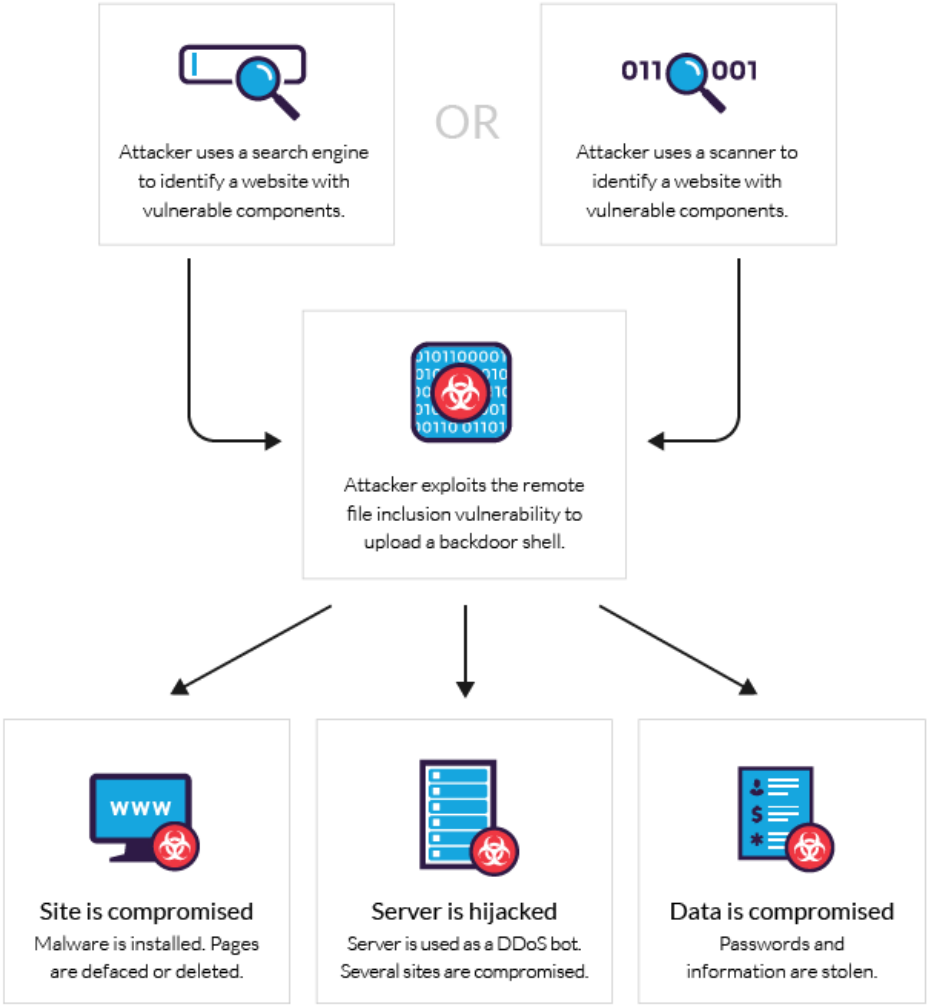
Task-4
29/8/2023

Hiya Sharma
21BCY10078

Any 10 Web Server Attacks-

1.Remote File Execution (RCE)-

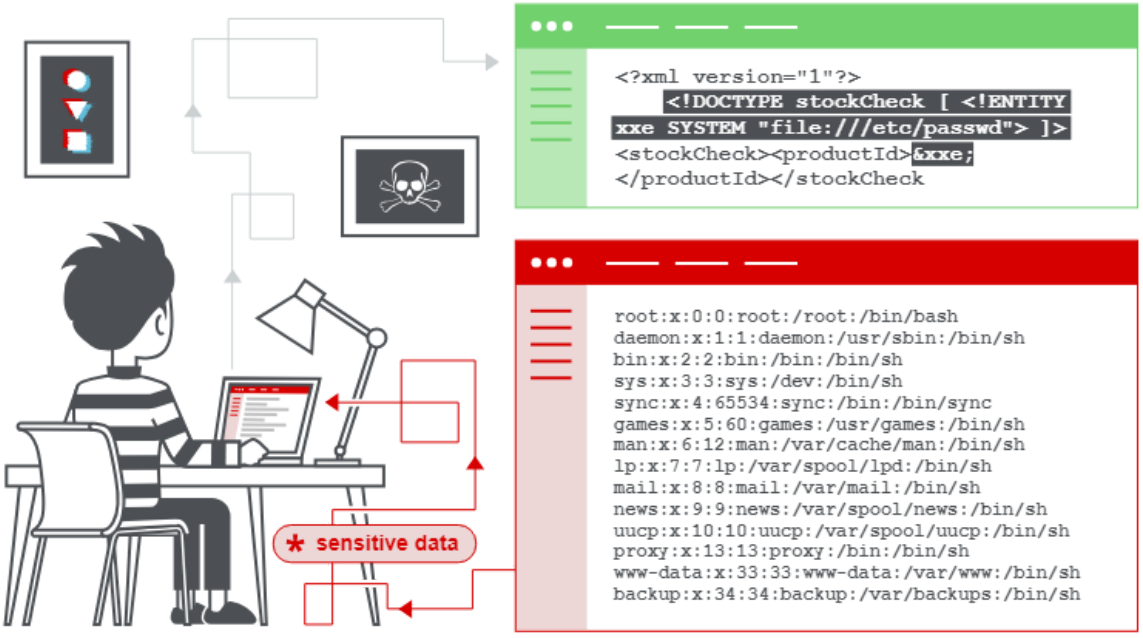
Attackers exploit vulnerabilities to execute malicious code on a web server, potentially taking control of the server or stealing data.
Remote file inclusion (RFI) is an attack targeting vulnerabilities in web applications that dynamically reference external scripts. The perpetrator's goal is to exploit the referencing function in an application to upload malware (e.g., backdoor shells) from a remote URL located within a different domain.
The consequences of a successful RFI attack include information theft, compromised servers and a site takeover that allows for content modification.



Remote File Inclusion (RFI) occurs when web applications allow user-input data, like URL parameters, to specify files to include. Attackers exploit this vulnerability to insert malicious files hosted on remote servers, potentially gaining control over the web server and its data. RFI can lead to arbitrary code execution and various malicious actions. Prevention measures include input validation, whitelisting allowed files, disabling remote includes, and setting proper file permissions. Combining security measures like Web Application Firewalls and regular audits helps detect and prevent RFI attacks. While there are legitimate uses for remote file inclusion, they should be carefully managed to ensure security. Developers and administrators should prioritize RFI prevention to safeguard web applications and servers.

2. XML External Entity (XXE) Attack:

XML external entity injection (also known as XXE) is a web security vulnerability that allows an attacker to interfere with an application's processing of XML data. It often allows an attacker to view files on the application server filesystem, and to interact with any back-end or external systems that the application itself can access.
In some situations, an attacker can escalate an XXE attack to compromise the underlying server or other back-end infrastructure, by leveraging the XXE vulnerability to perform server-side request forgery (SSRF) attacks.



How do XXE vulnerabilities arise?

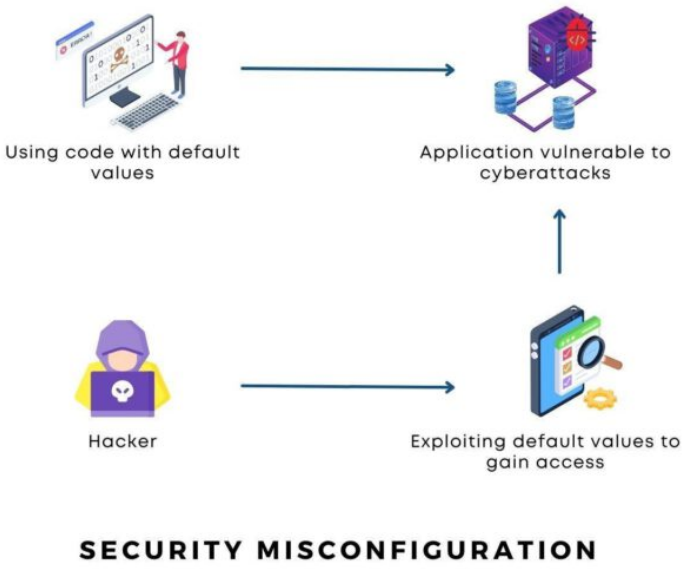
Some applications use the XML format to transmit data between the browser and the server. Applications that do this virtually always use a standard library or platform API to process the XML data on the server. XXE vulnerabilities arise because the XML specification contains various potentially dangerous features, and standard parsers support these features even if they are not normally used by the application.

An XML External Entity (XXE) attack is a security vulnerability that occurs when an application mishandles XML data from untrusted sources. Attackers craft malicious XML documents with external entity references to disclose sensitive files or disrupt the application. To prevent XXE attacks:

- Validate input sources.
- Use secure XML parsers.
- Whitelist safe entities.
- Keep software updated.
- Deploy Web Application Firewalls.
- Harden server configurations.
- Perform regular security testing.

3.Security misconfiguration

Security misconfiguration occurs when security settings are not properly defined, making systems or applications vulnerable to cyber attacks. It's a common cause of data breaches and can happen in various computing systems, including cloud infrastructure. Frameworks and open-source code can make development easier but may come with complex or insecure default configurations, increasing the risk. This vulnerability is often easy for hackers to detect and exploit, potentially affecting the entire application stack, from network services to databases and everything in between.



Common causes of security misconfiguration include:

- Inadequate Default Settings: Software and systems may come with default settings that prioritize ease of use over security. Failing to modify these defaults can leave systems vulnerable.
- Improper Permissions: Incorrectly assigning permissions or access controls can lead to unauthorized users gaining entry to sensitive data or functionality.
- Outdated Software: Failure to update and patch software can result in known vulnerabilities remaining unaddressed.
- Unused Features: Leaving unnecessary features and services enabled can increase the attack surface and expose potential vulnerabilities.
- Lack of Security Testing: Insufficient testing for security flaws during development can result in undetected misconfigurations.

4.Zero-Day Exploits-

A zero-day (0day) exploit is a cyber attack targeting a software vulnerability which is unknown to the software vendor or to antivirus vendors. The attacker spots the software vulnerability before any parties interested in mitigating it, quickly creates an exploit, and uses it for an attack. Such attacks are highly likely to succeed because defenses are not in place. This makes zero-day attacks a severe security threat. Attackers leverage previously unknown vulnerabilities (zero-days) in web server software or applications.

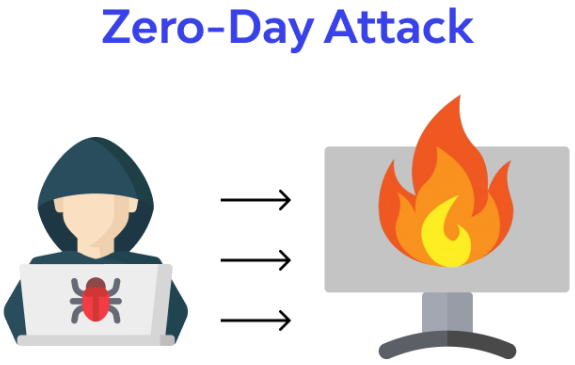
Typical attack vectors include Web browsers, which are common targets due to their ubiquity, and email attachments that exploit vulnerabilities in the application opening the attachment, or in specific file types such as Word, Excel, PDF or Flash.
A related concept is zero-day malware — a computer virus for which specific antivirus software signatures are not yet available, so signature-based antivirus software cannot stop it.

Zero-day vulnerability detection

Detecting zero-day vulnerabilities, which have no patches or antivirus signatures, can be challenging but essential for cybersecurity. Here are some ways to do it:

- Vulnerability Scanning: Use security tools to scan software code and attempt to find new vulnerabilities. It's not foolproof but can help identify some zero-day exploits.
- Patch Management: Quickly apply software patches and updates when vulnerabilities are discovered. This reduces the risk of attacks, but it relies on timely vendor response and patch deployment.
- Input Validation and Sanitization: Implement strict input validation to prevent vulnerabilities. Web Application Firewalls (WAFs) can help filter out malicious inputs.
- Runtime Application Self-Protection (RASP): Deploy RASP agents within applications to analyze request payloads and defend against threats in real-time.
- Zero-Day Initiative: Encourage security researchers to report vulnerabilities responsibly through a rewards program rather than selling them on the black market. This helps vendors patch vulnerabilities before hackers can exploit them.

These strategies help organizations proactively address zero-day vulnerabilities and enhance their overall security posture.



5.Man-in-the-Middle (MitM) Attack:

A Man-in-the-Middle (MitM) attack is a cybersecurity attack where an attacker secretly intercepts and possibly alters the communication between two parties without their knowledge. In a MitM attack, the attacker positions themselves between the communication path of the two parties, becoming an unauthorized intermediary.

Here's how a MitM attack typically works:

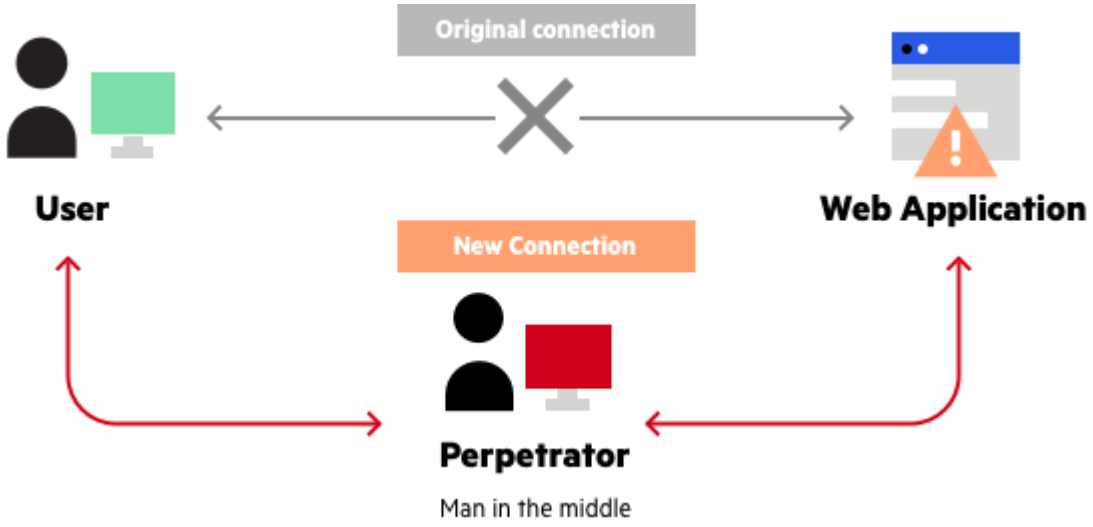
- Interception: The attacker secretly intercepts communication between a sender (Party A) and a receiver (Party B) without their knowledge.
- Interception Techniques: Attackers can employ various techniques to intercept communication, such as eavesdropping on unencrypted network traffic, exploiting vulnerabilities in network protocols, or setting up rogue Wi-Fi access points.
- Monitoring and Alteration: While intercepting the communication, the attacker can monitor the data being exchanged and, in some cases, modify or inject malicious content into the communication.
- Concealment: The attacker attempts to remain hidden, making it difficult for both parties to detect their presence.

MitM attacks pose significant risks, as they can lead to various malicious activities, including:

- Data Theft: Attackers can steal sensitive information, such as login credentials, financial details, or confidential documents, exchanged between Party A and Party B.
- Data Tampering: Attackers can alter the data being transmitted, potentially causing misinformation or damaging the integrity of the communication.
- Identity Theft: MitM attacks can facilitate impersonation, where the attacker pretends to be one of the parties involved in the communication.

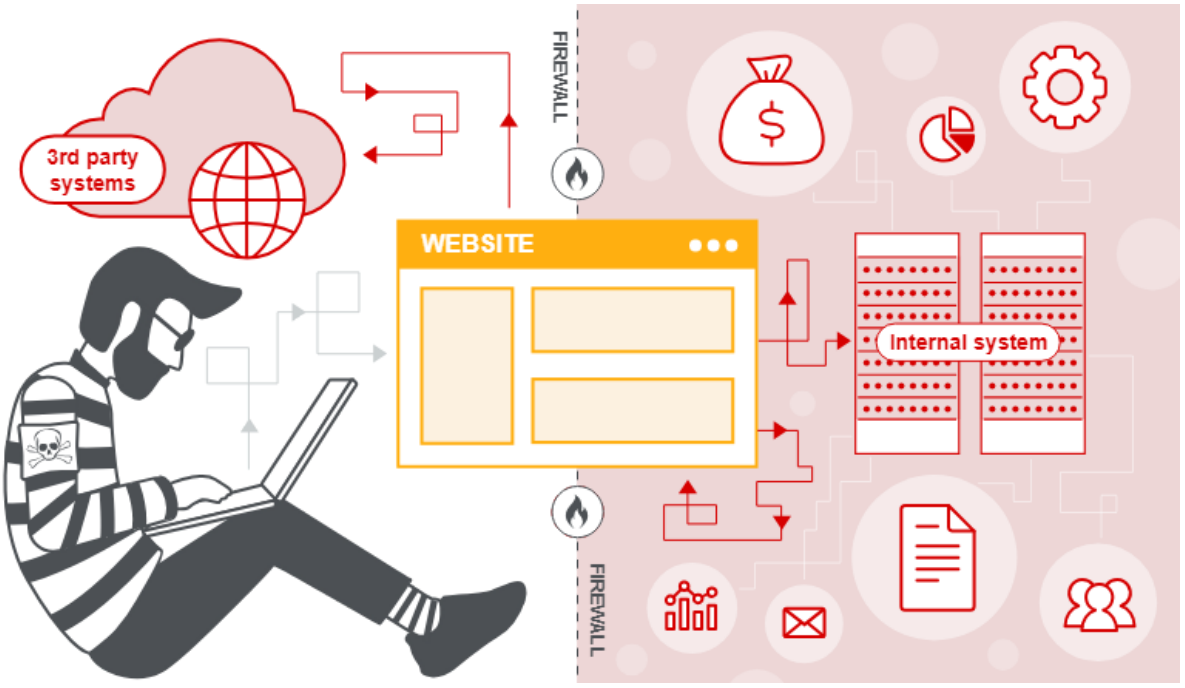
Mitigation and prevention of MitM attacks involve several security measures:

- Encryption: Implement strong encryption protocols (e.g., HTTPS for web communication) to protect data in transit, making it difficult for attackers to decipher intercepted information.
- Secure Authentication: Use secure authentication methods, such as two-factor authentication (2FA), to verify the identity of the parties involved.
- Certificate Validation: Ensure proper validation of digital certificates to prevent attackers from using rogue certificates to impersonate legitimate entities.
- Public Key Infrastructure (PKI): Implement PKI solutions to securely manage digital certificates and keys.
- Network Segmentation: Segment networks to limit the attack surface and reduce the chances of attackers gaining access to sensitive communication channels.
- Security Awareness: Educate users about the risks of MitM attacks and the importance of verifying the authenticity of communication channel.



6.Server-side request forgery (SSRF):

Server-side request forgery is a web security vulnerability that allows an attacker to cause the server-side application to make requests to an unintended location.
In a typical SSRF attack, the attacker might cause the server to make a connection to internal-only services within the organization's infrastructure. In other cases, they may be able to force the server to connect to arbitrary external systems. This could leak sensitive data, such as authorization credentials.



Attackers trick a web server into making requests to internal or external resources, often leading to unauthorized access to sensitive data or systems.

A successful SSRF attack can often result in unauthorized actions or access to data within the organization. This can be in the vulnerable application, or on other back-end systems that the application can communicate with. In some situations, the SSRF vulnerability might allow an attacker to perform arbitrary command execution.

This can lead to data access, attacks on external systems, port scanning, or even remote code execution. Prevention involves input validation, whitelisting allowed URLs, using security controls, network segmentation, least privilege access, and keeping software updated.

7.Directory Traversal-

Directory traversal is a type of HTTP exploit in which a hacker uses the software on a web server to access data in a directory other than the server's root directory. If the attempt is successful, the threat actor can view restricted files or execute commands on the server.

This type of attack is commonly performed using web browsers. Any server that fails to validate input data from web browsers is vulnerable to a directory traversal attack. Directory traversal is also known as directory climbing, backtracking and file path traversal vulnerabilities.

IT security professionals minimize the risk of a directory traversal with the following techniques:

- careful web server programming;
- installation of software updates and patches;
- filtering of input from browsers; and
- using vulnerability scanners.

How does directory traversal work?

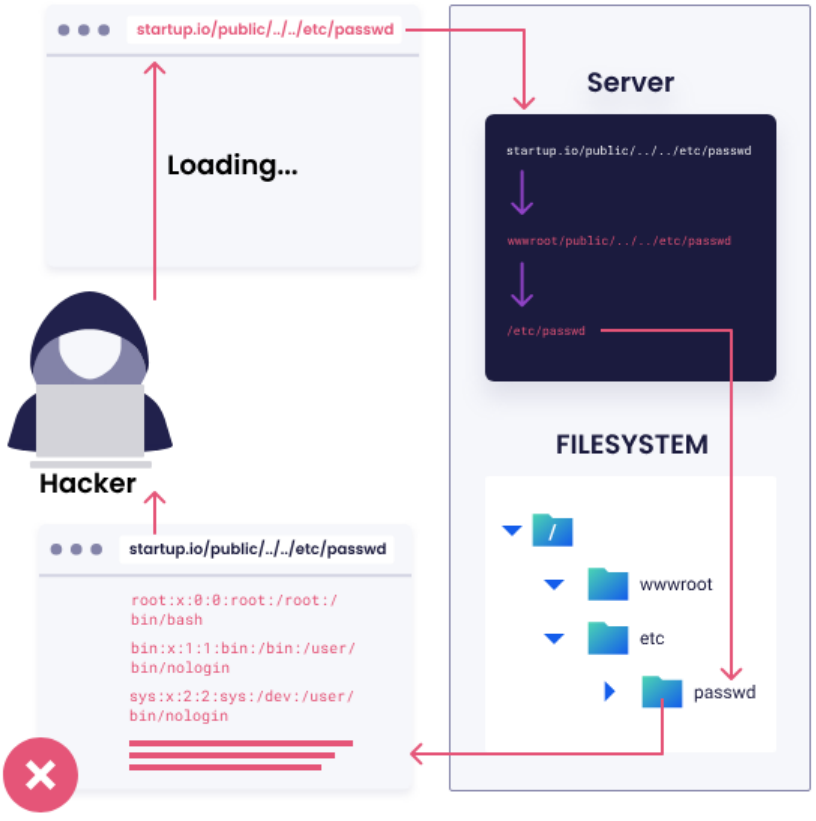
Hackers use guesswork to find paths to restricted files on a web server. However, a skilled hacker can search the directory tree and easily execute this type of attack on an inadequately protected server.
Only a few resources are needed to perform a directory traversal attack, including the following ones:

- access to a web browser;
- some knowledge about where to find directories; and

- basic knowledge of Hypertext Transfer Protocol (HTTP) requests.

What can an attacker do with directory traversal?

Once attackers access the root directory, they can enter other parts of the computer system. They may also be able to read and write arbitrary files on the server, enabling them to manipulate applications and associated data, read sensitive information like password files or take control of the server. Normally, users are unable to access any files outside of the web root folder.

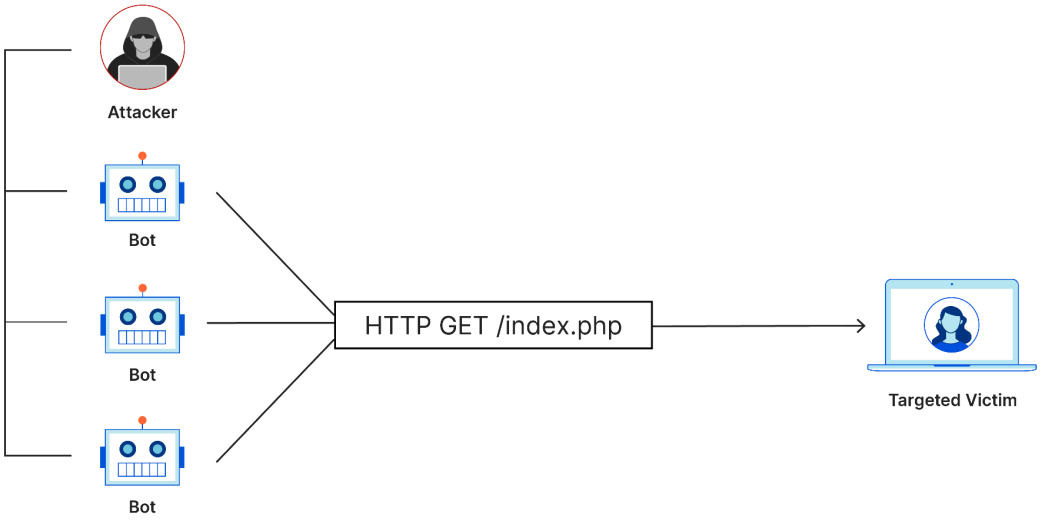


8.Distributed Denial of Service (DDoS)

A Distributed Denial of Service (DDoS) attack is a malicious attempt to disrupt the normal functioning of a targeted server, service, or network by overwhelming it with a flood of internet traffic. In a DDoS attack, multiple compromised computers or devices, often part of a botnet (a network of infected machines), are used to generate a massive volume of requests or traffic directed at the target, causing it to become unavailable to legitimate users.

Here's how a DDoS attack typically works:

- Botnet Coordination: The attacker gains control over a network of compromised devices, often using malware or other means. These devices can include computers, servers, Internet of Things (IoT) devices, or even smartphones.
- Traffic Generation: The attacker commands the compromised devices to flood the target with a large volume of requests or traffic. This can include HTTP requests, UDP or TCP packets, or other forms of network traffic.
- Overwhelm the Target: The sheer volume of traffic generated by the botnet overwhelms the target's resources, such as bandwidth, processing power, or memory. As a result, the target becomes unresponsive or extremely slow in serving legitimate requests.
- Service Disruption: During the DDoS attack, legitimate users may find it difficult or impossible to access the targeted website, application, or service, leading to service disruption.
- Mitigation and Defense: Organizations often employ various DDoS mitigation techniques, such as traffic filtering, rate limiting, or using content delivery networks (CDNs), to absorb or block the malicious traffic and maintain service availability.



9.Command Injection-

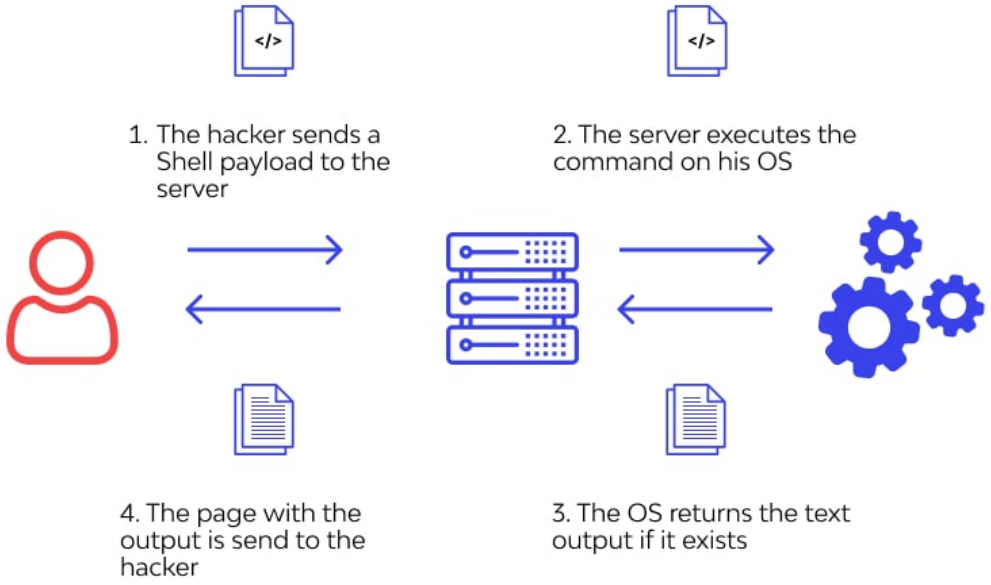
Command Injection is a web server attack that occurs when an attacker is able to manipulate or inject malicious commands into an input field or parameter of a web application. These commands are executed by the server, potentially leading to unauthorized access, data manipulation, or even full server compromise.

How Command Injection Works:

- Vulnerable Input: The web application allows user input, such as data submitted through forms or URL parameters, to be directly passed to a system command without proper validation or sanitization.
- Malicious Input: An attacker submits or injects malicious commands into the input fields. For example, in a search box, they might enter a command like ;ls in an attempt to list files on the server.
- Command Execution: The web application, unaware of the malicious input, concatenates or uses the user-supplied data to construct a system command. This command is then executed by the server.
- Exploitation: Depending on the attacker's skill and the server's configuration, the injected command can lead to various consequences, such as retrieving sensitive data, modifying files, executing arbitrary code, or compromising the entire server.

Prevention of Command Injection Attacks:
To prevent Command Injection attacks, web applications should implement the following security measures:

- Input Validation and Sanitization: Validate and sanitize all user inputs before using them in system commands. Ensure that only expected and safe characters are allowed.
- Parameterized Queries: Use parameterized queries or prepared statements when interacting with databases to avoid concatenating user input directly into SQL queries.
- Least Privilege: Configure the server to run with the least privilege necessary. Restrict the commands and operations that the web application can execute.
- Web Application Firewall (WAF): Implement a Web Application Firewall that can detect and block known Command Injection attack patterns.
- Security Awareness: Educate developers and users about the risks of Command Injection and the importance of secure coding practices.
- Regular Updates: Keep all software and libraries up to date to address known vulnerabilities and security issues.
- Logging and Monitoring: Set up logging and monitoring to detect and respond to unusual or suspicious activities that might indicate a Command Injection attempt.



10.Session Hijacking/Session Fixation:

Session fixation and session hijacking are both attacks that attempt to gain access to a user's client and web server session. In the session hijacking attack, the attacker attempts to steal the ID of a victim's session after the user logs in. In the session fixation attack, the attacker already has access to a valid session and tries to force the victim to use that particular session for his or her own purposes. The session fixation attack "fixes" an established session on the victim's browser, so the attack starts before the user logs in.

Session fixation attacks are designed to exploit authentication and session management flaws. Any system that allows one person to fixate another person's session identifier is vulnerable to this type of attack. Most session fixation attacks are web-based, and most rely on session identifiers being accepted from URLs or POST data.

Prevention and Mitigation:
To prevent and mitigate Session Hijacking and Session Fixation attacks, web applications can implement the following security measures:

-
- Secure Session Management: Use strong session management practices, including random and unpredictable session IDs, and regenerate session IDs upon login.
- Transport Layer Security (TLS): Encrypt all data transmitted between the user's browser and the web server using HTTPS to prevent session interception during transit.

- Secure Cookies: Use secure and HttpOnly flags for session cookies to prevent JavaScript from accessing session data and ensure that cookies are only transmitted over secure connections.
- Timeouts: Implement session timeouts and automatically log users out after a period of inactivity.
- Logout Functionality: Provide users with the ability to log out and invalidate their session.
- IP Verification: Monitor and check the user's IP address during a session to detect suspicious changes in IP addresses.
- User Education: Educate users about the risks of clicking on suspicious links or sharing their session IDs.