Name:Kaushik Kotian Roll No.:29 Div:D20B Batch:B

Experiment 5

Aim: Study of OWASP vulnerabilities.

Theory:

Study of OWASP

1) What is OWASP?

1. The Open Web Application Security Project (OWASP) is a non-profit organization founded in 2001, with the goal of helping website owners and security experts protect web applications from cyber attacks. OWASP has 32,000 volunteers around the world who perform security assessments and research.
2. Among OWASP's key publications are the OWASP Top 10, discussed in more detail below; the OWASP Software Assurance Maturity Model (SAMM), the OWASP Development Guide, the OWASP Testing Guide, and the OWASP Code Review Guide.

2) Why is the OWASP Top 10 Important?

1. OWASP Top 10 is a research project that offers rankings of and remediation advice for the top 10 most serious web application security dangers. The report is founded on an agreement between security experts from around the globe. The risks are graded according to the severity of the vulnerabilities, the frequency of isolated security defects, and the degree of their possible impacts.
2. The aim of the report is to provide web application security experts and developers with an understanding into the most common security risks so that they can use the findings of the report as part of their security practices. This can help limit the presence of such known risks within their web applications.
3. OWASP manages the Top 10 list and has been doing so since 2003. They update the list every 2-3 years, in keeping with changes and developments in the AppSec market. OWASP provides actionable information and acts as an important checklist and internal Web application development standard for a lot of the largest organizations in the world.
4. Auditors tend to see an organization's remiss to address the OWASP Top 10 as a sign that it may not be up-to-scratch regarding compliance standards. Employing the Top 10 into its software development life cycle (SDLC) shows a general valuing of the industry's best practices for secure development.

3) OWASP Top 10 security risks is:

1. Injection: Injection attacks occur when untrusted data is sent to an interpreter as part of a command or query. This can lead to unintended execution of malicious commands.

2. Broken Authentication: Weaknesses in authentication mechanisms can lead to unauthorized access to systems. Attackers might exploit vulnerabilities like weak passwords, session management flaws, and more.
3. Sensitive Data Exposure: Inadequate protection of sensitive data, such as personal information or financial details, can result in data breaches and identity theft.
4. XML External Entities (XXE): XXE attacks involve the exploitation of insecure XML parsers, potentially allowing attackers to read local files, perform remote requests, and gain unauthorized access.
5. Broken Access Control: Poorly enforced access controls can lead to unauthorized users gaining access to resources they shouldn't have access to, potentially leading to data leaks and other security breaches.
6. Security Misconfiguration: Incorrectly configured security settings can expose vulnerabilities and sensitive information. This includes default settings, open ports, and overly permissive access controls.
7. Cross-Site Scripting (XSS): XSS attacks involve injecting malicious scripts into web applications, which are then executed by unsuspecting users' browsers. This can lead to the theft of user data or the hijacking of user sessions.
8. Insecure Deserialization: Insecure deserialization can lead to remote code execution and other attacks by manipulating serialized data.
9. Using Components with Known Vulnerabilities: Integrating third-party components with known vulnerabilities can expose applications to attacks that exploit those vulnerabilities.
10. Insufficient Logging and Monitoring: Inadequate logging and monitoring can hinder timely detection of security breaches and limit the ability to respond effectively to incidents.

Mitigations and Vulnerabilities:

Mitigations and vulnerabilities are concepts related to security, specifically in the context of information technology and systems. Let's explore these terms in more detail:

Vulnerabilities:
A vulnerability is a weakness or flaw in a system, application, or network that can be exploited by attackers to compromise the security of that system. Vulnerabilities can arise from coding errors, misconfigurations, design flaws, or other issues in software or hardware. Exploiting a vulnerability can lead to unauthorized access, data breaches, service disruptions, and other security incidents.

For example, if a web application doesn't properly validate user input and is susceptible to SQL injection attacks, that application has a vulnerability.

Mitigations:
Mitigations are measures and strategies implemented to reduce the impact or likelihood of a vulnerability being exploited. These measures are put in place to enhance the security of a system and to minimize the potential damage that could result from a successful attack.

Mitigations can take various forms, including:

Patching and Updates: Keeping software, operating systems, and applications up to date with the latest security patches and updates can address known vulnerabilities.

Secure Coding Practices: Developers can write secure code by following best practices, avoiding common coding errors, and using security libraries and frameworks.

Access Control: Implementing strong access controls ensures that only authorized users can access certain resources, reducing the attack surface.

Firewalls and Intrusion Detection/Prevention Systems: These tools monitor and filter network traffic to prevent unauthorized access and detect potential attacks.

Encryption: Encrypting sensitive data helps protect it even if it's intercepted by unauthorized parties.

Regular Security Audits: Conducting regular security assessments and audits can help identify vulnerabilities and weaknesses before they're exploited.

Security Training and Awareness: Educating employees and users about security best practices helps them avoid inadvertently contributing to vulnerabilities.

Least Privilege: Assigning the minimum level of permissions necessary to perform a task reduces the potential impact of a compromise.

Multi-factor Authentication (MFA): Requiring multiple forms of authentication before granting access adds an extra layer of security.

Security Configuration: Properly configuring systems, applications, and devices helps prevent misconfigurations that could lead to vulnerabilities.

Mitigations aim to address vulnerabilities and make it more difficult or less rewarding for attackers to exploit them. It's important to have a comprehensive approach to security that includes both preventive and reactive measures, as well as a plan for responding to security incidents when they do occur.

SQL Injections

A SQL injection attack consists of insertion or “injection” of a SQL query via the input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system. SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to affect the execution of predefined SQL commands.

1. **Threat Modeling**

SQL injection attacks allow attackers to spoof identity, tamper with existing data, cause repudiation issues such as voiding transactions or changing balances, allow the complete disclosure of all data on the system, destroy the data or make it otherwise unavailable, and become administrators of the database server.

SQL Injection is very common with PHP and ASP applications due to the prevalence of older functional interfaces. Due to the nature of programmatic interfaces available, J2EE and ASP.NET applications are less likely to have easily exploited SQL injections.

2. **Risk Factors**

i) SQL Injection has become a common issue with database-driven web sites. The flaw is easily detected, and easily exploited, and as such, any site or software package with even a minimal user base is likely to be subject to an attempted attack of this kind.

ii) Essentially, the attack is accomplished by placing a meta character into data input to then place SQL commands in the control plane, which did not exist there before. This flaw depends on the fact that SQL makes no real distinction between the control and data planes.

3. **Mitigation**

a) Prepared Statements (with Parameterized Queries)

Prepared statements ensure that an attacker is not able to change the intent of a query, even if SQL commands are inserted by an attacker. Parameterized queries allow the database to distinguish between code and data, regardless of what user input is supplied.

b) Stored Procedures

Stored procedures are not always safe from SQL injection. However, certain standard stored procedure programming constructs have the same effect as the use of parameterized queries when implemented safely which is the norm for most stored procedure languages.

c) Allow-list Input Validation

Various parts of SQL queries aren't legal locations for the use of bind variables, such as the names of tables or columns, and the sort order indicator (ASC or DESC). In such situations, input validation or query redesign is the most appropriate defense. For the names of tables or columns, ideally those values come from the code, and not from user parameters.

d) Escaping All User-Supplied Input

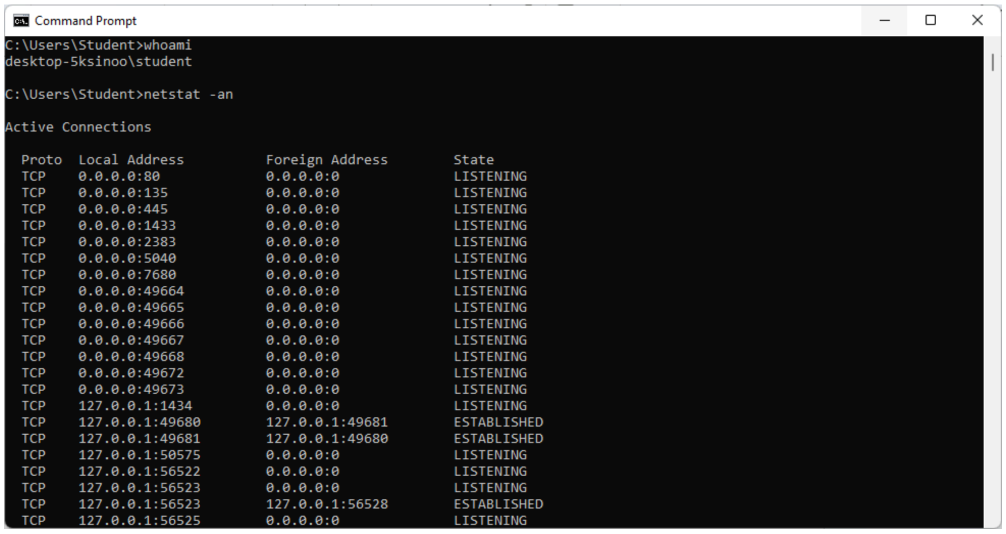
This technique should only be used as a last resort, when none of the above are feasible. Input validation is probably a better choice as this methodology is frail compared to other defenses and we cannot guarantee it will prevent all SQL Injections in all situations.

OS command injection

OS command injection (also known as shell injection) is a web security vulnerability that allows an attacker to execute arbitrary operating system (OS) commands on the server that is running an application, and typically fully compromise the application and all its data. Very often, an attacker can leverage an OS command injection vulnerability to compromise other parts of the hosting infrastructure, exploiting trust relationships to pivot the attack to other systems within the organization.

When you have identified an OS command injection vulnerability, it is generally useful to execute some initial commands to obtain information about the system that you have compromised. Below is a summary of some commands that are useful on Linux and Windows platforms:

Purpose of comman	Linux	Windows
Name of current user	whoami	whoami
Operating system	uname -a	ver
Network configuration	ifconfig	ipconfig /all
Network connections	netstat -an	netstat -an
Running processes	ps -ef	tasklist

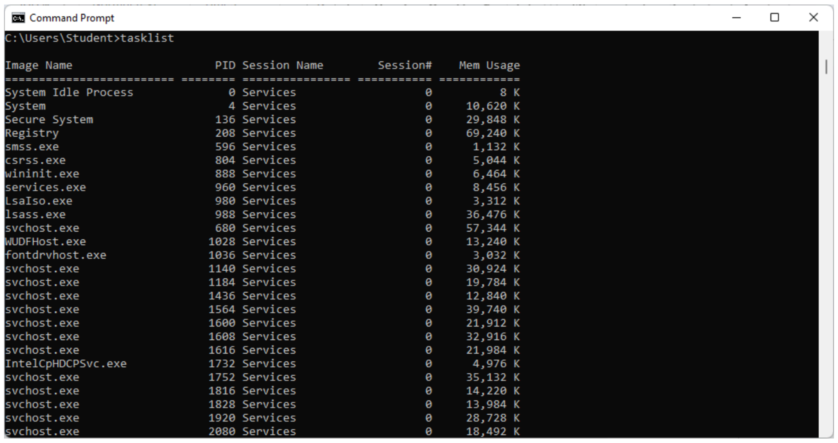


```
C:\Users\Student>whoami
desktop-5ksinoo\student

C:\Users\Student>netstat -an

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1433           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:2383           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:5040           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:7680           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49664          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49665          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49666          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49667          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49668          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49672          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49673          0.0.0.0:0              LISTENING
  TCP    127.0.0.1:1434         0.0.0.0:0              LISTENING
  TCP    127.0.0.1:49680        127.0.0.1:49681        ESTABLISHED
  TCP    127.0.0.1:49681        127.0.0.1:49680        ESTABLISHED
  TCP    127.0.0.1:50575        0.0.0.0:0              LISTENING
  TCP    127.0.0.1:56522        0.0.0.0:0              LISTENING
  TCP    127.0.0.1:56523        0.0.0.0:0              LISTENING
  TCP    127.0.0.1:56523        127.0.0.1:56528        ESTABLISHED
  TCP    127.0.0.1:56525        0.0.0.0:0              LISTENING
```



```
C:\Users\Student>tasklist

Image Name                     PID Session Name        Session#    Mem Usage
========================= ======== ================ =========== ============
System Idle Process              0 Services                   0          8 K
System                           4 Services                   0     10,620 K
Secure System                  136 Services                   0     29,848 K
Registry                       208 Services                   0     69,240 K
smss.exe                       596 Services                   0      1,132 K
csrss.exe                      804 Services                   0      5,044 K
wininit.exe                    888 Services                   0      6,464 K
services.exe                   960 Services                   0      8,456 K
LsaIso.exe                     980 Services                   0      3,312 K
lsass.exe                      988 Services                   0     36,476 K
svchost.exe                    680 Services                   0     57,344 K
WUDFHost.exe                  1028 Services                   0     13,240 K
fontdrvhost.exe               1036 Services                   0      3,032 K
svchost.exe                   1140 Services                   0     30,924 K
svchost.exe                   1184 Services                   0     19,784 K
svchost.exe                   1436 Services                   0     12,840 K
svchost.exe                   1564 Services                   0     39,740 K
svchost.exe                   1600 Services                   0     21,912 K
svchost.exe                   1608 Services                   0     32,916 K
svchost.exe                   1616 Services                   0     21,984 K
IntelCpHDCPSvc.exe            1732 Services                   0      4,976 K
svchost.exe                   1752 Services                   0     35,132 K
svchost.exe                   1816 Services                   0     14,220 K
svchost.exe                   1828 Services                   0     13,984 K
svchost.exe                   1920 Services                   0     28,728 K
svchost.exe                   2080 Services                   0     18,492 K
```

XML External Entities

XML External Entities (XXE) is a type of security vulnerability that occurs when an application processes XML input that contains a reference to an external entity. This vulnerability can lead to various attacks, including disclosure of internal files, denial of service, and remote code execution. The Open Web Application Security Project (OWASP) is a well-known organization that focuses on web application security and provides resources to help developers and security professionals understand and mitigate security risks.

Risk Factors

The application parses XML documents.
Tainted data is allowed within the system identifier portion of the entity, within the document type declaration (DTD).
The XML processor is configured to validate and process the DTD.
The XML processor is configured to resolve external entities within the DTD.

What Is the Impact of XXE Attacks?

Here are common consequences of XXE attacks:

1. **Disclosing local files**—threat actors can disclose files containing sensitive data, like passwords, using file: schemes or relative paths in the system identifier.
2. **Expanding the attack**—XXE attacks rely on the application that processes the XML document. Threat actors can use this trusted application to move to different internal systems.
3. **Remote code execution**—if the XML processor library is vulnerable to client-side memory corruption, a threat actor can dereference a malicious URI to allow arbitrary code execution under the application account.
4. I**mpacting application availability**—some XML attacks might allow actors to access local resources that do not stop returning data. If too many processes or threads are not released, it can negatively impact application availability.

Types of XXE Payloads:
There are several types of XML external entity attacks:

1. **XXE Exploit to Retrieve Files**
   An XXE attack can retrieve an arbitrary file from the target server's filesystem by modifying the submitted XML. The attacker introduces a DOCTYPE element defining an external entity that contains a path to the file. The attacker then edits the XML data value in the response.

2. **XXE Exploit to Perform SSRF**
   Attackers can use an XXE attack to perform server-side request forgery (SSRF), inducing the application to make requests to malicious URLs. This attack involves defining an external entity with the target URL and using the entity in the response's data value. It allows the attacker to view responses from the URL in an application's response, enabling interaction with the back end. In some cases, the attacker can only perform a blind SSRG attack and cause damage without viewing the response.

3. **Blind XXE Exploit to Exfiltrate Data**
   XXE vulnerabilities are often blind, meaning that the application doesn't return any values of external entities. Attackers cannot directly retrieve server-side files but can still detect and exploit blind XXE vulnerabilities with advanced methods like out-of-band data exfiltration or triggering an XML parsing error to disclose sensitive information.

4. **Blind XXE Exploit to Generate Error Messages**
   One way to exploit blind XXE vulnerabilities is to trigger parsing errors to generate an error message containing sensitive data. This approach is effective for applications that return a resulting error message in their response. Attackers can perform a blind XXE exploit by using malicious external DTDs to trigger an error message revealing the password file's contents.

XXE Prevention Best Practices
Here are popular XXE injections prevention techniques:

1. **Manual XXE Prevention**
   You can prevent vulnerabilities in entities outside of XML by configuring the XML parser to disallow custom DTDs. Since applications rarely require DTD, there are very few functional trade-offs. However, every parser in each programming language comes with its own requirements for setting this parameter.

2. **Disabling DTD Support**
   External DTDs are meant for use by trustworthy parties, but threat actors often exploit this legacy feature to attack web applications. You can disable DTD to prevent XXE attacks. However, if you cannot disable DTDs, you can still mitigate this risk by disabling the external entity functionality.

Sensitive Data Exposure

Sensitive data is anything that should not be accessible to unauthorized access, known as sensitive data. Sensitive data may include personally identifiable information (PII), such as Social Security numbers, financial information, or login credentials. Sensitive Data Exposure occurs when an organization unknowingly exposes sensitive data or when a security incident leads to the accidental or unlawful destruction, loss, alteration, or unauthorized disclosure of, or access to sensitive data. Such Data exposure may occur as a result of inadequate protection of a database, misconfigurations when bringing up new instances of datastores, inappropriate usage of data systems, and more.

1. **Types of Sensitive Data Exposure**
   Sensitive Data Exposure can of the following three types:

   Confidentiality Breach: where there is unauthorized or accidental disclosure of, or access to, sensitive data.
   Integrity Breach: where there is an unauthorized or accidental alteration of sensitive data.

   Availability Breach: where there is an unauthorized or accidental loss of access to, or destruction of, sensitive data. This will include both the permanent and temporary loss of sensitive data.

Conclusion:

A thorough understanding of OWASP vulnerabilities enables proactive identification and remediation of security weaknesses, reducing the risk of cyberattacks, data breaches, and unauthorized access.