**DAY 5 TASKS**

**Task 1: Understanding OWASP Top Ten**

**Objective:** To understand the most critical security risks outlined by OWASP.

**Steps:**

1. **Visit the OWASP website and read about the OWASP Top Ten vulnerabilities.**

This is a list of the top 10 most critical web application security risks. It helps developers identify and mitigate the most prevalent vulnerabilities in web applications.

1. **Identify the top 3 risks and write a brief explanation of each**.

The list includes threats like

* **Injection :** A type of attack where malicious code is inserted into a system's input to exploit vulnerabilities.
* **Broken Authentication :** Weak or compromised authentication mechanisms that allow unauthorized access.
* **Sensitive Data Exposure** : Accidental or intentional exposure of sensitive data, such as passwords, credit card numbers, or personal information.

1. **Research examples of each risk in real-world applications.**

Examples of each risk in real-world applicationsare :

* **Injection :**

SQL Injection

Command Injection

* **Broken Authentication :**

Session Hijacking

Insufficient Login Failure Handling

* **Sensitive Data Exposure:**

Data Breaches

Insufficient Logging and Monitoring

**Question: What are the top 3 most critical web security risks according to OWASP?**

the top 3 most critical web security risks according to OWASP

1. Broken Access Control: This occurs when an application fails to properly implement access controls, allowing unauthorized users to access sensitive data or functionality.
2. Cryptographic Failures: Weak or misconfigured cryptographic implementations can lead to data breaches and unauthorized access.
3. Injection: Injection attacks, such as SQL injection and cross-site scripting (XSS), allow attackers to inject malicious code into web applications to steal data, manipulate the application's behavior, or launch further attacks.

**Task 2: SSL/TLS Basics**

**Objective:** To understand SSL and its successor TLS, and how they secure web communications**.**

**Steps:**

**1. Research the differences between SSL and TLS.**

SSL, the older protocol, has been superseded by TLS, which offers enhanced security features. While SSL is technically deprecated, the term "SSL certificate" is commonly used to refer to TLS certificates due to historical reasons. TLS provides stronger encryption, improved key exchange methods, and overall better protection against attacks, making it the standard for secure web communication today.

**2. Learn how SSL/TLS works during a secure handshake.**

When you access an HTTPS website, a secure connection is established through an SSL/TLS handshake. The client (your browser) initiates the process by sending a message to the server, specifying supported protocols and encryption methods. The server responds with its chosen settings and a digital certificate, which is verified by the client to ensure authenticity. A shared secret key is then generated using public-key cryptography. This key is used to encrypt and decrypt data during the session, ensuring secure communication between the client and server.

**3. Examine a website’s SSL certificate by clicking on the padlock icon in the browser.**

To examine a website's SSL certificate, follow these steps:

1. **Locate the Padlock Icon:** Look for the padlock icon in your browser's address bar. It's usually located to the left of the website's URL.
2. **Click the Padlock:** Click on the padlock icon to open a detailed view of the website's security certificate.
3. **Inspect the Certificate Details:** The information displayed can vary slightly between browsers, but typically includes:
   * Issuer: The Certificate Authority (CA) that issued the certificate.
   * Valid From/To: The date range during which the certificate is valid.
   * Subject: The website or organization the certificate is issued to.
   * Public Key: The public key used for encryption.
   * Algorithm: The encryption algorithm used by the certificate.

**Question**: What is the main difference between SSL and TLS, and why is TLS preferred today?

The primary difference between SSL (Secure Sockets Layer) and TLS (Transport Layer Security) lies in their security features and overall robustness.

While SSL was a pioneering protocol, it has been superseded by TLS due to its numerous security vulnerabilities. 2 TLS, being the successor, offers significantly enhanced security protocols, stronger encryption algorithms, and improved key exchange methods. 3 This makes TLS the preferred choice for secure web communication today

**Task 3: Authentication vs Authorization**

**Objective**: To distinguish between authentication and authorization.

**Steps**:

1. Define authentication and authorization in your own words.

**Authentication** is the process of verifying a user's identity. It's like showing a valid ID to prove who you are. In digital terms, this often involves providing credentials like a username and password or using biometric methods like fingerprint or facial recognition.

**Authorization**, on the other hand, is about determining what a verified user is allowed to do. It's like granting specific permissions, such as access to certain files or the ability to perform particular actions. For instance, an administrator might have full access to a system, while a regular user might only be able to view certain data.

2. Create an example scenario where authentication and authorization are required.

Imagine you're trying to access your online banking account.

**Authentication:** You'll first need to prove your identity by entering your username and password. This is the authentication step.

**Authorization:** Once your identity is verified, the system checks your authorization level. Are you a regular user with limited access, or a bank employee with full privileges? Based on our authorization level, the system determines what actions we can perform, such as viewing your balance, transferring funds, or approving loans.

3. Research how both processes work in a typical web application.

**Authentication** is the process of verifying a user's identity. This usually involves the user providing credentials like a username and password. The web application then checks these credentials against a database or other authentication store. If the credentials are valid, the application creates a session for the user, assigning a unique session ID. This session ID is stored as a cookie in the user's browser and is sent back to the server with each subsequent request, allowing the server to identify the user.

**Authorization** is the process of determining what actions a user is allowed to perform. This is often based on the user's role or permissions. For example, an administrator might have full access to the application, while a regular user might only be able to view certain content.

**Question**: In a web application, if a user can log in but cannot access certain resources, which part of security (authentication or authorization) is responsible?

**Authorization**

**Task 4: Cookie-Based Authentication**

**Objective:** To understand how cookie-based authentication works.

**Steps:**

**1. Read about how cookies are used for user sessions in web applications**

Cookies are small text files used to store information about a user's session on a website. When a user visits a website for the first time, the server creates a unique session ID and sends it to the user's browser as a cookie.

With each subsequent request, the browser sends the cookie back to the server, allowing the server to identify the user's session and retrieve their information. This enables websites to provide personalized experiences, remember user preferences, and maintain state between page requests..

**2. Use developer tools in a browser to view cookies when logged into a website.**

**3. Examine the properties of the session cookie (e.g., domain, expiration).**

**Question: How do session cookies help maintain user authentication across multiple pages of a website?**

Session cookies play a crucial role in maintaining user authentication across multiple pages of a website. When a user logs in, the server generates a unique session ID and sends it to the user's browser as a cookie. This cookie is stored on the user's device and is sent back to the server with each subsequent request. The server uses this session ID to identify the user and retrieve their session data, including their authentication status. This allows the user to navigate through different pages of the website without having to re-enter their credentials.

For example, if a user logs into an online store and adds items to their shopping cart, the session cookie allows the store to remember the user's cart across multiple pages. When the user proceeds to checkout, the server can verify their identity and access their shopping cart information using the session cookie.

**Task 5: Token-Based Authentication**

**Objective:** To learn how token-based authentication (JWT) works**.**

**Steps:**

**1. Research how JSON Web Tokens (JWT) are used in authentication.**

JSON Web Tokens (JWTs) are a popular method for implementing authentication in web applications. After successful authentication, the server generates a JWT containing information about the user, such as their username and roles.

This JWT is sent to the client, who stores it and includes it in subsequent requests to protected resources. The server verifies the JWT's signature and claims to authorize the request. JWTs offer stateless authentication, improved security, flexibility, and scalability.

**2. Generate a simple JWT token using an online tool.**

To generate a simple JWT token, you can use an online tool like jwt.io. Here's a basic example:

1. Open your web browser and navigate to <https://jwt.io/>.
2. Header:
   * Algorithm: HS256 (a common algorithm for signing JWTs with a secret key)
3. Payload:
   * **iss:** Your application's issuer (e.g., "Your App")
   * **sub**: Subject (e.g., the user's username)
   * **exp:** Expiration time (e.g., 1 hour from now)
4. Secret:
   * Enter a secret key (keep it secure)
5. Click "Encode & Decode": This will generate the JWT token**.**

**3. Explore how tokens are transmitted in API requests.**

Tokens are typically transmitted in API requests through the Authorization header. This header specifies the authentication scheme and credentials, in this case, the token. The common format is:

**Authorization: Bearer <token\_value>**

Here, Bearer indicates the authentication scheme, and <token\_value> is replaced with the actual token string. The server then extracts the token from the header and validates it to authorize the request.

Sources and related content

**Question: What are the benefits of using token-based authentication over cookie-based authentication?**

Token-based authentication offers several advantages over cookie-based authentication:

1**. Statelessness:**

* Token-based authentication is stateless, meaning the server doesn't need to store session information for each user. This improves scalability and reduces server load.

**2. Enhanced Security:**

* Tokens can be signed and encrypted, making them more secure than plain text cookies.
* Tokens can be revoked or expired, limiting the potential damage of compromised tokens.

**3. Cross-Domain Support:**

* Tokens can be used across different domains, making them suitable for single sign-on (SSO) and microservices architectures.

**4. Flexibility:**

* Tokens can be customized to include additional information about the user, such as roles and permissions.

**5. Mobile-Friendly:**

* Token-based authentication is well-suited for mobile applications, as tokens can be stored securely on the device and used to authenticate requests.

**Task 6: Network Security Groups (NSGs)**

**Objective: To learn how Network Security Groups (NSGs) work in cloud environments.**

**Steps:**

1. Learn about NSGs and how they filter traffic in cloud services like Azure or AWS.

Network Security Groups (NSGs) are a fundamental security tool in cloud services like Azure and AWS. They act as firewalls that filter inbound and outbound network traffic to and from Azure resources within a virtual network.

NSGs contain a set of security rules that define which traffic is allowed or denied. Each rule specifies:

* **Source:** The IP address or IP address range of the source traffic.
* **Destination:** The IP address or IP address range of the destination traffic.
* **Port:** The port number or port range of the traffic.
* **Protocol:** The protocol used (e.g., TCP, UDP, or ICMP).
* **Action:** Whether to allow or deny traffic matching the rule.
* **Priority:** The order in which rules are evaluated (higher priority rules take precedence).

2. Create a simple NSG rule in a cloud platform.

Navigate to the Network Security Group in your cloud platform (Azure, AWS, etc.).

Click on "Add inbound security rule."

Configure the rule:

* **Name:** Give a descriptive name.
* **Priority:** Assign a priority number (lower numbers have higher priority).
* **Source:** Specify the source IP address or IP range.
* **Source port range:** Specify the source port range.
* **Destination port range:** Specify the destination port range (e.g., 22 for SSH).
* **Protocol:** Choose the protocol (TCP, UDP, ICMP).
* **Action:** Select "Allow" or "Deny."
* **Destination:** Specify the destination IP address or subnet.

Click "Save" to create the rule.

3. Test connectivity to a cloud resource by modifying NSG rules.

* Identify the resource you want to test connectivity to.
* Review the existing NSG rules for the resource's subnet or NIC.
* Modify or add NSG rules to allow inbound and outbound traffic to necessary ports.
* Test connectivity using appropriate tools (RDP, SSH, web browser, application-specific tools).
* Analyze the results and adjust NSG rules if necessary.

**Question: How do Network Security Groups help secure cloud resources?**

Network Security Groups (NSGs) act as firewalls for cloud resources, filtering inbound and outbound network traffic. By defining specific rules for allowed and denied traffic, NSGs help secure cloud resources in the following ways:

* **Preventing Unauthorized Access**: NSGs can block incoming traffic from untrusted sources, reducing the risk of unauthorized access.
* **Limiting Exposure:** By restricting access to specific ports and protocols, NSGs limit the attack surface of cloud resources.
* **Enforcing Security Policies**: NSGs can be used to enforce security policies, such as blocking certain types of traffic or limiting access to specific IP addresses.
* **Protecting Against DDoS Attacks:** NSGs can be used to mitigate DDoS attacks by limiting the amount of traffic that can reach a resource.
* **Segmenting Networks:** NSGs can be used to segment networks into smaller, more secure subnets, reducing the blast radius of potential security breaches.

**Task 7: Web Application Firewall (WAF)**

**Objective:** To understand the function of a Web Application Firewall (WAF).

**Steps:**

1. Research what a WAF is and how it protects web applications.

A Web Application Firewall (WAF) is a security tool that protects web applications from various attacks by filtering and monitoring HTTP traffic. It acts as a shield between a web application and the internet, analysing incoming and outgoing requests to identify and block malicious activity.

WAFs protect against common web application vulnerabilities like:

* **SQL Injection:** Prevents malicious SQL queries from being injected into the application's database.
* **Cross-Site Scripting (XSS):** Blocks malicious scripts from being executed in a user's browser.
* **Cross-Site Request Forgery (CSRF):** Prevents unauthorized actions on a user's behalf.
* **Brute-Force Attacks:** Detects and blocks repeated login attempts.
* **Other Web Application Attacks:** Protects against a wide range of other web-based threats.

2. Check if your web hosting service includes a WAF.

3. Test the protection provided by a WAF against common attacks like SQL injection.

To test WAF protection against SQL injection, you can use:

1. **Web Application Vulnerability Scanners**: Tools like Acunetix, Burp Suite, or OWASP ZAP can automate vulnerability scanning and WAF testing.
2. **Manual Testing (with Caution):** Use simple and complex SQL injection payloads on a non-production environment to observe the WAF's response**.**

**Question: How does a WAF differ from a traditional network firewall in terms of its security focus?**

A WAF and a traditional network firewall differ in their primary security focus:

* **Network Firewall:**
  + Operates at the network layer (Layer 3) of the OSI model.
  + Focuses on blocking network traffic based on IP addresses, port numbers, and protocols.
  + Protects against network-level threats like DDoS attacks, port scans, and unauthorized access.
* **Web Application Firewall (WAF):**
  + Operates at the application layer (Layer 7) of the OSI model.
  + Analyzes HTTP traffic and protects against web application vulnerabilities like SQL injection, XSS, and CSRF.
  + Focuses on application-specific attacks that target the logic and data of web applications.

**Task 8: Encryption of Data**

**Objective: To understand how encryption works to protect data.**

**Steps:**

1. Learn the difference between symmetric and asymmetric encryption.

**Symmetric Encryption:**

* Uses a single key for both encryption and decryption.
* Faster and more efficient for large data sets.
* Key distribution is a challenge, as the same key must be securely shared between parties.

**Asymmetric Encryption:**

* Uses a pair of keys: a public key for encryption and a private key for decryption.
* Slower than symmetric encryption but offers better security.
* Secure key distribution, as the public key can be shared widely.
* Commonly used for digital signatures and secure communication.

2.Research how encryption is used in data protection, both in transit and at rest.

Encryption is a crucial technique to safeguard data both in transit and at rest.

**Data in Transit:**

* **TLS/SSL:** This protocol encrypts data as it travels between a client and a server, ensuring that eavesdroppers cannot intercept sensitive information.
* **VPN:** Virtual Private Networks encrypt data transmitted over public networks, creating a secure private connection.

**Data at Rest:**

* **File-Level Encryption:** Individual files can be encrypted using strong encryption algorithms like AES.
* **Disk-Level Encryption:** Entire disks or partitions can be encrypted, protecting all data stored on them.
* **Database Encryption:** Databases can be encrypted at various levels, including table-level, column-level, or full-disk encryption.

1. Test encryption using a tool like OpenSSL or similar.

Basic example of how to encrypt and decrypt a text file using OpenSSL:

* **Generate a Key:**

openssl genrsa -out private.pem 2048

* **Encrypt a File:**

openssl rsautl -encrypt -pubin -inkey public.pem -in plaintext.txt -out ciphertext.txt

* **Decrypt the File:**

openssl rsautl -decrypt -inkey private.pem -in ciphertext.txt -out decrypted.txt

**Question: Why is it important to encrypt both data at rest and data in transit?**

Encrypting both data at rest and in transit is essential for comprehensive data protection. Encrypting data at rest, such as data stored on hard drives or in databases, safeguards it from unauthorized access in case of physical theft or security breaches.

Encrypting data in transit, such as data transmitted over networks, protects it from interception and eavesdropping by malicious actors. By implementing both types of encryption, organizations can significantly enhance their security posture and minimize the risk of data breaches.

**Task 9: Identity and Access Management (IAM)**

**Objective**: To understand the role of IAM in securing cloud resources.

**Steps**:

1. Research IAM concepts like roles, permissions, and access policies.

2. Create a sample IAM policy that allows access to a cloud resource for a specific role.

3. Test how IAM controls user access and permissions within a cloud platform.

**Question**: What is the principle of least privilege, and why is it important in IAM?

IAM (Identity and Access Management) is a framework for managing digital identities and controlling access to resources. It ensures that the right people have the right access to the right resources.

1. **Identity:** Represents a user, group, or service account.
2. **Role:** A collection of permissions that define what actions an identity can perform.
3. **Permission:** An individual right to perform a specific action on a resource.
4. **Access Policy:** A set of rules that determine who can access what resources and under what conditions.

**Task 10: Vulnerability Testing with OWASP ZAP**

**Objective**: To perform vulnerability testing on a website using OWASP ZAP.

**Steps**:

1. Install OWASP ZAP and set it up for scanning.

2. Run a scan on a publicly available test site (e.g., OWASP Juice Shop).

3. Review the scan results and identify any critical vulnerabilities.

**Question**: What types of vulnerabilities can OWASP ZAP detect, and how can it help improve website security?

OWASP ZAP can detect a wide range of web application vulnerabilities, including:

* **Injection vulnerabilities:** SQL injection, command injection, and others.
* **Broken authentication and session management:** Weak password policies, session hijacking, and other vulnerabilities.
* **Sensitive data exposure:** Exposure of sensitive data like passwords or credit card numbers.
* **Broken access control:** Improper access controls that allow unauthorized access.
* **Security misconfiguration:** Misconfigured servers, frameworks, or libraries.
* **Cross-site scripting (XSS):** Malicious scripts injected into web pages.
* **Insecure deserialization:** Deserializing untrusted data.
* **Using components with known vulnerabilities:** Using outdated or vulnerable libraries and frameworks.
* **Missing function level access control:** Lack of authorization checks at the function level.