**Name: Lakshmi Likitha Malemarpuram**

**Reg no: 21BCE8963**

**Email:** [**likitha.21bce8963@vitapstudent.ac.in**](mailto:likitha.21bce8963@vitapstudent.ac.in)

**DATE: 29-08-2023**

**DAY: TUESDAY**

**TASK 4: 10 Web Server Attacks**

Web Server Attack:

Any attempt by a malicious actor to undermine the security of a Web-based application is referred to as a Web Application Attack or Web Server Attack. Web application attacks can either target the application itself in order to get access to sensitive data, or they can use the application as a staging area for attacks against the program’s users.

There are 5 types of major Web Attacks:

* Denial-of-Service (DoS) / Distributed Denial-of-service (DDoS)
* Web Defacement Attack
* SSH Brute Force Attack
* Cross-site scripting (XSS)
* Directory Traversal
* DNS Server Hijacking
* MITM Attack
* HTTP Response Splitting Attack\
* Server-Side Request Forgery (SSRF)
* XML External Entity (XXE) Injection

1. DENIAL-OF-SERVICE (DOS) / DISTRIBUTED DENIAL-OF-SERVICE (DDOS):

* Definition: A Denial-of-Service (DoS) attack is a deliberate attempt to disrupt the normal functioning of a computer network, service, or website by overwhelming it with a flood of traffic. In a Distributed Denial-of-Service (DDoS) attack, multiple compromised computers (botnets) are used to launch the attack, making it more powerful and harder to mitigate.
* Attack Vector: Attackers flood the target network or server with a high volume of incoming traffic, consuming its resources and causing it to become slow or unavailable.
* Motivation: DoS attacks can be carried out for various reasons, including financial gain, political motivations, revenge, competitive advantage, or even simply to cause chaos.
* Impact: DoS and DDoS attacks can lead to service downtime, loss of revenue, reputational damage, and can disrupt critical online services.
* Attack Types: DoS and DDoS attacks come in various forms, including:
  + Volumetric Attacks: Overwhelm the target's bandwidth with a flood of traffic.
  + TCP/UDP Exhaustion: Exploit limitations in the target's ability to handle TCP or UDP connections.
  + Application Layer Attacks: Target specific applications, exploiting vulnerabilities or overwhelming application resources.
* DDoS Amplification: Attackers often use techniques like DNS amplification, NTP amplification, or memcached amplification to make their attacks more powerful by sending small requests that trigger large responses from vulnerable servers.
* Mitigation: To mitigate DoS and DDoS attacks, organizations can:
  + Use traffic filtering and rate limiting mechanisms.
  + Employ Intrusion Detection and Prevention Systems (IDS/IPS).
  + Use content delivery networks (CDNs) to distribute traffic and absorb attacks.
  + Work with Internet service providers (ISPs) to filter attack traffic.
* Botnets: DDoS attacks often involve botnets—networks of compromised computers controlled remotely by the attacker. These botnets amplify the attack's impact.
* Protection: Businesses should implement strong security measures, such as firewalls, load balancers, and proactive monitoring, to detect and respond to DoS and DDoS attacks.
* Legal Implications: Carrying out DoS or DDoS attacks is illegal in many jurisdictions and can result in severe legal consequences for the perpetrators.

2. WEB DEFACEMENT ATTACK:

* Definition: A Web Defacement Attack is a cyberattack in which an attacker unauthorizedly alters the content of a website, replacing it with their own messages, images, or malicious content.
* Motivation: Attackers may deface websites for various reasons, including political statements, ideological reasons, showcasing hacking skills, or spreading misinformation.
* Methods: Attackers exploit vulnerabilities in the website's security, such as weak passwords, outdated software, or poor access controls, to gain unauthorized access and modify the content.
* Impact: Web defacement can lead to damage to an organization's reputation, loss of customer trust, financial losses, and the compromise of sensitive data if the attacker gains access to the website's backend.
* Visible Changes: Attackers typically alter the homepage or prominent pages of a website to display their own messages, images, or banners.
* Prevention and Mitigation: To prevent web defacement attacks, follow these practices:
  + Regularly update and patch web software and plugins.
  + Implement strong and unique passwords for all accounts.
  + Enforce proper access controls and permissions.
  + Monitor website logs and traffic for unusual activity.
  + Regularly backup website data to quickly restore in case of an attack.
* Recovery: If a website is defaced, organizations should:
  + Temporarily take the website offline to prevent further damage.
  + Analyze logs to understand how the attack occurred.
  + Remove the malicious content and replace it with the original content.
  + Close the vulnerability that was exploited by the attacker.
* Legal Implications: Web defacement is illegal and can lead to legal consequences for the attacker, depending on the jurisdiction.
* Security Audits: Conduct regular security audits and penetration testing to identify vulnerabilities that could be exploited for defacement.
* Content Management Systems (CMS): If using a CMS, keep it updated, use reputable plugins and themes, and follow security best practices to prevent vulnerabilities that attackers could exploit.

3. SSH BRUTE FORCE ATTACK:

* Definition: An SSH Brute Force Attack is a type of cyberattack in which attackers systematically attempt to guess the correct username and password combinations to gain unauthorized access to a remote server via SSH (Secure Shell).
* Attack Vector: Attackers use automated tools to iterate through a list of possible usernames and passwords, repeatedly trying to authenticate to the target server.
* Impact: Successful SSH brute force attacks can result in unauthorized access to the server, compromising data integrity, confidentiality, and potentially leading to further network exploitation.
* Common Targets: Servers that allow SSH access, such as Linux servers, network devices, and cloud instances, are common targets for brute force attacks.
* Dictionary Attacks: Attackers often use dictionaries of common passwords, known as password lists, to systematically guess passwords. These lists can include variations, common words, and leaked passwords from previous breaches.
* Slow and Low-Key Attacks: To avoid detection, attackers may employ slow and low-key attack techniques that space out login attempts over time, making it less likely for security systems to trigger alerts.
* SSH Key Authentication: Strongly recommended over passwords, SSH key authentication uses cryptographic key pairs instead of passwords, making brute force attacks ineffective if properly configured.
* Fail2Ban and Intrusion Detection Systems (IDS): These tools automatically block or temporarily ban IP addresses that make multiple failed login attempts, helping to mitigate brute force attacks.
* IP Whitelisting: Restrict SSH access to known and trusted IP addresses, reducing the attack surface for potential attackers.
* Mitigation and Best Practices: Use strong and unique passwords, enforce account lockout policies, disable unused SSH access points, regularly update and patch SSH software, and implement multi-factor authentication (MFA) for an additional layer of security.

4. CROSS SITE SCRIPTING (XSS):

* Definition: Cross-Site Scripting (XSS) is a web security vulnerability where attackers inject malicious scripts into web applications that are then executed in the browsers of unsuspecting users.
* Attack Vector: Attackers inject malicious code (usually JavaScript) into web pages, often through input fields, comments, or URLs. When other users visit the infected page, the code runs in their browsers.
* Impact: XSS attacks can lead to theft of sensitive information (like cookies or session tokens), session hijacking, defacement of websites, or even the spread of malware.
* Types of XSS:
  + Stored XSS: The injected script is permanently stored on the server and served to users who access the affected page.
  + Reflected XSS: The injected script is reflected off a web server and executed immediately in the user's browser.
  + DOM-based XSS: The vulnerability occurs within the Document Object Model (DOM), where client-side scripts manipulate the page's structure.
* Payloads: Attackers use various payloads to exploit XSS, including:
  + Persistent Scripts: Stored in databases, forums, or comments to affect multiple users.
  + Phishing Scripts: Trick users into providing sensitive information.
  + Session Hijacking Scripts: Steal session cookies and impersonate users.
* Mitigation: Preventing XSS requires input validation and output encoding. Input validation checks user inputs for malicious content, and output encoding ensures that any user-generated content is rendered safely.
* Content Security Policy (CSP): Implementing CSP headers can restrict the sources from which scripts can be loaded, mitigating the risk of unauthorized scripts running.
* HTTP Only and Secure Cookies: By setting cookies as HTTP-only and secure, you can limit the potential impact of session theft through XSS attacks.
* Web Application Firewalls (WAFs): WAFs can detect and block known patterns of XSS attacks, offering an additional layer of defense.

Security Best Practices: Regularly audit and test your web applications for XSS vulnerabilities, educate developers about secure coding practices, and keep software and libraries up to date.

5. DIRECTORY TRAVERSAL:

* Definition: Directory Traversal, also known as Path Traversal or Directory Climbing, is a web security vulnerability where an attacker manipulates input to access files and directories that are located outside the intended directory structure.
* Attack Vector: Attackers exploit poorly sanitized user inputs, often through URLs or file paths, to navigate through directories and access files they shouldn't have permission to access.
* Impact: Directory Traversal attacks can lead to unauthorized disclosure of sensitive files, data theft, and potential remote code execution if the attacker can access and execute executable files.
* Traversal Characters: Attackers use special characters like "../" to move up a directory level in the file system. By repeating these characters, they can navigate through directories.
* Web Application Context: Directory Traversal is common in web applications that allow users to access files or resources using user-provided input, such as file download functionality.
* File Inclusion Vulnerabilities: Directory Traversal is often associated with file inclusion vulnerabilities (like Local File Inclusion or Remote File Inclusion) where attackers exploit paths to include malicious files.
* Mitigation: To prevent Directory Traversal attacks, validate and sanitize user inputs thoroughly. Use whitelists to specify allowed file paths and ensure that users have access only to intended resources.
* Input Validation: Implement strict input validation, especially for file paths and directory names, to prevent the injection of traversal characters.
* Access Controls: Ensure that the web server's file system permissions are properly configured to prevent unauthorized access to sensitive files.
* Security Best Practices: Regularly test applications for Directory Traversal vulnerabilities through security assessments, follow secure coding practices, and stay informed about the latest attack techniques.

6. DNS SERVER HIJACKING:

* Definition: DNS Server Hijacking, also known as DNS Spoofing or DNS Cache Poisoning, is a cyberattack where attackers redirect legitimate domain name requests to malicious IP addresses, leading users to fraudulent websites.
* Attack Vector: Attackers compromise DNS servers, routers, or network infrastructure to manipulate the DNS responses that users receive when querying domain names.
* Impact: DNS Server Hijacking can lead to users unknowingly visiting malicious websites, exposing them to phishing attacks, malware downloads, or other forms of exploitation.
* Domain Impersonation: Attackers can direct users to fake websites that closely resemble legitimate sites, tricking users into entering sensitive information.
* DNS Cache Poisoning: Attackers exploit vulnerabilities in DNS caching mechanisms to inject malicious entries into DNS caches. Subsequent requests from users can then lead to redirection.
* Pharming Attacks: DNS Server Hijacking is often used in pharming attacks, where attackers target a large number of users to steal sensitive information or spread malware.
* TTL Exploitation: Attackers can manipulate the Time-to-Live (TTL) value in DNS records to prolong the duration of a hijacked entry in the DNS cache.
* Mitigation: Implement DNSSEC (Domain Name System Security Extensions) to ensure the authenticity and integrity of DNS responses. Use reputable DNS resolvers and regularly update router firmware.
* Subdomain Takeover: Attackers identify unconfigured or abandoned subdomains and hijack them, redirecting them to malicious content.
* Security Best Practices: Regularly monitor DNS logs, implement intrusion detection systems, educate users about phishing risks, and maintain up-to-date DNS server software to prevent DNS Server Hijacking.

7. MITM ATTACK:

* Definition: A Man-in-the-Middle (MitM) attack is a type of cyberattack where an attacker intercepts and possibly alters the communication between two parties without their knowledge.
* Attack Vector: Attackers can execute MitM attacks by placing themselves between the sender and receiver, often on public Wi-Fi networks or by compromising routers or switches.
* Packet Sniffing: MitM attackers use packet sniffing tools to intercept and read the data transmitted between parties, which can include sensitive information like passwords or credit card numbers.
* Data Alteration: Attackers can modify the intercepted data before forwarding it to the intended recipient. This alteration can include injecting malicious content, changing transaction details, or redirecting traffic.
* Session Hijacking: By intercepting session tokens or cookies, attackers can impersonate a legitimate user and gain unauthorized access to a system or application.
* SSL/TLS Stripping: Attackers can force a connection to use unencrypted HTTP instead of secure HTTPS, making it easier to intercept and manipulate data.
* Public Wi-Fi Vulnerability: Public Wi-Fi networks are common environments for MitM attacks due to their lack of encryption and the ease of intercepting traffic.
* Secure Connections: MitM attacks are more challenging on secure connections that use SSL/TLS encryption. However, attackers can still exploit vulnerabilities in outdated encryption protocols or weak configurations.
* Prevention: To prevent MitM attacks, always use secure and updated protocols (like TLS 1.2 or later), avoid public Wi-Fi for sensitive transactions, and be cautious of certificate warnings.
* Security Best Practices: Employing certificate pinning, using virtual private networks (VPNs), implementing HTTPS Everywhere, and staying informed about the latest encryption standards are crucial for mitigating MitM attacks.

8. HTTP RESPONSE SPLITTING ATTACK:

* Definition: HTTP Response Splitting is a web security vulnerability where an attacker manipulates input to insert extra HTTP headers or control characters into a server's response. This can lead to various security issues.
* Attack Vector: Attackers exploit this vulnerability by injecting newline characters and other specially crafted inputs into the application's input fields or URLs.
* Impact: HTTP Response Splitting can lead to multiple security issues, including cache poisoning, cross-site scripting (XSS), session fixation, and exposure of sensitive information.
* Caching Exploitation: Attackers can manipulate cache systems to store malicious content that gets served to other users, potentially leading to XSS attacks or unauthorized data exposure.
* Cookie Manipulation: Attackers can split HTTP responses to inject malicious cookies into the user's browser, which can then be used for session fixation or other attacks.
* Redirection Attacks: Attackers can manipulate the Location header in the response to redirect users to malicious websites or phishing pages.
* Cross-Site Scripting (XSS) via Splitting: Attackers can split responses to inject malicious scripts that get executed in the user's browser, leading to XSS attacks.
* Denial of Service: By manipulating headers and responses, attackers can cause caching or content-related issues, leading to service disruption or degraded performance.
* Mitigation: Preventing HTTP Response Splitting requires proper input validation and encoding of user inputs. Security libraries and frameworks often have built-in protections against this vulnerability.
* Security Best Practices: Developers should follow secure coding practices, perform security testing (such as vulnerability scanning and penetration testing), and keep software up to date to mitigate HTTP Response Splitting and related issues.

9. Server-Side Request Forgery (SSRF):

* Definition: SSRF is a web security vulnerability where an attacker tricks a server into making unauthorized requests to internal resources or other external systems, often within the same organization's network.
* Attack Vector: Attackers manipulate user inputs (such as URLs) that are used to make requests from the server. By controlling these inputs, they can target specific resources.
* Impact: SSRF can lead to data exposure, unauthorized access to internal services, remote code execution, and potentially facilitate attacks like port scanning on internal systems.
* Blind SSRF: In some cases, attackers might not receive direct responses from the requests made, making it "blind." However, they can still infer sensitive information based on the behavior of the server's responses.
* Exploitation Scenarios: Attackers can use SSRF to bypass firewalls, access internal databases, fetch sensitive files, perform actions on internal services, and exploit other services that are not meant to be directly accessible.
* Mitigation: Preventing SSRF requires proper input validation and sanitization. Whitelisting allowed URLs, using security controls like Web Application Firewalls (WAFs), and restricting network access are common mitigation strategies.
* Proxy Bypass: Some SSRF attacks can bypass security measures if the server's outgoing requests are routed through a proxy server. Attackers can craft requests to bypass the proxy's controls.
* Cloud SSRF: Cloud environments are also vulnerable to SSRF attacks. Attackers can target cloud metadata services to extract sensitive information or access internal resources.
* Indirect Impacts: SSRF can be used as part of a larger attack chain. For example, gaining access to internal systems might help attackers in other attacks, such as privilege escalation.
* Security Best Practices: Regular security assessments, secure coding practices, input validation, and educating developers and users about the risks of SSRF are crucial in minimizing this vulnerability's impact.

10. XML External Entity (XXE) Injection:

* Definition: XML External Entity (XXE) Injection is a vulnerability that occurs when an attacker injects malicious XML entities into an XML input processed by an application, leading to information disclosure, denial of service, or remote code execution.
* Attack Vector: Attackers exploit XXE by injecting specially crafted XML content into input fields, XML documents, or API calls that are parsed by the server.
* Impact: XXE attacks can lead to the disclosure of sensitive files, data from internal systems, DoS by resource consumption, and even remote code execution depending on the vulnerability's severity.
* XML Entities: Attackers often use external entities in XXE attacks. These are declarations that reference external resources, which the attacker can control, leading to unintended behaviors.
* Blind XXE: In some cases, attackers might not receive direct responses from the server that indicates success or failure. This is known as "blind" XXE, and attackers infer information from the application's behavior.
* Exploitation Scenarios: Attackers can exploit XXE vulnerabilities to retrieve files, fetch sensitive data from internal systems, execute arbitrary code on the server, and carry out DoS attacks.
* Mitigation: To prevent XXE, developers should disable external entity processing, validate and sanitize user inputs, and use parsers that are not vulnerable to XXE. Web Application Firewalls (WAFs) can also help.
* Out-of-Band XXE: In this variant, attackers trigger the exploitation using external entities, which initiate requests to external servers controlled by the attacker. This can help bypass certain defenses.
* Content-Type Attacks: Attackers can manipulate the Content-Type header to force the server to process the input as XML, even if it wasn't intended to be treated as XML.
* Security Best Practices: Regular code reviews, secure coding guidelines, and security testing (e.g., vulnerability scanning, penetration testing) are important to identify and address XXE vulnerabilities in web applications.