Farah Anany 19103787

Cyber Final Report

## Introduction

There are 10 Top OWASP 2021 Vulnerabilities. They were organized according to severity.

## Broken Access Control

* Every application has a mechanism that is known by access control, and it’s used to allow certain users to access certain functionalities, databases, files, directories, resources…
* When a certain user that is not authorized to access certain files manages to access these files, broken access control has taken place.
* Broken Access Control is a type of security vulnerability that has to do with access control.
* When the developer doesn’t properly limit access to sensitive data, a web application becomes prone to this type of vulnerability.
* Broken Access Control is basically bypassing authorization.
* One type of this vulnerability is **Not limiting URL access**: This is when the developer has an ID for the user that is logged in the application and the attacker might change this ID while intercepting the log in process and via that the attacker might be able to
* Another type of Broken Access Control is IDOR (Insecure Object References). This happens when an attacker can extract the page source and gains access to the admin page.
* Mitigation includes properly implementing authentication methods, implementing Role-Based Access Control (RBAC), parametrizing data queries, and encrypting session data.
* The severity of this vulnerability: Exposure of Information

## Cryptographic Failures

* Cryptographic Failures happen when the web application has little or no encryption of data.
* This can threaten the confidentiality of the web application as an eavesdropper might steal the data if the data that is being inputted and transmitted is not encrypted.
* If poorly encrypted or not encrypted at all, usernames, passwords, financial data, emails, accounts might be prone to being stolen.
* Another fail of cryptography is when a developer relies on the encryption of the database. It is automatically decrypted when it is retrieved which makes it prone to SQL injections.
* Attackers can also crack the hash if there’s no salting involved in the creation of the hashed passwords.
* Mitigation: To protect the data, it must be encrypted at rest and in transit and encryption keys must be created cryptographically, secure coding
* Encrypting data in transit can help prevent man in the middle attacks.
* Severity: Extremely Dangerous

## Injection

* This type of vulnerability happens when the user can input anything in an input field and then the web application interprets it as a command or parameter.
* Two very common examples include SQL injections and command injections.
* **A SQL injection** is when an attacker uses queries to gain access to information that should not be shown, like dumping all the data from the database or even wiping out the entire database.
* **A Command injection** is when an attacker inputs a system command and can gain access to the user’s system.
* **Examples of Command Injections:**
* whoami
* id
* ifconfig/ip addr
* uname -a
* ps -ef
* For mitigation, controlling what the user inputs and allowing certain things to be input can help prevent injections. Stripping input is also useful as removing characters that might be deemed as dangerous can help secure the web application.
* Severity: High Impact Severity

## Insecure Design

* This happens when there is a lack of security control when it comes to the life cycle of the web application. This might lead to plenty of security holes.
* When there are no secure design patterns in the web application, the security of the design deteriorates.
* For example, a developer might build an insecure design if he/she disables the OTP validation in the development phase of the software cycle. If he/she doesn’t enable it again and sends it to production, this means that he/she built an insecure design.
* Mitigation: Perform threat modeling
* Severity: Severe. This vulnerability can lead to data breaches, unauthorized access, and system compromise

## Security Misconfiguration

* Poor Security Configurations are vulnerabilities which could make installation vulnerable.
* Examples include poorly configured permissions on cloud services and default accounts with unchanged passwords.
* One type is displaying a detailed error message to the user. This might open an easy way to the system.
* Mitigation: Updating the software on a regular basis and automate repetitive tasks
* Severity: It has a severe impact

## Vulnerable and Outdated Components

* When pen testers find vulnerabilities in programs or applications, they report them, and they get saved in the DB Exploit. Therefore, they became well-known.
* However, if the developers don’t update the vulnerable components or web applications or programs, attackers might exploit these vulnerabilities.
* Mitigation: Updating the web application to remove the vulnerability, perform auditing or pen testing before releasing the product
* Severity: Moderate

## Identification and Authentication Failures

* Authentication and session management are an integral part of any web application.
* Authentication is used to confirm the identity of people when they gain access to certain web pages. A very simple and known form is when a user inputs the username and password.
* The server is used to verify the identity of the user and it provides the user’s browser with the session cookie if they are correct.
* If the authentication system is flawed, the attacker might be able to gain access to other users’ accounts and their data.
* This can be done because of brute force attacks, or if the web application allows the user to register with weak credentials or using weak session cookies.
* Mitigation: Ensuring strong policy rules helps prevent password guessing attacks and logging the user out of a certain number of attempts prevents brute force attacks. Multi-factor authentication is crucial.
* Severity: This might allow brute force access and session hijacking.

## Software and Data Integrity Failures

* For software or data to have integrity, they must not have any modifications or alterations.
* To ensure integrity, a hash is added and is checked to see if the original file, for example want corrupted or modified during transmission.
* An example of this vulnerability would be missing support for integrity checks. Sometimes, software cannot check the integrity of a file, and this might be a threat as the software is unable to see if the file is the correct file. An attacker might exploit this fact and execute a malicious code.
* WannaCry is an example of ransomware and is an example of a program that was able to exploit the software and data integrity failure vulnerability.
* Mitigation: Updating, using up-to-date components, doing security checks regularly

## Security Logging and Monitoring Failures

* This vulnerability can arise when the system or program fails to login or monitor security events, and this can lead the attacker to exploit this and gain unauthorized access and enter the system or web application without leaving a trace or getting detected.
* Without logging, you can never tell what actions were taken when the attacker gained unauthorized access.
* If the attacker’s logs are undetected, he/she can attack the infrastructure or steal credentials.
* Logs contain status codes, usernames, IP addresses, and timestamps.
* Logs are important for detecting suspicious behavior.
* Logging failures can happen if the logging and monitoring system is not secure which might enable the attacker to erase or hide his/her tracks making it impossible to find who the attacker is.
* The incapability to store logs for a long time is also considered a failure.
* Mitigation: Logging and Monitoring should be secured, and the monitoring of the logs should be on a regular basis.
* Severity: The average incident rate is 6.51%.

## Server-Side Request Forgery (SSRF)

* This is a vulnerability where the attacker leads the server-side application to make requests to peculiar and unintended locations.
* Leakage of data and remote code execution might be possible because of this vulnerability.
* Sanitized API requests maybe an entry point for this vulnerability
* Mitigation: Implement proper input validation and sanitation and this includes URL
* Severity: This can lead to unauthorized access to sensitive data to the compromise of entire cloud environments.

## More About Broken Access Control

* There are four types of access control: Discretionary Access Control (DAC), Mandatory Access Control (MAC), Role-Based Access Control (RBAC), and Attribute-Based Access Control (ABAC)
* The Discretionary Access Control is a type of access control that is used in operating and file systems where the administrator determines which users are allowed access to which resources.
* Mandatory Access Control is governed by certain rules or policies and is used in highly secured systems.
* Role-Based Access Control is a type that is used in enterprise systems where a user is allowed access to resources according to the level, he/she is in.
* Attribute-Based Access Control is a type that is used in clouds environments and web applications and the access that the user gets is determined by attributes like location, role, and time.
* Implementing access control can help prevent security breaches but it is not bulletproof.
* There are four main types of Broken Access Control, and they include horizontal privilege escalation, vertical privilege escalation, insufficient access control checks, and insecure direct object references.
* Horizontal privilege escalation is when a user can access another user’s account simply by changing the ID in the URL
* Vertical privilege escalation is when a regular user gains access to administrative functions if he/she can manipulate hidden form fields or certain parameters.
* Insufficient access control checks are when the checks don’t happen very often or correctly.
* Insecure direct object references happen when an attacker can access data or resource by exploiting a vulnerability in the access control mechanism.
* Mitigation: Implement Role-Based Access Control helps mitigate this vulnerability as it specifies which user of which role has a right to which access. Another important step is to use parameterized queries toa void SQL injections. Using hashing can help mitigate this vulnerability, too. Managing sessions securely is important as well.
* Severity: Its severity is extremely high as it affects the CIA triad negatively.

## References

<https://nordlayer.com/learn/access-control/broken-access-control/>

<https://portswigger.net/kb/issues/00100850_broken-access-control#:~:text=Broken%20access%20controls%20often%20present,should%20not%20be%20able%20to>.

<https://www.softwaresecured.com/post/introduction-to-cryptographic-failures#:~:text=As%20per%20OWASP%2C%20cryptographic%20failure,be%20considered%20a%20cryptographic%20failure>.

<https://synopsys.com/glossary/what-is-owasp-top-10.html>

<https://tryhackme.com/r/room/owasptop102021>

<https://foresite.com/blog/owasp-top-10-insecure-design/>

<https://owasp.org/Top10/A04_2021-Insecure_Design/>

<https://hadrian.io/blog/understanding-insecure-design-a-deep-dive#:~:text=The%20consequences%20of%20insecure%20design,damage%20reputations%20and%20user%20trust>.

<https://foresite.com/blog/owasp-top-10-vulnerable-and-outdated-components/>

<https://info.veracode.com/rs/790-ZKW-291/images/vulnerable-and-outdated-components-prevention-guide-en.pdf>

<https://www.securityjourney.com/post/owasp-top-10-software-and-integrity-failures-explained>

<https://www.securityjourney.com/post/owasp-top-10-security-logging-and-monitoring-failures-explained#:~:text=Security%20logging%20and%20monitoring%20failures%20are%20security%20vulnerabilities%20that%20can,systems%20and%20data%20without%20detection>.

<https://owasp.org/Top10/A09_2021-Security_Logging_and_Monitoring_Failures/>

<https://portswigger.net/web-security/ssrf>

<https://www.freecodecamp.org/news/defending-against-ssrf-attacks/>

<https://www.evolvesecurity.com/blog-posts/how-to-prevent-server-side-request-forgery#:~:text=To%20mitigate%20the%20risk%20of,malicious%20input%20from%20being%20processed>.