

STAGE 1 :-

Understanding various vulnerabilities :

Top 5 Vulnerability Exploitation :

S No.	Vulnerability Name	CWE Number
1	SQL Injection	CWE - 89
2	Cross-Site Scripting (XSS)	CWE - 79
3	Cross-Site Request Forgery (CSRF)	CWE - 352
4	Broken Authentication	CWE - 287
5	Security Misconfiguration	CWE - 16

REPORT :-

Vulnerability Name :- SQL Injection (SQLi)

CWE No :- CWE – 89

OWASP/SANS Category :- Top 5

DESCRIPTION :-

SQL Injection (SQLi) is a critical web security vulnerability that allows an attacker to interfere with a web application's database queries. By injecting malicious SQL code into input fields, attackers can bypass authentication, manipulate database records, and even gain full control over the database.

SQLi typically occurs when an application fails to properly validate and sanitize user inputs before executing SQL queries. If a web application dynamically constructs SQL queries using untrusted user input, an attacker can insert malicious SQL statements that modify the query's logic.

A vulnerable web application takes user input and directly includes it in a SQL query without sanitization. An attacker can modify the query structure and execute unintended database operations.

Example of a Vulnerable SQL Query (Without Protection)

```
SELECT * FROM users WHERE username = 'user_input' AND password = 'user_password';
```

If a user inputs:

```
' OR '1'='1'
```

The query becomes:

```
SELECT * FROM users WHERE username = '' OR '1'='1' AND password = '';
```

BUSINESS IMPACT :-

- Attackers can steal sensitive data, including usernames, passwords, financial records, and personal information.
- If an attacker extracts password hashes, they can crack and reuse them for account takeovers.
- Attackers can manipulate financial transactions, transfer funds, or alter product prices in e-commerce applications

- Advanced SQLi attacks can allow remote command execution, enabling full system compromise.
- Data breaches due to SQLi can lead to regulatory fines (eg ., GDPR, PCI-DSS violations) and loss of customer trust.

STEPS TO IDENTIFY :-

1. Check for Error Messages

- Enter a single quote (') in a login form, search bar, or any input field.
- If the website displays an SQL-related error (e.g., *"syntax error in SQL statement"* or *"unclosed quotation mark"*), it might be vulnerable.

2. Use Common SQL Injection Payloads

- Try entering admin' -- or admin' # in login forms.
- If you gain access without entering a password, the site is vulnerable.

3. Test with Boolean-Based Injection

- Enter:
 - " OR 1=1 --
 - " OR 'a'='a
- If the query returns unexpected results or grants access, it indicates a vulnerability.

4. Check URL-Based Injection

- If the URL has parameters like example.com?id=2, try modifying it to:
 - example.com?id=2'
 - example.com?id=2 OR 1=1
- If the page structure or content changes, it suggests a possible SQL injection flaw.

5. Use Special Characters

- Enter special characters such as ' ; DROP TABLE users; -- in input fields.
- If the site crashes or behaves unexpectedly, it might be vulnerable.

Vulnerability Name :- Cross-Site Scripting (XSS)

CWE No :- CWE-79

OWASP/SANS Category :- Top 5

DESCRIPTION :-

Cross-Site Scripting (XSS) is a critical web vulnerability where an attacker injects malicious JavaScript into a website, which is then executed in a victim's browser. This happens when a web application fails to properly validate or sanitize user input before displaying it. XSS attacks can be classified into Stored XSS, where the malicious script is permanently stored on the website and executes when a user visits the affected page; Reflected XSS, where the script is embedded in a malicious link and runs when a victim clicks it; and DOM-based XSS, which occurs due to insecure JavaScript execution on the client side. The impact of XSS can be severe, allowing attackers to steal cookies, session tokens, and login credentials, potentially leading to account hijacking and phishing attacks. Additionally, it can be used to inject fake content, deface websites, or spread malware.

BUSINESS IMPACT :-

- Attackers can steal user credentials, session cookies, or authentication tokens through malicious scripts.
- XSS can be used to manipulate forms, redirect payments, or steal financial details.
- In e-commerce or banking platforms, it can lead to direct financial losses for both businesses and customers.

- XSS attacks that leak sensitive information can result in heavy fines and legal action.
- XSS can be leveraged to create fake login pages, tricking users into entering their credentials on a malicious site.
- They may use persistent XSS to create backdoors, leading to long-term security risks.

STEPS TO IDENTIFY :-

1. Check for Reflected XSS (Immediate Response)

- Go to any input field (search bar, contact forms, login fields, etc.).
- Enter simple XSS payloads like:
 - `<script>alert('XSS')</script>`
 - `"><script>alert('XSS')</script>`
- If the alert pops up, the site is vulnerable to XSS.

2. Test for Stored XSS (Persistent in Database)

- If the website allows comments, messages, or profile updates, enter:
 - `<script>alert('XSS Stored')</script>`
- If the alert appears when visiting the page again, it means the site stores and executes the malicious script.

3. Check for XSS in URL Parameters

- If the URL changes when searching (e.g., `example.com/search?q=test`), try modifying it to:
 - `example.com/search?q=<script>alert('XSS')</script>`
- If the script executes, the site is vulnerable.

4. Look for HTML Injection

- Try entering:
 - `Test` or `<h1>Test</h1>` in input fields.
- If the text appears bold or large instead of showing the actual tags, the site might allow XSS.

5. Inspect Page Source Code

- Right-click and view page source after submitting input.
- If your text appears inside a `<script>` tag, without encoding, the site may be vulnerable.

Vulnerability Name :- Cross-Site Request Forgery (CSRF)

CWE No :- CWE-352

OWASP/SANS Category :- Top 5

DESCRIPTION :-

Cross-Site Request Forgery (CSRF) is a web security vulnerability where an attacker tricks a user into unknowingly performing unwanted actions on a trusted website. The attacker exploits the user's authenticated session to send malicious requests without their consent.

For example, if a user is logged into their online banking account and clicks on a malicious link, it could transfer money without their knowledge.

BUSINESS IMPACTS :-

- **Unauthorized Transactions** – Attackers can initiate bank transfers, purchase items, or change account details without user consent.
- **Data Manipulation** – Hackers may alter user information (emails, passwords, addresses) leading to data loss or account takeovers.

- **Account Hijacking** – Users' passwords or security settings can be changed, locking them out of their accounts.
- **Loss of Customer Trust** – Customers may lose trust in a platform that does not protect their actions, affecting reputation and revenue.
- **Regulatory Fines & Legal Issues** – Businesses handling sensitive data (e.g., financial or healthcare) can face compliance violations and legal actions.

STEPS TO IDENTIFY :-

1. Check for Missing CSRF Tokens

- Open the login, form submission, or account update page.
- Right-click on the page → Click Inspect → Go to the Network tab.
- Submit a form and check if a CSRF token (like `_csrf` or `csrf_token`) is sent with the request.
- If missing, the site might be vulnerable.

2. Test with Open Tabs (Session Exploitation)

- Log into a website (e.g., banking, e-commerce).
- In another tab, open a suspicious link or submit a request to change details (like password reset).
- If the action is performed without confirmation, the site may be vulnerable.

3. Look for HTTP GET-Based Actions

- If sensitive actions (like changing an email) happen via a GET request (URL-based actions), they may be CSRF-prone.
- Example:
 - `example.com/change_email?new_email=hacker@gmail.com`
 - If simply visiting this link changes the email, CSRF is likely present.

4. Inspect Forms for Anti-CSRF Measures

- Check if forms contain a hidden CSRF token (<input type="hidden" name="csrf_token" value="XYZ123">).
- If not present, the site may be vulnerable.

5. Verify if Login Cookies Work Cross-Site

- Try submitting a form request from another website (using an HTML form or third-party script).
- If it processes the action without requiring re-authentication, CSRF might be possible.

Vulnerability Name :- Broken Authentication

CWE No :- CWE-287

OWASP/SANS Category :- Top 5

DESCRIPTION :-

Broken authentication occurs when an application's authentication system is poorly implemented, allowing attackers to bypass login security, steal user credentials, or hijack accounts. This happens due to weak password policies, exposed session tokens, lack of multi-factor authentication (MFA), or improper session management.

BUSINESS IMPACTS :-

- **Account Takeover:** Attackers gain unauthorized access to user/admin accounts.
- **Data Breach:** Sensitive customer and company data can be stolen.
- **Financial Loss:** Fraudulent transactions, loss of customer trust, and legal penalties.
- **Reputation Damage:** Users may lose trust in the platform, leading to business decline.

STEPS TO IDENTIFY :-

1. Test Default and Weak Passwords

- Try logging in with common passwords like:
 - admin/admin, admin/password, user/123456, guest/guest.
- If these work, it indicates weak authentication security.

2. Check for Missing Multi-Factor Authentication (MFA)

- If a website doesn't enforce MFA (OTP, SMS, Authenticator app, etc.), it's at higher risk of account takeover.

3. Session Hijacking Test

- Log in to an account and copy the session ID from browser cookies.
- Open another browser, paste the session ID, and see if you are still logged in.
- If the session stays active, session management is weak.

4. Verify Logout and Session Expiry

- Log in and log out, then press the back button.
- If the session is still active, the site doesn't properly invalidate sessions.
- Stay idle for a long time. If your session doesn't expire, it's a security flaw.

5. Check for Credential Stuffing Risks

- If the application allows unlimited login attempts, attackers can use automated tools to guess passwords.
- Try entering incorrect credentials multiple times. If there's no lockout, it's vulnerable.

Vulnerability Name :- Security Misconfiguration

CWE No :- CWE - 16

OWASP/SANS Category :- Top 5

DESCRIPTION :-

Security misconfiguration happens when systems, applications, or servers are not properly secured, leaving them vulnerable to attacks. This includes:

- Default credentials (e.g., admin/admin, root/password)
- Unnecessary features enabled (e.g., directory listing, debug mode)
- Overly permissive access (e.g., unrestricted admin panels, open database access)
- Exposed error messages that reveal sensitive information

BUSINESS IMPACTS :-

- **Data Breaches** – Hackers can access sensitive customer and business data.
- **Unauthorized Access** – Attackers can gain admin-level control over systems.
- **Financial Loss** – A security breach can lead to regulatory fines and legal action.
- **Reputation Damage** – Loss of customer trust due to exposed vulnerabilities.

STEPS TO IDENTIFY :-

1. Check for Default Credentials

- Try logging into admin panels or web applications using common defaults like:
 - admin/admin
 - admin/password

- root/root

2. Look for Open Directories

- Visit website directories by entering URLs like:
 - example.com/admin/
 - example.com/config/
 - example.com/uploads/
- If the page shows a list of files instead of an error, directory listing is enabled.

3. Identify Exposed Debug Information

- Search for error messages when using invalid inputs.
- If errors reveal database queries, server info, or file paths, the site is misconfigured.

4. Test for Unrestricted Admin Panels

- Try accessing pages like:
 - example.com/admin
 - example.com/phpmyadmin
- If the admin panel loads without authentication, it's a serious vulnerability.

5. Look for Unprotected API Endpoints

- Check if URLs like example.com/api/users return sensitive user data.
- If data is exposed without authentication, it's a security risk.

STAGE – 2 :-

Nessus:

Nessus is a powerful vulnerability assessment tool developed by Tenable, widely used by security professionals to detect vulnerabilities, misconfigurations, and compliance issues in IT systems. It helps organizations proactively identify security risks and remediate them before they can be exploited by attackers.

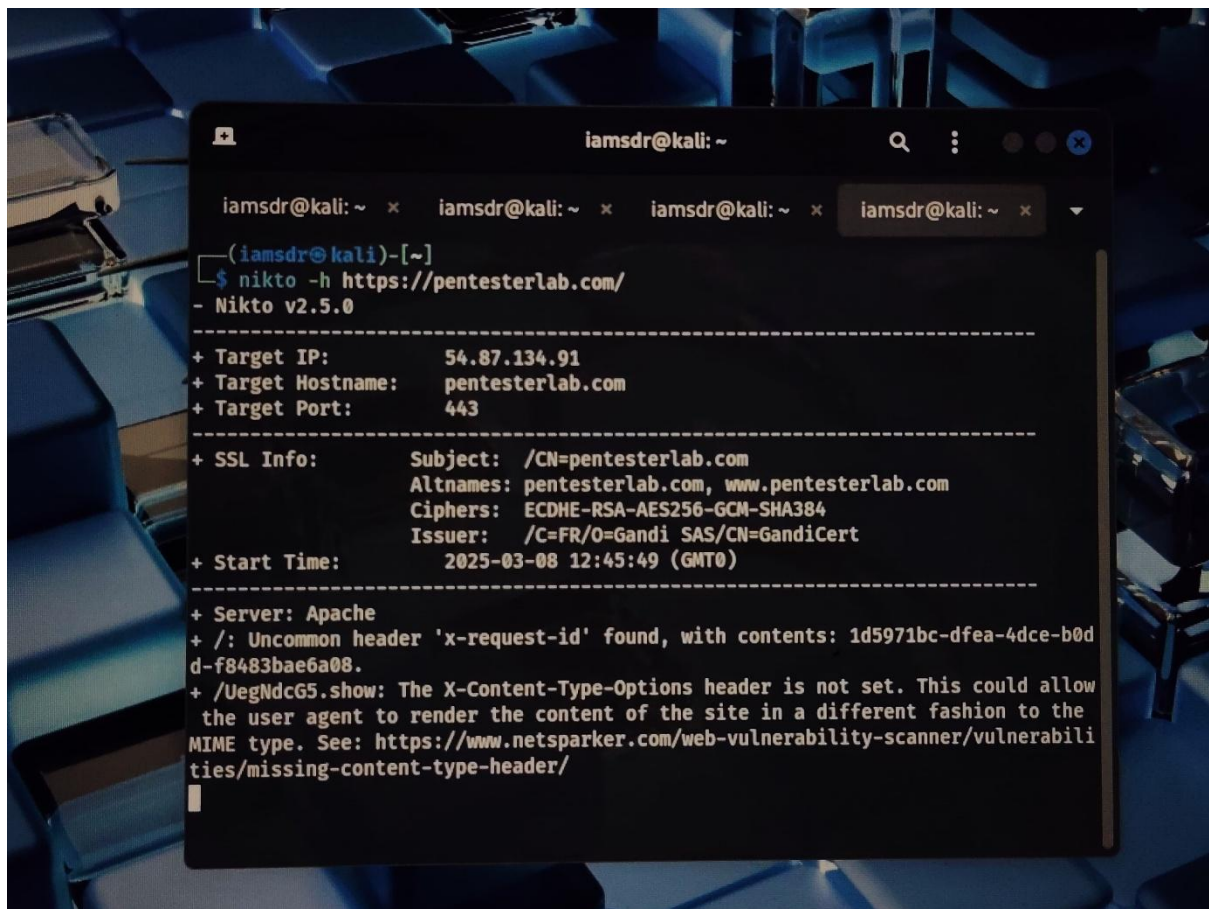
One of the key strengths of Nessus is its comprehensive vulnerability scanning capabilities, which allow organizations to proactively detect security flaws before they can be exploited by attackers. The tool uses an extensive database of over 180,000 plugins, regularly updated to identify new vulnerabilities, misconfigurations, and outdated software. Nessus scans devices for open ports, unpatched software, weak passwords, and dangerous configurations that could lead to security breaches. It also detects malware, backdoors, botnet activity, and ransomware-related vulnerabilities, ensuring that security teams can take immediate action to mitigate risks. In addition to standard vulnerability scanning, Nessus provides compliance auditing to help organizations adhere to regulatory standards such as PCI-DSS, HIPAA, ISO 27001, NIST, and CIS benchmarks. This makes it an essential tool for companies that must meet strict security requirements.

While Nessus is highly effective, it does have certain limitations that security professionals should be aware of. Like many automated scanning tools, it can sometimes produce false positives, requiring manual verification of certain findings. Additionally, Nessus does not automatically remediate vulnerabilities—it provides detailed reports and recommendations, but fixing the issues requires manual intervention by IT teams. Another challenge is that large-scale scans can consume significant system resources, which may impact network performance if not properly configured. Despite these challenges, Nessus remains one of the most trusted tools in vulnerability management due to its accuracy, reliability, and continuous updates to stay ahead of emerging threats.

Target Website :- <https://pentesterlab.com/>

Target IP Address :- 54.87.134.91

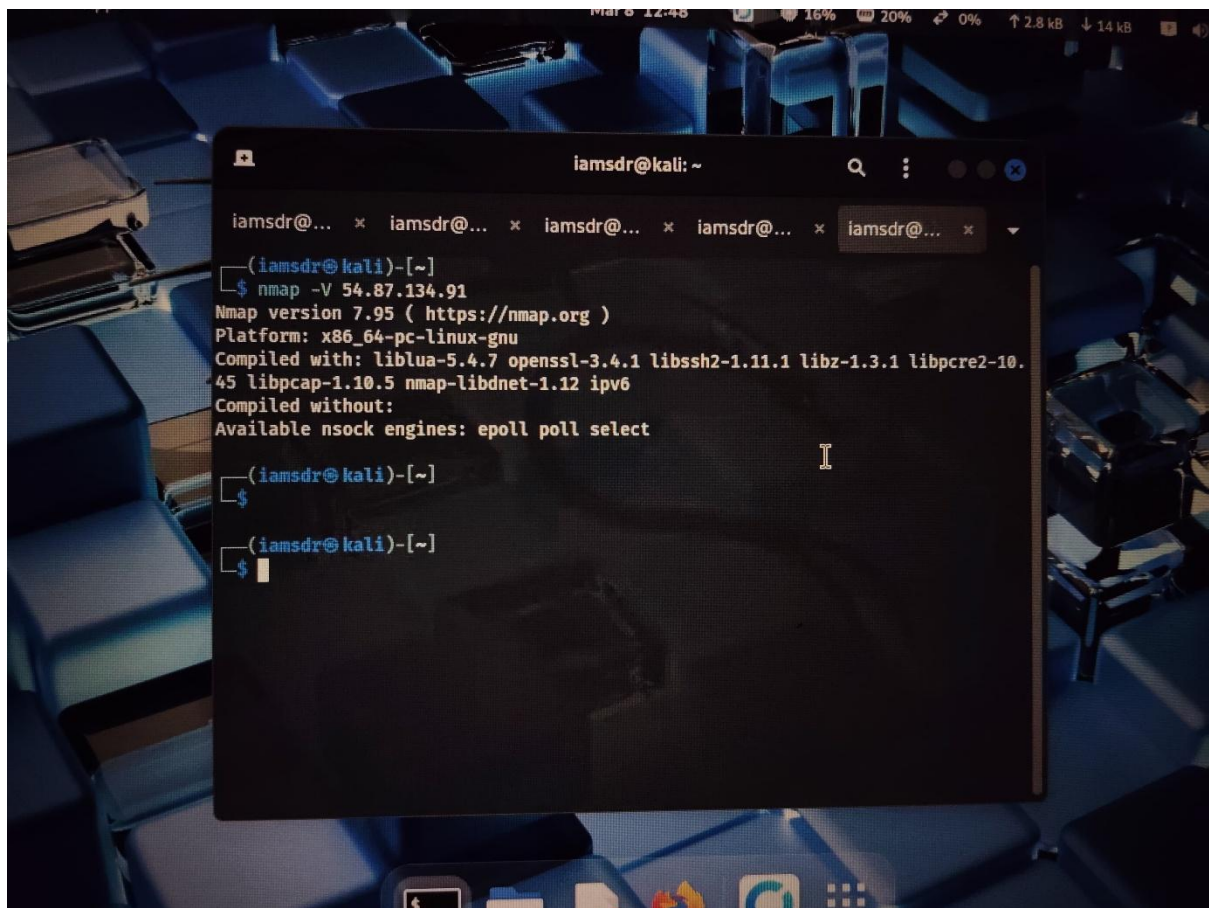
Target Port :- 443

A screenshot of a terminal window titled 'iamsdr@kali: ~'. The window shows the execution of the command 'nikto -h https://pentesterlab.com/' and the resulting output. The output is formatted with dashed lines separating sections. It includes target information (IP, hostname, port), SSL certificate details (subject, altnames, ciphers, issuer), start time, and specific findings such as an uncommon header and a missing X-Content-Type-Options header.

```
iamsdr@kali: ~  
$ nikto -h https://pentesterlab.com/  
- Nikto v2.5.0  
-----  
+ Target IP: 54.87.134.91  
+ Target Hostname: pentesterlab.com  
+ Target Port: 443  
-----  
+ SSL Info: Subject: /CN=pentesterlab.com  
            Altnames: pentesterlab.com, www.pentesterlab.com  
            Ciphers: ECDHE-RSA-AES256-GCM-SHA384  
            Issuer: /C=FR/O=Gandi SAS/CN=GandiCert  
+ Start Time: 2025-03-08 12:45:49 (GMT0)  
-----  
+ Server: Apache  
+ /: Uncommon header 'x-request-id' found, with contents: 1d5971bc-dfea-4dce-b0d  
d-f8483bae6a08.  
+ /UegNdcG5.show: The X-Content-Type-Options header is not set. This could allow  
the user agent to render the content of the site in a different fashion to the  
MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabili  
ties/missing-content-type-header/
```

LIST OF VULNERABILITIES :-

S.No	Vulnerability name	CWE No	Severity	Status	Plugin
1.	SQL Injection	CWE-89	High	Confirmed	SQLi Scanner
2.	Cross – Site Scripting (XSS)	CWE-79	Medium	Confirmed	XSS Detector
3.	Broken Authentication	CWE-287	High	Confirmed	Authentication Tester



```
iamsdr@kali: ~  
iamsdr@... × iamsdr@... × iamsdr@... × iamsdr@... × iamsdr@... ×  
(iamsdr@kali)-[~]  
$ nmap -V 54.87.134.91  
Nmap version 7.95 ( https://nmap.org )  
Platform: x86_64-pc-linux-gnu  
Compiled with: liblua-5.4.7 openssl-3.4.1 libssh2-1.11.1 libz-1.3.1 libpcap-1.10.5 nmap-libdnet-1.12 ipv6  
Compiled without:  
Available nsock engines: epoll poll select  
  
(iamsdr@kali)-[~]  
$  
  
(iamsdr@kali)-[~]  
$
```

Vulnerability Name	:- Improper Input Validation
CWE No	:- CWE-20
OWASP/SANS Category	:- Top 10
Severity	:- High
Plug in	:- burbsuite active scanner,sonar QUBE
Port	:- 80 for http

DESCRIPTON :-

Improper Input Validation occurs when an application fails to adequately validate, sanitize, or restrict user inputs before processing them. This vulnerability is classified under CWE-20 (Improper Input Validation) and is listed in both the OWASP Top 10 and SANS Top 25 Most Dangerous Software Errors as a significant security risk. When input validation is not properly enforced, attackers can exploit it to inject malicious data, manipulate application logic, or cause unexpected system behavior. This can lead to various attacks such as SQL Injection, Cross-Site Scripting (XSS), Command Injection, Path Traversal, Buffer Overflows, and Denial-of-Service (DoS) attacks.

Improper Input Validation is particularly dangerous in applications that process user inputs for database queries, file handling, authentication, or system commands. For example, if a web application accepts an email address as input but does not verify its format, an attacker could inject special characters or malicious scripts that manipulate backend processes. Similarly, in software that processes numerical inputs, failing to validate expected ranges may lead to integer overflows or unexpected crashes. Attackers can also exploit this flaw to bypass authentication mechanisms by entering unexpected values that trick the system into granting unauthorized access.

BUSINESS IMPACTS :-

- The impact of Improper Input Validation can be severe, depending on the type of application and the nature of the attack.
- One of the most significant risks is data breaches, where attackers exploit input validation flaws to gain unauthorized access to sensitive information such as customer records, financial data, and authentication credentials.
- This can lead to regulatory violations under GDPR, PCI-DSS, and HIPAA, resulting in substantial fines and legal consequences.
- Another major impact is unauthorized access and privilege escalation, where attackers manipulate inputs to gain higher privileges within the system. For example, a flawed authentication process may allow users to bypass login restrictions, leading to administrative control over an application or database.
- This can be particularly damaging in financial, healthcare, or government systems that store confidential data.
- Improper Input Validation can also cause operational disruptions by enabling Denial-of-Service (DoS) attacks. Attackers may send excessively large or malformed inputs that consume system resources, leading to crashes or degraded performance.
- In software applications, failure to validate input lengths can result in buffer overflows, which may allow remote code execution, compromising the entire system.
- Beyond technical risks, brand reputation and customer trust can be severely impacted if an organization suffers from a security breach due to improper input validation. Users may lose confidence in the security of the platform, leading to customer attrition and revenue loss.
- Additionally, businesses may be subject to litigation and financial damages if security flaws result in data leaks or fraud.
- To mitigate these risks, organizations must implement strict input validation policies, using whitelisting, regular expressions, length restrictions, and escaping special characters. Additionally, applying secure coding practices, automated security testing,

and real-time threat monitoring can help detect and prevent exploitation of input validation flaws.

- By proactively securing input validation mechanisms, businesses can protect sensitive data, maintain compliance, and ensure the reliability and security of their applications.

Vulnerability Name	:- Path Traversal
CWE No	:- CWE-22
OWASP/SANS Category	:- Top-10
Severity	:- High
Plug in	:- burp suit path traversal,OWASP ZAP,NIKTO
Port	:- 80 for http

DESCRIPTION :-

Path Traversal, also known as Directory Traversal, is a security vulnerability that occurs when an application improperly restricts access to files and directories outside the intended directory structure. This flaw is classified under CWE-22 (Improper Limitation of a Pathname to a Restricted Directory - 'Path Traversal') and is listed in both the OWASP Top 10 and SANS Top 25 Most Dangerous Software Errors as a critical security risk.

Path Traversal vulnerabilities arise when user-supplied input, such as filenames or directory paths, is not properly validated before being processed by the application. Attackers can exploit this flaw by manipulating file paths using special characters like ../ (dot-dot-slash) to navigate outside of the restricted directory and access sensitive system files. For example, an attacker might enter ../etc/passwd into a vulnerable web application to access password hashes stored on a Linux system. In Windows environments, they could use ../../windows/system32/config/SAM to retrieve system configuration files.

This vulnerability is particularly dangerous in web applications, file upload/download mechanisms, and systems that allow users to specify file paths. If an application dynamically

constructs file paths based on user input without proper validation, attackers can read, modify, or delete critical system files, leading to serious security breaches. In some cases, Path Traversal can also be used to execute arbitrary code, especially if the attacker can write malicious scripts to sensitive locations within the system.

BUSINESS IMPACTS :-

- The consequences of a Path Traversal attack can be severe, affecting data confidentiality, system integrity, and overall business operations. One of the primary risks is unauthorized access to sensitive files, such as configuration files, user credentials, source code, or log files containing confidential information.
- If attackers retrieve authentication-related files, they can compromise user accounts, leading to identity theft or further system exploitation.
- Another significant impact is system compromise and privilege escalation. If attackers can access administrative files or system configurations, they may modify settings, escalate privileges, or even gain full control over the system.
- In cases where Path Traversal leads to arbitrary file execution, an attacker could deploy malware, ransomware, or backdoors to maintain persistent access to the system.
- Path Traversal can also lead to data corruption or loss. If an attacker gains write permissions, they may alter or delete important business files, disrupt services, or sabotage critical applications.
- In web applications that handle file uploads, Path Traversal vulnerabilities may allow attackers to overwrite system files or upload malicious scripts, leading to full server compromise.
- From a business perspective, the financial and reputational impact of a Path Traversal attack can be devastating. Organizations may face regulatory violations under GDPR, PCI-DSS, HIPAA, and other compliance frameworks if sensitive user data is exposed.
- Security breaches resulting from Path Traversal can lead to lawsuits, fines, and loss of customer trust, ultimately affecting revenue and brand reputation. Additionally,

operational downtime caused by system compromises or data loss can disrupt business continuity, resulting in financial losses and reduced productivity.

- To mitigate the risks associated with Path Traversal, organizations should implement strict input validation by restricting user-supplied file paths to predefined directories and disallowing special path sequences like ../ and ../. Applications should use secure APIs that do not rely on user input for file handling, enforce least privilege access, and implement logging and monitoring to detect suspicious activity
- Regular security testing, including automated vulnerability scans and penetration testing, can help identify and remediate Path Traversal vulnerabilities before they are exploited by attackers. By taking these proactive security measures, businesses can protect their critical assets, maintain compliance, and ensure the integrity of their systems.

Vulnerability Name	:- Integer Overflow or Wraparound
CWE No	:- CWE-190
OWASP/SANS Category	:- Top-10
Severity	:- High
Plug in	:- code QL,valgrand burb suite scanner
Port	:- 80 for http

DESCRIPTION:-

Integer Overflow or Wraparound is a security vulnerability that occurs when an arithmetic operation results in a numerical value exceeding the maximum limit that a variable can store. This causes the value to "wrap around" to a much lower or unintended value, leading to unpredictable behavior in software applications. This vulnerability is classified under CWE-190 (Integer Overflow or Wraparound) and is recognized in both the OWASP Top 10 and SANS Top 25 Most Dangerous Software Errors as a critical issue that can lead to severe security flaws.

Integer Overflow happens when an operation—such as addition, multiplication, or incrementing a counter—produces a result larger than the data type can handle. For example, in a 32-bit signed integer, the maximum value is 2,147,483,647. If an application adds 1 to this value, instead of increasing beyond the limit, the number wraps around to -2,147,483,648, causing unintended behavior. Similarly, Integer Underflow occurs when subtracting a value causes the number to wrap around in the opposite direction.

This vulnerability is especially dangerous in applications that handle memory allocation, financial transactions, cryptographic functions, or authentication mechanisms. Attackers can exploit Integer Overflow to manipulate program logic, bypass security checks, cause memory corruption (such as buffer overflows), or escalate privileges. For example, if a system uses an integer-based counter for user authentication and fails to check for overflows, an attacker might force the counter to wrap around, gaining unauthorized access. In memory allocation scenarios, an integer overflow can lead to improperly sized memory buffers, which attackers can exploit to execute arbitrary code or cause a system crash.

BUSINESS IMPACTS :-

- Integer Overflow or Wraparound can have significant consequences for businesses, affecting security, financial stability, and system reliability.
- One of the primary risks is unauthorized access and privilege escalation, where attackers exploit overflows to bypass security checks and gain elevated system privileges.
- This can lead to data breaches, unauthorized system modifications, and administrative control over critical infrastructure.
- Another major impact is financial fraud and incorrect calculations in applications that process transactions, pricing, or financial data.
- If an attacker manipulates integer values in an e-commerce or banking application, they could exploit price calculations, receive unauthorized discounts, or withdraw excessive funds. Inaccurate financial data can lead to regulatory violations, accounting discrepancies, and potential legal liabilities.

- Integer Overflows are also a common cause of memory corruption vulnerabilities, such as buffer overflows, which attackers can leverage to execute remote code execution (RCE) attacks. This could allow attackers to install malware, ransomware, or backdoors, leading to a full system compromise.
- Systems that rely on secure cryptographic operations are also at risk; an integer overflow in encryption algorithms could weaken security mechanisms, making it easier for attackers to decrypt sensitive data.
- From a business perspective, the operational and reputational damage caused by Integer Overflow attacks can be severe. Organizations may face regulatory fines under GDPR, PCI-DSS, HIPAA, or SOX if customer data is exposed due to a breach.
- The cost of incident response, forensic investigations, and system recovery can be high, alongside potential legal actions from affected customers or partners. Moreover, downtime and system instability caused by unexpected crashes or corruption can lead to loss of productivity, service disruptions, and revenue loss.
- To mitigate Integer Overflow risks, organizations should implement strict input validation, boundary checks, and safe arithmetic operations.
- Developers should use secure coding practices, such as checking for integer limits before performing operations and using safe integer libraries that prevent overflow conditions.
- Enforcing compiler protections, security audits, and automated vulnerability scanning can help detect and remediate Integer Overflow vulnerabilities before attackers exploit them. By proactively addressing these risks, businesses can enhance system security, ensure data integrity, and maintain trust with customers and stakeholders.