Web Application Security

Nassec.io

Background

Web Apps are highly functional applications that rely on two way information between server and browser.

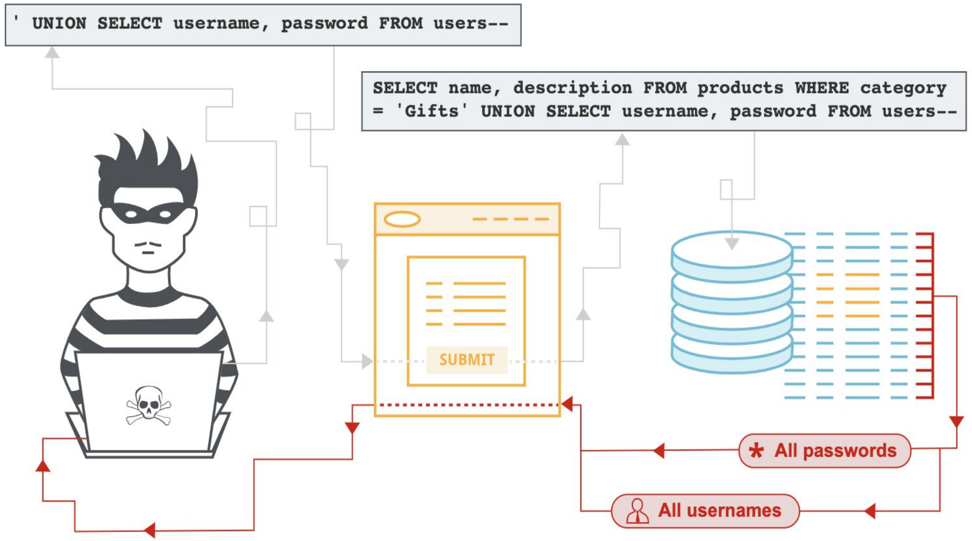
Web applications, if not secured properly can cause massive cyber risks.

Web Application Security Risks

The OWASP® Foundation has categorized web application security risks into ten different categories.

OWASP Foundation works to improve the security of software through its community-led open source software project.

OWASP Top 10 - Injection



```
' UNION SELECT username, password FROM users--
```

```
SELECT name, description FROM products WHERE category
= 'Gifts' UNION SELECT username, password FROM users--
```

Injection flaws, such as SQL, NoSQL, and OS injection, occur when untrusted data is sent to an web server as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

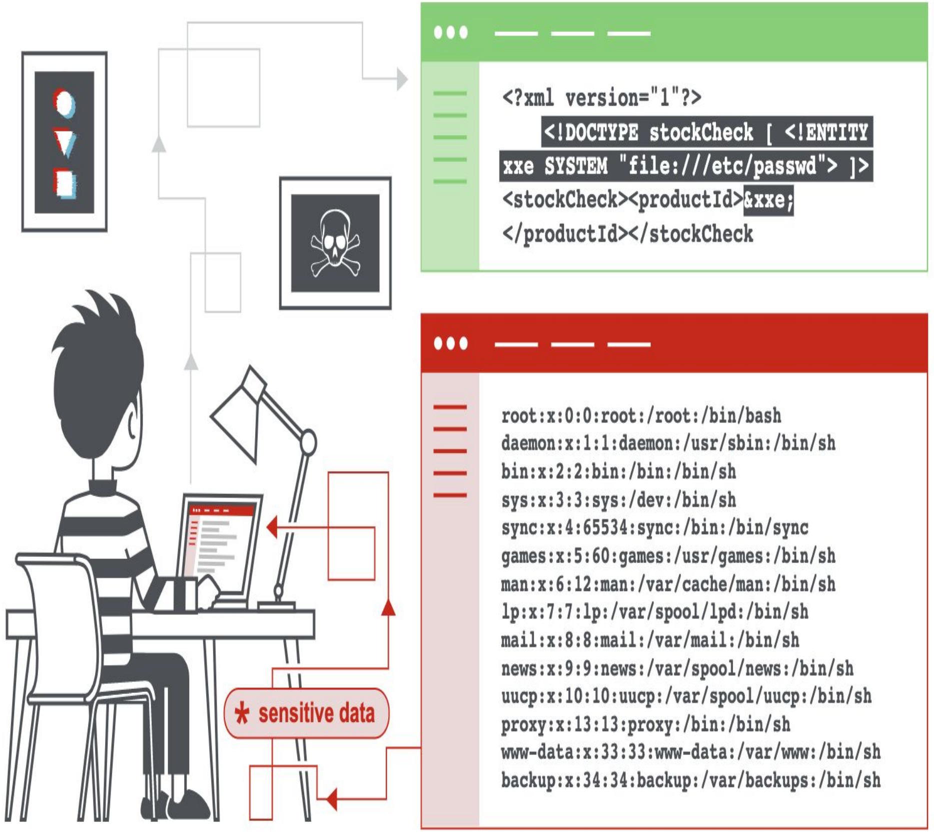
OWASP Top 10 - Broken Authentication

Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.

OWASP Top 10 - Sensitive Data Exposure

Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.

OWASP Top 10 - XML External Entities

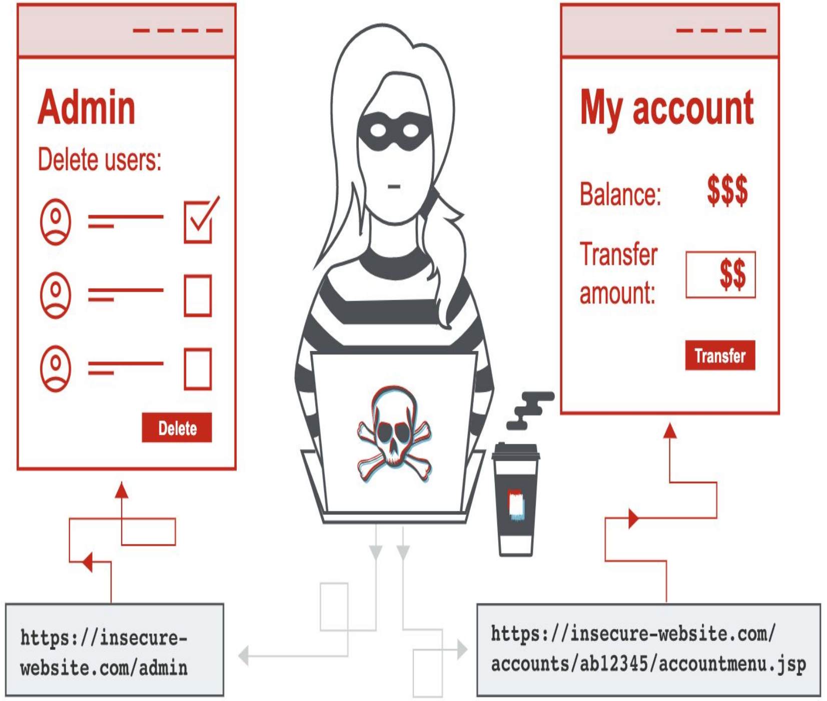


```
<?xml version="1"?>
    <!DOCTYPE stockCheck [ <!ENTITY
xxe SYSTEM "file:///etc/passwd"> ]>
<stockCheck><productId>&xxe;
</productId></stockCheck
```

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
```

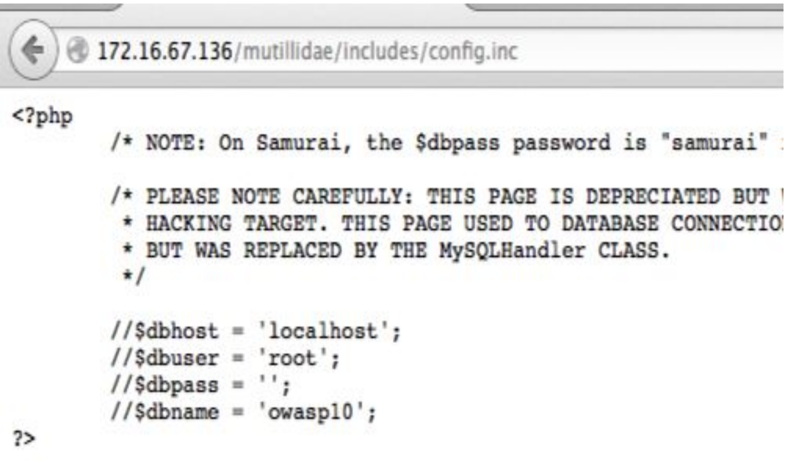
Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.

OWASP Top 10 - Broken Access Control


Admin
Delete users:
Delete
My account
Balance: $$$
Transfer amount: $$
Transfer
https://insecure-website.com/admin
https://insecure-website.com/accounts/ab12345/accountmenu.jsp

Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.

OWASP Top 10 - Security Misconfiguration



172.16.67.136/mutillidae/includes/config.inc

```
<?php
        /* NOTE: On Samurai, the $dbpass password is "samurai"

        /* PLEASE NOTE CAREFULLY: THIS PAGE IS DEPRECIATED BUT
         * HACKING TARGET. THIS PAGE USED TO DATABASE CONNECTIO
         * BUT WAS REPLACED BY THE MySQLHandler CLASS.
         */

        //$dbhost = 'localhost';
        //$dbuser = 'root';
        //$dbpass = '';
        //$dbname = 'owasp10';
?>
```

Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched and upgraded in a timely fashion.

OWASP Top 10 - Cross-site Scripting

XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

OWASP Top 10 - Insecure Diserialization

Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks.

OWASP Top 10 - Using Components with known Vulnerabilities

Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.

OWASP Top 10 - Insufficient Logging and Monitoring

Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

60% of breaches occur due to Unpatched Vulnerability.

Challenges

- Evolving nature of vulnerabilities.
- Cyber security is not a priority in organizations.
- Insufficient enabling security technologies.
- Lack of in-house expertise.

VAPT can help reduce cyber risks associated with web applications.

What is a VAPT?

Technical Testing process to determine the security stature of software application

Helps detect and resolve loopholes that exists in software applications

Important part of compliances such as PCI DSS and GDPR Regulation.

A regular process - needs to be done again and again

VAPT Process

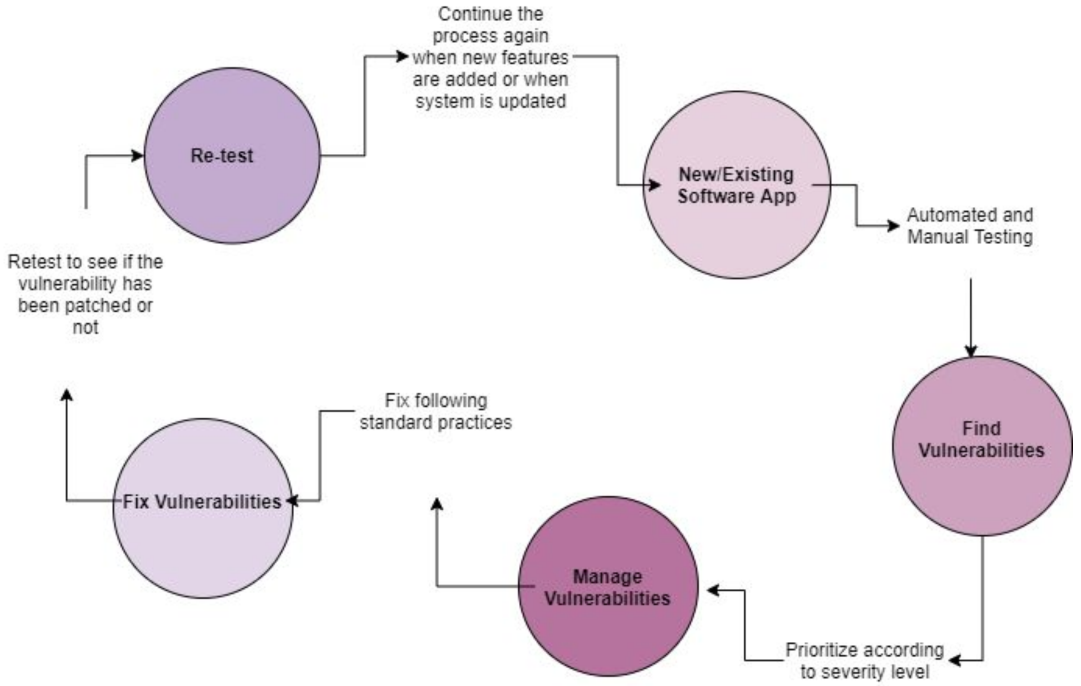
Information Gathering - gather preliminary information of software application

Vulnerability Assessment - Assess how secure/insecure software application is

Exploitation - Exploit the system from hacker's find

Fixing - Fix/patch loopholes found in the system

Vulnerability Management Process


Re-test
Continue the process again when new features are added or when system is updated
New/Existing Software App
Automated and Manual Testing
Find Vulnerabilities
Prioritize according to severity level
Manage Vulnerabilities
Fix following standard practices
Fix Vulnerabilities
Retest to see if the vulnerability has been patched or not

Methodology

Automated Tests - Using tools such as ReconwithMe, Burpsuite, Nesus, Acunetix and others.

Manual Tests - Planning and testing from a hacker’s perspective

Why VAPT?

- **Helps monitor and patch vulnerabilities proactively**
- Exploits keep coming, so vigilance and routine patching is vital.
- **Helps address security gaps of your organization**
- VAPT helps maintain compliance
- **Helps keep hackers at bay**

Thank you.