OWASP Top Ten

Nathan Hinebaugh

1. **Broken Access Control**

   This is an issue that arises when Access Control Lists are incomplete or not configured properly, allowing potential attackers to gain access to networks. Attackers request access to files or functionalities they shouldn't have. This includes administrative interfaces that can remotely manage sites on the Internet. Attackers who gain access to this can change or delete files, steal sensitive data, or perform ransomware attacks.

   To mitigate this threat, Access Control Lists should be reviewed frequently to make sure there are no unnecessary permissions, as well as inactive accounts being left open. Access should only be granted when absolutely necessary for job functions. When in doubt, access should be denied until it is clear that it is needed.

2. **Cryptographic Failures**

   This vulnerability occurs when data being transmitted over the internet is not properly encrypted, or not encrypted at all. Some examples include weak cryptography, sending messages over the network in plaintext, use of outdated cryptographic methods, or improper key management, among others.

   To mitigate this, all messages sent over the network or the internet at large should be encrypted, with special attention given to sensitive or confidential information. Authenticated encryption should be used rather than plain encryption. Keys should be rotated regularly, and only the most up-to-date cryptography methods should be employed.

3. **Injection**

   This involves interfering with a network connection in such a way that it appears to be normal network traffic, but is, in fact, exploiting a vulnerability. This can take the form of SQL Injection, where the wrong data is input into a comment or input field. This tricks the database into performing functions that are unintended, often allowing attackers to gain access to the data stored in the database. This same basic principle can be applied to web servers, and even the code itself. In each case, the attacker is exploiting the fact that the database, OS, or server is expecting only certain inputs, and when an unexpected input is encountered; it leads to undesirable behavior that was not originally intended.

To mitigate this, web servers, operating systems, and databases should be programmed so that any unexpected commands or inputs are disregarded or blocked so that only the intended outputs can be produced.

4. **Insecure Design**

   This vulnerability occurs when applications are designed without proper regard for security vulnerabilities that may be present. When businesses are too focused on accessibility but not integrity or confidentiality, apps might be built solely for functionality while ignoring the need to protect data and secure networks. This allows attackers to exploit flaws in the code, allowing them to access passwords or other data, or even gain partial or full control of a system or network.

   This can be mitigated by bringing cybersecurity experts in on the design phases of app development, ensuring that security measures are built-in, rather than attempting to find and patch vulnerabilities later as they arise.

5. **Security Misconfiguration**

   Security misconfiguration is any error or vulnerability present in the configuration of code that allows attackers access to sensitive data. This often occurs when security measures are in place but not correctly utilized. An example would be using the default credentials that came with a piece of software or hardware. Another example would be installing a firewall, but leaving it at the default settings, or only using it to blacklist a few IP addresses or URL's while remaining vulnerable to other unknown threats.

   To mitigate this, permissions should be granted only as needed. Credentials should be changed as soon as new software applications or hardware is installed, and IPS systems should be configured correctly (ex: firewall set to allow traffic only on certain ports, and only known traffic should be allowed, i.e., "whitelisting" instead of "blacklisting".)

6. **Vulnerable and Outdated Components**

   This refers to hardware or software that has known or unknown vulnerabilities, as well as those that are using outdated or unpatched versions. When a vulnerability is identified, it should be patched as soon as possible to avoid zero-day exploits.

   To mitigate these attacks, only the most recent versions of software should be used, and hardware should be updated with firmware updates whenever available. Legacy software that no longer supports updates should be discarded and replaced, as well as hardware that has physical components that may have outlived their lifespan.

7. **Identification and Authentication Failures**

   This vulnerability is a result of such things as allowing weak and/or reused passwords, lack of multi-factor authentication, sessions staying open too long instead of timing out, as well as encryption flaws.

   Several things can be done to mitigate this vulnerability, including the use of multi-factor authentication, good encryption and session management, checking for and disallowing the use of weak passwords, as well as limiting the number of times a wrong password can be entered before locking the user out of the system.

8. **Software and Data Integrity Failures**

   The danger here is often when downloading new software or patches, or firmware updates. Often these are set up so they are done automatically without any human intervention so that there is no way to verify that the updates are legitimate and free of known vulnerabilities. Attackers can exploit this by sending out fake software updates that contain malware or other malicious tools.

   This is mitigated by verifying any software or firmware updates, and by making sure that these updates come from trusted sources. Additionally, it is advisable to verify the supply chain of the hardware itself to make sure that there are no known vulnerabilities and that the hardware has not been tampered with in any way.

9. **Security Logging and Monitoring Failures**

   This does not speak to a specific vulnerability, but rather the inability or reduced ability to detect such vulnerabilities when and where they may occur. Networks that are not being scanned for vulnerabilities will often be blind to any attack that doesn't cause obvious and immediate damage. Similarly, when an attack is or has occurred, failure to log network activity leads to an inability to understand the source and nature of such attacks.

   This is obviously mitigated by doing the opposite of what is described above. Regular network scanning is crucial for detecting attacks early. Logs should be in a format that will be easily usable later, there should be enough data to be of use in determining what actually happened, and steps should be taken to make sure that the logs are not tampered with or incomplete.

10. Server-side Request Forgery

This happens when an attacker sends a request for data to a public server. That public server then sends a request to an end server in order to fulfill the request. The end server trusts the public server, and so the requested information is sent back to the public server. By pretending to act on behalf of a trusted public server, the attacker can gain access to unauthorized data.

Since I've used the word mitigate in the last nine sections, I might as well stick with it. To mitigate this, always make sure that you are making requests to back end server on the behalf of public server, not from the browser. Also, verify that the request is not coming from the local host, but from the actual server.

Sources:

Broken Access Control | OWASP Foundation. (n.d.). https://owasp.org/www-community/Broken_Access_Control

Sengupta, S. (2022, November 17). Cryptographic Failures Vulnerability - Examples & Prevention. *Crashtest Security*. https://crashtest-security.com/owasp-cryptographic-failures/#:~:text=Security%20flaws%20that%20commonly%20lead%20to%20cryptography%20failures,8%20Use%20of%20deprecated%20hash%20functions%20More%20items

Odogwu, C. (2021). What Are Injection Attacks and How Can You Prevent Them? *MUO*. https://www.makeuseof.com/what-are-injection-attacks-how-to-prevent-them/

Sengupta, S. (2022a, November 16). Insecure Design. *Crashtest Security*. https://crashtest-security.com/insecure-design-vulnerability/

Brathwaite, S. (2023). The Risks in Vulnerable and Outdated Components. Software Secured. https://www.softwaresecured.com/the-risks-in-vulnerable-and-outdated-components/#:~:text=Vulnerable%20and%20Outdated%20components%20are%20components%20of%20a,an%20organization%20will%20be%20older%20versions%20of%20software.

Owasp. (n.d.). Top10/A07_2021-Identification_and_Authentication_Failures.md at master · OWASP/Top10. GitHub. https://github.com/OWASP/Top10/blob/master/2021/docs/A07_2021-Identification_and_Authentication_Failures.md

A08 Software and Data Integrity Failures - OWASP Top 10:2021. (n.d.). https://owasp.org/Top10/A08_2021-Software_and_Data_Integrity_Failures/

Clements, J. (2022). OWASP Top Ten: #9 Security Logging and Monitoring Failures. Foresite. https://foresite.com/blog/owasp-top-ten-9-security-logging-and-monitoring-failures/

GeeksforGeeks. (2022). Server Side Request Forgery SSRF in Depth. GeeksforGeeks. https://www.geeksforgeeks.org/server-side-request-forgery-ssrf-in-depth/