**A1: INJECTION**

Injection attacks is a type of attack that involves obtaining of sensitive data (passwords, usernames, credit card numbers, etc.). Injection attacks has been 1st in the Top 10 OWASP web application vulnerabilities for many years and considered as the most dangerous among the vulnerabilities. Confidentiality, Authentication, Authorization and Integrity are the main consequences of injections. Generally, when attackers use injection, they target clients and insert malicious codes into the input fields to trick the database into sending back sensitive data and gives the attacker the ability to read and manipulate the sensitive data in the database. Injection also gives the attacker administrative control over the database (such as inserting, deleting and shutting down the database). There are multiple types of injections such as Code Injection, Carriage Return and Line Feed Injection (CRLF), Cross-site Scripting (XSS), Email (Mail Command/STMP) injection, Host header injection, Lightweight Directory Access Protocol (LDAP) injection, OS Command Injection, XPath injection, SQL Injection (SQLi) and many more. SQL injection is the most common talked about type of injection which uses SQL commands in an input type to change the execution of programmed SQL commands in the web application.

**MEASURES AND MITIGATION**

There are multiple measures in mitigating injection attacks such as:

* **Query Parameterization**
* An SQL query type that involves the separation of the SQL statement and the parameter. Since the SQL will only execute when there is an existing parameter, a place holder is used as a substitute for the parameter.
* **Proper Input Validation**
* A technique for adding limitations in text fields such as not allowing special characters ($, @, #, %, ^, &, \*, etc.) and numbers.
* **Building the Web Application Properly**
* Since all input from users are considered to be untrusted data, adding proper access controls, having firewalls and defense mechanisms is highly advisable.

**A2: BROKEN AUTHENTICATION**

Broken authentication is a vulnerability that involves the attacker being able to access the web application through the use of a user’s account. Attackers use different methods in getting pass through authentication. One of them is through the exploitation of the session ID which involves…. Credential stuffing is a method where the attacker makes use of username and passwords that were obtained from an old exploitation and continuously trying them in the authentication process until the attacker gains access. Automated Attacks is similar to credential stuffing but there is no pre-gathered data used and the attacker keeps using random usernames and passwords. Lastly, default password is a method where the attacker makes use of the default username and passwords provided when the web application is set up.

**MEASURES AND MITIGATIONS**

There are multiple measures in mitigating injection attacks such as:

* **Multi-factor Authentication**
* Besides username and password authentication there are added measures such as online tokens, thumb scanner, barcode scanner, etc.
* **Password Checking**
* Checking each password in the database for possibility of being one of the commonly used passwords and notifying the user to change the password.
* **Password Complexity**
* Require a password length, special characters and numbers when the user is creating his/her password. Add a meter that will show the user the password strength.
* **Limit Failed Logins**
* Add a security measure that when a user fails to login a certain number of times the account will lock and can only be unlocked by contacting the administration.
* **Server Side Session Management**
* In the back end, set up server side session manager that will throw an old session ID and replace another session ID when the another user logs in. This will allow each user to have a unique session ID.

**A3: SENSITIVE DATA EXPOSURE**

When sensitive data such as credit card details, customer details, usernames, passwords, etc. are stored into a database, the data can be obtained and read directly by attackers. Attackers are able to obtain data even if the database is not accessible through a web application. All data transferred from and to the web application can also lead to the exposure of sensitive data. Once sensitive data is exposed, the data will be available to everyone and can lead to different kinds of crimes such as theft, false identity, etc.

**MEASURES AND MITIGATION**

* **Data Encryption and Limit Accessibility**
* Encryption is a coding process where the data is only readable to users with the proper authorization. Adding a key-based encryption can harden and add extra security to the sensitive data.
* **Secure Authentication Gateways**
* Using technologies like SSL or TSL to prevent third person involvement and ensure a private transmission of data between the server and browser. Adding a secure communication protocol is also recommended.
* **Prevent Password Attacks**
* Avoid making use of weak passwords and ad **Access controls are security measures added to a web application.** d password hashing. Constantly changing the password and using unique passwords for each platform is also some ways to prevent password attacks.
* **Conduct Regular Risk Assessment**
* Frequent monitoring and updating of the security system of the web application and always having a backup for the sensitive data.

**A4: XML External Entities (XXE)**

XML External Entity is a server attack method that steals data or just deliberately crash the website with denial of service. The hacker targets servers that is discovered to have an XML parser that has been weakly configured. This attack is simple as the attacker inputs an XML that is linked to an external entity that is programmed to steal or destroy a targeted server.

**MEASURES AND MITIGATION**

* Add codes that will prevent external entities
* A Cheat Sheet with the codes is available in the OWASP website.

**A5: BROKEN ACCESS CONTROLS**

When these measures/restrictions are not properly configured and added to the web application this can lead to attackers being able to penetrate through the web application. A common mistake is when an admin side of a web application utilizes javascript on the functions particularly creating an admin account. Most programmers remove the javascript code that requests for the API and this can lead to an attacker being able to access the admin panel without proper authentication.

**MEASURES AND MITIGATION**

* **Add Proper Access Control Documentation**
* Documenting each operation, request and function can help in avoiding broken access controls.
* **Multifactor Authentication**
* Adding different types of authentication methods such as biometrics, cards, scanners.
* **Limiting to Strong Passwords Only**
* **Password Encryption**
* All passwords data must be encrypted. Using HTTPS can help an organization strengthen security.

**A6: SECURITY MISCONFIGURATION**

Security Misconfiguration talks about the errors and mistakes done by the administrator when creating the web application which leads to the web application being vulnerable and exposed to attackers. Keep in mind that with every code, server, software and features used there are possible risks and vulnerabilities. There are many factors that contribute to security misconfiguration such as enabling unnecessary features can add potential risks and exposure to attackers. Default accounts is another factor which leaves the accounts easily accessible to attackers. Giving too much information to the user can allow the attacker to learn and easily find vulnerabilities of the web application. Lastly, a server responding to the client without security headers.

**MEASURES AND MITIGATION**

* **Automated Scanning Capability**
* A tool that checks if there are default accounts and updates. Notifies the administrator of the errors and updates available.
* **Adding Repeatable Hardening Processes**
* Adding a process that can “harden” the web application. Hardening a web application means that the web application’s vulnerability to attackers is reduced.
* **All Servers, Same Configuration**
* All servers used in the web application must have the same configuration which can make the adding of new codes and features easier.
* **Security Directives/Security Headers**
* Adding security headers can add another security layer to the web application. Some examples are HSTS (HTTP Strict Transport Security), HPKP (HTTP Public Key Pinning) and X-Frame Options.

**A7: CROSS - SITE SCRIPTING (XSS)**

Attackers usually target the client side script of a web application by inserting a malicious script and tricking the user into executing the script. This attack can lead to access controls being bypassed and attackers using the victim’s account. Victims are not only limited to the users but the administrator’s account is also vulnerable to this attack.

**MEASURES AND MITIGATION**

* **Escaping**
* “Escaping” is a method of rendering data received from a sender and checking for malicious code before sending it to receiver.
* **Input Validation**
* Limiting the user’s input and not allowing certain characters to be inputted in any fields.
* **Data Sanitizing**
* Removing unwanted and malicious user input and markups and keeping your database “clean” or “sanitized”.

**A8: INSECURE DESERIALIZATION**

A vulnerability where attackers target Java and PHP applications. This attack occurs when untrusted data is used to change the logic of a code or application. Deserialization is the changing serialized data into an object. The attack usually occurs when deserializing untrusted input.

**MEASURES AND MITIGATION**

* A Cheat Sheet with the codes is available in the OWASP website each platform has specific measures.

**A9: USING COMPONENTS WITH KNOWN VULNERABILITIES**

Attackers usually tries to find the vulnerabilities of each software. Whether the version of the software is old or new the hackers will always find vulnerabilities and exploit it. For example, if a web application is using an older version and fails to update it for years, then the attacker would have an easier time breaking through the security measures and exploit each vulnerability.

**MEASURES AND MITIGATION**

* **Update All Software Used**
* Check if all the versions of the software used in the web application are up to date.

**A10: INSUFFICIENT LOGGING AND MONITORING**

Monitoring hacking attempts and detecting breaches takes too much time to do. This is why most organizations are no longer doing this as often as possible. Without the proper logs of both the admin and user activities in the web application, there is no possible way an admin can detect who is doing the attack.

**MEASURES AND MITIGATION**

* **Application Logging**
* Logging events such as:
* Identifying security incidents
* Monitoring policy violations
* Establishing baselines
* Assisting non-repudiation controls
* Providing information about problems and unusual conditions
* Contributing additional application-specific data for incident investigation which is lacking in other log sources
* Helping defend against vulnerability identification and exploitation through attack detection
* Security events
* Business process monitoring e.g. sales process abandonment, transactions, connections
* Anti-automation monitoring
* Audit trails
* Performance monitoring
* Compliance monitoring
* Data for subsequent requests for information, freedom of information, litigation, police and other regulatory investigations
* Legally sanctioned interception of data
* Other business-specific requirements