary Cookie Stored
Phony Web Page Displayed
Cookie Disclosed
Script Executed
5. Targeted information (e.g. cookie) is leaked
4. Scripts are executed on the user's web browser
3. Web application generates the web page tainted with the attacker-manipulated header and/or response body

Figure 3. Http header injection

Possible Threats to HTTP Header in Web Application

This vulnerability could allow malicious attackers to:

- Present the same threats posed by the cross-site scripting vulnerability.
- If an arbitrary response body is injected, the user's browser may result in displaying the false information or be forced to execute arbitrary scripts. Those are the same threats discussed earlier in "Cross-Site Scripting".
- Create arbitrary cookies.
- When an HTTP Set-Cookie header is inserted, an arbitrary cookie is created and stored in the user's browser.
- Poison web cache

HTTP response splitting forces a web server to generate multiple HTTP responses and could inflict cache poisoning 27, which results in web page falsification, by having a proxy server cache an arbitrary HTTP response and replacing the original cached web page with it. The users visiting the victimized website are to view the replaced phony web page. Compared to the cross-site scripting attack, in which only a targeted individual would fall victim just once right after the attack, the threat cache poisoning poses would affect a larger number of users and last long time.

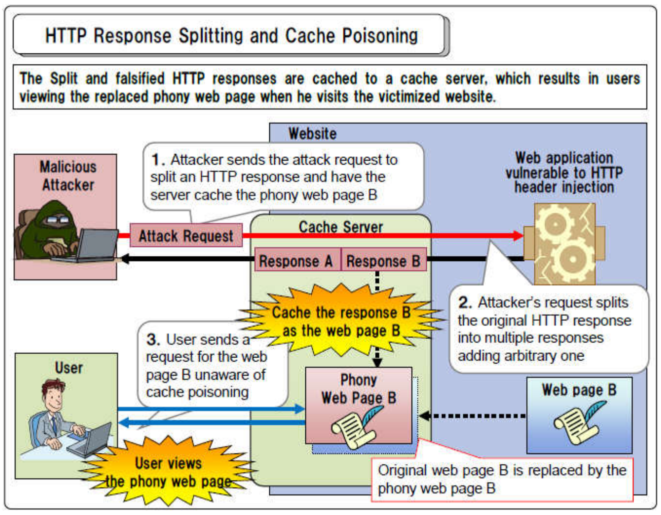
HTTP Response Splitting and Cache Poisoning
The Split and falsified HTTP responses are cached to a cache server, which results in users viewing the replaced phony web page when he visits the victimized website.
Website
Malicious Attacker
1. Attacker sends the attack request to split an HTTP response and have the server cache the phony web page B
Web application vulnerable to HTTP header injection
Attack Request
Cache Server
Response A
Response B
Cache the response B as the web page B
2. Attacker's request splits the original HTTP response into multiple responses adding arbitrary one
3. User sends a request for the web page B unaware of cache poisoning
User
Phony Web Page B
Web page B
User views the phony web page
Original web page B is replaced by the phony web page B

Figure 4. Threats HTTP header in web application

Solutions Problems HTTP Header in Web Application

- Do not print out HTTP header directly and do it through an HTTP header API provided by execution environment or programming language.
- If HTTP header API that offers line feed neutralization is not available for use, implement it manually.
- Remove all line feed characters that appear in the external text input.

Mail Header Injection: Some web applications provide a function that sends emails to the particular email addresses about, for example, the merchandise the users have purchased or survey replies.

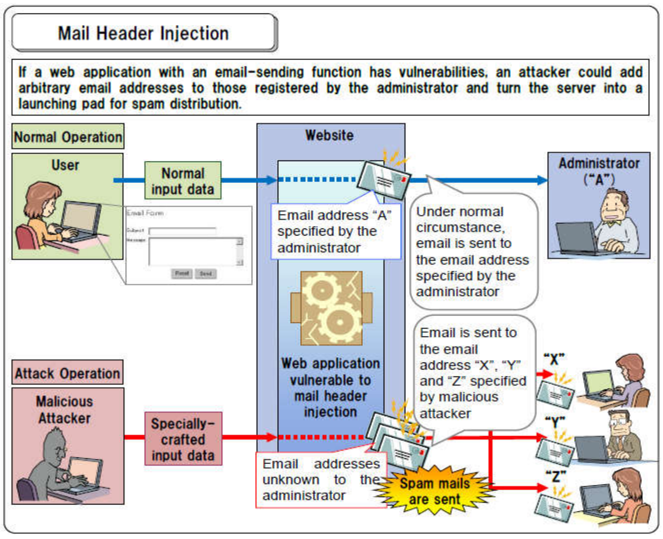
Mail Header Injection
If a web application with an email-sending function has vulnerabilities, an attacker could add arbitrary email addresses to those registered by the administrator and turn the server into a launching pad for spam distribution.
Normal Operation
Website
User
Normal input data
Administrator ("A")
Email address "A" specified by the administrator
Under normal circumstance, email is sent to the email address specified by the administrator
Web application vulnerable to mail header injection
Email is sent to the email address "X", "Y" and "Z" specified by malicious attacker
"X"
"Y"
"Z"
Attack Operation
Malicious Attacker
Specially-crafted input data
Email addresses unknown to the administrator
Spam mails are sent

Figure 5. Mail header injection

In general, these email addresses are prespecified and only the web administrator can change. Depending on how it is implemented, however, an attacker may be able to set and change them to arbitrary email addresses. This vulnerability is called "Mail Header Injection" and the attacking method exploiting this vulnerability is called "Mail Header Injection attack".

Possible threats to Mail header in web application:

This vulnerability could allow malicious attackers to:

- Third party mail relay. Used as a launching pad for spam distribution.

Solutions problems Mail header in web application

- Use the fixed values for the header elements and output all external input to the email body.
- If the fixed values cannot be used for the header, use an email-sending API offered by the web application's execution
- Do not specify the email addresses in HTML.
- Remove all line feed characters that appear in the external text input.

Lack of Authentication and Authorization

There are some inadequately designed websites in operation due to lack of the operator's security awareness. In this chapter, the vulnerabilities is reported to us that stem form the lack of important functions such as "authentication" and "authorization".

Solutions problems Lack of Authentication and Authorization in web application

- Lack of Authentication:
- When a web site needs access control, implement an authentication mechanism that requires users to enter some kind of secret information, such as password.
- Lack of Authorization Control:
- Implement authorization as well as authentication to make sure that a login user cannot pretend to be other users and access their data.
- Do not specify the email addresses in HTML.
- Remove all line feed characters that appear in the external text input.

Improper Session Management: Some web applications issue session ID, which is the information to identify the user, to manage sessions. If session ID is not created and managed properly, an attacker could steal the session ID of a legitimate user and gain unauthorized access to the services pretending to be the legitimate user. The attacking method exploiting this vulnerability in session management is called "Session Hijacking". In addition to guessing or stealing session IDs, there is another attack exploiting improper session management called "Session Fixation". It occurs when an attacker prepares a session ID and has a target user use the session ID in some way11 and the target user who is unaware of it logs into the website.

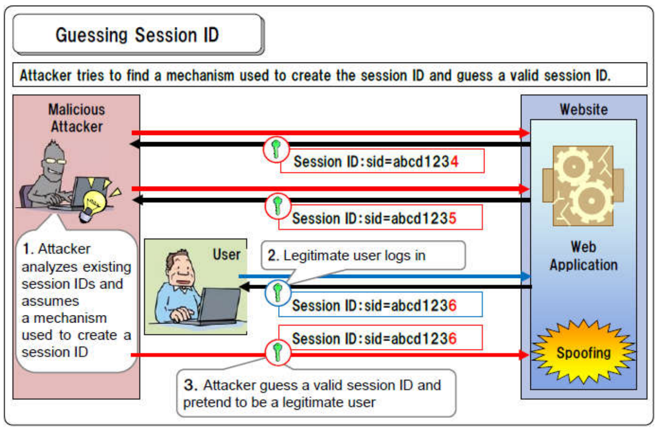
Guessing Session ID
Attacker tries to find a mechanism used to create the session ID and guess a valid session ID.
Malicious Attacker
Website
Session ID:sid=abcd1234
Session ID:sid=abcd1235
1. Attacker analyzes existing session IDs and assumes a mechanism used to create a session ID
User
2. Legitimate user logs in
Session ID:sid=abcd1236
Session ID:sid=abcd1236
Web Application
Spoofing
3. Attacker guess a valid session ID and pretend to be a legitimate user

Figure 6. Guessing Session ID

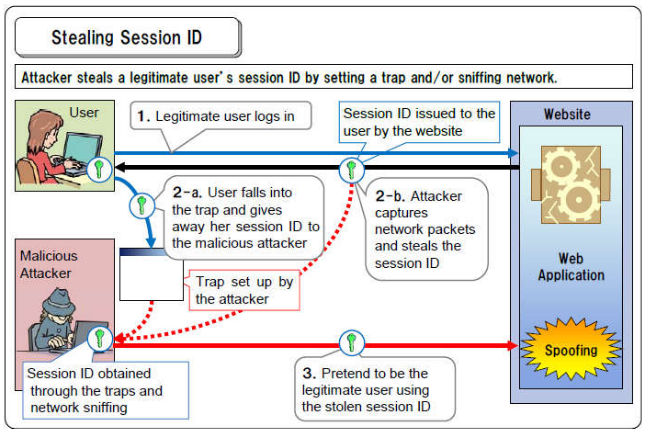
Stealing Session ID
Attacker steals a legitimate user' s session ID by setting a trap and/or sniffing network.
User
1. Legitimate user logs in
Session ID issued to the user by the website
Website
2-a. User falls into the trap and gives away her session ID to the malicious attacker
2-b. Attacker captures network packets and steals the session ID
Malicious Attacker
Trap set up by the attacker
Web Application
Spoofing
Session ID obtained through the traps and network sniffing
3. Pretend to be the legitimate user using the stolen session ID

Figure 7. Stealing Session ID

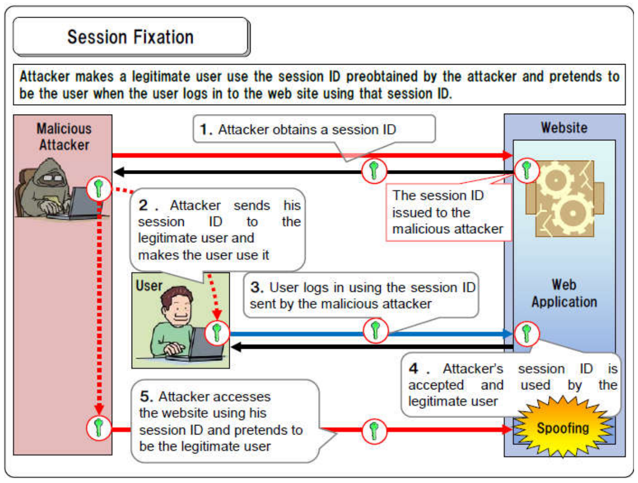
Session Fixation
Attacker makes a legitimate user use the session ID preobtained by the attacker and pretends to be the user when the user logs in to the web site using that session ID.
Malicious Attacker
1. Attacker obtains a session ID
Website
2. Attacker sends his session ID to the legitimate user and makes the user use it
The session ID issued to the malicious attacker
User
3. User logs in using the session ID sent by the malicious attacker
Web Application
4. Attacker's session ID is accepted and used by the legitimate user
5. Attacker accesses the website using his session ID and pretends to be the legitimate user
Spoofing

Figure 8. Session Fixation

If successful, the attacker could pretend to be the targeted user using his or her session ID, which has been set up by the attacker, and access the website.

Possible Threats to Improper Session in Web Application: If an attack exploiting improper session management succeeds, an attacker could pretend to be a legitimate user and do the operations permitted to that user.

For example, it could allow to:

- Access the services normally available only for the users who have properly logged in.
- e.g. Unauthorized money transfer, purchasing unintended goods, canceling the membership against the user's will
- Add and modify information normally permitted only for the users who have properly logged in
- When e.g. Unauthorized change of application settings (passwords, administrator functions etc.),writing inappropriate entries
- View information normally available only for the users who have properly logged in
- e.g. Unauthorized access to personal information, webmails, members-only bulletin board

Solutions problems improper session in web application

- Make session ID hard to guess.
- Do not use URL parameter to store session ID.
- Set the secure attribute of the cookie when using HTTPS.
- Start a new session after successful login.
- Issue a secret after login and authenticate the user with it whenever the user moves around the web site.

Mitigation Measures

- Use random session ID.
- Set the cookie's expiration date with care when storing session ID in cookie.

Conclusions

There are some problems in the web application, and there are threats information security. The Solutions in this article help to decrease threats information security.

REFERENCES

Network Working Group , "Requirements for internet Hosts-Application and Support ",RFC 1123, (http://tools.ietf.org/html/rfc1123),May 1996.

Sebastian Bauer," Knorr.de SQL Injection and XSS Vulnerabilities ", (http://blog.gjl-network.net/ blog/index. php?/ archives /78-Knorr.de-SQL-Injection -and -XSS-Valnerabilities.html), 01.12.2007.

Gmail uses Google's innovative technology to keep spam out of your inbox", gmail .com, (http://www.google.com/ mail/help/fightspam/spamexplained .html),December, 2007.

****