TASK-4

K. Naveen Abhiram

21BCE7357

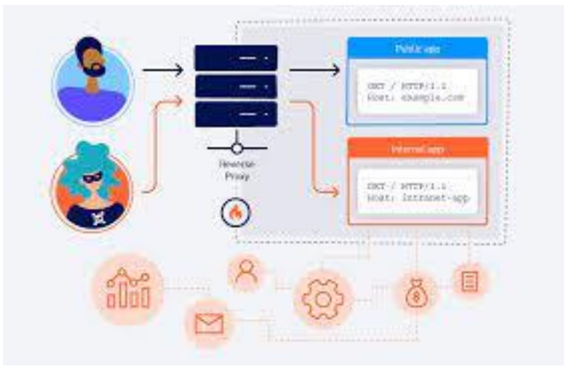
Understanding any Top 10 web applications Vulnerabilities (other than Top 10 OWASP) write a paragraph about that and add an image to the respective vulnerability

1. HTTP Header Injection:

A web vulnerability known as HTTP Header Injection enables attackers to change HTTP response headers that are transmitted from a web application to a user's browser. Attackers are able to bypass security measures, create cookies, lead visitors to dangerous websites, inject malware, and more by inserting malicious text into these headers. For instance, they could alter headers to appear to be a reliable source, which might result in data breaches or phishing scams. In order to prevent this kind of injection and guarantee the integrity and security of HTTP headers and the entire web application, proper input validation and output encoding are crucial.

Mitigation:

Use online security frameworks or libraries that automatically encrypt or escape header information to reduce the risk of HTTP Header Injection. Update and patch web servers and programs often to fix known header-related vulnerabilities.

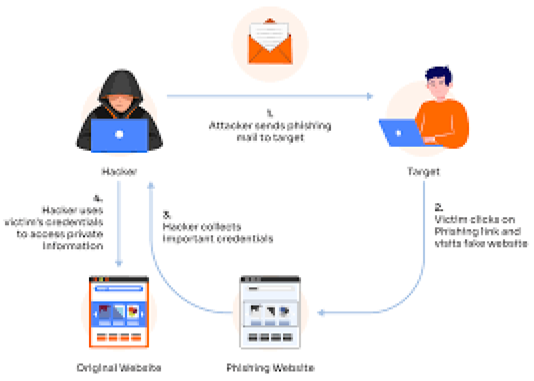


2. Content Spoofing:

An online security flaw called content spoofing, often referred to as content injection or text injection, enables attackers to change the information that visitors see on a website. This may entail inserting false or deceptive content onto online pages, such as phishing links or bogus messages. Attacks that spoof content can undermine confidence, deceive people, and trigger a variety of destructive behaviours. Strict input validation, output encoding, and security headers must all be used by site developers in order to prevent this issue and assist safeguard users from misleading or malicious material. To successfully identify and mitigate content spoofing vulnerabilities, routine security testing and code reviews are essential.

Mitigation:

Implement stringent user-generated content input validation and encoding, and utilize strong security headers to prevent malicious insertion to reduce the risk of content spoofing. To proactively find and fix vulnerabilities, routine security testing and code reviews are crucial.



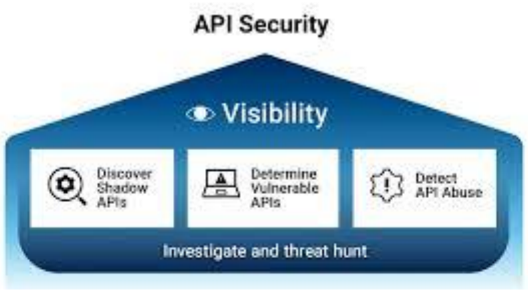
Hacker
1. Attacker sends phishing mail to target
Target
2. Victim clicks on Phishing link and visits fake website
3. Hacker collects important credentials
4. Hacker uses victim's credentials to access private information
Original Website
Phishing Website

3. API Security Vulnerabilities:

Application Programming Interfaces (APIs) are used in online and mobile apps, however there are security flaws and hazards that come with their use. These flaws have the potential to reveal private information, expose user accounts, or facilitate malevolent actions. Insufficient authentication, excessive data exposure, and a lack of rate limits are just a few examples of common security problems with APIs. Developers should build robust authentication and authorisation systems, authenticate user inputs, monitor and limit API usage, do extensive security testing and code reviews to discover and resolve any holes proactively in order to reduce API security risks. Additionally, adopting API security frameworks and tools can improve API security and guard against hostile actors' exploitation.

Mitigation:

Implementing robust authentication, input validation, rate restriction, doing security testing, and keeping documentation for continuous security awareness are all part of mitigating API security issues.

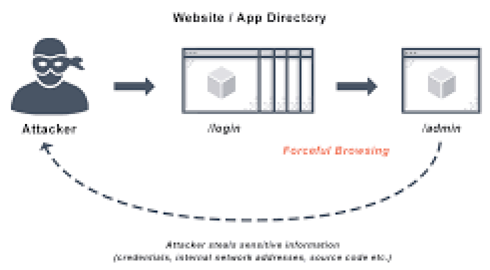


4. Predictable Resource Location:

A web application vulnerability known as Predictable Resource Location occurs when resource identifiers such as URLs follow a pattern or are simple to guess. By foreseeing or brute-forcing resource locations, attackers might take advantage of this to get unauthorized access to confidential information or functionality.

Mitigation:

To mitigate this kind of attack, robust access constraints, unexpected resource naming practices, and appropriate authentication and permission systems are used.

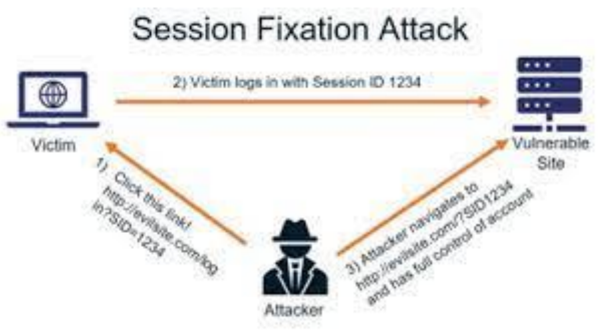

Website / App Directory
Attacker
/login
/admin
Forceful Browsing
Attacker steals sensitive information
(credentials, internal network addresses, source code etc.)

5. Session Fixation:

A web application vulnerability called "Session Fixation" allows an attacker to force the victim to utilize a preset session by setting the user's session identifier (such as a session cookie) before the victim checks in. Once the victim logs in, the attacker can take over their session and possibly acquire access to the victim's account without the victim's knowledge.

Mitigation:

To mitigate this type of attack, use strong session management procedures and generate new session identifiers at login.

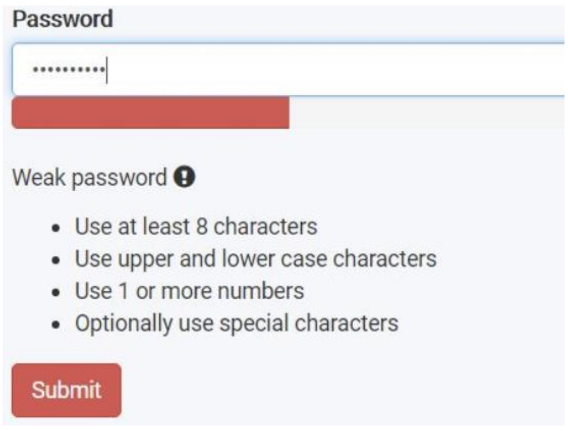

Session Fixation Attack
2) Victim logs in with Session ID 1234
Victim
Vulnerable Site
1) Click this link! http://evilsite.com/login?SID=1234
3) Attacker navigates to http://evilsite.com/?SID1234 and has full control of account
Attacker

6. Inadequate Error Handling:

A web application vulnerability known as inadequate error handling occurs when error messages or responses disclose private information or internal facts about the design and functionality of the application. Attackers can make use of this information to identify the system's vulnerabilities and possibly conduct focused attacks.

Mitigation:

Implementing bespoke error handling allows users to receive generic error messages while safely logging in-depth error data for system administrators to evaluate. This improves overall security by preventing the disclosure of sensitive data.



Password

••••••••••

Weak password

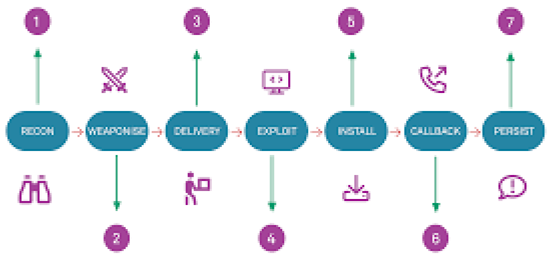
- Use at least 8 characters
- Use upper and lower case characters
- Use 1 or more numbers
- Optionally use special characters

7. Information Disclosure:

Web application vulnerabilities known as information disclosure occur when private information or system specifics are unintentionally made available to users or attackers. This can involve disclosing internal file locations, error messages from databases, or private user data. Such releases can be used by attackers to gather information for specialized assaults.

Mitigation:

To prevent unauthorized access to critical information and to make sure that error messages and responses reveal as little as possible about the internal workings of the program, mitigation solutions entail providing strong access controls, error handling, and input validation.

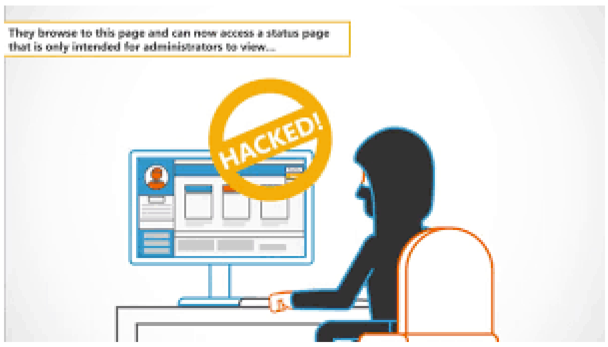


8. Missing Function-Level Access:

A web application vulnerability called Missing Function-Level Access Control allows users to access features or functions they shouldn't be allowed to utilize. Attackers can take advantage of this to perform operations only approved by privileged users or obtain unauthorized access to critical portions of the program.

Mitigation:

In order to ensure that users may only access resources or functionalities they are permitted to use, and to stop unauthorized activities and data exposure, adequate access controls and permission checks must be implemented at both the front-end and back-end levels.



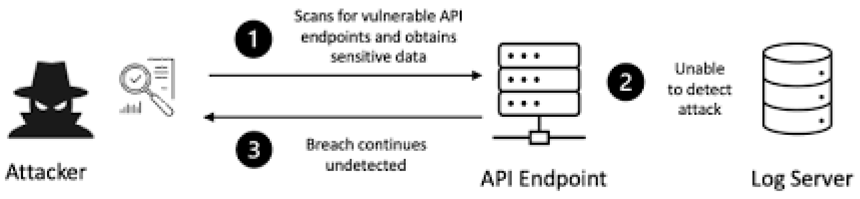
They browse to this page and can now access a status page that is only intended for administrators to view...
HACKED!

9. Insufficient Logging and Monitoring:

A web application vulnerability is when an application does not sufficiently record and monitor security events and user activity. This is known as inadequate logging and monitoring. Due to this lack of visibility, it is challenging to immediately identify and address security problems and suspicious activity. Attackers can take advantage of this by sneaking up on targets and committing crimes unnoticed. To quickly identify and address security risks and improve overall application security, mitigation calls for the implementation of thorough recording of pertinent security events, real-time monitoring, and incident response procedures.

Mitigation:

Implement thorough recording of security events, including authentication, access, and error logs, to mitigate insufficient logging and monitoring. Create incident response protocols to deal with security events successfully and set up real-time monitoring alerts to quickly identify and handle suspicious activity. To keep up with changing risks, examine and improve recording and monitoring procedures often.



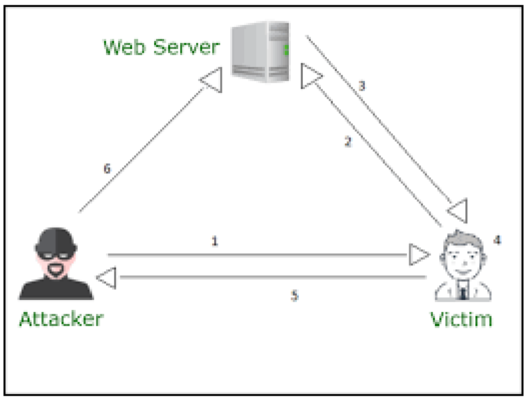
1
Scans for vulnerable API endpoints and obtains sensitive data
2
Unable to detect attack
3
Breach continues undetected
Attacker
API Endpoint
Log Server

10. DOM-Based Cross Site Scripting (DOM-XSS):

An online application vulnerability known as DOM-Based Cross-Site Scripting (DOM XSS) occurs when client-side JavaScript code modifies the Document Object Model (DOM) to insert malicious scripts. DOM XSS is harder to identify and defend against than standard XSS assaults because it happens when the browser interprets user-supplied data in the DOM. Attackers take use of this to run malicious scripts in a user's browser, perhaps stealing data or taking over user sessions.

Mitigation:

To mitigate these threats, one must do extensive input validation and output encoding, use security libraries, and teach developers secure DOM manipulation.


Web Server
Attacker
Victim
1
2
3
4
5
6