Ethical Hacking

Module – 6

- **What is Session Hijacking Explain with Techniques?**

Session Hijacking is a Hacking Technique. In this, the hackers (the one who perform hacking) gain the access of a target's computer or online account and exploit the whole web session control mechanism. This is done by taking over an active TCP/IP communication session by performing illegal actions on a protected network.

Types of Session Hijacking: -

Active Session Hijacking: An Active Session Hijacking occurs when the attacker takes control over the active session. The actual user of the network becomes in offline mode, and the attacker acts as the authorized user. They can also take control over the communication between the client and the server. To cause an interrupt in the communication between client and server, the attackers send massive traffic to attack a valid session and cause a denial-of-service attack (DoS).

Passive Session Hijacking: In Passive Session Hijacking, instead of controlling the overall session of a network of targeted user, the attacker monitors the communication between a user and a server. The main motive of the hacker is to listen to all the data and record it for the future use. Basically, it steals the exchanged information and use for irrelevant activity. This is also a kind of man-in-middle attack (as the attacker is in between the client and the server exchanging information.

Hybrid Hijacking: The combination of Active Session Hijacking and Passive Session Hijacking is referred to as Hybrid Hijacking. In this the attackers monitors the communication channel (the network traffic), whenever they find the issue, they take over the control on the web session and fulfill their malicious tasks.

To perform these all kinds of **Session Hijacking attacks**, the attackers use various methods. They have the choice to use a single method or more than one method simultaneously to perform Session Hijacking. Those methods are:

- ✓ Brute-forcing the Session ID
- ✓ Cross-Site Scripting (XSS) or Misdirected Trust
- ✓ Man-in-the-browser
- ✓ Malware infections
- ✓ Session Fixation
- ✓ Session side-jacking
- ✓ These all-Session Hijacking methods can be elaborated as:

Brute-forcing the Session ID: As the name suggests, the attack user uses guessing and trial method to find Session ID depending on its length. This is due to lack of security and shorter length. The introduction of a strong and long session key made this method increase in a slow rate.

Cross-Site Scripting (XSS) or Misdirected Trust: In Cross-Site-Scripting, the attacker tries to find out the flaws and the weak point in the web server and injects its code into that. This activity of the attacker will help the attacker to find out the Session ID.

Man-in-the-browser: Man-in-the-browser uses a Trojan Horse (program that uses malicious code) to perform its required action. The attacker puts themselves in the communication channel of a server and a client. The main purpose of performing these attacks by the attacker is to cause financial fraud.

Malware infections: In Malware Infections, attacker can deceive the user to open a link that is a malware or Trojans program which will install the malicious software in the device. These are programmed to steal the browser cookies without the user's knowledge.

Session Fixation: Attackers create a duplicate or another disguised session in Session Fixation. It simply motivates or trick the user into authenticating the vulnerable server. This can be done by sending an email to the user, which on clicking directs to the attacker session.

Session side-jacking: In Session side-jacking, the attackers try to get access over a session using the network traffic. This becomes easy when the user is using an insecure Wi-Fi. The reading of network traffic and stealing of session cookie is done by packet sniffing. Packet Sniffing is a technique by which the data flowing across a network is observed.

- **Find DoS/DDoS Attack Tools.**
  - ✓ GoldenEye
  - ✓ Slowloris
  - ✓ LOIC (Low Orbit Ion Cannon)
  - ✓ HOIC (High Orbit Ion Cannon)
  - ✓ THC-SSL-DoS
  - ✓ HULK (http Unbearable Load King)
  - ✓ Pyloris
  - ✓ TOR's Hammer
  - ✓ XOIC
  - ✓ RUDY (R U Dead Yet?)
  - ✓ DAVOSET
  - ✓ OWASP HTTP POST

- **Explain SYN Flooding Attack with example**

→ TCP SYN flood is a type of Denial of Service (DDoS) attack that exploits part of the normal TCP three-way handshake to consume resources on the targeted server and render it unresponsive.

- ✓ First, select your target's IP address. (ping www.website.com)
- ✓ So now I know the victim's IP Address nnn.nnn.nnn.hhh
- ✓ Launching Metasploit by typing msfconsole -q in your kali terminal
- ✓ Msf6 > use auxiliary/dos/tcp/synflood
- ✓ Msf6> show options
- ✓ Now you can see you have all the available options that you can set.
- ✓ To set an option just you must typeset and the option name and option.
- ✓ You must set two main options
- ✓ RHOST= target IP Address
- ✓ RPORT=target PORT Address
  - Set RPORT 18.192.182.30

- Set RPORT 80

✓ To launch the attack just type.
✓ Exploit
✓ to see the packets, you can open Wireshark.

- **List of Web App Hacking Methodology**

There are several common web application attacks that hackers often exploit to compromise the security of web applications.

Cross-Site Scripting (XSS)

XSS attacks involve injecting malicious scripts into web pages viewed by other users. This can occur when the application fails to properly sanitize user input or output, allowing attackers to execute arbitrary code in the victim's browser. XSS attacks can be used to steal sensitive information, hijack user sessions, or deface websites.

SQL Injection

SQL injection attacks occur when an attacker manipulates a web application's database queries by inserting malicious SQL code. This can enable unauthorized access to the database, data theft, or modification of data. SQL injection vulnerabilities commonly arise when user input is not properly validated or sanitized before being used in database queries.

Cross-Site Request Forgery (CSRF)

CSRF attacks trick authenticated users into unknowingly executing unwanted actions on a web application. This is achieved by crafting malicious requests and exploiting the trust placed in the user's browser sessions. CSRF attacks can lead to actions being performed without the user's consent, such as changing passwords, making financial transactions, or deleting data.

Remote File Inclusion (RFI) and Local File Inclusion (LFI)

RFI and LFI attacks involve exploiting vulnerabilities that allow the inclusion of external or local files in a web application. Attackers can manipulate these vulnerabilities to execute arbitrary code, read sensitive files, or gain unauthorized access to the server.

XML External Entity (XXE) Attacks

XXE attacks target applications that parse XML input insecurely. By exploiting this vulnerability, attackers can retrieve sensitive information, execute remote code, or perform denial-of-service attacks.

Server-Side Request Forgery (SSRF)

SSRF attacks occur when an attacker tricks a web application into making requests to other internal or external resources on behalf of the application server. This can lead to unauthorized access to internal systems, data leakage, or further exploitation of vulnerabilities.

File Upload Vulnerabilities

Insecure file upload functionalities can be abused by attackers to upload malicious files onto a server. These files can then be executed to gain unauthorized access, escalate privileges, or perform other malicious activities.

Session Hijacking and Session Fixation

These attacks target weaknesses in session management mechanisms. Session hijacking involves stealing or impersonating valid user sessions, while session fixation involves forcing a user to use a predetermined session ID. Both attacks can lead to unauthorized access to user accounts and sensitive data.

- **SQL Injection Methodology**

SQL injection, also known as SQLI, is a common attack vector that uses malicious SQL code for backend database manipulation to access information that was not intended to be displayed. This information may include any number of items, including sensitive company data, user lists or private customer details.

The impact SQL injection can have on a business is far-reaching. A successful attack may result in the unauthorized viewing of user lists, the deletion of entire tables and, in certain cases, the attacker gaining administrative rights to a database, all of which are highly detrimental to a business.

When calculating the potential cost of an SQLi, it's important to consider the loss of customer trust should personal information such as phone numbers, addresses, and credit card details be stolen.

While this vector can be used to attack any SQL database, websites are the most frequent targets.

- **Explain SQL injection with any tool**

SQL Injection Based on 1=1 is Always True

Look at the example above again. The original purpose of the code was to create an SQL statement to select a user, with a given user id.

If there is nothing to prevent a user from entering "wrong" input, the user can enter some "smart" input like this:

UserID: - _____

Then, the SQL statement will look like this:

SELECT * FROM Users WHERE UserId = 105 OR 1=1;

The SQL above is valid and will return ALL rows from the "Users" table, since OR 1=1 is always TRUE.

Does the example above look dangerous? What if the "Users" table contains names and passwords?

The SQL statement above is much the same as this:

SELECT UserId, Name, Password FROM Users WHERE UserId = 105 or 1=1;

A hacker might get access to all the user names and passwords in a database, by simply inserting 105 OR 1=1 into the input field.

SQL Injection Based on ""="" is Always True

Here is an example of a user login on a web site:

Username: - _____

Password: - _____

Example

uName = getRequestString("username");
uPass = getRequestString("userpassword");

sql = 'SELECT * FROM Users WHERE Name ="' + uName + '" AND Pass ="' + uPass + '"'

Result

SELECT * FROM Users WHERE Name ="John Doe" AND Pass ="myPass"

A hacker might get access to user names and passwords in a database by simply inserting " OR ""=" into the username or password text box:

Username: - _____

Password: - _____

The code at the server will create a valid SQL statement like this:

Result

SELECT * FROM Users WHERE Name ="" or ""="" AND Pass ="" or ""=""

The SQL above is valid and will return all rows from the "Users" table, since OR ""="" is always TRUE.

SQL Injection Based on Batched SQL Statements

Most databases support batched SQL statement.

A batch of SQL statements is a group of two or more SQL statements, separated by semicolons.

The SQL statement below will return all rows from the "Users" table, then delete the "Suppliers" table.

Example

SELECT * FROM Users; DROP TABLE Suppliers

Look at the following example:

Example

txtUserId = getRequestString("UserId");
txtSQL = "SELECT * FROM Users WHERE UserId = " + txtUserId;

And the following input:

User id: - _____

The valid SQL statement would look like this:

Result

SELECT * FROM Users WHERE UserId = 105; DROP TABLE Suppliers;