SAFE SECURITY

Top Web Application Security

Threats & Standards

APPROACH NOTE

SHLOK GUPTA

SAFE SECURITY

Table of Content

1.0 OWASP TOP 10

1.1 Introduction to OWASP:

The Open Web Application Security Project or OSWAP is an international non-profit organization devoted to web application protection. Two of the key concepts of OWASP is that all the content is open to download on their pages so that anyone should enhance the protection of their Web application. The papers, tools, videos, and forums they include. Perhaps the OWASP TOP 10 is their most famous project.

1.2 OWASP TOP 10:

The OWASP Top 10 is a frequently published guide that identifies safety issues regarding web application protection and discusses the ten most important dangers. The OWASP Top 10 is a frequently published guide that identifies safety issues regarding web application protection and discusses the ten most important dangers. To minimize/or eliminate security risks, OWASP refers to the Top Ten as an "AWARENESS GUIDE" which proposes that every organization integrate the report into its processes.

1.3 What are OWASP TOP 10 2013:

1. Injection
2. Broken Authentication
3. Cross-Site Scripting (XSS)
4. Insecure Direct Object References
5. Security Misconfiguration
6. Sensitive Data Exposure
7. Missing Function Level Access Control
8. Cross-Site Request Forgery (CSRF)
9. Using Components with Known Vulnerabilities
10. Unvalidated Redirects and Forward

1.4 What are OWASP TOP 10 2017:

1. Injection
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities (XXE)
5. Broken Access Control
6. Security Misconfiguration
7. Cross-Site Scripting (XSS)
8. Insecure Deserialization
9. Using Components with Known Vulnerabilities
10. Insufficient Logging and Monitoring

1.5 What changed from 2013 to 2017:

The Open Web Application Security Project (OWASP) publishes a list of 10 critical security risks for web applications every few years. This article outlines key issues impacting the digital web and the actions you should take to protect your web applications. OWASP produces these lists with input from web creation and security groups, as well as data obtained from over 100,000 live web applications.

1.6 We will be discussing OWASP top 10 2017:

A1:2017 Injections:

Exploitability: Easy

Prevalence: Common

Detectability: Easy

Technical Impact: Severe

Injections are among the oldest and most damaging threats targeted at web applications. They will lead to data-stealing, data lack, lack of credibility, denial of service, and complete network compromise. The primary explanation for flaws in injections is typically a lack of validity of user data. Injection attacks apply to a broad class of attack vectors. In an injection attack, the attacker provides untrusted program data. This input is interpreted by an interpreter as part of an order or query, that, in effect, modifies the execution of the program.

Injection bugs require attackers to pass malicious code to another device via an application. Such threats include machine-based calls to the operating system, the use of external programs using shell commands, as well as calls to backend databases using SQL (i.e. SQL injection). Whole scripts are written in Perl, Python, and other languages can be embedded and implemented in badly built programs. Whenever a program uses an interpreter of some kind, there is a possibility that input may become vulnerable.

Attack Vector:

The injection vector can be virtually any database, including applications, criteria, external and internal web resources, and all types of users. Injection vulnerabilities exist where an intruder can transmit offensive information to an interpreter.

Security Weaknesses:

Faults in the injection are easy to find while analyzing code. Scanners and flushes can allow attackers to identify injection flaws. Injection bugs, particularly in the legacy code, are very prevalent. Also used in SQL, XPath, or NoSQL databases, Operating Systems, SMTP headers, languages for voice, or ORM databases.

Technical Impacts:

The unwanted access to confidential data (for example passwords, credit card details, or personal user information) could lead to a successful Injection attack. Many extremely serious data breaches in recent years have led to reputational harm and legal penalties as the result of SQL injection attacks. In other situations, an attacker can have a permanent way into the processes of an entity, resulting in an irreversible breach, which will go unseen for a longer period.

Types of Injection Attacks:

1. Code Injections:

The unwanted access to confidential data (for example passwords, credit card details, or personal user information) could lead to a successful Injection attack. Many extremely serious data breaches in recent years have led to reputational harm and legal penalties as the result of SQL injection attacks. In other situations, an attacker can have a permanent way into the processes of an entity, resulting in an irreversible breach, which will go unseen for a longer period.

2. CRLF Injection:

CRLF is the abbreviation for CARRIAGE RETURN and LINE FEED. A CRLF injection attack is one of many forms of injection attacks. It is often to intensify to additional malicious attacks reminiscent of Cross-site Scripting (XSS), page injection, internet cache poisoning, cache-based hurt, and more. A vulnerability to CRLF injection exists when an attacker can inject the CRLF characters into a web application, e.g. using a user input form or an HTTP request.

3. SMTP Header Injection:

If the user is put within the email header, without adequate sanitization, an attacker will insert additional headers of unknown value. SMTP header injection vulnerabilities occur. This behavior can be used to send third-party copies of the emails, attach viruses, make phishing attacks and often alter email content. Spammers are usually used to exploit the credibility of the insecure organization and grant their emails legitimacy.

4. HTTP/Host Header Injection:

Using the HTTP host header available on the client-side HTTP request, developers often use the URI to build a link in Web applications. A remote attacker may exploit this by sending a fake header under his control with a domain name which, for instance, enables him to poison web cache or reset emails with passwords.

5. LDAP Injections:

LDAP is the shortened form of Lightweight Directory Access Protocol. LDAP injection is a vulnerability where queries are built without prior validation or sanitization from untrusted input. LDAP employs questions constructed from different characters, for instance, brackets, asterisks, ampersands, or quotes, which require the use of syntax.

6. OS Command Injection:

Shell Injection or OS command injection is a web security vulnerability that enables an attacker to execute the commands arbitrarily to the OS server running an app, typically compromising the software and all of its data. Very often an attacker can exploit a vulnerability to OS command injecting to jeopardize other parts of the hosting infrastructure and exploit relationships of confidence to pivot the attack on other systems.

7. SQL Injection:

SQL injection (SQLi) is a security flaw that enables attackers to manipulate the storage of an application by allowing them to view or erase data, alter the data-driven actions of an application, and do other inappropriate stuff–tricking the application into sending inattentive SQL commands. SQL injection vulnerabilities arise where a program uses insecure information such as data inserted in fields of the web-based type.

8. Xpath Injections:

Web pages that use user-support knowledge to create an XPath database for XML data are used for XPath Injections. By intentionally submitting malformed information to the database, an attacker would be able to identify the structuring of the XML data or access data that he usually cannot access.

Preventions:

Avoiding access to other interpreters as far as possible is the easiest way to defend against injection. There are language libraries that perform the same functions for many shell commands and some system calls. The operating system shell interpreter is not involved in the use of such libraries, which prevents numerous shell-command problems. Another strong injection attack protection is to ensure that only the privileges required to perform the web application are available.

A2:2017 Broken Authentication:

Exploitability: Easy

Prevalence: Common

Detectability: Average

Technical Impact: Severe

Broken Authentication is typically due to poor session management and authentication functions. Broken authentication attacks aim to gain the same privileges as a user in one or several accounts to the attacker. Authentication shall be "broken" if attackers are capable of compromising user identity passwords, keys, or tokens. User account and other details.

The Common Risk factor includes:

- User credentials are predictable
- Vulnerability of session IDs to session fixation attacks
- No time out and proper logout function in sessions
- Expose of session IDs in URLs
- Insufficient security to stored user authentication credentials
- User credentials sent over unencrypted connections/paths
- Same session IDs with each Logins

Attack Vector:

Attackers can browse hundreds of millions of correct login stuffing usernames and password combinations, default corporate account lists, automatic brute force and dictionary attack methods, and sophisticated GPU cracking software.

Security Weakness:

- User credentials are predictable
- Vulnerability of session IDs to session fixation attacks
- No time out and proper logout function in sessions
- Expose of session IDs in URLs
- Insufficient security to stored user authentication credentials
- User credentials sent over unencrypted connections/paths
- Same session IDs with each Logins

Technical Impacts:

Attackers only have access to some accounts or just an administrative account to adversely affect the system. It can facilitate cash laundering of fraud and identity stealing depending on the scope of the app, or expose highly confidential, legally protected details.

Prevention:

- User Credentials should be stored using encryption or hashing
- The logout should correctly invalidate user sessions or tokens.
- Even after successful login, session IDs should be recreated.
- Password, user credentials, or session Ids shouldn't be visible in URLs and always sent through encrypted connections.
- Disabling account after a set number of invalid login attempts
- Authentication failure responses do not reveal the correct location of the authentication data.

A3:2017 Sensitive Data Exposure:

Exploitability: Average

Prevalence: Widespread

Detectability: Average

Technical Impact: Severe

Publicly available Leakage happens where a program does not encrypt confidential info adequately. The data can vary from passwords, sessions, credit card information to personal health data and more. More commonly, vulnerable applications simply do not encrypt sensitive information and store it in a database where SQL injection or any other type of attack can compromise.

Sensitive data become revealed when confidential information is accidentally released by an individual, business, or other organization. Data breach differs from sensitive exposure, in which an attacker accesses and steals information.

As the result of insufficient protection of a database in which data are stored, sensitive data exposure occurs. This could be the product of a host of issues such as weak encryption, no encryption, bugs in applications, or whether anyone mistakenly transfers data to the wrong computer.

Attack Vector:

Sensitive Data Leakage vulnerabilities arise when a web application may not properly safeguard confidential information from being exposed to attackers. This can involve details such as credit card records, medical background, session tokens, or other authentication certificates. With the introduction of powerful GPUs and ASICs, attackers are now able to attack weak cryptographic algorithms like MD5, many of which saw a password-hating algorithm as suitable.

Security Weakness::

The most prominent flaw is that confidential data are not encrypted. When encryption is used, weak key generation and management and weak algorithms are common, especially for the weak hacking of passwords. For server-side faults in-transit data, it is mainly easy to detect, but it is difficult for data to rest.

Technical Impacts:

The result of insufficient protection of a database in which information is stored is sensitive data exposure. This could be the consequence of several things, such as poor coding, no coding, software faults, or mistaken data uploading to an incorrect database.

A sensitive data exposure may expose various types of data. The following kinds of information can be disclosed, e.g. banking account numbers, credit card numbers, health information, session tokens, social security number, home address, telephone numbers, the date of birth, and user account information such as usernames and passwords.

Prevention:

Sensitive Data Exposure can be prevented, if the following measures are inherited:

- Installation of SSL certificates
- Use the latest algorithms for encoding.
- Deactivate the cache of data collection forms.
- Encrypt data transportation.
- Prevent critical data storage by browsers.
- Implementation of Web Application Firewall (WAF) & Vulnerability/Malware Scanners.

A4:2017 XML External Entities (XXE):

Exploitability: Average

Prevalence: Common

Detectability: Easy

Technical Impact: Severe

Extensible Markup Language (XML) originally was created that can be used with digital design but is now a popular way to exchange data with different types of applications and is typically more than HTML used for data interchange, in many situations. This makes XML incredibly popular in many types of web applications, services, and documents. This permits the communication and exchange of data between two systems running different technologies. To interpret XML data, applications require some form of XML or XML parser that understands the format of the XML processor to either transfer the data to another format or simply output the result.

Attack Vector:

XML External Entity injection, also known as XXE, is a web safety vulnerability that allows an attacker to interfere with the processing of XML data on an application. It often allows an intruder to view files on an application server filesystem and interact with any server-side or external system accessible to the application.

Sometimes an attacker can escalate an XXE attack by using the XXE vulnerability to perform server-side request forgery (SSRF) attacks to compromise the underlying server or other backend facilities.

Security Weakness::

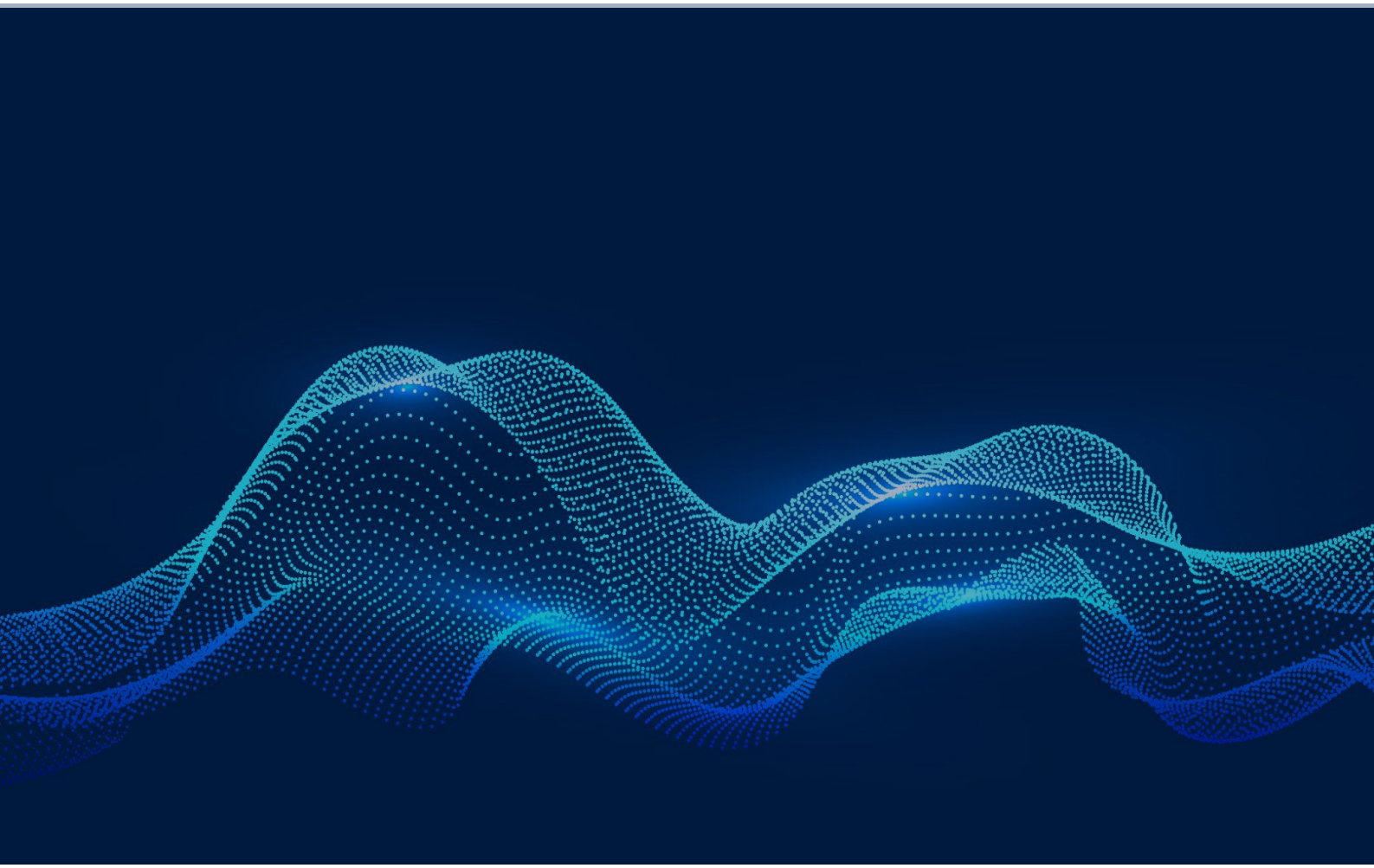
The XXE error allows an attacker to turn the XML parser into a proxy that can be used on request for local or remote contents. The key issue with all these attacks is that there is no proper input validation of the data, which allows the intruder to execute malicious commands on the insecure computer. Moreover, the default norm XML "external entities" allows the possibility of such an attack occurring in many production contexts.

Technical Impacts:

In injection system attack categories similar to command injection and SQL injection, there are XXE vulnerabilities. For XXE the attack focuses on the XML language that gives an attacker the chance to use the back-end framework to run the application which parses or interprets XML documents. An attacker can also access local files or remotely access objects that an attacker chooses to use dynamically for example external files or scripts. The security vulnerability of external entities may very much be similar to Local File Inclusion (LFI) and Remote File Inclusion (RFI) exploits.

Prevention:

- Disable additional XML and DTD loading in all the application's XML parsers.
- To prevent hostile data within XML documents, headers or nodes, implement a whitelisting of the servers' input validation, filtering, or sanitization.
- Use fewer complex formats like JSON, and prevent sensitive data serialization.
- Use the application or operating system to upgrade or patch all XML processors and libraries in use.
- Make sure that one verifies Extensible Markup Language (XML) or Extensible Stylesheet Language (XSL) File Upload Functions with XML Schema Definition (XSD) Validation or the like.



A5:2017 Broken Access Control:

Exploitability: Average

Prevalence: Common

Detectability: Average

Technical Impact: Severe

The web server penetration tests discover failed permission checks as one of the most popular bugs. Vulnerabilities to access control occur if users can act outside their intended permissions. This typically leads to unwanted entry, leakage of knowledge, and computer alteration or loss. Such flaws emerge from unclear coding and insecure application of systems for authentication and authorization.

Attack Vector:

Authorization is the implementation of restrictions on who can carry out or use services that are needed. Broken Access control is based on authorization and session management in the sense of web applications:

- The management of the session identifies which subsequent HTTP requests the same user makes.
- Regulation of access decides how the consumer will execute the operation he is attempting to execute.
- Authentication acknowledges the individual and reveals that they are.

Security Weakness::

Controls of broken access are a common security vulnerability, often critical. The creation and maintenance of access controls is a complicated and important problem that relates to the software application market, organization, and regulatory restrictions. Technical, and not machines, choices on access control architecture must be taken so there is a significant probability of mistake.

Technical Impacts:

The technical impact would be the use of anonymous attackers, users who use privileged functions to build, view, edit to erase any record. The other effects of Broken Access Control depend heavily on what type of information or functionality the attacker can access. This can range from seemingly useless information to complete system acquisition.

Prevention:

To prevent Broken Access Control, these things should be kept in mind:

- When a tool is not meant to be publicly available, refuse access by default.
- Strictly auditing and testing access controls to ensure that they work as intended.
- Using Lists of access controls and authentication methods dependent on function.
- At the application level, developers must announce the permissible access to might resource and deny access by default.
- Use a common application-wide access compliance system whenever possible.

A6:2017 Security Misconfiguration:

Exploitability: Easy

Prevalence: Widespread

Detectability: Easy

Technical Impact: Moderate

Security Misconfiguration is clearly described as not implementing all security checks for a server or web application or security checks with errors. What a company considered a secure environment actually carries dangerous errors which put the company at risk. This form of fault is the number 6 in the list of essential safety threats for the Web application, according to the top 10 of OWASP.

Attack Vector:

Security Misconfiguration happens when default security configurations are configured, enforced, and maintained. A secure, application-defined, web server, database server, and network architecture require strong protection. The updating of the program is equally critical.

Security Weaknesses:

IMisconfiguration of protection can occur at any application stack level including portal, web user, user client, databases, software, and custom code. Automated scanners are useful for error detection, default accounts or configurations, redundant facilities, heritage choices.

Technical Impacts:

If you have these vulnerability gaps in your web server, depending on the program and the exact fault, the effects can be very serious.

It usually gives attackers entry or exposure to unauthorized functionality of data that they don't want to share. It can also lead to a complete system compromise by allowing an attacker to do anything he wishes on your system, such as robbing the database, modifying rules for business processing, installing worms, stopping or deleting your website.

Prevention:

- Deactivate default usage of accounts/passwords.
- Try running checks and performing routine assessments to spot potential mistakes or missed updates.
- Server Configuration to stop unwanted entry lists files, etc.
- Disable interfaces for administration.

A7:2017 Cross-Site Scripting (XSS):

Exploitability: Easy

Prevalence: Widespread

Detectability: Easy

Technical Impact: Moderate

Cross-site Scripting (XSS) is an intrusion attack by the client. The intruder seeks to perform malicious scripts on a victim's web browser by including malicious code on a legal website or web application. If the developers have unsensitized user input in the data they create a web page or a software application that is vulnerable to XSS. This user input is then evaluated by the browser of the victim. Forum, message boards, and web pages that allow comment are vulnerable vehicles commonly in use for cross-site scripting attacks.

Cross-Site Scripting is divided into 3 different types:

- **Reflected Cross-Site Scripting:**

The simplest cross-site scripting is reflected XSS. This happens if a device collects data in an HTTP request and uses this data in an insecure manner in the immediate response.

- **Stored Cross-Site Scripting:**

XSS is processed when a device collects data from an untrusted source and incorporates the data in its corresponding HTTP answers in an unclear manner.

- **DOM-Based Cross-Site Scripting:**

DOM-based XSS is generated when an application contains a JavaScript client which processes data from an unauthorized user in a dangerous manner, typically by returning the data to the DOM.

Attack Vector:

Cross-Site Scripting (XSS) is a common attack vector that injects malicious code into an insecure Web application. XSS differs from other network attack vectors as the program itself is not attacked directly. The web server consumers are at risk instead.

A successful cross-site scripting attack can devastatingly impact the reputation and relationship of an online company with its customers.

Security Weaknesses:

If the developers have unsensitized user input in the data they create a web page or a software application that is vulnerable to XSS.

This user input is then evaluated by the browser of the survivor. VBScript, ActiveX, Flash, and even CSS can be used to attack XSS. But they are most common in JavaScript, particularly because JavaScript is crucial for most browsing experiences.

Technical Impact:

If an attacker fails to bypass XSS bugs, it can gain passwords. They can also spread webworms, hack the user's device to show the browsing history of the user, remotely monitor the browsing. Attackers can also evaluate and utilize certain intranet software until the victims' device has been monitored.

An attacker may perform malicious acts, such as by leveraging XSS vulnerabilities:

- Remotely inspect the browser.
- Application, as well as Appliances connected to the internet, can be scanned or even exploited.
- Transmit webworms.
- User Accounts can be Compromised.
- Enable history of the browser and content of the clipboard.

Prevention:

Cross-Site Scripting can be prevented by adopting following measures:

- Input data should be filtered upon arrival.
- Output data should also be encoded before sending.
- Policies including content security should be imposed.
- Appropriate response headers must be used.
- Data should be sanitized at all levels.

A8:2017 Insecure Deserialization:

Exploitability: Difficult

Prevalence: Common

Detectability: Average

Technical Impact: Severe

Insecure deserialization is a weakness that occurs if untrusted data is used for misuse of an application's logic, for performing a denial of service, or even for deserializing arbitrary text. The Insecure Deserialization has a rank position at 8 in OWASP Top 10 2017.

Attack Vector:

Insecure deserialization is a flaw in which insecure or undefined data are used to hack, execute code, circumvent authentication or further exploit the logic of an application to either a denial-of-service attack (DOS attacks). Serialization is the mechanism by which the object may be transformed into a file later on. Deserialization is the opposite mechanism that passes data to an entity from a register, stream, or network.

Security Weaknesses:

Web apps routinely use serialization and deserialization, and even native data serialization features, especially in popular formats like JSON and XML, are available in the majority of programming languages. It is important to understand that safe object deserialization in software development is a normal practice. Moreover, the problem starts when untrusted user input is deserialized.

Technical Impacts:

When an application remains vulnerable, an attacker can abuse the deserialization mechanism. An attacker can use a web app for hostile serialized objects to initiate deserialization of the hostile data by the victim's computer. The attacker will then adjust the angle of attack, allowing the original entry point to a victim's device with insecure deserialization.

Prevention:

- Encrypting mechanisms in serialization.
- The usage of a firewall to prevent serious damage.
- Track the cycle of deserialization.
- Limiting access permissions to the deserialization code.

A9:2017 Using Components with Known Vulnerabilities:

Exploitability: Average

Prevalence: Widespread

Detectability: Average

Technical Impact: Moderate

Nearly all applications are insecure because the code relies on vulnerabilities in components or libraries. There are also intentional flaws. Vendors were known to remotely access a system by leaving backdoor vulnerabilities in their systems. Most vulnerabilities, however, are unintentional. The safety deficiencies inherent in the conception of the product are responsible.

Having the modules and libraries current and updating as long as the newest release is available would greatly minimize the number of known bugs placing the program at risk, but it is more normal for developers not to learn the modules that their applications currently use.

Most suppliers deal with vulnerabilities by identifying the vulnerabilities and addressing them with product updates and patches or by releasing a new version of the product.

Attack Vector:

This type of threat comes with almost always executing components such as libraries and framed books with full privileges within the app. This makes it possible for the hacker to inflict significant data loss or servers' acquisition when a weak part is used.

Security Weaknesses:

This problem has a widespread prevalence. Component-heavy programming practices cannot help engineering teams realize the features they need in their SDK or applications and can be left even less up to date. This issue may be identified with the use of scanners such as retire.js and header inspection.

Technical Impacts:

- Various weaknesses, including injection, broken access control, XSS, are possible.
- The effect may vary from negligible to complete host acquisition and data compromise.
- This may be negligible or a complete compromise.

Prevention:

- Maintain database objects access limit.
- Maintain the secured network and firewall.
- Keep the default and recommended test tools to validate and update these vulnerabilities on time.
- Retain all resources up to date, such as files, project mailing list, etc.
- Add protection wrappers for fragile parts.

A10:2017 Insufficient Logging & Monitoring:

Exploitability: Average

Prevalence: Widespread

Detectability: Difficult

Technical Impact: Moderate

There is insufficient security-critical logging and monitoring vulnerability when the system does not monitor current events. The lack of such tasks will certainly make it impossible to track criminal behavior which can impair the efficacy of emergency response in the case of an attack. The OWASP classified this as one of the Top 10 Essential Network Security Threats 2017 based on the outcome of the risk review below.

Attack Vector:

If an attacker wants to hack a flaw, they spend a lot of time testing a program or device to find such vulnerabilities. In the case that a program lacks sufficient tracking and control, the attacker can discover bugs and vulnerabilities at pleasure, raising the likelihood of a known bug being discovered and abused. Ideally, you will have software to monitor you to this pernicious test; otherwise, you will at least need a mechanism to detect the intrusion that you are targeted in.

Security Weaknesses:

If a malicious insider is left powerless to identify what happened when it has authentic reasons for querying databases, accessing applications, modifying system configurations, and obscure data. A comprehensive recording of consumer access to important company information has now become an integral factor for protecting the brand identity and digital properties of businesses. Within this paper, we will discuss the vulnerability of so-called insufficient reporting and tracking.

Technical Impact:

Although the tracking and reporting inadequate to be a clear attack vector are too vague, this affects the identification and reaction to each violation. If web application and server events are tracked improperly, inappropriate activities may be missed quickly. If security issues are not adequately logged – or records are improperly maintained or impossible to obtain – the faults may not be resolved.

Prevention:

Ensure that high-value transactions provide an audit trail of quality protection, such as connection tables or equivalent, to avoid tampering or deletion.

- Ensure that logs are generated by central log management solutions, in a format that can be easily consumed.

- Ensure that all authentication errors, server-side input validation, and access control errors are logging in with enough user context in which suspicious or malicious accounts are detected.

- Establish effective monitoring and warning to identify and respond to suspicious activities promptly.

- Create or adopt a response to incidents and recovery plan.

2.0 OWASP ASVS

2.1 Introduction:

The OWASP ASVS is the short abbreviation for Application Security Verification Standards, which is A framework to test technical web application security checks and also provides development developers with a list of requirements for secure development.

The OWASP Application Security Verification Standards (ASVS) project's principal goal is to standardize the scope array and validity required in the industry in carrying out security tests for web applications using an open general that is technically workable. This standard is used to test application technological security controls and all technical environmental security controls relating to vulnerabilities such as cross-site scripting (XSS) and SQL injection protection. This standard can be used to establish a level of trust in web applications' safety.

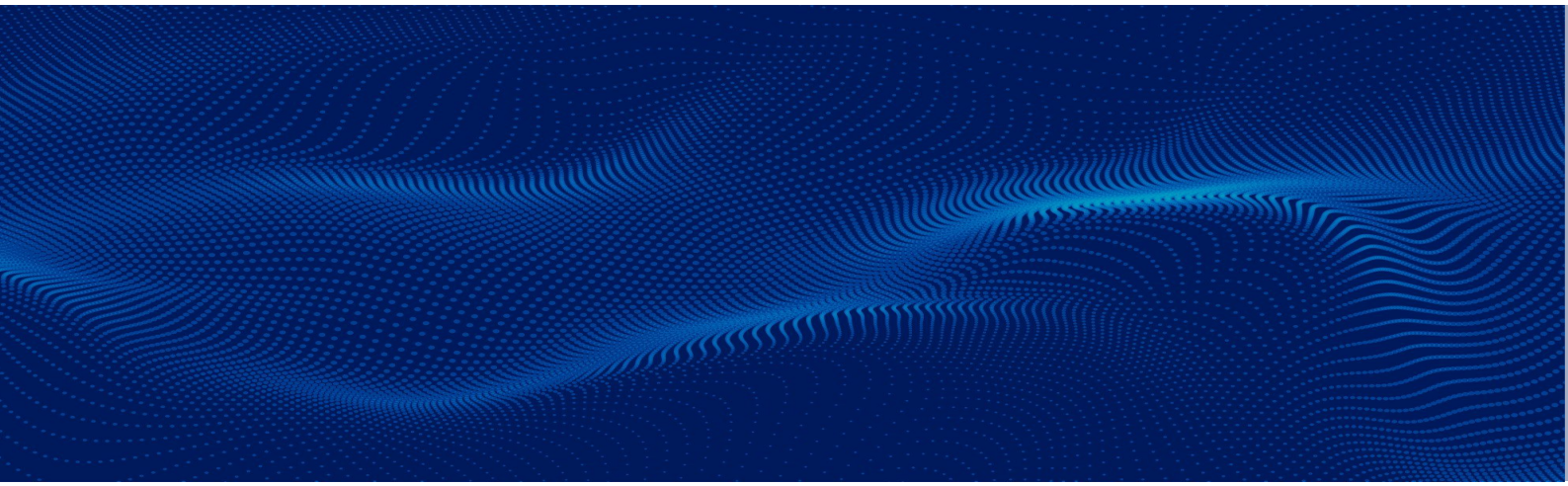
2.2 Requirement:

With the following objectives, these requirements were developed:

- Use as a metric - Provide the developer and owner of applications with a yard to evaluate the level of confidence placed on their Web applications

- Use ad Guidance - Provide guidelines for the implementation of security controls as to how to integrate security controls to meet security criteria for the application

- Usage during acquisition - Set the basis for defining specifications in contracts for application security verification.



2.3 Use of ASVS:

Usage of OWASP ASVS:

Several "levels" are used by the OWASP ASVS to classify and determine the level of security checks for the the web application. This makes it easy for developers to recognize and see real-world security technology needs. For a web application that does not need any other authentication level, a Level 1 requirement could, for example, be enough. This hierarchical level structure simplifies the definition of the requisite application safety and avoids fewer stable applications.

2.4 Application:

Applications & Importance of OWASP ASVS:

- Help architects and engineers create safe applications by planning and constructing protection and ensuring that it is in place and efficient by the use of ASVS testing unit and integration tests.

- Support organizations implement or develop high-quality stable coding standards.

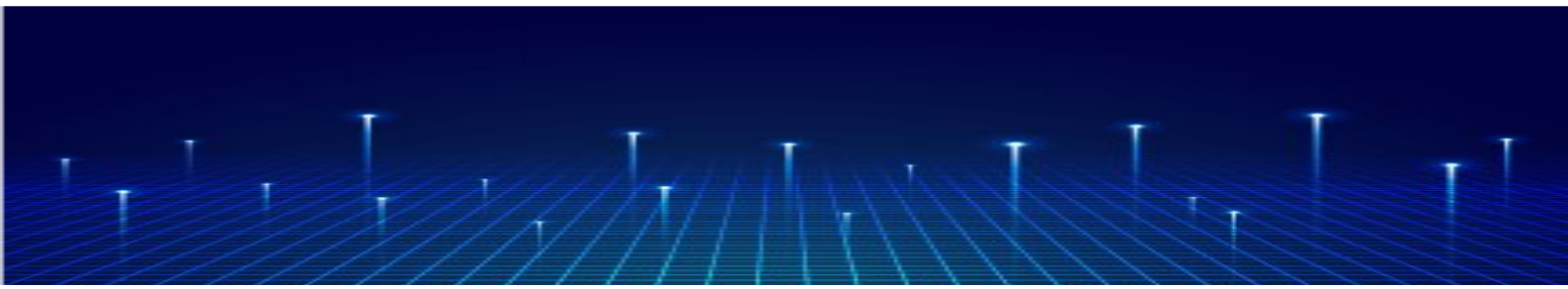
- Support software testers use a systematic, reliable, high-quality framework for integrated code reviews, stable code reviews, user code reviews, interviews, and collaborate with developers to create software unit and implementation checks. It is also possible to use this model for level 1 penetration checks.

- Help to use testable, secure builds to implement secure software.

- Support tool providers by ensuring the readable version of CWE mappings is generated easily.

- Assist organizations in setting the application safety tools for dynamic, interactive, and statistical analysis of tools by the number of availabilities for ASVS.

- Minimize conflicting and competing criteria from other specifications, either by aligning closely with them or by having strict supersets that can help minimize compliance costs, energy, and wasted time in recognizing needless discrepancies as threats.



3.0 WASC

3.1 Introduction to WASC:

WASC Stands for Web Application Security Consortium. The WASC Risk Assessment is a joint initiative aimed at clarifying and coordinating security risks to a website. The project was developed by the members of the Web Application Security Consortium to create and facilitate industry standards for defining these problems. Developers of apps, technology professionals, suppliers of products, and enforcement auditors should be able to use a common vocabulary and Network security description-related concerns.

3.2 What is WASC:

WASC is a non-profit group consisting of a Global Association of practitioners and specialists from different fields of the World Wide Web. Web Security Application Consortium (WASC). They create standards of protection in open-source best practices that are broadly agreed upon and implemented across the Internet, especially in the field of web apps.

3.3 Function Of WASC:

The WASC was founded in 2004 and includes Individual Participants, businesses, federal departments, and university officials. In 2004, WASC is a network application protection consortium. They have the mandate to review, debate and publish information, particularly on security issues with web apps, and have also supported vulnerability disclosure in their consortium. They consistently publish technical information, best practices, and guidelines on security and contribute security articles. Such tools are used for the further development of network security by companies, states, application creators, technology specialists, and educational institutions.

3.4 List of Security Threats issued by WASC:

1. Brute force:

The Brute attack involves potentially millions of concurrent connections on the website. Because no web server can manage that much traffic concurrently, it opens the door for an intruder to join.

User awareness in choosing the right type of password and built-in authentication mechanisms, such as having upper- and lower-case numbers or special characters, will only go so far as to secure the user's identity. Security tests must also be carried out on the server-side if brute force attacks are to be effectively mitigated.

2. Buffer Overflow:

Buffer Overflow is a type of WASC module that is dealt with as a malfunction in blocks of the data memory. The buffer that has to assign a specific amount of memory to this block overflows if more data is written in a memory block than it can hold. Using these vulnerability loopholes, several different forms of threats can be leveraged. The primary objective of the attacker is to monitor the target process by the buffer overflow.

3. Content Spoofing:

This manipulative content fools users into redirects to install malware, to viruses that could be disguised as normal apps or which may later be used to jeopardize victims. Content spoofing is widely used on bogus pages where purchases and related customer financial information may be hacked, including PayPal spoofs, airline ticket reservation sites, parcel distribution companies, banks, and more.

4. Directory Indexing:

The server handles this URL by displaying the appropriate website directory to display the contents as an entry. If the homepage or the website's main page is not available, the server reports the directory listing, and the full result is forwarded to the end-user. The findings are mainly reasoning for processes if the server uses Unix OS or Dir where the server uses Windows OS.

Nevertheless, only HTML representation of the tests is available in the client/user calling the URL. To track these flaws and to build programs, the attacker uses countermeasures.

5. Failure to restrict URL Access:

Protection plugins for blogs and domains do not always encrypt posts. It might make matters even harder if a program needs to be installed, or if an incorrect application implementation occurs.

It would be incredibly easy for attackers to violate all website records, to reach all user accounts, and vice versa.

6. Http Request/Response Splitting:

A client sends HTTP notifications to websites, allowing users to access content in seconds. When an attack is in progress, the same URL request is mixed with XSS attacks to infect browser-cache damage. In the HTTP request/response scattering attack.

After being infected/poisoned, the browser of the user is prompted to load the attack. Consequently, two requests are made, because the latter is the malicious type, rather than one HTTP request.

7. Improper file system permission:

Incorrect access to the file system is more of a threat than a vulnerability. These permission errors occur when developers are insufficient care in implementing file access permissions during project work.

8. Improper Input Handling:

Input handling is the most prevalent type of vulnerability on any online network when it is wrong or malfunctioning. Such vulnerabilities are found mainly on blogs, mobile apps, and so on.

Input handling includes inputting validation functions, sanitizing data, filtering information, encoding, handling arguments, and so on.

9. Information Leakage:

Leakage of information is a flaw in a software program or a full website where directory or files are not sufficiently concealed, thereby exposing the secret and covered sensitive information.

Leakage attacks on any online or offline site have a huge impact. The details may be exploited in different ways by criminals, including bribery, blackmail, credit card theft to the identity stealing, or even worse.

10. Insecure Cookie:

Cookies have huge potential for cyber aggressors with expertise in exploiting their properties. This is why we allow web creators and suppliers of online security services to erase server-side and local-side cookies.

The attacker attacks other customers, servers, and any cookie-storage program. The attacker senses the form, calculates the expiration date of such cookies, and then attacks accordingly. The intruder thus knows when cookies were created, for what reasons they will be used, and why they will be used for that reason.

11. Insecure Cryptography Storage:

The program is designed to protect against users accessing confidential data or privately-run websites. In this type of attack, the attackers raise no alarms. They continue their activities in silence; they break information, blow the whistle, rob precious data.

Uncertain cryptography can be easily avoided by encrypting files, spinning keys, powerful algorithms, and applying these protocols. If such protocols fail to be followed, it can lead to costly recovery plans and reputation protection.

12. Insecure Indexing:

In the event of unsecured indexing conditions, data security is threatened. Normally, the public view is meant to cover website files or archives. Similarly, they are deemed privileged to have access to the site not for all users.

Such indexed files can contribute to the device integrity of attackers. If such files are indexed by a website or the intruder, unauthorized parties capture the information they hold or that they hold in a sequence of URL queries.

13. Insufficient Authentication:

It is when a company has applied Password Authentication Protocols carelessly or inappropriately. While care has been taken to optimize compliance with account login protocols, new strategies are often created to try to subvert the program. To mitigate the effect of these attacks, enhancements to modifications may always be required.

14. Insufficient Authorization:

Consider your website user doing an action that is consistent with your company's security policies. In this context, it is necessary to call the relevant authorization levels to make it easy for users to proceed with business. The user will not see any content limited or privileged information that is not available at his current level. Proper authorization procedures are so deeply developed in our everyday web operations that something that has gone wrong may quickly be overlooked.

When the same user has poorly established authentication/access thresholds and has a malicious purpose in mind, he/she can use, use, use, or sell the information to his / her gain without being allowed.

15. Integer Overflow:

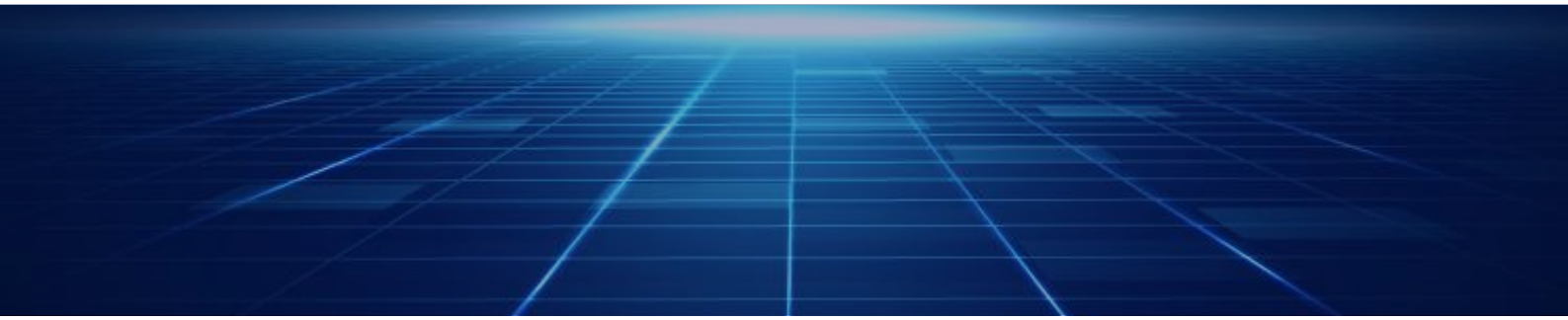
Integer overflow is a type of arithmetic overflow error if it does not fit into the allocated memory space. It normally allows the outcome to be unpredictable instead of a mistake in the software. For most programming languages, a certain number of bits in the memory are typically assigned integer values.

16. LDAP Injection:

LDAP is the shortened form of Lightweight Directory Access Protocol. LDAP injection is a vulnerability where queries are built without prior validation or sanitization from untrusted input. LDAP employs questions constructed from different characters, for instance, brackets, asterisks, ampersands, or quotes, which require the use of syntax.

17. Local File Inclusion (LFI):

An attacker can read files to the victim's computer via LFI vulnerabilities. That can be very risky as the intruder will obtain access to confidential information if the Web server is misconfigured and runs in high privileges. The attacker will then execute arbitrary commands if he can put code on the webserver through any means.



18. Path Traversal:

The Path Traversal techniques are available on an unauthorized basis to files located outside the network archive. The attacker manipulates or runs a website's URL to expose files that extend through a folder directory structure.

The Path Traversal attacks are used in many areas of the web application as well as in other parts of the filesystem where the webserver can interpret, for access to the confidential information contained inside arbitrary files. A successful Path Traversal attack may allow an attacker to continue recognition or exploit other vulnerabilities to the security of an application since files containing sensitive information might contain secrets such as passwords, access tokens, etc.

19. Remote File Inclusion (RFI):

Remote file inclusion (RFI) is an attack that targets web application vulnerabilities that reference external scripts dynamically. The attacker aims to use the reference feature in an application to upload malware from a remote URL from a specific area.

The effects of a successful RFI attack include identity stealing, corrupted computers, and a website change takeover.

20. Sensitive Data:

Typically, the repository manages most files, the configuration of the directory, and the program function on its side. Because often an app will be modified or even the main website page, as a result of these activities a series of batch files or backup files are generated. Because of their sensitive nature, these backup files are targeted by cyber attackers.

21. Server Misconfiguration:

User identity information, login information, and many other items intended for developers or web administrators are typically filled out on servers. If such information is not locked on the server-side by misconfiguration, unforeseen consequences could occur. Server malfunctions arise from vulnerabilities and flaws discovered on the server-side of every online or offline business.

22. Server Response:

When the cyber-attack is correct, a server can send a single HTTP request to the attacker using this split response opportunity. These demands can be submitted once or several times, irrespective of what is needed to impact the victim. Instead of one HTTP response, the webserver is forced to develop an output data stream interpreted by the attacker/objective in two different ways.

The first server-side response is likely managed by the intruder. But what matters is how the second server response stream is handled by the attacker. The HTTP status line is managed by the attacker attacking the machine, server, or some other unit of the victim. The answer is controlled by the HTTP body's last byte.

23. Session Prediction:

The intruder creates trust by communicating as a frequent user with the victims' website/community. Then sensitive information is more or less hijacked by identifying particular logs of the session with a little impersonation. It is also known as Session Hijacking Attack, where the domain requests with compromised account accounts are likely to be submitted by attackers. Even when you access your accounts, users that visit websites that require a username and password put their security at stake. They have passwords that are distributed for authentication as a session to the server-side. The attacker selects and accesses these session IDs and calls a legal user.

24. SSI Injection:

SSI is the 'Server Side Include' terminology abbreviation referring to the failure of improperly deployed sanitation procedures on a website. The server-side attacker uses data packets to inject later interpreted as valid arguments by website administrators or the site background.

These points can also be inserted in an HTML file by modifying, which enables attackers to execute their business more effectively. Before it is supplied to the clients who order, the website server analyzes details. The attacker gains the ability to compromise a system by submitting a server-side with statements.

25. URL Redirect:

Webmasters look at the details of all URLs which are responsible for transferring users to specific website pages when creating or managing the website.

To mess with web load balancing and monitor the outgoing connections, the attacker uses URL redirectors to name a few types of attack. URL redirects are overlooked as they are not treated as a vulnerability to security. As a result, travelers to a website frequently find themselves in contact with other websites with malicious codes and so on.

4.0 SANS CWE 25

4.1 About CWE:

Common Weakness Enumeration (CWE) is a community-developed list of common security-related device and hardware vulnerabilities. Weaknesses are bugs, defects, flaws, or other failures in the operation of software or hardware, programming, specification, or architecture that can be vulnerable to attacks if left unaddressed. The CWE List and related classification taxonomy was used as a vocabulary in which these deficiencies in terms of CWE can be defined and represented.

4.2 What is SANS CWE 25:

The Top 25 Common Weakness Enumeration (CWE) is a demonstrative list of the most common and important flaws that can result in severe program vulnerabilities. Often these shortcomings can be easily identified and exploited. It is dangerous as they often encourage opponents to take over program execution, steal data, or stop the program. The CWE Top 25 is a community resource that can be used to provide input on some of the most prevalent security threats to software developers, software testing providers, software clients, software project managers, security investigators, and trainers.

4.3 List of CWE Top 25:

- CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
- CWE-89: SQL Injection
- CWE-94: Code Injection
- CWE-434: Unrestricted Upload of File with Dangerous Type
- CWE-120: Buffer Copy Without Checking Size of Input ('Classic Buffer Overflow')
- CWE-494: Download of Code Without Integrity Check
- CWE-829: Inclusion of Functionality from Untrusted Control Sphere
- CWE-306: Missing Authentication for Critical Function
- CWE-307: Improper Restriction of Excessive Authentication Attempts
- CWE-798: Use of Hard-coded Credentials
- CWE-807: Rely on Untrusted Inputs in a Security Decision
- CWE-862: Missing Authorization
- CWE-863: Incorrect Authorization

- CWE-311: Missing Encryption of Sensitive Data
- CWE-319: Cleartext Transmission of Sensitive Information
- CWE-22: Path Traversal
- CWE-285: Improper Authorization
- CWE-250: Execution with Unnecessary Privileges
- CWE-676: Use of Potentially Dangerous Function
- CWE-732: Incorrect Permission Assignment for Critical Resource
- CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-Site Scripting')
- CWE-134: Use of Externally-Controlled Format String
- CWE-190: Integer Overflow or Wraparound
- CWE-327: Use of a Broken or Risky Cryptographic Algorithm
- CWE-759: Use of a One-way Hash Without a Salt

4.4 Comparing SANS CWE 25 with OWASP Top 10:

A list of 25 key vulnerabilities in software. A similar listing is provided in the Top 10 Project, which includes a community-based collection of software vulnerabilities, the Open Web Application Security Project (OWASP). Although they are different from CWE/25 and OWASP Top 10, they contain many of the same flaws. The OWASP Top 10 entrants for 2017 and their corresponding CWEs are listed below.

OWASP Top 10	SANS CWE 25
A1: Injection	• CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection') • CWE-89: SQL Injection • CWE-94: Code Injection • CWE-434: Unrestricted Upload of File with Dangerous Type • CWE-120: Buffer Copy Without Checking Size of Input ('Classic Buffer Overflow') • CWE-494: Download of Code Without Integrity Check • CWE-829: Inclusion of Functionality from Untrusted Control Sphere
A2: Broken Authentication	• CWE-306: Missing Authentication for Critical Function • CWE-307: Improper Restriction of Excessive Authentication Attempts • CWE-798: Use of Hard-coded Credentials • CWE-807: Reliance on Untrusted Inputs in a Security Decision • CWE-862: Missing Authorization • CWE-863: Incorrect Authorization
A3: Sensitive Data Exposure	• CWE-311: Missing Encryption of Sensitive Data • CWE-319: Cleartext Transmission of Sensitive Information

A4: XML External Entities	• WE-22: Path Traversal • CWE-285: Improper Authorization
A5: Broken Access Control	• WE-22: Path Traversal • CWE-285: Improper Authorization
A6: Security Misconfiguration	• CWE-250: Execution with Unnecessary Privileges • CWE-676: Use of Potentially Dangerous Function • CWE-732: Incorrect Permission Assignment for Critical Resource
A7: Cross-Site Scripting (XSS)	• CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-Site Scripting')
A8: Insecure Deserialization	• CWE-134: Use of Externally-Controlled Format String
A9: Using Components with Known Vulnerabilities	• CWE-190: Integer Overflow or Wraparound • CWE-327: Use of a Broken or Risky Cryptographic Algorithm • CWE-759: Use of a One-way Hash Without a Salt
A10: Insufficient Logging and Monitoring	NONE

4.5 Top 25 CWE In Details:

1. CWE-78: Improper Neutralization of Special Elements Used in an OS Command ('OS Command Injection'):

Applications that use external input when executing a command on a host system are vulnerable to command injection. Misrepresented input might cause the command to work unaware, potentially compromise the whole operating system.

2. CWE-89: SQL Injection:

SQL injection takes place while a program is using foreign data in a SQL database. It can be used by an attacker to execute any number of SQL commands in the database which leads to data recovery or manipulation.

3. CWE-94: Code Injection:

Code injection is the insertion of malicious code into a program. The code that has been introduced or injected can compromise the integrity of the database and/or compromise the privacy properties, security, and even the correctness of data. It can also steal records, and circumvent access control and authentication.

4. CWE-434: Unrestricted Upload of File with Dangerous Type:

Certain types of files may be viewed as binary files by the application. An intruder uploads and runs arbitrary PHP code for example, with a PHP server accepting and running PHP scripts.

5. CWE-120: Buffer Copy Without Checking Size of Input ('Classic Buffer Overflow'):

The buffer overflow is the most common code error built-in developers and can have severe security effects if it is taken advantage of. A buffer overflow is an executable error when the buffer index has access to the buffer boundary, sequence, string, etc.

6. CWE-494: Download of Code Without Integrity Check:

The use of code from external sources without verification of its integrity by software is susceptible to injection code. For example, if the server in the library suddenly begins to supply another library, then this change is not known until it has been integrated with the library.

7. CWE-829: Inclusion of Functionality from Untrusted Control Sphere:

It requires the use of foreign source code. For this situation, though, the code cannot be counted on to carry out the actions it needs to take. For example, if your website uses a script to an untrustworthy website, the entirety of your website will suffer any harm done by the external script.

8. CWE-306: Missing Authentication for Critical Function:

Authentication is always necessary for commands performing privileged actions or communicating with sensitive data. If the program lacks the mechanism of authentication, attackers may execute protected actions without their identity being checked.

9. CWE-307: Improper Restriction of Excessive Authentication Attempts:

If a user fails to authenticate several times within a short period, the application should not retry the user until a certain amount of time has passed. An attacker can try different credentials continuously if not, which leads to a brute force attack.

10. CWE-798: Use of Hard-coded Credentials:

During testing or authentication of an app with another service hard-coded credentials are often used. An attacker who finds this information can use them to log in, access sensitive information, or impersonate the application.

11. CWE-807: Rely on Untrusted Inputs in a Security Decision:

Any secret fields can be modified to include specific values from the specification anticipated. Attackers can change certain fields to fool or circumvent security measures like authentication or management of sessions.

12. CWE-862: Missing Authorization:

The application should always review the authorization of the user who requested the action when carrying out any privileged action. Failure to access sensitive information or to take risky acts may allow unprivileged users.

13. CWE-863: Incorrect Authorization:

Authorization checks not properly carried out can be tricked by attackers or bypassed.

14. CWE-311: Missing Encryption of Sensitive Data:

During travel and at rest, confidential data will be encrypted. Anyone using it would therefore not be able to view confidential data without the decryption key.

15. CWE-319: Cleartext Transmission of Sensitive Information:

Data should also be encrypted during the transmission via a network-like communication channel. Data transmitted through a network could be accessed by a third party and would allow an intruder to access the unencrypted data.

16. CWE-22: Path Traversal:

If the user, file, or network data is used for building a pathname, a proper input check may pass through the directory structure of the device. This vulnerability could result in the leakage of private and sensitive information, library substitution and malware execution, or corrupt system configurations.

17. CWE-285: Improper Authorization:

Users can execute commands without checking the defining user inappropriate authorization. This happens when an attacker carries out an action requiring but not an authorization check.

18. CWE-250: Execution with Unnecessary Privileges:

Requests should carry out operations with minimum permits required. For example, it can overwrite critical application and system files by writing to a file as an administrator instead of a normal user.

19. CWE-676: Use of Potentially Dangerous Function:

If used incorrectly, some functions may have negative effects allowing an attacker to access system resources or cause mistakes.

20. CWE-732: Incorrect Permission Assignment for Critical Resource:

The right permissions are to be allocated to operations that create and manage resources (such as files). Unintentional access by other users would be permitted if this is not done.

21. CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-Site Scripting'):

Cross-site scripting (XSS) occurs when an attacker submits an input which is then made available as output by the application. If there is code in the input for example JavaScript, the code is executed in the browser provided by the page.

22. CWE-134: Use of Externally-Controlled Format String:

Externally managed format strings can allow an attacker to trigger or even write to a stack execution error. Only standard file strings can be used by programs.

23. CWE-190: Integer Overflow or Wraparound:

When the value of a variable is too large for its type, an integer overflow occurs, causing it to wrap to the edge of the type. In various operations including basic arithmetic, control looping, and memory operations, this can have unexpected consequences.

24. CWE-327: Use of a Broken or Risky Cryptographic Algorithm:

There are many potential vectors of attack for cryptographic algorithms considered to be weak. With an unsecured algorithm, these attacks open up your application, while safer algorithms are less prone to these attacks.

25. CWE-759: Use of a One-way Hash Without a Salt:

Salt is a random string used in hacking data like a password to improve security. The hash can easily be reversed with brute force or rainbow tables without salt.

SAFE
SECURITY

www.safe.security | info@safe.security

Palo Alto
3000, El Camino Real,
Building 4, Suite 200, CA
94306