NAME: NANDIGAM KAMALI HARIPRIYA
REG NO: 21BCE2746

AI for Cyber Security with IBM Qradar (AI for Web Security)
28th August, 2023

Task 4: 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. **XML INJECTION**
   XML injection, or XML code injection, is a vulnerability where an application fails to properly validate user input before using it in an XML document or query. XML, or extensible markup language, is a widely used data storage format. XML injection vulnerabilities allow attackers to exploit weaknesses in applications and front-end services, allowing them to deploy malicious payloads and gain access to sensitive stored data.

   Unvalidated user data can be used to construct queries that allow attackers to read or modify XML documents or execute commands in an XML-enabled database. This allows attackers to bypass the application's front end to access stored data by exploiting vulnerabilities in input fields like user's name, password, and search input fields.An attacker might attempt to read the contents of an organization's stored data files by entering improperly formatted queries on the web app's front end. If the app is properly coded and configured, the unvalidated request will be sent to the app's XML parser and then to the web server and XML-enabled database for execution.

   Source: https://www.thesslstore.com/blog/xml-injection-attacks-what-to-know-about-xpath-xquery-xxe-more/

2. **XML EXTERNAL ENTITY (XXE) INJECTION**
   XML external entity injection (XXE) is a web security vulnerability that allows attackers to interfere with an application's processing of XML data. Common examples include viewing files on the application server filesystem and interacting with back-end or external systems. In some cases, an attacker can escalate an XXE attack to compromise the underlying server or back-end infrastructure by leveraging the vulnerability to perform server-side request forgery (SSRF) attacks. Preventing XXE injection attacks is crucial to protect against potential security threats.

   Source: https://portswigger.net/web-security/xxe

3. **INSECURE DESERIALIZATION**
   Insecure deserialization is a vulnerability where a website deserializes user-controllable data, allowing an attacker to manipulate the objects and pass harmful data into the application code. This vulnerability, also known as an "object injection" vulnerability, can arise due to a lack of understanding of the dangers of deserializing user-controllable data. Website owners may implement additional checks on the

deserialized data, but these checks are often ineffective and flawed. Deserialized objects are often assumed to be trustworthy, especially when using binary serialization formats. However, it is just as possible for an attacker to exploit binary serialized objects as string-based formats. Modern websites have a massive pool of dependencies, making it difficult to manage securely. As an attacker can create instances of any class, it is difficult to predict which methods can be invoked on the malicious data, making it almost impossible to anticipate the flow of malicious data and plug every potential hole. Therefore, it is not possible to securely deserialize untrusted input.

Source: https://portswigger.net/web-security/deserialization

4. SERVER-SIDE TEMPLATE INJECTION (SSTI)

SSTI is a vulnerability where an attacker injects malicious input into a template engine, leading to remote code execution (RCE). Template engines combine templates with data models to produce result documents, populating dynamic data into web pages. Popular template engines include PHP (Smarty, Twigs), Java (Velocity, Freemaker), Python (JINJA, Mako, Tornado), JavaScript (Jade, Rage), and Ruby (Liquid). When input validation is not properly handled on the server side, a malicious server-side template injection payload can be executed, resulting in remote code execution.

Source:
https://www.cobalt.io/blog/a-pentesters-guide-to-server-side-template-injection-ssti

5. SECURITY MISCONFIGURATION

Security misconfiguration is an error or vulnerability in code that allows attackers to access sensitive data. There are various types of security misconfiguration, but most pose a significant danger of data breaches and unauthorized access. These flaws can lead to system compromises, affecting a business significantly. Attackers can exploit or modify applications, leaving businesses exposed to potential attacks. The value of compromised data can significantly impact a business, making it crucial to address security misconfigurations promptly.

Souce: https://www.crowdstrike.com/cybersecurity-101/security-misconfiguration/

6. FILE INCLUSION VULNERABILITIES

A file inclusion vulnerability is a security flaw that allows an attacker to access or execute arbitrary files on a target system. It is often found in web applications that dynamically include files based on user input. The lack of appropriate checks could allow the attacker to gain unauthorized access to sensitive data. File inclusion vulnerabilities are difficult to detect and protect against, making them a common target for hackers. An example of a file inclusion vulnerability is an application that allows users to upload their profile pictures and generates a link to the uploaded picture. If the application does not properly validate the file type or location, an attacker could exploit this by uploading a malicious file and tricking the application into executing it.

Souce: https://www.crowdstrike.com/cybersecurity-101/security-misconfiguration/

7. CORS MISCONFIGURATION

Cross-origin resource sharing (CORS) is a browser mechanism that allows controlled access to resources outside a domain. It adds flexibility to the same-origin policy (SOP) but also exposes potential for cross-domain attacks if poorly configured. CORS is not a protection against cross-origin attacks like cross-site request forgery (CSRF). PortSwigger Research popularized this attack class with their presentation Exploiting CORS misconfigurations for Bitcoins and bounties. Protecting against CORS is crucial for maintaining security and preventing potential attacks.

Source: https://portswigger.net/web-security/cors

8. INSECURE FILE UPLOAD

File upload vulnerability is a common security issue in web applications where a web server accepts a file without validation or restriction, allowing a remote attacker to upload malicious content and execute unrestricted code. This vulnerability can be exploited through crafted multipart form-data POST requests with specific filenames or mime types. The consequences include system acquisition, overloaded file systems or databases, attacks diverting to backend systems, and simple defamation.

Source: https://beaglesecurity.com/blog/vulnerability/insecure-file-upload.html

9. CLICKJACKING

Clickjacking is an attack where a user is tricked into clicking on actionable content on a hidden website by clicking on other content in a decoy website. This can occur when a user clicks on a button to win a prize on a decoy website, only to be deceived into pressing an alternative hidden button, leading to the payment of an account on another site. The technique involves incorporating an invisible, actionable web page or multiple pages containing a button or hidden link within an iframe, overlaid on top of the user's anticipated decoy web page content. This attack differs from a Content Security Risk (CSRF) attack, as the user is required to perform an action, while CSRF attacks forge an entire request without the user's knowledge or input.

Source: https://portswigger.net/web-security/clickjacking

10. BUSINESS LOGIC VULNERABILITIES

Business logic vulnerabilities are security vulnerabilities that can be exploited by attackers to compromise an application's functionality. These flaws occur when developers fail to consider all possible scenarios while coding the application's functionality. An attacker can exploit these vulnerabilities to manipulate the application's logic, performing unauthorized actions like accessing confidential data, bypassing security controls, or executing fraudulent transactions. This article will discuss the causes of business logic vulnerabilities, their impact, real-world attacks, and preventive measures. By understanding and implementing these measures,

businesses can minimize security breaches, protect customer data, and safeguard their reputation, ultimately ensuring the security and integrity of their applications.

Source: https://www.scaler.com/topics/cyber-security/what-are-business-logic-vulnerabilities/