ASSIGNMENT -1

Name : C Devaki Nandan Reddy

Top 5 OWASP Vulnerabilities perform in the demo websites and show proof of contact

What is OWASP?

OWASP stands for the Open Web Application Security Project, an online community that produces articles, methodologies, documentation, tools, and technologies in the field of web application security.

The Top 10 OWASP vulnerabilities in 2021 are:

- Injection
- Broken authentication
- Sensitive data exposure
- XML external entities (XXE)
- Broken access control
- Security misconfigurations
- Cross site scripting (XSS)
- Insecure deserialization
- Using components with known vulnerabilities
- Insufficient logging and monitoring

Injection

A code injection happens when an attacker sends invalid data to the web application with the intention to make it do something that the application was not designed/programmed to do.

Perhaps the most common example around this security vulnerability is the **SQL query consuming untrusted data**. You can see one of OWASP's examples below:

String query = "SELECT * FROM accounts WHERE custID = '" + request.getParameter("id") + "'";

This query can be exploited by calling up the web page executing it with the following URL: https://example.com/app/accountView?id=' or '1'='1 causing the return of all the rows stored on the database table.

Broken Authentication

A broken authentication vulnerability can allow an attacker to use manual and/or automatic methods to try to gain control over any account they want in a system – or even worse – to gain complete control over the system.

Websites with broken authentication vulnerabilities are very common on the web. Broken authentication usually refers to logic issues that occur on the application authentication's mechanism, like bad session management prone to username enumeration – when a malicious actor uses brute-force techniques to either guess or confirm valid users in a system.

To minimize broken authentication risks avoid leaving the login page for admins publicly accessible to all visitors of the website:

- /administrator on Joomla!
- /wp-admin/ on WordPress
- /index.php/admin on Magento
- /user/login on Drupal

Sensitive Data Exposure

Sensitive data exposure is one of the most widespread vulnerabilities on the OWASP list. It consists of compromising data that should have been protected.

Examples of Sensitive Data

Some sensitive data that requires protection is:

- Credentials
- Credit card numbers
- Social Security Numbers
- Medical information
- Personally identifiable information (PII)
- Other personal information

Not encrypting sensitive data is the main reason why these attacks are still so widespread. Even encrypted data can be broken due to weak:

- Key generation process
- Key management process
- Algorithm usage
- Protocol usage
- Cipher usage
- Password hashing storage techniques

XML External Entities (XXE)

According to Wikipedia, an XML External Entity attack is a type of attack against an application that parses XML input. This attack occurs when XML input containing a reference to an external entity is processed by a weakly configured XML parser.

Most XML parsers are vulnerable to XXE attacks by default. That is why the responsibility of ensuring the application does not have this vulnerability lays mainly on the developer.

What are the XML external entity attack vectors?

According to the OWASP Top 10, the XML external entities (XXE) main attack vectors include the exploitation of:

- Vulnerable XML processors if malicious actors can upload XML or include hostile content in an XML document
- Vulnerable code
- Vulnerable dependencies
- Vulnerable integrations

How to prevent XML external entity attacks

Some of the ways to prevent XML External Entity attacks, according to OWASP, are:

- Whenever possible, use less complex data formats ,such as JSON, and avoid serialization of sensitive data.
- Patch or upgrade all XML processors and libraries in use by the application or on the underlying operating system.

- Use dependency checkers (update SOAP to SOAP 1.2 or higher).
- Disable XML external entity and DTD processing in all XML parsers in the application, as per the OWASP Cheat Sheet 'XXE Prevention.'
- Implement positive ("allowlisting") server-side input validation, filtering, or sanitization to prevent hostile data within XML documents, headers, or nodes.
- Verify that XML or XSL file upload functionality validates incoming XML using XSD validation or similar.
- SAST tools can help detect XXE in source code – although manual code review is the best alternative in large, complex applications with many integrations.

Broken Access Control

In website security, access control means putting a limit on what sections or pages visitors can reach, depending on their needs.

For example, if you own an ecommerce store, you probably need access to the admin panel in order to add new products or to set up a promotion for the upcoming holidays. However, hardly anybody else would need it. Allowing the rest of your website's visitors to reach your login page only opens up your ecommerce store to attacks.

And that's the problem with almost all major content management systems (CMS) these days. By default, they give worldwide access to the admin login page. Most of them also won't force you to establish a two-factor authentication method (2FA).

The above makes you think a lot about software development with a security-first philosophy.

Examples of Broken Access Control

Here are some examples of what we consider to be "access":

- Access to a hosting control / administrative panel
- Access to a server via FTP / SFTP / SSH
- Access to a website's administrative panel
- Access to other applications on your server
- Access to a database

Attackers can exploit authorization flaws to the following:

- Access unauthorized functionality and/or data
- View sensitive files

- Change access rights

What are the risks of broken access control?

According to OWASP, here are a few examples of what can happen when there is broken access control:

- **Scenario #1:** The application uses unverified data in a SQL call that is accessing account information: **pstmt.setString(1,request.getParameter("acct")); ResultSetresults =pstmt.executeQuery( );**

An attacker simply modifies the 'acct' parameter in the browser to send whatever account number they want. If not properly verified, the attacker can access any user's account.

https://example.com/app/accountInfo?acct=notmyacct

- **Scenario #2:** An attacker simply force browses to target URLs. Admin rights are required for access to the admin page.https://example.com/app/**getappInfo**

https://example.com/app/**admin_getappInfo**