**ASSIGNMENT 1**

**21BCY10123 – HARSHITAA ASHISH**

**OWASP VULNERABILITIES**

**Broken access control**

The OWASP (Open Web Application Security Project) Broken Access Control is a category of security vulnerabilities that refers to improper implementation of access controls, leading to unauthorized access to resources, data, or functionalities within a web application. Access control mechanisms are designed to ensure that users have the appropriate permissions to access certain resources, and when these mechanisms are not implemented correctly, it can result in serious security breaches.

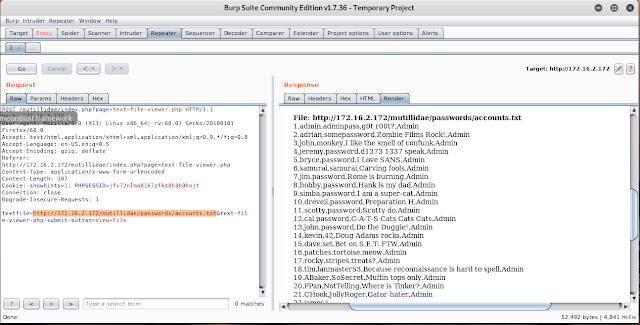
Impact:

* Unauthorized Data Exposure: Sensitive data might be accessed by unauthorized users, leading to data breaches and privacy violations.
* Data Manipulation: Attackers can modify data, compromise integrity, and perform unauthorized actions.
* Business Logic Compromise: Attackers can abuse improperly restricted functionalities, affecting the intended operation of the application.
* Financial Loss: Unauthorized access can lead to financial loss, legal liabilities, and reputation damage.

Prevention:

* Implement Strong Access Controls: Apply the principle of least privilege, ensuring users only have access to what is necessary for their roles.
* Use Frameworks and Libraries: Leverage established security libraries and frameworks to implement access control mechanisms.
* Secure APIs: Properly implement authentication and authorization mechanisms in APIs to prevent unauthorized data exposure.
* Validate Input: Validate and sanitize user inputs to prevent attackers from exploiting direct object references.
* Test: Regularly perform security testing, including penetration testing, to identify and fix access control vulnerabilities.
* Regular Auditing: Regularly review and audit access control configurations to identify and correct misconfigurations.





we can see the output of the accounts.txt file in the response window. It means this file is not restricted to get access and view the content. Using this method a hacker can easily go through the directory and get access to other files too. So the best practice is, implement access control on server directories and files.

**Cryptographic failure**

OWASP Cryptographic Failures refer to vulnerabilities and weaknesses that arise from improper implementation or use of cryptographic techniques within applications. Cryptography plays a crucial role in ensuring data confidentiality, integrity, and authentication. When cryptography is not correctly applied, it can lead to security breaches and data exposure.

Impact:

* Data Exposure: Sensitive data might be exposed due to weak encryption or insecure storage.
* Data Tampering: Attackers can modify encrypted data or inject malicious content if data integrity is not ensured.
* Data Breaches: Inadequate encryption can lead to unauthorized access to sensitive information.
* Replay Attacks: Without proper security measures, attackers might capture and replay encrypted data.
* Loss of Trust: Cryptographic failures can lead to a loss of user trust, reputation damage, and legal liabilities.

Prevention:

* Use Strong Algorithms: Use well-established, up-to-date cryptographic algorithms that are considered secure.
* Key Management: Implement proper key management practices, such as securely generating, storing, and rotating encryption keys.
* Secure Transport: Use encryption (e.g., HTTPS) to secure data in transit.
* Random Number Generation: Use strong random number generators for cryptographic operations.
* Data Integrity: Implement integrity checks (e.g., hash functions) to ensure data hasn't been tampered with.
* Secure Storage: Store sensitive data in encrypted form using strong encryption algorithms.



Identifying that the website uses weak ciphers, testssl.sh can be used again to find that this causes it to be exploitable via Sweet32. With this information, an attacker now knows that a Man in the Middle attack could be used to capture sensitive data, then easily be cracked using the Sweet32 vulnerability. This vulnerability showcases how important it is to ensure that the strongest, most up to date encryption algorithms be used whenever possible, especially considering that encryption algorithms are constantly evolving.

**Injection**

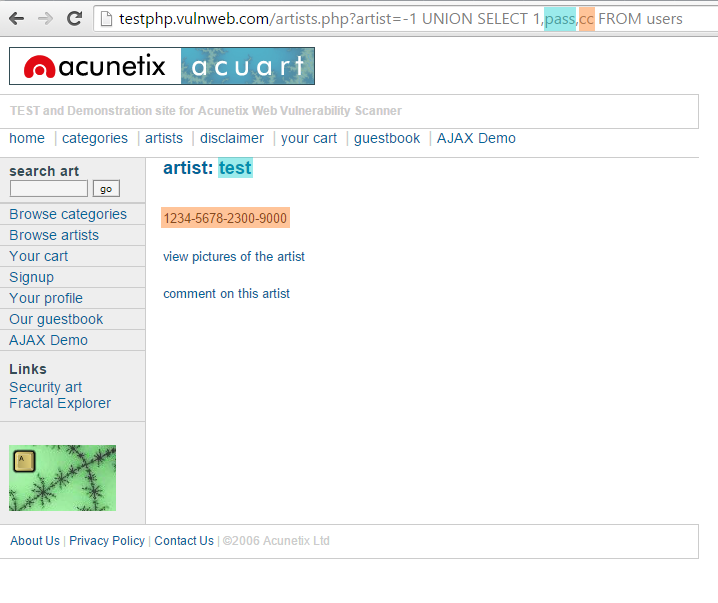
OWASP Injection vulnerabilities refer to security weaknesses that occur when untrusted data is improperly handled and subsequently executed as part of a command or query within an application. These vulnerabilities can allow attackers to manipulate the behavior of an application, potentially leading to unauthorized access, data leakage, or even complete system compromise.

Impact:

* Unauthorized Data Access: Attackers can access sensitive data or manipulate application behaviour to gain unauthorized access.
* Data Leakage: Sensitive information might be leaked, leading to privacy violations and potential regulatory fines.
* Application Defacement: Attackers can inject malicious content that defaces or modifies the appearance of web pages.

Prevention:

* Input Validation: Implement strong input validation by sanitizing and validating user inputs before processing them.
* Parameterized Queries: Use parameterized queries or prepared statements in database interactions to prevent SQL injection.
* Output Encoding: Properly encode output data to prevent Cross-Site Scripting (XSS) attacks.
* Whitelisting: Use whitelists to define valid input patterns and reject inputs that don't adhere to these patterns.
* Contextual Escaping: Apply context-based escaping when rendering dynamic content to mitigate XSS vulnerabilities.



The artist parameter is vulnerable to SQL Injection. The following payload modifies the query to look for an inexistent record. It sets the value in the URL query string to -1. Of course, it could be any other value that does not exist in the database. However, a negative value is a good guess because an identifier in a database is rarely a negative number. In SQL Injection, the UNION operator is commonly used to attach a malicious SQL query to the original query intended to be run by the web application. The result of the injected query will be joined with the result of the original query. This allows the attacker to obtain column values from other tables.

**Insecure design**

OWASP Insecure Design refers to a category of security vulnerabilities that stem from poor or insecure design decisions made during the development of software applications. These vulnerabilities can lead to fundamental weaknesses that are difficult to address later in the development lifecycle. Insecure design can result in vulnerabilities that attackers can exploit to compromise the security and integrity of an application.

Impact:

* Widespread Vulnerabilities: Insecure design decisions can lead to vulnerabilities that affect multiple parts of an application.
* Difficulty in Remediation: Insecure design flaws can be challenging to fix later in the development process, often requiring substantial rework.
* Data Breaches: Insecure design can lead to unauthorized access to sensitive data, resulting in data breaches and privacy violations.

Prevention:

* Secure Software Development Lifecycle (SDLC): Implement a secure SDLC that includes security reviews and assessments during the design phase.
* Security by Design: Incorporate security principles and practices into the design phase of the development process.
* Threat Modeling: Identify potential threats and risks during the design phase to address them proactively.
* Secure Architecture: Design a secure system architecture that includes defense-in-depth, proper isolation, and appropriate access controls.
* Authorization and Authentication: Implement strong authorization and authentication mechanisms, including multi-factor authentication.
* Input Validation: Ensure proper validation and sanitization of user inputs to prevent injection vulnerabilities.

**Security misconfiguration**

OWASP Security Misconfiguration is a category of security vulnerabilities that occur when security settings, configurations, or defaults are improperly implemented or left in their default state. These misconfigurations can lead to security weaknesses that attackers can exploit to gain unauthorized access, compromise data, or disrupt the application's functionality.

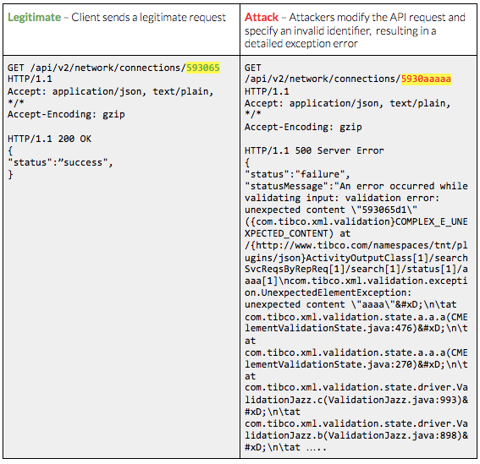
Impact:

* Unauthorized Access: Attackers can exploit misconfigurations to gain unauthorized access to systems, applications, or data.
* Data Exposure: Sensitive data might be exposed due to misconfigured permissions or unprotected data storage.
* System Disruption: Misconfigurations can lead to application downtime, service disruptions, and compromised availability.

Prevention:

* Secure Defaults: Configure systems and applications with secure defaults to minimize the attack surface.
* Regular Security Audits: Regularly review and audit configurations to identify and address misconfigurations.
* Minimal Privileges: Assign the principle of least privilege, giving users and applications only the permissions, they need.
* Patch and Update: Keep software, frameworks, libraries, and components up-to-date to mitigate known vulnerabilities.
* Secure Configuration Management: Implement version control for configurations and apply them consistently across environments.

In the example above, the attacker modified the connectionId parameter of the GET request to an API,  causing the application server to respond with a detailed exception error with stack trace information. This type of error can include information about the application environment, such as software vendor names, software packages used, software versions, and lines of code within the backend server-side code that the error resulted from.  All of this information is invaluable to an attacker who is performing reconnaissance in order to gain an understanding of the infrastructure that serves applications and APIs, as well as the application code itself, in order to discover other exploitable vulnerabilities.



**Vulnerable and outdated components**

OWASP Vulnerable and Outdated Components refers to a category of security vulnerabilities that occur when software applications use outdated or known vulnerable third-party components, libraries, or frameworks. These components might have known security flaws that attackers can exploit to compromise the security of the application.

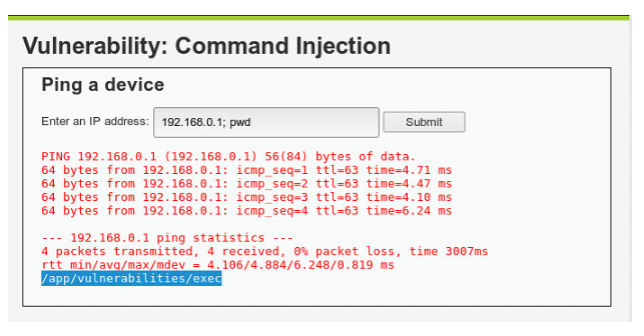
Impact:

* Exploitation: Attackers can exploit known vulnerabilities in components to compromise the application's security.
* Data Breaches: Vulnerabilities in components can lead to unauthorized access to sensitive data.
* Loss of Trust: Using outdated components might erode user trust due to the perception of inadequate security.

Prevention:

* Component Management: Maintain an inventory of all third-party components and libraries used in the application.
* Regular Updates: Keep all components and libraries up-to-date with the latest security patches and updates.
* Automated Tools: Use automated vulnerability scanning tools to identify known vulnerabilities in components.
* Component Monitoring: Subscribe to security alerts and updates for components and libraries.

Command injection is a type of attack that allows an attacker to execute arbitrary commands on a target system. This can be done by submitting malicious input to a program that executes system commands. If the input is not properly validated, the attacker can execute arbitrary commands on the system. An attacker can use command injection to take control of a system, install malware, or even delete files. Below is an example from the open source project, damn vulnerable web application (DVWA). Here the attacker inputs the present work directory (pwd) command into a web input form and the command is executed, returning the highlighted text.



**Identification and Authentication Failures**

OWASP Identification and Authentication Failures refer to a category of security vulnerabilities that arise from weaknesses or flaws in the identification and authentication mechanisms of software applications. These vulnerabilities can lead to unauthorized access, data breaches, and compromised user accounts due to inadequate or improperly implemented identification and authentication processes.

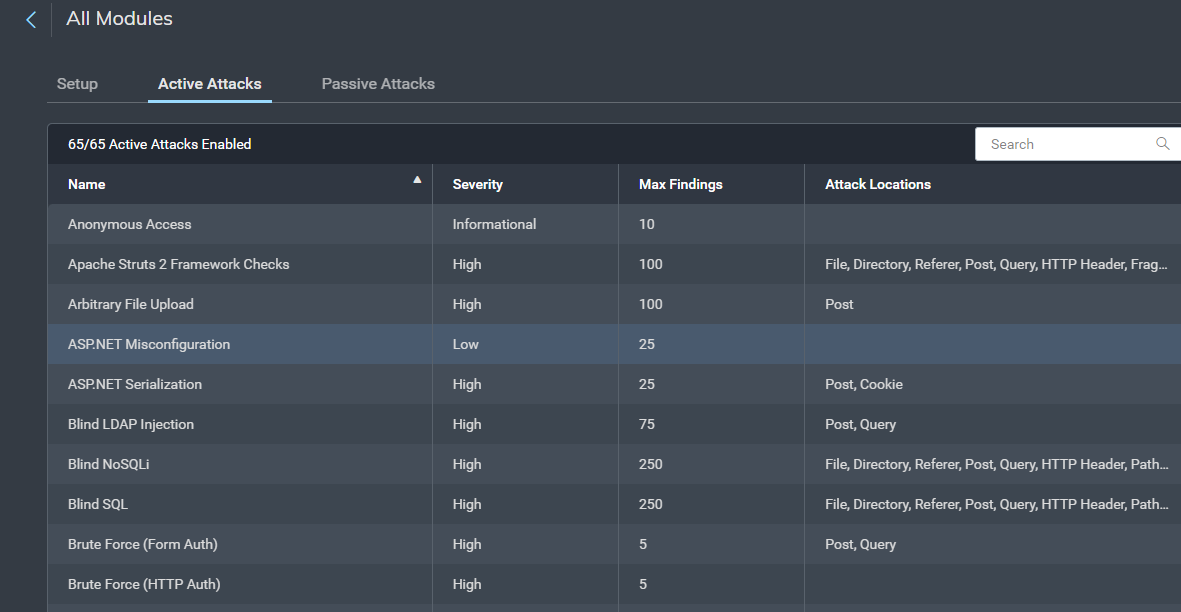
Impact:

* Unauthorized Access: Attackers can gain unauthorized access to user accounts and sensitive data.
* Data Breaches: Weak authentication can lead to data breaches and unauthorized exposure of sensitive information.
* Identity Theft: Attackers can impersonate legitimate users, potentially leading to identity theft or fraud.

Prevention:

* Strong Password Policies: Enforce strong password policies, including complexity requirements and regular password updates.
* Secure Authentication Mechanisms: Implement secure authentication methods, such as OAuth, OpenID Connect, and token-based authentication.
* Secure Password Storage: Use strong hashing algorithms with salts to securely store passwords.
* Secure Session Management: Implement proper session management practices to prevent session fixation and hijacking.

As organizations continue to protect their applications from identification and authentication issues, there will be added mechanisms for authentication in place for protection. As a part of any good application security program, these applications will still be to be scanned and tested. Having a scanning solution in place to be able to authenticate appropriately through these security protections is essential for organizations to address their identity- and access-related vulnerabilities.



**Security login and monitoring failures**

OWASP Security Login and Monitoring Failures refer to vulnerabilities and weaknesses related to the login mechanisms and monitoring practices within software applications. These vulnerabilities can result in unauthorized access, improper use of credentials, and inadequate tracking of user activities, compromising the security of the application.

Impact:

* Unauthorized Access: Weak authentication can lead to unauthorized access to user accounts and sensitive data.
* Data Breaches: Inadequate login security can result in data breaches and unauthorized exposure of sensitive information.
* Identity Theft and Fraud: Unauthorized access might lead to identity theft, fraud, or misuse of user accounts.

Prevention:

* Strong Authentication: Implement strong authentication methods, such as password policies and multi-factor authentication (MFA).
* Secure Session Management: Implement proper session management practices to prevent session-related vulnerabilities.
* Account Lockout: Implement account lockout mechanisms to prevent brute-force attacks.
* Proper Logging and Monitoring: Implement robust logging and monitoring practices to track user activities and security events.
* Login Failure Handling: Implement rate limiting or CAPTCHA mechanisms to deter brute-force attacks.

Open Event Viewer in Active Directory and navigate to Windows Logs> Security. The pane in the center lists all the events that have been setup for auditing. You will have to go through events registered to look for failed logon attempts. Once you find them, you can right click on the event and select Event Properties for more details. In the window that opens, you can find the IP address of the device from which the logon was attempted.

