Don Bosco Institute of Technology, Kurla (W), Mumbai – 400070
Department of Information Technology
Secure Application Development Lab

Roll no.: 52 **Date**: 30 / 09 / 2024

Name: Afzal Siddiquie

Experiment No. 8

Aim: Study of OWASP Top 10 Security Risks or Vulnerabilities by OWASP

Objectives: The objective of this experiment is to

- Describe a standard awareness document for developers and web application security.
- Understand the Owasp methodologies and standards

Outcomes: After study of this experiment, the student will be able to

- Illustrate the Owasp methodologies and standards.

Prerequisite: Knowledge of Vulnerabilities.

Requirements: PC and Internet

Brief Theory:

OWASP 2023 is a big deal because this list of the 10 most serious web app security vulnerabilities ranks them in order of risk. It's an important checklist of *threats to guard against* for web developers as well as anyone who is responsible for website security or web app development.

OWASP stands for ***Open Worldwide Application Security Project***, and it's a non-profit foundation that creates web application security resources. It produces a risk assessment framework, industry standards, best practices, tools, and more, and anyone in its community can contribute, so it has a vast pool of expertise on tap.

To compile its top 10 list of security vulnerabilities OWASP regularly gathers data from more than 200,000 organizations and from surveys of industry professionals. With a new update yet to surface (we're expecting one sometime in the next couple of years), OWASP 2023 inevitably relies on the 2021 list, but make no mistake, these vulnerabilities are still very relevant and everyone in web development and security needs to be alert to the threats they pose.

Top Ten Vulnerabilities for OWASP 2023

1. Broken Access Control

Rising from 5th place in 2017 to top the list in 2021, broken access control remains a significant, ongoing threat. Access controls limit users to the resources and functionalities they are authorized to use, and broken access control is the term used when a system fails to enforce appropriate restrictions.

This can happen for various reasons, including a misconfiguration, IDOR (Insecure Direct Object References) which is where an application exposes direct references to resources like files or database records, insecure session management, where flaws in session management can allow attackers to hijack users' sessions, and others.

Developers and system administrators should follow the Principle of Least Privilege here, which means only granting users the minimum set of permissions that are required for them to perform their tasks and nothing more.All user input should be validated and sanitized to prevent attackers from injecting malicious data, access controls should be applied to APIs, and authorization checked for every request.Regular security audits and code reviews are a must to identify and fix access control issues, and multi-factor authentication should be enforced to limit unauthorized access.

2. Cryptographic Failures

This was previously in the number three spot and was called "Sensitive Data Exposure" but it's since been relabeled because the old name described a symptom rather than the cause. Cryptography is used to protect highly sensitive data like credit card numbers and PII while it's in transit, but it can fail due to factors like weak encryption algorithms or short encryption keys which can make it easier for attackers to decrypt sensitive data.

Other examples of cryptography failures include insecure password storage, insufficient transport layer security (which can lead to man-in-the-middle attacks), weak SSL/TLS protocols, and insecure cipher suites that can expose the web application to attacks.

Implement regular security testing (including code reviews and vulnerability assessments) to identify and fix cryptographic weaknesses, and also consider using secure cryptographic libraries too.

3. Injection

Injection occupied the 2017 number one spot and cross-site scripting the number seven. Now they've been consolidated under this one umbrella term which occupies the current number three position for OWASP 2023.

Injection attacks exploit vulnerabilities in input validation and inadequate data handling. Attackers inject data such as SQL queries, code snippets, or commands into web application forms or URLs. They allow adversaries to access sensitive data and manipulate an application's behavior.

Examples include:

- SQL Injection for database attacks
- Cross-Site Scripting (XSS)—usually JavaScript-based browser attacks launched via infected web pages, leading to session hijacking, cookie theft, or other attacks on users.
- Command Injection—attackers inject malicious commands into system commands executed by the application, potentially gaining control of the server or executing unauthorized operations.
- LDAP Injection—attackers manipulate LDAP queries used for authentication and authorization to gain access.
- XML Injection—attackers insert malicious content into XML data, potentially disrupting the application's parsing process to gain access.
- Server-Side Template Injection (SSTI)—where attackers inject malicious code into server-side templates to execute code on the server.

4. Insecure Design

This OWASP 2023 category was new in 2021, and it covers faulty application design and flaws in architecture that hackers can exploit. Insecure design vulnerabilities occur when teams don't adhere to security best practices, and they fail to adequately anticipate and evaluate potential threats during the code design phase of creating the application.

An example of insecure design is an app that produces overly detailed error messages. If it reports on error conditions in too much detail and offers diagnostic clues about the application environment, or other associated data, it could be revealing potentially useful information to attackers. They could then use it to launch other attacks like path transversal or SQL injection. It's important to mitigate design vulnerabilities by using consistent threat modeling to shut down known methods of attack.

5. Security Misconfiguration

This category now includes 2017's XML External Entities (XXE) category. Security misconfigurations encompass a variety of potential vulnerabilities, but these are the most common ones:

- Unpatched vulnerabilities
- Default configurations
- Unused pages
- Unprotected files and directories
- Unnecessary services
- Use of vulnerable XML files

A common mistake that webmasters commit is leaving CMS (Content Management System) default configurations unchanged. While CMS apps are user-friendly, they can pose security risks for end-users. Many attacks are entirely automated and rely on exploiting default settings, which makes changing these settings during CMS installation crucial for mitigating a significant number of potential attacks. Adjusting settings to control comments, user access, user information visibility, and default file permissions can bolster security.

6. Vulnerable and Outdated Components

Even the simplest of websites have many dependencies like frameworks, libraries, extensions, and plugins, and every one of them must be kept up to date. Attackers are actively looking for websites with vulnerable components which they can exploit to spread malware, launch phishing attacks, and more, so failing to install updates for whatever reason is a bad idea.

The latest version of any software is going to contain the latest security updates, but if your website relies on a lot of dependencies that can be easier said than done. The first step to fixing this is to create an inventory that lists all the connected components in your environment and keeps you up to date.

7. Identification and Authentication Failures

Authentication and identity management failures expose applications to the risk of malicious actors posing as genuine users. A session ID configured without a validity period can run and run. Weak passwords can be susceptible to guessing and with no rate limits imposed on login attempts automated attacks keep doing that until they succeed. To address these issues, implement multi-factor authentication (MFA) within applications. Also, developers should be made aware of the need to adhere to recommended password length, complexity, and rotation policies.

8. Software and Data Integrity Failures

These are a type of design flaw. The complexity of modern architectures means that developers often add plugins and libraries to the pipeline from various sources without verifying their integrity. If any of them fail, this can leave applications susceptible to unauthorized information disclosure, system compromise, or malicious code insertion. This is another reason why it's important to have an active inventory of all third-party and open-source plugins and libraries, along with continuous threat monitoring.

9. Security Logging and Monitoring Failures

Poor logging and monitoring capabilities mean that incidents are missed and alerts aren't generated, and they could remain unnoticed for long enough to do substantial damage.

Login attempts and failures need to be logged, and logs need to be backed up in case of server failure. Logs need to be accurate so that monitoring systems can detect suspicious activities or raise timely alerts, and they also need to be protected against tampering. To mitigate vulnerabilities, record all login attempts (including failures), maintain copies of logs, use anti-tamper mechanisms, and test monitoring systems regularly.

10. Server-Side Request Forgery (SSRF)

This vulnerability allows attackers to make unauthorized requests from the server to other internal or external resources. In SSRF attacks, the attacker can manipulate input fields or parameters in the application to trick the server into sending requests to arbitrary URLs, often without the user's knowledge. Attackers can abuse this vulnerability to access sensitive data, interact with internal resources, or perform actions on behalf of the server, potentially leading to a complete compromise of the application or its infrastructure.

To mitigate SSRF vulnerabilities, developers should follow best practices such as:

Input Validation: Properly validate and sanitize user-supplied input to prevent malicious URLs or IP addresses from being used in requests.

Whitelisting: Implement whitelisting for allowed URLs or IP ranges to restrict the server's ability to make requests to known safe resources.

Firewall Rules: Configure firewalls and network settings to restrict outgoing requests from the server to specific resources and protocols.

Use of Safe APIs: If the application needs to make requests to external resources, use safe APIs or specific endpoints that are intended for public access.

Least Privilege: Ensure that the server has the least privileges necessary to access external resources to limit potential damage if an SSRF attack occurs.

Top 10 Web Application Security Risks

There are three new categories, four categories with naming and scoping changes, and some consolidation in the Top 10 for 2021.

- **A01:2021-Broken Access Control** moves up from the fifth position; 94% of applications were tested for some form of broken access control. The 34 Common Weakness Enumerations

(CWEs) mapped to Broken Access Control had more occurrences in applications than any other category.

- **A02:2021-Cryptographic Failures** shifts up one position to #2, previously known as Sensitive Data Exposure, which was a broad symptom rather than a root cause. The renewed focus here is on failures related to cryptography which often leads to sensitive data exposure or system compromise.
- **A03:2021-Injection** slides down to the third position. 94% of the applications were tested for some form of injection, and the 33 CWEs mapped into this category have the second most occurrences in applications. Cross-site Scripting is now part of this category in this edition.
- **A04:2021-Insecure Design** is a new category for 2021, with a focus on risks related to design flaws. If we genuinely want to "move left" as an industry, it calls for more use of threat modeling, secure design patterns and principles, and reference architectures.
- **A05:2021-Security Misconfiguration** moves up from #6 in the previous edition; 90% of applications were tested for some form of misconfiguration. With more shifts into highly configurable software, it's not surprising to see this category move up. The former category for XML External Entities (XXE) is now part of this category.
- **A06:2021-Vulnerable and Outdated Components** was previously titled Using Components with Known Vulnerabilities and is #2 in the Top 10 community survey, but also had enough data to make the Top 10 via data analysis. This category moves up from #9 in 2017 and is a known issue that we struggle to test and assess risk. It is the only category not to have any Common Vulnerability and Exposures (CVEs) mapped to the included CWEs, so a default exploit and impact weights of 5.0 are factored into their scores.
- **A07:2021-Identification and Authentication Failures** was previously Broken Authentication and is sliding down from the second position, and now includes CWEs that are more related to identification failures. This category is still an integral part of the Top 10, but the increased availability of standardized frameworks seems to be helping.
- **A08:2021-Software and Data Integrity Failures** is a new category for 2021, focusing on making assumptions related to software updates, critical data, and CI/CD pipelines without verifying integrity. One of the highest weighted impacts from Common Vulnerability and Exposures/Common Vulnerability Scoring System (CVE/CVSS) data mapped to the 10 CWEs in this category. Insecure Deserialization from 2017 is now a part of this larger category.
- **A09:2021-Security Logging and Monitoring Failures** was previously Insufficient Logging & Monitoring and is added from the industry survey (#3), moving up from #10 previously. This category is expanded to include more types of failures, is challenging to test for, and isn't well represented in the CVE/CVSS data. However, failures in this category can directly impact visibility, incident alerting, and forensics.
- **A10:2021-Server-Side Request Forgery** is added from the Top 10 community survey (#1). The data shows a relatively low incidence rate with above average testing coverage, along with above-average ratings for Exploit and Impact potential. This category represents the scenario where the security community members are telling us this is important, even though it's not illustrated in the data at this time.

Conclusion: In this experiment, we studied the OWASP Top 10 security risks for 2023, gaining insight into common web application vulnerabilities, their causes, and how to mitigate them to enhance security in web development.

Mention your References:

01. https://owasp.org/www-project-top-ten/
02. https://owasp.org/Top10/