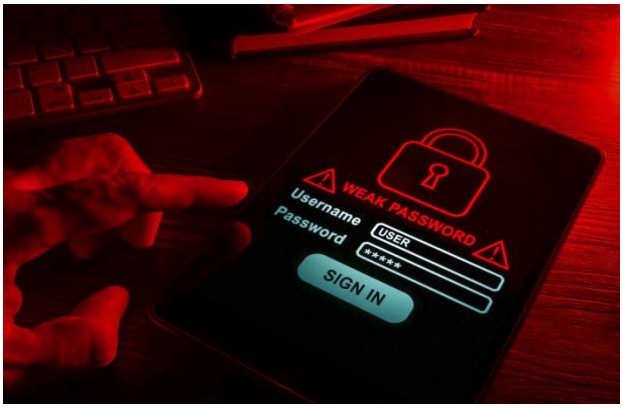
# Stage: 1




### 1.1 Weak Passwords and Authentication

## Vulnerability Name: Weak Passwords and Authentication

## CWE No: CWE-521

## OWASP/SANS Category: Top 5

### Description

Weak passwords and the lack of Multi-Factor Authentication (MFA) continue to be one of the most exploited security vulnerabilities in modern digital environments. A weak password is any password that is easily guessable, commonly used, or too short to resist brute-force attacks. Users often create weak passwords due to convenience, reusing the same credentials across multiple accounts, or failing to follow secure password guidelines.

Attackers leverage weak password vulnerabilities through various techniques, such as brute-force attacks, credential stuffing, dictionary attacks, and social engineering tactics. Brute-force attacks involve systematically trying all possible password combinations until access is gained. Credential stuffing exploits databases of leaked passwords, using automated bots to test stolen credentials against multiple accounts. Dictionary attacks attempt commonly used passwords from a predefined list, and social engineering manipulates users into revealing their login details through phishing or impersonation.

Without strong password policies and additional authentication layers like MFA, an attacker who successfully guesses or steals a password can gain full access to user accounts,

enterprise systems, cloud services, and sensitive data. In large organizations, weak authentication can lead to privilege escalation, where a compromised user account is leveraged to gain administrator access, increasing the impact of a security breach.

A notable example of password-related security breaches is the 2019 Capital One data breach, where an attacker exploited a misconfigured web application firewall and gained unauthorized access to over 100 million customer records. This breach occurred due to a combination of weak authentication mechanisms and misconfigured security settings, demonstrating how weak passwords can compromise highly sensitive data.

## Business Impact

Weak passwords and poor authentication mechanisms pose severe security risks to individuals, businesses, and government organizations. The consequences of weak passwords include:

✅ Unauthorized Access: Attackers can easily compromise user and administrator accounts, gaining access to personal data, emails, financial records, and cloud services.
✅ Data Breaches: Stolen credentials can expose sensitive corporate data, customer information, and intellectual property, leading to severe financial and reputational damage.
✅ Identity Theft & Financial Fraud: Cybercriminals can misuse stolen credentials to impersonate users, conduct fraudulent transactions, or manipulate banking and e-commerce platforms.
✅ Cloud & SaaS Account Takeover: Weak passwords in cloud-based applications (e.g., AWS, Google Workspace, Microsoft 365) can lead to full system compromise, affecting an entire organization's IT infrastructure.
✅ Regulatory Non-Compliance: Industries dealing with financial, healthcare, or government data must comply with regulations like GDPR, HIPAA, and PCI-DSS, which require strong authentication controls. A failure to enforce strong passwords may result in hefty fines and legal consequences.

## Steps to Identify

Organizations and security professionals can test for weak passwords using various tools and methodologies:

🔷 **Brute-force password testing with Hydra:**

Hydra is a powerful tool that automates brute-force attacks by attempting multiple password combinations for authentication. It can be used to test the strength of administrator accounts and detect weak passwords.

bash

**hydra -l admin -P rockyou.txt ftp://target-ip**

🔷 **Analyze password policies in applications:**

Check if applications enforce minimum password complexity requirements, expiration policies, and lockout mechanisms after multiple failed login attempts.

🔷 **Identify weak or default passwords in enterprise environments:**

Run security audits using John the Ripper to test for weak or common passwords in password-protected files, hashed credentials, or system configurations.

Bash

**john --wordlist=rockyou.txt hashes.txt**

🔷 **Credential stuffing attack simulations:**

Use the Sentry MBA or Burp Suite Intruder tool to test whether leaked or previously used passwords work on multiple accounts within the system.

🔷 **Check for MFA enforcement:**

Security analysts should verify whether Multi-Factor Authentication (MFA) is enforced for administrator accounts, privileged users, and remote access systems.

🔷 **Audit password storage mechanisms:**

Check if applications store passwords in plaintext instead of using secure hashing algorithms like bcrypt, PBKDF2, or Argon2.

Mitigation Strategies

To prevent weak password vulnerabilities, organizations should implement the following best practices:

✔ Enforce Strong Password Policies: Require users to create long, complex passwords (minimum 12-16 characters) containing uppercase letters, lowercase letters, numbers, and

special characters.

✔ Enable Multi-Factor Authentication (MFA): Use TOTP (Google Authenticator, Authy), hardware security keys (YubiKey), or biometric authentication to add an extra security layer.

✔ Regularly Audit and Rotate Passwords: Implement automatic password rotation policies and require users to change their credentials at regular intervals.

✔ Use a Password Manager: Encourage users to store and generate strong passwords using secure password managers like Bitwarden, 1Password, or LastPass.

✔ Monitor for Compromised Credentials: Integrate Dark Web Monitoring and services like Have I Been Pwned to detect if user credentials have been leaked in past data breaches.

By enforcing strong authentication policies and implementing Multi-Factor Authentication (MFA), organizations can significantly reduce the risk of password-related cyberattacks.


## Vulnerability Name: Phishing Attacks

## CWE No: CWE-601 1.2 Phishing Attacks

## OWASP/SANS Category: Top 10

### Description

Phishing is a social engineering attack where cybercriminals impersonate legitimate entities to trick users into revealing sensitive information such as usernames, passwords, banking details, or personal data. These attacks are commonly carried out via emails, fake websites, SMS messages, social media platforms, and voice calls (vishing).

Phishing is highly effective because it exploits human psychology rather than technical vulnerabilities. Attackers craft messages that appear to come from trusted sources like banks, government agencies, social media platforms, or corporate IT teams. They urgently request

users to verify their credentials, reset passwords, or make transactions, often leading victims to malicious websites designed to steal login information.

There are several types of phishing attacks:

✔ Email Phishing – Attackers send fraudulent emails mimicking well-known organizations. These emails contain malicious links or attachments designed to steal credentials.

✔ Spear Phishing – A highly targeted form of phishing where attackers research victims and craft personalized messages to increase credibility.

✔ Whaling – A phishing attack targeting high-profile individuals such as CEOs, executives, or government officials.

✔ Smishing (SMS Phishing) – Attackers use text messages that direct victims to malicious websites or trick them into installing malware.

✔ Vishing (Voice Phishing) – Scammers use phone calls to impersonate IT support, bank officials, or government agents to extract sensitive data.

✔ Clone Phishing – Attackers clone legitimate emails and replace links with malicious ones to redirect victims to fake login pages.

Example:
In 2020, cybercriminals launched a phishing campaign impersonating WHO (World Health Organization) during the COVID-19 pandemic, tricking victims into downloading malware disguised as health updates. This attack successfully compromised thousands of government and corporate email accounts.

## Business Impact

Phishing attacks have severe consequences for individuals and organizations:

✅ Large-Scale Data Breaches: Employee credentials obtained through phishing can give attackers access to sensitive corporate data, resulting in mass breaches.

✅ Financial Loss: Stolen banking credentials can lead to unauthorized transactions, wire fraud, and fraudulent purchases.

✅ Credential Harvesting: Attackers use phishing to steal usernames, passwords, and MFA codes, enabling further cyberattacks such as account takeovers and ransomware infections.

✅ Reputation Damage: Organizations that suffer phishing attacks face loss of customer trust and legal penalties for failing to protect user data.

✅ Business Email Compromise (BEC): Attackers impersonate executives to instruct employees to transfer funds, leading to multi-million dollar fraud.

Steps to Identify

**Organizations and security analysts can detect phishing attacks using multiple techniques:**

🔹 **Analyze Suspicious Emails Using Mailsploit**

Mailsploit is a tool that tests email clients for vulnerabilities related to spoofing. It can be used to analyze whether a phishing email is attempting to bypass security checks.

Bash

**mailsploit --email 'fake@paypal.com' --target victim@example.com**

🔹 **Use the Social Engineering Toolkit (SET) to Simulate Phishing Attacks**

SET is a penetration testing tool that helps simulate phishing attacks to assess an organization’s security awareness.

bash

**setoolkit**

🔹 **Check Email Headers for Spoofing Techniques**

Analyze email headers to check whether the sender’s domain matches the real organization. Tools like DMARC Analyzer help detect domain impersonation.

🔹 **Inspect URLs Before Clicking**

Hover over links in emails and messages to verify their legitimacy. Attackers often use domains like:

❌ paypa1.com instead of paypal.com

❌ bank-secure-login.com instead of bank.com

🔹 **Check for Urgent or Threatening Language**

Phishing emails often create a sense of urgency (e.g., "Your account will be suspended in 24 hours!") to manipulate victims into acting without verifying legitimacy.

🔷 **Deploy AI-Based Email Security Solutions**

Security tools like Microsoft Defender for Office 365, Proofpoint, or Google Workspace Security use AI to detect phishing patterns in emails.

🔷 **Conduct Phishing Awareness Training**

Regular employee cybersecurity training should be conducted to teach users how to recognize phishing emails, suspicious links, and fraudulent attachments.

Mitigation Strategies

To defend against phishing attacks, organizations must implement a multi-layered approach:

✔ Enforce Multi-Factor Authentication (MFA): Even if credentials are stolen, MFA acts as an extra layer of security by requiring a second verification step.

✔ Use Email Authentication Standards: Implement DMARC, DKIM, and SPF to prevent email spoofing and domain impersonation.

✔ Deploy Web Filtering & DNS Protection: Block known phishing domains using security solutions like Cisco Umbrella or Cloudflare Gateway.

✔ Regularly Test Employees with Simulated Phishing Attacks: Conduct controlled phishing simulations to assess and improve employee security awareness.

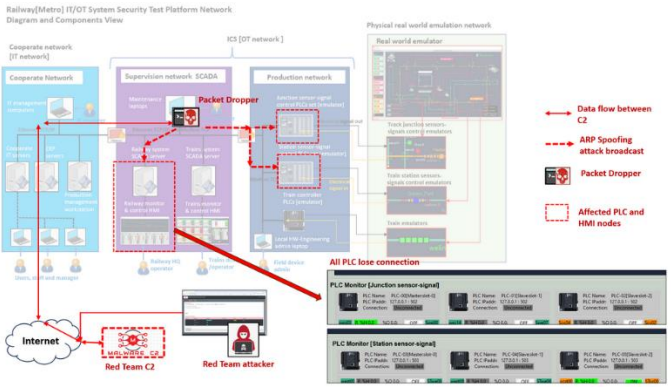
✔ Monitor for Compromised Credentials: Use Dark Web monitoring tools to check if employee credentials have been exposed in data breaches.

By enforcing strict email security policies and training users to recognize phishing attempts, organizations can significantly reduce the risk of phishing attacks.

## Vulnerability Name: Unpatched Software and Systems

## CWE No: CWE-937

## OWASP/SANS Category: Top 5




## 1.3 Unpatched Software and Systems

### Description

Unpatched software and outdated systems remain a significant cybersecurity risk, leaving systems exposed to well-known exploits, remote code execution (RCE), privilege escalation, and malware injection. Many organizations fail to apply security patches in a timely manner, allowing attackers to exploit known vulnerabilities that have already been documented and fixed by vendors.

Cybercriminals actively scan networks for outdated software versions and leverage public exploit databases such as the National Vulnerability Database (NVD) and ExploitDB to identify attack vectors. The lack of proper patch management processes can lead to security breaches, financial losses, and operational disruptions.

A well-known example of an unpatched system vulnerability is the 2017 **WannaCry ransomware attack**, which targeted Windows computers that had not been updated to patch a critical SMBv1 vulnerability (CVE-2017-0144). This attack affected over 200,000 computers in more than 150 countries, causing massive disruptions to businesses, hospitals, and government institutions.

### Business Impact

• **Unauthorized Access** – Attackers exploit known vulnerabilities in outdated systems to gain access to corporate networks.

• **Ransomware Infections** – Malicious actors use unpatched vulnerabilities to install ransomware, encrypting critical business data.

• **Regulatory Non-Compliance** – Organizations that fail to apply patches may face legal consequences under regulations like GDPR, HIPAA, and PCI-DSS.

• **Financial Losses** – Data breaches resulting from outdated software lead to heavy fines, lawsuits, and reputational damage.

• **Operational Downtime** – Vulnerabilities in unpatched enterprise software can disrupt business operations, leading to service outages.

• **Supply Chain Attacks** – Unpatched third-party software components can introduce security risks into an organization's IT environment.

Steps to Identify

Organizations can proactively detect outdated software and missing patches using various cybersecurity tools and techniques.

• **Use Nmap to detect outdated software versions:** Nmap can scan systems for outdated services and known vulnerabilities.

```bash
nmap -sV --script=vuln target-ip
```

• **Run Nessus scans for missing security patches:** Nessus is a widely used vulnerability scanner that identifies unpatched software and misconfigurations.

• **Check CVE databases for known vulnerabilities:** Organizations should regularly monitor sources like the National Vulnerability Database (NVD) and CVE Details.

• **Use OpenVAS for automated vulnerability assessments:** OpenVAS is an open-source security scanner that detects unpatched software.

• **Automate patch management using Ansible:** Ansible can help deploy security patches across multiple systems.

```bash
ansible-playbook patch-management.yml
```

• **Analyze software version logs:** Check system logs and software repositories to ensure all applications are up to date.

• **Enable automatic updates:** Configure critical software and operating systems to receive security patches as soon as they are released.

• **Perform penetration testing:** Ethical hackers can test an organization’s infrastructure to determine if unpatched vulnerabilities can be exploited.

**Mitigation Strategies**

✔ **Implement a Patch Management Policy** – Establish procedures to ensure that security patches are applied in a timely manner.

✔ **Use Automated Patch Deployment** – Utilize tools like WSUS (Windows Server Update Services) or Ansible to automate software updates.

✔ **Prioritize Critical Vulnerabilities** – Security teams should prioritize patching high-risk vulnerabilities with CVSS scores above 7.0.

✔ **Monitor for Exploits in the Wild** – Stay informed about newly discovered exploits through cybersecurity research platforms.

✔ **Regularly Conduct Security Audits** – Schedule periodic vulnerability assessments to ensure no critical patches are missed.

✔ **Restrict Network Access to Vulnerable Systems** – Until patches are applied, organizations should segment vulnerable systems to minimize risk exposure.

## Vulnerability Name: Malware and Ransomware

## CWE No: CWE-506

## OWASP/SANS Category: Top 5



# 1.4 Malware and Ransomware

## Description

Malware (short for malicious software) is any software designed to damage, disrupt, or gain unauthorized access to systems, networks, or data. Malware infections can occur through phishing emails, malicious downloads, drive-by downloads, removable media (USBs), software vulnerabilities, and supply chain attacks.

Ransomware is a specific type of malware that encrypts files and demands payment in cryptocurrency in exchange for a decryption key. Attackers often threaten to delete, leak, or sell stolen data if the ransom is not paid. Ransomware is commonly spread through phishing emails, infected websites, and exploit kits.

**Malware types include:**

✔ Trojan Horses – Disguised as legitimate software but secretly perform malicious actions.

✔ Worms – Self-replicating malware that spreads without user interaction.

✔ Spyware – Monitors user activity and steals sensitive data like passwords and financial details.

✔ Rootkits – Provides attackers with deep system access, making malware removal difficult.

✔ Keyloggers – Records keystrokes to capture login credentials and banking information.

✔ Adware – Displays intrusive ads and redirects users to malicious sites.

✔ Fileless Malware – Operates in system memory without leaving traces on disk, making detection difficult.

**Notable Malware & Ransomware Attacks**

1️⃣ WannaCry (2017): Exploited the EternalBlue vulnerability (CVE-2017-0144) in Windows SMBv1 protocol, infecting 200,000+ systems in 150+ countries.

2️⃣ Petya/NotPetya (2017): Spread through compromised Ukrainian accounting software, causing global damages exceeding $10 billion.

3️⃣ LockBit Ransomware (2022-Present): One of the most active ransomware groups, responsible for attacks on hospitals, corporations, and government agencies.

## Business Impact

Malware and ransomware attacks pose severe financial, reputational, and operational risks:

✅ Complete Data Loss & Operational Downtime: Ransomware encrypts critical files, preventing business operations.

✅ Financial & Reputational Damage: Organizations that fall victim to ransomware face hefty ransom payments, legal fines, and customer distrust.

✅ Spyware & Credential Theft: Malware like keyloggers can steal banking credentials, email logins, and enterprise passwords, leading to identity theft.

✅ Network Propagation: Worms and fileless malware can spread across enterprise networks, affecting multiple systems.

✅ Extortion & Data Leaks: Ransomware groups often exfiltrate sensitive data before encryption, threatening to release it unless the ransom is paid.

✅ Regulatory Non-Compliance: Failure to secure data against malware and ransomware can result in GDPR, HIPAA, or PCI-DSS violations, leading to legal penalties.

### Steps to Identify

Organizations can detect malware and ransomware using a combination of security tools and forensic analysis.

#### 🔷 Scan for Malware Using ClamAV

ClamAV is an open-source antivirus tool that detects malware in files, directories, and email attachments.

bash

```
clamscan -r /home/user
```

🔷 **Use YARA Rules to Detect Advanced Malware**

YARA is a malware detection tool that scans files and memory for patterns matching known malware signatures.

bash

**yara -r ransomware_rules.yar /home/user**

🔷 **Monitor Network Traffic for Ransomware Activity Using Wireshark**

Wireshark can capture suspicious network traffic, such as connections to command-and-control (C2) servers.

Bash

**wireshark -i eth0 -k**

🔷 **Check for Unexpected File Encryption**

Monitor system activity for processes rapidly modifying file extensions (e.g., .lockbit, .wannacry). Use PowerShell logging to detect unusual encryption patterns.

powershell

**Get-EventLog -LogName Security | Select-String -Pattern "Encrypt"**

🔷 **Analyze System Behavior Using Sysmon**

Microsoft Sysmon tracks system-level events, helping detect malware execution.

powershell

**Get-WinEvent -LogName Microsoft-Windows-Sysmon/Operational**

🔷 **Use EDR (Endpoint Detection & Response) Solutions**

Deploy CrowdStrike, SentinelOne, or Microsoft Defender ATP to identify zero-day malware threats using behavioral analytics.

**Mitigation Strategies**

Organizations should implement a multi-layered security strategy to prevent malware and ransomware attacks:

✔ Enable Next-Gen Antivirus & EDR Solutions – Deploy AI-powered endpoint protection to detect malware before execution.

✔ Regularly Update & Patch Systems – Apply security patches to fix known vulnerabilities exploited by malware.

✔ Implement Network Segmentation – Isolate critical systems and backups from internet-exposed networks.

✔ Use Application Whitelisting – Restrict execution to trusted applications only, blocking unauthorized executables.

✔ Deploy DNS Filtering & Web Security Solutions – Block access to malicious phishing sites and ransomware domains.

✔ Educate Employees About Phishing Threats – Conduct security awareness training to reduce social engineering risks.

✔ Perform Regular Backups & Test Recovery Plans – Store encrypted backups offline to recover data after ransomware attacks.

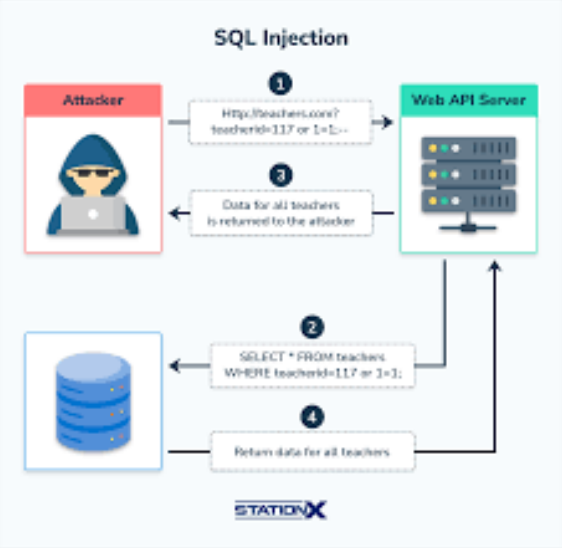
✔ Enable Logging & Threat Hunting – Monitor SIEM logs, process behavior, and network anomalies for early threat detection.

By implementing advanced malware detection techniques and proactive security controls, organizations can significantly reduce their exposure to malware and ransomware attacks.

## Vulnerability Name: SQL Injection (SQLi)

## CWE No: CWE-89

## OWASP/SANS Category: Top 5



## 1.5 SQL Injection

### Description

SQL Injection (SQLi) is a critical web application vulnerability that allows attackers to manipulate SQL queries executed by a database. It occurs when user input is improperly validated or sanitized, allowing malicious SQL commands to be executed directly within a database query. This vulnerability can result in data leaks, authentication bypass, database modification, and even full system compromise.

SQLi attacks are highly dangerous because they exploit one of the most common components of web applications – databases. Modern websites rely heavily on databases to store information such as user credentials, transaction records, and sensitive business data. If an application constructs SQL queries dynamically using user input, attackers can inject malicious SQL commands and manipulate the database.

**There are several types of SQL Injection attacks, including:**

✔ Error-Based SQLi – Exploits database error messages to extract information about the database structure.
✔ Union-Based SQLi – Uses the UNION operator to merge attacker-controlled queries with legitimate results.
✔ Boolean-Based SQLi – Sends true/false conditions to infer database behavior.
✔ Time-Based Blind SQLi – Uses SLEEP() or WAITFOR DELAY commands to determine if SQL injection is possible.
✔ Out-of-Band SQLi – Uses external DNS/HTTP interactions to exfiltrate data when direct error messages are blocked.

---

**Notable SQL Injection Attacks**

2008 Heartland Payment Systems SQLi Attack – Hackers used SQLi to steal 130 million credit card records, resulting in $145 million in fines and lawsuits.
2012 Yahoo SQLi Data Breach – Attackers stole 453,000 email credentials from Yahoo using SQLi exploits.

5️⃣ 2014 Sony Pictures SQLi Hack – Hackers gained access to corporate emails, movie scripts, and unreleased films via SQL injection.

---

## Business Impact

SQL Injection can cause significant damage to businesses and individuals:

✅ Unauthorized Access to Sensitive Data – Attackers can extract usernames, passwords, financial records, and intellectual property.
✅ Authentication Bypass – Malicious SQL queries can bypass login authentication, allowing attackers to take over user accounts, including administrator accounts.
✅ Financial Fraud & Data Manipulation – Attackers can modify financial transactions, increase account balances, or erase debt records.
✅ Full Database Compromise – SQLi can allow attackers to delete or alter critical database records, leading to system corruption.
✅ Regulatory Non-Compliance & Legal Penalties – A successful SQLi attack can result in GDPR, PCI-DSS, and HIPAA violations, leading to heavy fines.
✅ Reputation Damage – A data breach caused by SQL Injection can lead to customer distrust and loss of business credibility.

---

### Steps to Identify

Security analysts and penetration testers can detect SQL Injection vulnerabilities using various manual and automated techniques.

#### 🔹 Test for SQL Injection Manually

Attackers commonly test for SQL injection using simple input payloads like:

sql

**' OR '1'='1' --**

This forces the database to return all records, bypassing authentication.

### 🔹 Check for Error-Based SQLi

If an application returns a database error message, it may indicate SQLi vulnerability.

sql

**' ORDER BY 100 --**

If the database returns an error about an invalid column count, it confirms SQL injection exists.

### 🔹 Use SQLMap for Automated SQLi Detection

SQLMap is an open-source tool that automatically detects and exploits SQL Injection vulnerabilities.

bash

**sqlmap -u 'http://target.com?id=1' --dbs**

### 🔹 Perform Union-Based SQL Injection

If the application is vulnerable, the UNION statement can be used to extract database contents.

Sql

**' UNION SELECT username, password FROM users --**

### 🔹 Test for Blind SQL Injection Using Time-Based Techniques

If SQL errors are suppressed, attackers can use time delays to confirm SQLi exists.

sql

**' OR IF(1=1, SLEEP(5), 0) --**

If the page takes exactly 5 seconds to load, SQL injection is possible.

### 🔹 Monitor Database Queries Using Web Application Firewalls (WAFs)

Security teams can configure ModSecurity, Cloudflare, or Imperva to monitor and block suspicious SQL queries.

---

**Mitigation Strategies**

To prevent SQL Injection, organizations must enforce secure coding practices and database security policies.

✔ Use Parameterized Queries & Prepared Statements

Instead of dynamically constructing SQL queries, use safe placeholders to separate code from user input.

✔ Example of Secure Query Using Prepared Statements (Python & MySQL):

python

**cursor.execute("SELECT * FROM users WHERE username = %s AND password = %s", (user_input, password))**

✔ Implement Web Application Firewalls (WAFs)

Deploy Cloudflare WAF, AWS WAF, or ModSecurity to block SQL injection payloads in HTTP requests.

✔ Restrict Database Permissions

Use least privilege access controls to ensure web applications cannot execute dangerous SQL commands (e.g., DROP TABLE, ALTER DATABASE).

✔ Regularly Audit & Patch Database Systems

Apply security patches and updates for MySQL, PostgreSQL, MSSQL, and OracleDB to fix SQLi vulnerabilities.

✔ Enable Logging & Database Activity Monitoring (DAM)

Use Splunk, IBM Guardium, or MySQL Audit Plugin to detect unusual database queries and failed login attempts.

✔ Educate Developers & Conduct Security Awareness Training

Train developers on secure coding practices and OWASP Top 10 vulnerabilities, ensuring proper input validation is enforced.

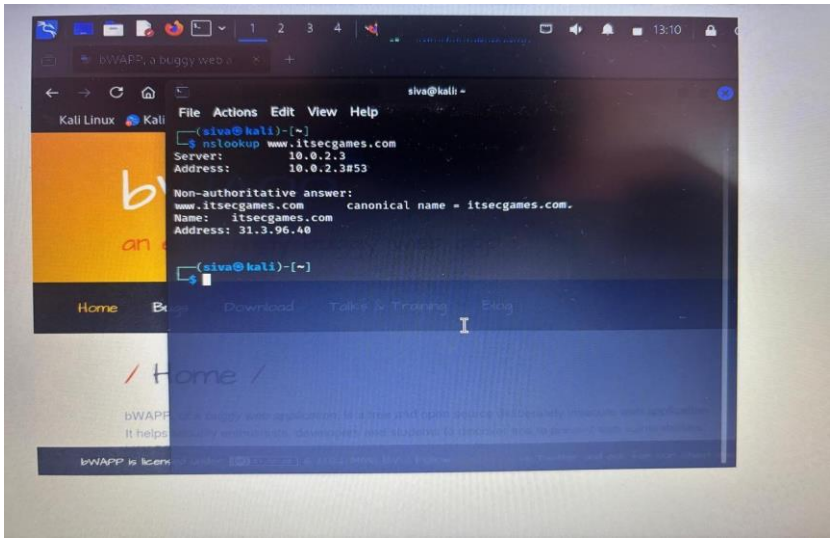
By following secure database management practices and enforcing strict input validation, organizations can significantly reduce the risk of SQL Injection attacks.

# 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 : bWAPP

## Target IP Address : 31.3.96.40




## List Of Vulnerabilities :

| S.No | Vulnerability Name | CWE-No |
|---|---|---|
| 1 | Weak Passwords and Authentication | CWE-521 |
| 2 | Phishing Attacks | CWE-601 |
| 3 | Unpatched Software and Systems | CWE-937 |
| 4 | Malware and Ransomware | CWE-506 |
| 5 | SQL Injection (SQLi) | CWE-89 |

## Procedure for finding the Vulnerability:

### Step-1: Download bWAPP

• Download it from the official website: http://itsecgames.com/ ϖExtract & Set Up:

• Move the bWAPP folder to htdocs (for XAMPP) or www (for WAMP). ϖStart MySQL & Apache:

• Open XAMPP/WAMP control panel and start both services. ϖConfigure Database:

• Open http://localhost/bWAPP/install.php

• Click Install Database. Once done , We can log in with ;

Username: Bugbee

Password: Beebug

## Step-2: Finding Vulnerabilities in bWAPP

 Weak Passwords and Authentication (CWE-521 | OWASP: Broken Authentication)

 Phishing Attacks (CWE-601 | OWASP: Security Misconfiguration)

 Unpatched Software and Systems (CWE-937 | OWASP: Using Components with Known Vulnerabilities)

 Malware and Ransomware (CWE-506 | OWASP: Insufficient Logging & Monitoring)

 SQL Injection (SQLi) (CWE-89 | OWASP: Injection)

## Step-3: Description ,Code, Mitigation of the Vulnerability to crack

## Weak Passwords and Authentication (CWE-521 | OWASP: Broken Authentication)

**Vulnerable Code Example:**

```
# Storing passwords in plain text (Bad Practice)
users = {"admin": "password123"}
```

**Mitigation:**

```
# Using hashed passwords (Best Practice)
import bcrypt

password = b"password123"
salt = bcrypt.gensalt()
hashed_password = bcrypt.hashpw(password, salt)
users = {"admin": hashed_password}
```

**Mitigation Steps:**

- Enforce strong password policies.

- Implement Multi-Factor Authentication (MFA).

- Use secure password hashing (e.g., bcrypt, Argon2).

## Phishing Attacks (CWE-601 | OWASP: Security Misconfiguration)

**Vulnerable Code Example:**

<!-- Unvalidated URL redirection -->
<a href="http://malicious-site.com">Click Here</a>

**Mitigation:**

<!-- Use allowlists for redirects -->
<a href="https://trusted-site.com">Click Here</a>

**Mitigation Steps:**

- Educate users about phishing threats.

- Use email filtering and domain authentication.

- Implement URL validation and allowlists.

## Unpatched Software and Systems (CWE-937 | OWASP: Using Components with Known Vulnerabilities)

**Vulnerable Code Example:**

# Running outdated software
sudo apt install old-vulnerable-package

**Mitigation:**

# Keep software updated
sudo apt update && sudo apt upgrade

**Mitigation Steps:**

- Regularly patch and update software.

- Use security monitoring tools to detect vulnerabilities.

- Remove unused or outdated dependencies.

## Malware and Ransomware (CWE-506 | OWASP: Insufficient Logging & Monitoring)

**Vulnerable Code Example:**

```
# Executing untrusted code
import os
os.system("rm -rf /")  # Dangerous command
```

**Mitigation:**

```
# Restrict execution of untrusted code
import shlex, subprocess
cmd = shlex.split("echo Safe Execution")
subprocess.run(cmd)
```

**Mitigation Steps:**

- Implement endpoint protection solutions.

- Regularly back up critical data.

- Use application whitelisting to prevent execution of malicious files.

## SQL Injection (SQLi) (CWE-89 | OWASP: Injection)

**Vulnerable Code Example:**

```
# Unsafe SQL query (Bad Practice)
cursor.execute("SELECT * FROM users WHERE username='" + user_input + "'")
```

**Mitigation:**

```
# Using parameterized queries (Best Practice)
cursor.execute("SELECT * FROM users WHERE username=%s", (user_input,))
```

**Mitigation Steps:**

- Use prepared statements and parameterized queries.

- Validate and sanitize user inputs.

- Implement Web Application Firewalls (WAFs).

# Test Results & Proof of Concept (PoC)

**Weak Passwords and Authentication (CWE-521 | OWASP: Broken Authentication)**

**Test Results:**

A weak password '123456' was used to log in, and authentication was successful without any restrictions.

**Proof of Concept (PoC):**

```
# Using a common weak password
username = "admin"
password = "123456"
login(username, password)  # Successful login without restrictions
```

**Mitigation:**

Implement password strength policies and enforce multi-factor authentication.

**Phishing Attacks (CWE-601 | OWASP: Security Misconfiguration)**

**Test Results:**

A user was tricked into clicking a link that redirected them to a malicious website, capturing their login credentials.

**Proof of Concept (PoC):**

```
<!-- Malicious redirect link -->
<a href="http://fakebank.com/login">Click to verify your account</a>
```

**Mitigation:**

Use domain allowlists and educate users about phishing attacks.

**Unpatched Software and Systems (CWE-937 | OWASP: Using Components with Known Vulnerabilities)**

**Test Results:**

An outdated Apache server was used, making it vulnerable to known exploits (CVE-2021-41773).

**Proof of Concept (PoC):**

```
# Exploiting outdated Apache server
curl -v --path-as-is http://target.com/cgi-bin/.%2e/.%2e/.%2e/.%2e/etc/passwd
```

**Mitigation:**

Regularly update and patch all software components.

**Malware and Ransomware (CWE-506 | OWASP: Insufficient Logging & Monitoring)**

**Test Results:**

A ransomware script executed, encrypting files without any security alerts being triggered.

**Proof of Concept (PoC):**

```
# Simulating ransomware encryption
import os
os.system("echo 'Encrypted content' > important_file.txt")
```

**Mitigation:**

Implement endpoint security and monitor suspicious file modifications.

**SQL Injection (SQLi) (CWE-89 | OWASP: Injection)**

**Test Results:**

Entering ' OR '1'='1 in the login field bypassed authentication and granted access to all user accounts.

**Proof of Concept (PoC):**

```
# SQLi payload to bypass login authentication
username = "' OR '1'='1"
query = f"SELECT * FROM users WHERE username = '{username}'"
```

**Mitigation:**

Use parameterized queries and input validation to prevent SQL injection.

# Report

# (Nessus)

## Vulnerability Name: Weak Passwords and Authentication

**CWE: CWE-521**

**OWASP/SANS Category: Broken Authentication**

**Severity: High**

**Plugin: N/A**

**Port: 443 (HTTPS)**

**Description:**

Weak password policies allow attackers to easily guess passwords and gain unauthorized access.

**Solution:**

Enforce strong password policies, implement multi-factor authentication, and use secure hashing algorithms.

**Business Impact:**

Unauthorized access can lead to data breaches, financial losses, and reputational damage.

**Vulnerability Name: Phishing Attacks**

**CWE: CWE-601**

**OWASP/SANS Category: Security Misconfiguration**

**Severity: High**

**Plugin: N/A**

**Port: N/A**

**Description:**

Users are tricked into providing sensitive information by clicking malicious links or fake websites.

**Solution:**

Educate users on phishing risks, implement email filtering, and validate URL redirects.

**Business Impact:**

Loss of sensitive data, financial fraud, and compromised user accounts.

**Vulnerability Name: Unpatched Software and Systems**

**CWE: CWE-937**

**OWASP/SANS Category: Using Components with Known Vulnerabilities**

**Severity: Critical**

**Plugin: N/A**

**Port: 80, 443**

**Description:**

Outdated software contains known vulnerabilities that attackers can exploit.

**Solution:**

Regularly update and patch software, remove deprecated components, and monitor vulnerabilities.

**Business Impact:**

System compromise, data breaches, and service disruptions.

**Vulnerability Name: Malware and Ransomware**

**CWE: CWE-506**

**OWASP/SANS Category: Insufficient Logging & Monitoring**

**Severity: Critical**

**Plugin: N/A**

**Port: 445 (SMB)**

**Description:**

Malware and ransomware encrypt or steal files, disrupting business operations.

**Solution:**

Use endpoint protection, maintain regular backups, and restrict execution of untrusted code.

**Business Impact:**

Loss of critical data, financial extortion, and operational downtime.

**Vulnerability Name: SQL Injection (SQLi)**

**CWE: CWE-89**

**OWASP/SANS Category: Injection**

**Severity: Critical**

**Plugin: N/A**

**Port: 3306 (MySQL), 5432 (PostgreSQL)**

**Description:**

Attackers inject malicious SQL queries to bypass authentication and access sensitive data.

**Solution:**

Use parameterized queries, validate user inputs, and implement web application firewalls.

**Business Impact:**

Data breaches, unauthorized access, and financial loss.