Vulnerabilities

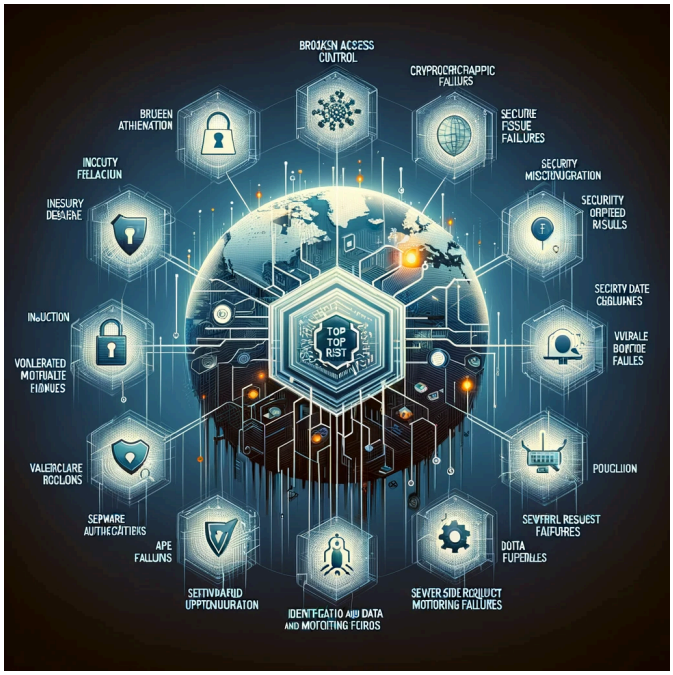

Image Generated by Dalle-E 2 from chat gpt

Prepared for

Kokub Sultan

Prepared by

Valentine Jingwa, Jean-Pierre Nde-Forgwang

Security System, Cohort E

School of Advance Digital Technology

SAIT

10 January 2024

Table of Contents

Software Vulnerabilities

Definition of vulnerability

Weaknesses in software that could be exploited maliciously to cause harm to the users of that system.

List of vulnerabilities

This list of vulnerabilities was extracted from the Owaps website for the year 2021.

Broken Access Control (1)

In terms of understanding broken access control, such an occurrence becomes prevalent in scenarios in which restrictions have not been enforced due to authentication. The aftermath is end users being able to access unauthorized pieces of information and privileges. Instances such as users gaining access to administrative functionality, users accessing other end users' data, access to files, and backend databases that would otherwise not have been possible.

Prevention:

To ensure effective prevention of access control issues, it is essential to employ trusted server-side code or server-less APIs that cannot be altered by attackers. Default settings should deny access unless specified otherwise. Access controls must validate record ownership, and business-specific limits should be enforced. Access control testing should be a standard part of both development and quality assurance processes.

Cryptographic Failures (2)

Cryptographic Failure occurs predominantly when the system handling sensitive user data could be accessed due to mistakes during the encryption and decryption process of that data. Either due to using a poor encryption process or not properly setting up necessary steps preventing the data access.

previously known as Sensitive Data Exposure, which was a broad symptom rather than a root cause (2).

Prevention:

- Ensure the encryption method or protocol being used is not out of date.
- Disable Caching for responses that contain sensitive data.
- Use the right encryption protocol for the right data.
- Use authenticated encryption instead of encryption.
- Keys for encryption should be generated cryptographically randomly and stored as byte arrays.

Injection: (3)

In terms of Injections, it is a scenario in which an attacker can “inject” malicious data into a command. This entails that the command is executed without validation, leading to united behavior by the application.

Step by Step: (3)
User Input - The attacker inputs data entry which bypasses detection.

Input Submission - The input exploits the syntax of SQL, these input is intended to change the behavior of the application.

Input Processed - The input is not properly validated nor distinguished the input, and the query is treated as trustworthy.

Execution - The database executes all queries including the malicious one.

Unintended Action - In such a case unauthorized actions can be performed as per the injection command.

Insecure Design: (4)

This arises from poor design principles and logic that leave a certain system vulnerable. One major contributing factor that leads to an insecure design is the lack of business

risk profiling inherent in the software or system being developed, and thus the failure to determine what level of security design is required (4).

Prevention:

- Assign the user a specific amount of resources.
- Identify threats and model design to accommodate for possible threats.
- Implement Unit and integration testing at each level of the design process.

Security Misconfiguration: (5)

In terms of security misconfiguration, the conceptualization of such an idea stems from not appropriately securing settings. This allows end users to potentially access admin settings and sensitive data. Ranging from default credentials, unprotected files and directories, and verbose error messages, to outdated software. Such vulnerabilities allow end users to access united information.

Vulnerable and Outdated Components: (6)

Just as its name indicates this vulnerability arises from the usage of outdated software components. With more powerful machines these outdated/Legacy components such as encryption methods could be easily broken and accessed by malicious users.

Prevention:

- The designers of the system should use proper SDLC and update their software with time.
- Continuously Inventory the version of both client-side components and their dependencies (6).

- Obtain official components from official sources over secure links.
- Remove all unused dependencies, unnecessary features, components, files, and documentation (6).

Identification and Authentication Failures: (7)

In terms of identification and authenticity, such instances happen in which the application does not properly identify the end user that authenticates, or mistakes a user with another. Issues such s weak passwords, lack of multi-factor authentication, and flawed session management.

Prevention:

Strong Password Policies - A requirement for complex and strong passwords that are hard to guess.

Multi-Factor Authentication - Including another level and layer of security that grants access.

Secure Session Management - After login and later inactivity from the end user's point of view, your session closes securely.

Regular Update - To bolster such an application, regularly updating and strengthening the application for vulnerabilities is essential.

Software and Data Integrity Failures: (8)

This occurs when the infrastructure isn't protected against integrity violations. A very great example of this is when the system uses plugins from a private source and relies on auto updates from that source. The creator of such a plugin can rewrite and access the data from any target system if they want to.

Prevention:

- Design a mechanism that checks for signatures before integration into the system.
- Ensure that there is a review process for code and configuration changes to minimize the chance that malicious code or configuration could be introduced into your software pipeline.
- Ensure that your CI/CD pipeline has proper segregation, configuration, and access control to ensure the integrity of the code flowing through the build and deploy processes.
- Ensure that unsigned or unencrypted serialized data is not sent to untrusted clients without some form of integrity check or digital signature to detect tampering or replay of the serialized data.

Security Logging and Monitoring Failures: (9)

In security logging and monitoring in relation to an absence of tracking and monitoring of user activities within a system. In such occurrence of a data breach, or vulnerability within an application, the admin can look back on the instance in which a problem occurred and deliberate finding a solution.

Prevention:

Comprehensive Logging - Ensuring all significant actions are logged.

Active Monitoring - An active monitoring of system and logs in real-time, and alerts if suspicious and unauthenticated activities are detected.

Regular Reviews - An active review of the logs within a system, checking for suspicious activities and potential security issues and threats

Protect Logs - It is imperative that the logs within an application is securely housed and are not accessible by unauthorized parties.

Server-Side Request Forgery: (10)

SSRF is a type of security problem in web applications. It happens when an application gets a URL from a user but doesn't check if the URL is safe. Because of this, a hacker could trick the application into accessing parts of the internet or internal networks that it shouldn't, like private databases or internal services. This can happen even if there are security systems like firewalls in place.

As more web applications offer features that involve grabbing information from different internet locations, the chances of SSRF happening are going up. This is becoming a bigger problem, especially with the use of cloud services and more complicated computer systems, because it could let hackers get into areas that are usually well-protected.

Prevention:

- Segment remote resource access functionality in separate networks to reduce the impact of SSRF
- Enforce “deny by default” firewall policies or network access control rules to block all but essential intranet traffic.

Scenario: Ashley Madison data breach: (11)

The Ashley Madison data breach, which took place in July 2015, was a major cybersecurity incident. Ashley Madison, a website designed for people seeking extramarital relationships, was targeted by a group calling themselves "The Impact Team." This group managed to steal a vast amount of personal information from the site's users.

Broken Access Control: The hackers were able to gain access to data that should have been restricted. This indicates a failure to implement proper access controls to sensitive user data and administrative functions.

Cryptographic Failures: Previously known as "Sensitive Data Exposure". Ashley Madison's failure to properly protect users' sensitive data, including real names and credit card information, falls under this category. Despite claims of deleting user data upon request, the breach revealed that the data was not properly erased.

Security Misconfiguration: This risk is evident in the breach, as the company had misconfigurations in their system, such as retaining data that was supposed to be deleted and possibly having inadequate security settings that allowed the breach.

Security Logging and Monitoring Failures: The company's failure to detect and respond to the breach promptly indicates inadequate monitoring and logging of security events.

References

[1] O. Top 10 team, “A01:2021 – Broken Access Control,” A01 Broken Access Control - OWASP Top 10:2021, https://owasp.org/Top10/A01_2021-Broken_Access_Control/ (accessed Jan. 10, 2024).

[2] OWASP Top 10 team, "A02 Cryptographic Failures - OWASP Top 10:2021," OWASP Top 10:2021, https://owasp.org/Top10/A02_2021-Cryptographic_Failures/ (accessed Jan. 10, 2024).

[3] OWASP Top 10 team, "A03 Injection - OWASP Top 10:2021," OWASP Top 10:2021, https://owasp.org/Top10/A03_2021-Injection/ (accessed Jan. 10, 2024).

[4] OWASP Top 10 team, "A04 Insecure Design - OWASP Top 10:2021," OWASP Top 10:2021, https://owasp.org/Top10/A04_2021-Insecure_Design/ (accessed Jan. 10, 2024).

[5] OWASP Top 10 team, "A05 Security Misconfiguration - OWASP Top 10:2021," OWASP Top 10:2021, https://owasp.org/Top10/A05_2021-Security_Misconfiguration/ (accessed Jan. 10, 2024).

[6] OWASP Top 10 team, "A06 Vulnerable and Outdated Components - OWASP Top 10:2021," OWASP Top 10:2021, https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/ (accessed Jan. 10, 2024).

[7] OWASP Top 10 team, "A07 Identification and Authentication Failures - OWASP Top 10:2021," OWASP Top 10:2021, https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/ (accessed Jan. 10, 2024).

[8] OWASP Top 10 team, "A08 Software and Data Integrity Failures - OWASP Top 10:2021," OWASP Top 10:2021, https://owasp.org/Top10/A08_2021-Software_and_Data_Integrity_Failures/ (accessed Jan. 10, 2024).

[9] OWASP Top 10 team, "A09 Security Logging and Monitoring Failures - OWASP Top 10:2021," OWASP Top 10:2021, https://owasp.org/Top10/A09_2021-Security_Logging_and_Monitoring_Failures/ (accessed Jan. 10, 2024).

[10] OWASP Top 10 team, "A10 Server Side Request Forgery (SSRF) - OWASP Top 10:2021," OWASP Top 10:2021, https://owasp.org/Top10/A10_2021-Server_Side_Request_Forgery_(SSRF)/ (accessed Jan. 10, 2024).

[12] Google Play Store, "Ashley Madison - Apps on Google Play," https://play.google.com/store/apps/details?id=com.ashleymadison.mobile (accessed Jan. 10, 2024).