|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
|  |  |  | |  |
| PROJECT REPORT  cybersecurity Internship | | |
| EXTION INFOTECH | | NIBIL MATHEW |

**PROJECT 1**

**Vulnerability Assessment Report**

**Target System:** Metasploitable2 Virtual Machine  
**Assessment Date:** 20/01/25  
**Scanner Used:** OpenVAS  
**Report Prepared By:** Nibil Mathew

**1. Overview**

The security assessment of the **Metasploitable2** virtual machine was conducted using OpenVAS, which identified several critical vulnerabilities. The top five vulnerabilities are analyzed in this report, including their severity, impact, and step-by-step mitigation strategies.

**2. Critical Vulnerabilities Identified**

**1. TWiki XSS and Command Execution Vulnerabilities**

* **Severity:** 10.0 (High)
* **Description:** TWiki is vulnerable to Cross-Site Scripting (XSS) and command execution attacks. Attackers can execute arbitrary commands on the server, leading to complete system compromise.
* **Potential Impact:** Remote attackers can execute commands, steal credentials, and deploy malware.
* **Mitigation Steps:**
  1. **Disable TWiki** if not required.
  2. Update TWiki to the latest secure version.
  3. Restrict access to trusted users.
  4. Apply input sanitization to prevent XSS.
* **Estimated Remediation Time:** 2-3 hours
* **Required Resources:** System administrator, patch management tools.

**2. Distributed Ruby (dRuby/DRb) Multiple Remote Code Execution Vulnerabilities**

* **Severity:** 10.0 (High)
* **Description:** The dRuby (DRb) service allows remote code execution due to insecure configurations.
* **Potential Impact:** Attackers can execute arbitrary Ruby code remotely, leading to full system takeover.
* **Mitigation Steps:**
  1. Disable the dRuby service if it is not required (systemctl stop drb on modern systems).
  2. Configure access control to allow only trusted IPs.
  3. Apply firewall rules to block unauthorized access to port **8787/tcp**.
* **Estimated Remediation Time:** 1-2 hours
* **Required Resources:** System administrator, network firewall configuration.

**3. DistCC Remote Code Execution Vulnerability**

* **Severity:** 9.3 (High)
* **Description:** The **DistCC** service is running with no authentication, allowing attackers to execute arbitrary commands remotely.
* **Potential Impact:** Attackers can gain remote shell access and execute malicious payloads.
* **Mitigation Steps:**
  1. Disable DistCC service if unnecessary:

systemctl stop distccd

systemctl disable distccd

* 1. If needed, configure DistCC to **allow only trusted IPs**.
  2. Implement network segmentation to limit exposure.
  3. Monitor logs for unauthorized access attempts.
* **Estimated Remediation Time:** 2-3 hours
* **Required Resources:** System administrator, network engineer.

**4. MySQL / MariaDB Weak Password Vulnerability**

* **Severity:** 9.0 (High)
* **Description:** The MySQL/MariaDB service is using weak or default credentials, allowing attackers to gain unauthorized database access.
* **Potential Impact:** Attackers can steal or manipulate sensitive database records.
* **Mitigation Steps:**
  1. Change default MySQL root password immediately:

ALTER USER 'root'@'localhost' IDENTIFIED BY 'Strong\_Password';

* 1. Disable remote root login in MySQL config (/etc/mysql/my.cnf):

[mysqld]

skip-networking

* 1. Enable **account lockout policies** to prevent brute-force attacks.
  2. Apply **regular database updates** to patch vulnerabilities.
* **Estimated Remediation Time:** 1-2 hours
* **Required Resources:** Database administrator, security analyst.

**5. VNC Brute Force Login Vulnerability**

* **Severity:** 9.0 (High)
* **Description:** The VNC (Virtual Network Computing) service is susceptible to brute force attacks due to weak authentication mechanisms.
* **Potential Impact:** Attackers can gain remote desktop access, leading to full system compromise.
* **Mitigation Steps:**
  1. Disable VNC if not required:

systemctl stop vncserver

systemctl disable vncserver

* 1. Enable strong password authentication for VNC.
  2. Restrict access using firewall rules (iptables or ufw).
  3. Configure **fail2ban** to detect and block brute-force attempts.
  4. Use **SSH tunneling** for secure VNC connections instead of exposing it to the internet.
* **Estimated Remediation Time:** 2-3 hours
* **Required Resources:** System administrator, network security engineer.

**3. Conclusion & Recommendations**

The scan identified multiple high-severity vulnerabilities, most of which can lead to **remote code execution and unauthorized access**. To improve security, the following measures should be implemented:

1. **Immediate Actions:**
   * Disable or patch vulnerable services.
   * Apply network segmentation to restrict access to critical services.
   * Update all software and enforce strong authentication policies.
2. **Long-Term Security Enhancements:**
   * Regular security audits and vulnerability scans.
   * Deploy Web Application Firewalls (WAFs) to block exploitation attempts.
   * Implement a **Security Information and Event Management (SIEM)** system for real-time monitoring.

By following these mitigation steps, the security posture of the **Metasploitable2** virtual machine can be significantly improved, reducing the risk of exploitation.

**End of Report**

**PROJECT 2**

**Forensic Investigation Report**

**Incident Title:** Investigation of Data Breach at ABC SecureBank

**Investigation Date:** 20/01/25

**Report Prepared By:** Nibil Mathew

### **1. Incident Overview**

**Breach Discovery:** During a routine security audit on 20/01/25, the security team at ABC SecureBank identified unusual activity on the company’s web server. Further investigation revealed a data breach involving potential exposure of sensitive customer account information, including names, account numbers, and transaction history.

**Vulnerability Identified:** The breach was caused by an **SQL Injection (SQLi)** vulnerability in the website’s login portal. This is a common attack vector that allows malicious actors to manipulate SQL queries to gain unauthorized access to sensitive data.

### **2. Incident Analysis**

**Point of Entry:** The investigation revealed that the attacker exploited an SQL Injection vulnerability in the login endpoint:

* **Endpoint Affected:** /login
* **Payload Used:** ' OR '1'='1' --
* **Query Impacted:**

SELECT \* FROM users WHERE username = '[INPUT]' AND password = '[INPUT]';

* The above query was manipulated to bypass authentication checks and access the database directly.

**Extent of the Breach:**

* Customer data, including names, account numbers, and transaction history for approximately **10,000 users**, was exposed.
* The attacker exfiltrated data by executing unauthorized SELECT queries.

**Timeframe:**

* Breach occurred between **15/01/25** and **17/01/25**.
* Logs show the first unauthorized query execution on **2025-01-15 22:40:00.**.

### **3. Forensic Analysis**

**Digital Forensic Actions Taken:**

1. **Log Analysis:**
   * Examined web server logs, database query logs, and application logs.
   * Identified multiple unauthorized SELECT queries originating from IP address **192.168.1.10**.
   * Example log entry:

[2025-01-27 14:35:12] IP: 192.168.1.10, Query: SELECT \* FROM accounts WHERE 1=1 --

1. **Malware Detection:**
   * No malware was found on the web server.
   * Suspicious activity was purely query-based, indicating manual exploitation.
2. **Evidence Collection:**
   * Extracted database logs, HTTP access logs, and a memory dump of the web server.
   * Preserved all evidence in a tamper-proof format for legal proceedings.

### **4. Data Recovery and Containment**

**Data Exposed:**

* **Personal Identifiable Information (PII):** Names, account numbers.
* **Financial Information:** Transaction history for affected users.

**Recovery Plan:**

* Immediately patched the SQL Injection vulnerability by implementing parameterized queries.
* Reset credentials for all affected users.
* Implemented stricter access controls on the database.

**Incident Containment:**

* Isolated the compromised server from the network.
* Conducted a full security review of the application codebase.
* Updated Web Application Firewall (WAF) rules to detect and block SQL Injection attempts.

### **5. Regulatory Compliance**

**Legal and Regulatory Considerations:**

* Notified the regulatory body [Insert Name] within **72 hours** as per **GDPR** Article 33.
* Drafted and sent breach notification letters to all affected customers.
* Assured customers that steps were taken to mitigate the issue and prevent future breaches.

### **6. Communication and Notification**

**Internal Communication:**

* Informed senior management and key stakeholders about the breach.
* Conducted an internal briefing for employees to enhance vigilance and prevent similar incidents.

**Customer Notification:**

* Sent personalized notifications to affected customers with:
  + Details of the breach.
  + Steps taken by the bank.
  + Recommendations to reset passwords and monitor account activity.

**Media Statement:**

* Released a public statement acknowledging the breach and outlining measures taken to secure systems.

### **7. Post-Incident Review**

**Root Cause Analysis:**

* The SQL Injection vulnerability was due to insecure coding practices, specifically failing to validate and sanitize user inputs.

**Recommendations:**

1. Implement parameterized queries and stored procedures for all database interactions.
2. Conduct regular vulnerability assessments and penetration testing.
3. Deploy a robust Web Application Firewall (WAF).
4. Provide secure coding training to development teams.
5. Establish a 24/7 Security Operations Center (SOC) for continuous monitoring.

**Follow-Up Actions:**

* Conducted a company-wide security awareness training session.
* Engaged a third-party auditor to review and certify the updated security measures.

**Conclusion:** The breach at ABC SecureBank highlighted critical vulnerabilities in its web application security. Immediate action was taken to contain the breach, notify affected parties, and improve security controls. Moving forward, adherence to secure coding practices and continuous monitoring will be prioritized to prevent similar incidents.

**Attachments:**

1. Log files and evidence snapshots.
2. Communication templates.
3. Vulnerability assessment report.

**End of Report**