10 Oswap Tools:

- A1 Broken Access Control:

It occurs when a web application can act in ways that are not intended. A malevolent user may communicate as the original user if they have the ability to assume the identity of another user. This wicked person is capable of deceiving other users with ulterior motives. This regulation applies such as that users cannot conduct contrary to the permissions they have been granted. Failures frequently result in the illegal disclosure of information, the change or deletion of all data, or the performance of business functions that are outside the user's scope. enhancement of privilege A CORS configuration error allows unauthorized or untrusted origins to access APIs.

It can be prevented by Only in trustworthy server-side programs or server-less APIs, where an attacker cannot alter the access control check or metadata, is access control effective. Deny by default, excluding resources owned by the state. Reduce the utilization of Cross-Origin Resource Sharing (CORS) by implementing access control measures once and reusing them across the application. Instead of allowing the user to create, read, update, or delete any record, model access restrictions should guarantee record ownership. Domain models should impose specific business restriction constraints for applications. Make sure backup files and file metadata (like.git) are not present in web roots and disable web server directory listing. Record access control errors and notify administrators as necessary (e.g., repeated failures). To lessen the damage from automated attack tooling, rate limit API and controller access.

- A2 Cryptographic failures:

Sensitive information is exposed since it is not encrypted.

The data is accessible to users. The focus is on failures linked to cryptography at position 2, formerly known as Sensitive Data Exposure, which is more of a general symptom than a core cause (or lack thereof). can frequently cause sensitive data to be exposed. In the absence of a password base key derivation function, the question of whether passwords are being used as cryptographic keys arises. First, avoid using old protocols like SMTP and FTP, avoid caching responses that contain sensitive data, and lastly, don't keep unneeded sensitive data.

In plain English, a cryptography failure is a security blunder that happens when a third party (apps, web pages, or various websites) leaks important information. Specifically, it occurs when the thing acts in a certain way without a clear intention. A cryptography failure, whether because to carelessness, stupidity, or poor judgment, can have disastrous effects on both the individual and the company. Sometimes the database protection is insufficient. When they start up new datastore instances, it can also be because of configuration errors. Sometimes improper data system utilization results in the exposing of sensitive data, bugs in software, shoddy encryption, Absolutely no encryption, an unintentional upload to the wrong database. There are numerous ways for businesses to expose themselves and reveal private information, and this does happen occasionally.

- A3 Injection

When user-supplied data is not verified, filtered, or sanitized by the program, the application is open to attack. When malevolent users learn the online application is susceptible to injection, they may try to insert malicious code into text fields. In order to retrieve more, sensitive records, object-relational mapping (ORM) search criteria are utilized with hostile data. Isn't user input the source of all evil? Because of this, if user input is implicitly trusted and not subjected to any sanitization, filtering, or escape, a malicious user may be able to manipulate the application's output and possibly even take over the servers. This occurs when the user's data is combined with the interpreter (such as a database engine, shell, template engine, etc.), which can result in assaults like:

SQL/NoSQL Injection

Command Injection

Server Side Template Injection

Header Injection

Content Injection (XSS, HTML Injection, CSS Injection) and more

Here are some steps you could take to reduce your sensitivity to this risk: Never rely on user feedback It is the source of all bad. Keep a safelist of the permitted characters that are anticipated in the input and discard any incorrect or unexpected input. Prior to processing, clean up, filter, and/or escape user input as appropriate. Before sending the output back, encode it (depending on the context). Set restrictions on the output that is sent back. If SQLi/NoSQLi were present in your applications, this would prevent them from returning millions of database records. A connection timeout would also stop any persistent shell sessions.

- A4 Insecure Design:

A general term used to describe a variety of control structure flaws in a web project is "insecure design." These consist of missing or dysfunctional controls. A refining session should incorporate a threat modeling tool to check for changes in the data flows and other security controls in order to find these. The hazards connected with design and architectural faults are the main topic of insecure design. It focuses on the necessity of secure design patterns, concepts, and threat modeling. The shortcomings of an unsafe design cannot be fixed by implementation. Insecure design is distinguished by OWASP from security implementation and controls in the following ways:

-A perfect implementation cannot remedy an unsafe design since, by definition, the necessary security safeguards were never developed to protect against certain attacks.

-Attackers can use workflows in the program to threat model insecure design to expose a variety of flaws and vulnerabilities.

Insecure Design is a comprehensive vulnerability that features more than 40 CWE and covers vulnerabilities that resulted from known or unknowable defects at the application/software architecture level. Developers are compelled to move past the shift-left approach used in the coding process and adopt Secure by Design-approved pre-code activities. A wide range of flaws are explained by the ineffective and absent control design susceptibility of insecure design. It's important to realize that insecure designing differs greatly from insecure implementation as one attempts to understand its meaning. Even with a safe design, there may be instances of insecure implementation. Similar to this, a flawed design might result in weaknesses in a secure implementation.

- A5 Security Misconfiguration:

One of the most prevalent problems with online applications. It applies to infrastructure setup settings and other situations as well. It demonstrates an incomplete configuration, a disregard for the firewall rules, or the openness of a vulnerable port that you aren't using. Another illustration is default accounts, which have activated and unaltered passwords. Making ensuring the environment is uniform throughout is one way to prevent. Make sure the testing and production environments are identical if you have them. As a result, you're reducing the component's quantity. Automated systems should be avoided and recognized. the design and certification of environments. Security The likelihood of setup errors is quite high, and this risk can be easily taken advantage of. Nevertheless, the impact of the risk would be minimal. The severity of the security misconfiguration totally determines the attacker's ability to reach an unprotected areas (Insecure weblogs,Internal error,Insecure admin page).

It's best to configure the various services with security in mind in order to eliminate these problems, and you should also make sure that the components being used—whether they be software or OS packages or any services you rely on—are kept up to date. This is somewhat general guidance, and some practical measures might be:

1. Turn off features that are not necessary in production scenarios.
2. Observe the "Principle of Least Privilege." Give the entities only the rights they need in order to function properly.
3. Set up the security headers correctly; they shouldn't be too permissive or they'll lose their effectiveness.
4. Ensure that the used services and packages are updated.
5. Security must be considered when configuring services.
6. Follow the patching and repeated hardening procedure.
7. To maintain excellent security of your apps and services, audit your infrastructure and get periodical pen tests. It's a general tip that will assist in identifying and minimizing the majority.

- A6 Vulnerable and Outdated Components:

Open-source or proprietary code that is old or has security flaws is referred to as having vulnerable and vulnerable components. For online applications, this code can take the shape of libraries or frameworks, such as Laravel (PHP), Angular (JavaScript), Django (Python), and many more. Unfortunately, this code is frequently developed with little to no thought given to security, which could have serious repercussions for application users and jeopardize the reputation of businesses. While it is occasionally possible to compromise critical systems using zero-day vulnerabilities found in third-party components, the majority of breaches are caused by flaws that IT experts are already aware of. Unfortunately, resolving the problem is not as straightforward as executing an update command or downloading new packages. It can often be fairly complicated. This can be a major problem if you don't have the most recent versions of the components you utilize, even those you directly use. Your web or application server's functionality may be hampered by outdated or vulnerable software. Additionally, it may have an impact on your system's databases, apps, and APIs. To speed up and simplify development, almost all applications rely on open-source libraries rather than developing everything from scratch. Since software engineers are frequently under intense time constraints, these components are frequently not sufficiently tested before implementation. Applications utilizing a vulnerable component can be recognized and exploited once it has been found by criminals. Although the vulnerability may appear to be a minor flaw in the code, in some circumstances it can result in a complete system compromise. Such a breach might seriously harm client data, result in lost sales, and seriously harm an organization's brand.

It is frequently simple to identify recognized dangers in obsolete and vulnerable components, and both MITRE and NIST have extensive databases of CVEs, or Common Vulnerabilities and Exposures. Many of these CVEs can be easily exploited, and Offensive Security has a sizable collection of attack code that is accessible to the public. Applications should be regarded as at risk until they have been fixed if they have vulnerabilities that are listed on a public database. These third-party components can be extremely time- and resource-intensive to manage. Although you might be tempted to believe that installing updates or upgrading to newer software versions are not major issues, this is not the case. The deprecation of features, the renaming of functions, the possibility of some applications breaking following changes, and a number of other potential difficulties all exist. In the best-case scenario, the improvements will function flawlessly or just necessitate small code modifications, but these changes could equally well end up consuming a sizable amount of time and money, resulting in severe issues for everyone.

- A7 Identification and Authentication failures

The authentication of the user's identity is crucial for preventing identity and session management attacks. An attacker may be able to access a user's account through a weak point in the authentication procedure. It is possible to automate brute force assaults or other tasks using this vulnerability. A user's account may potentially be accessible to an attacker without the need for a special password. A user's account could be compromised by an attacker utilizing shoddy or inadequate password recovery procedures by taking advantage of this vulnerability. Passwords can be entered in plain text or hashed form. A user's account could be compromised by an attacker utilizing shoddy or inadequate password recovery procedures by taking advantage of this vulnerability. Additionally, the URL may reveal a session's specific ID. Multi-factor authentication should be used to stop automated credential reuse and other attacks. Make an unified set of standards in order to harmonize the diverse password management policies and procedures with the NIST 800-63b recommendations. Make sure that the registration, API, and credential recovery processes are fortified against account-based attacks to avoid them. Even if it's crucial to restrict or postpone unsuccessful login attempts, a denial-of-service situation should be avoided. Log all errors to avoid brute force, credential stuffing, and other attacks. After a user logs in, a secure built-in session manager can produce a fresh, highly entropy random session ID. To safeguard against threats linked to authentication, it is essential to confirm the user's identity, authenticate them, and manage their sessions. There might be issues with authentication if the program:

Allows for automated attacks like credential stuffing, in which the attacker possesses a list of legitimate users and passwords allows automated or brute force attacks permits commonly used passwords like "Password1" or "admin/admin" that are either weak or default. It uses flimsy or ineffective techniques for forgotten passwords and recovering credentials, such as "knowledge-based answers" that cannot be made secure, uses passwords that are plain text, encrypted, or have weak hashing (see A02:2021-Cryptographic Failures), lacks or is not working with multi-factor authentication, the URL exposes the session identification, after a successful login, reuse the session identification.

- A8 Software and Data integrity failures

The inability of the infrastructure and code to prevent unauthorized access and manipulation is the root cause of data integrity problems and software problems. For instance, a program that depends on plugins and libraries from unreliable sources may be open to attack. Many programs now have auto-update functionality, which allows updates to be downloaded and applied without enough verification to previously trusted applications. As a result, attackers might be able to disseminate and activate their own updates across all installations. Another problem is the insecure way in which data and objects are encoded and stored. Use digital signatures or other measures to confirm the authenticity of the information to make sure the program and data are from the anticipated source. Data or objects that are encoded and stored in a manner that is susceptible to deserialization are another problem. Use digital signatures or other measures to confirm the authenticity of the information to make sure the program and data are from the anticipated source. Make that the libraries and dependencies the program uses are from reliable sources. For known vulnerabilities, a good software supply chain security tool can also be used. The OWASP CycloneDX or Dependency Check can be used for this. Reviewing the application's settings and code alterations to make sure they don't create any detrimental changes is another crucial step.

- A9 Security Logging and Monitoring

Failures in security logging and monitoring are frequently a contributing element in serious security events. The BIG-IP system offers security capabilities to guard against attacks that may be caused by insufficient system and application logging and monitoring, as well as enhanced logging and monitoring functionality. The difficulty in identifying suspicious behavior and the risk that an attacker will be successful in exploiting your application are both considerably increased by inadequate logging, monitoring, or reporting of security events, such as login attempts. An attacker might, for instance, continuously check your program or application for known flaws. If such probes are let to go unnoticed, there is a greater chance that the attacker will eventually identify a weakness and successfully exploit it. Developers that want to assist in locating and addressing live breaches in their applications should select this category. Inadequate monitoring and logging make it impossible to find breaches. This is why it's crucial that they put in place a variety of safeguards to assist thwart attacks of this nature. Make sure that the various security measures are correctly applied in order to prevent unauthorized access and manipulation of the data recorded in the logs. Make sure the logs are formatted such that log management systems may easily access and consume them. Your application is vulnerable to attacks that target any layer of the application stack if there is insufficient logging, monitoring, or reporting. For instance, failing to record, monitor, or report security events may lead to the following attack types:

1. Code injection
2. Buffer overflow
3. Command injection
4. Cross-site scripting (XSS)
5. Forceful browsing

- A10 Server Side Request Forgery

It is a specific type of cyberattack in which a knowledgeable hacker utilizes a server's pre-existing vulnerability against the server. A hacker may utilize SSRF to introduce a corrupted code or link into mission-critical server data that is being saved or traversed, depending on their objectives. The SSRF attack can generate problems that go above and beyond what was initially anticipated if it is neglected for a long time. The following are the main effects:

1. A hacked server can be used by attackers as an IP scanner or port to gather data from front- and back-end systems.
2. On systems that are being targeted, internal protocols like Gopher can be enabled, which gives hackers easy access to additional opportunities for reconnaissance.
3. Threat actors are given access to the reverse proxy's disguised IP addresses through a successful SSRF cyberattack.
4. By making XCE a possibility, SSRF establishes a foundation for several cyber issues because malicious software, viruses, ransomware, and other types of malicious components can be added.

example of SSRF: Attackers have a chance because to unsegmented network design, which they can utilize to determine whether or not internal server ports are open. Ports can readily carry an SSRF connection if they are open.

You can identify this type of vulnerability by:

1. Take a look at the request's incomplete URLs

   An application or program frequently keeps a portion of the URL or host information when forwarding a request. When a server receives one of these requests, a full URL is added. In order to find the simple signs of an SSRF cyber-attack, you must carefully check for the presence of GET-type form inputs or URLs, for which the input can expose the obvious.
2. Be mindful of the data formats

   Applications that transfer data in rich formats typically allow the URL to be present in order to parse the data in the most efficient way. For instance, structured data is frequently sent and received using XML. Thus, it creates a pathway for SSRF and XXE injections.
3. Referer header is present

   It goes without saying that the program will record the Referer header in requests if you use server-side analytics to monitor server visitors. For the efficient tracking of received links, this is crucial.

Additionally, it's possible that this server's analytics software is accessing a URL from a third party because it showed up in the Referer header. This form of Referer Header demonstrates a favorable environment for intrusions similar to SSRF.