Penetration Testing

Report

Client: Kioptrix

Project: Penetration Testing on Kioptrix Level 1.3 Network

Report Date: May 30, 2023

Prepared By: Ifeanyi Moses

Tester: Ifeanyi Moses

Contact Information:
https://www.linkedin.com/in/ifeanyimoses/

CONFIDENTIALITY STATEMENT

This report and its contents are confidential and intended solely for the use of Kioptrix. Any unauthorized use or disclosure of this report, in whole or in part, is strictly prohibited. The findings, conclusions, and recommendations contained in this report are based on the results of a penetration testing exercise conducted by Ifeanyi Moses and should be treated as sensitive and confidential information. If you have received this report in error, please notify us immediately and destroy all copies of the report.

TABLE OF CONTENTS

EXECUTIVE SUMMARY

The purpose of this penetration testing report was to assess the security of the target system and identify any vulnerabilities or weaknesses that could potentially be exploited by attackers. The testing revealed several critical findings that require immediate attention to protect the system from potential threats.

The identified vulnerabilities include an outdated operating system, a weakness in the system's password field that allows unauthorized access, a security misconfiguration that allows root login without a password, a vulnerability in the Samba software, the presence of browsable web directories, and a lack of secure communication settings.

These vulnerabilities could potentially lead to unauthorized access to sensitive information, data breaches, and compromise of the system's integrity. It is important to understand that these vulnerabilities pose risks to the confidentiality, integrity, and availability of the system and its data.

To address these vulnerabilities, it is recommended to update the operating system to a supported version, implement strong security measures for user passwords, ensure proper authentication and access controls for system administration, apply necessary patches and updates to address the Samba vulnerability, secure the web directories to prevent unauthorized access, and enforce secure communication protocols.

It is crucial to understand that security is an ongoing process, and addressing these vulnerabilities is just the first step. Regular monitoring, updating, and proactive security practices are essential to maintain a strong defense against potential threats.

By implementing the recommended measures, the organization can significantly improve its security posture, reduce the risk of potential breaches, and protect its valuable data. It is important to prioritize these recommendations and allocate necessary resources to address the vulnerabilities promptly.

Overall, this penetration testing report serves as a valuable guide to enhance the security of the system, ensuring a safer environment for the organization, its systems, and its users.

1. TECHNICAL SUMMARY

1.1 Scope

During the testing phase, the scope was strictly limited to the target IP Address (192.168.73.131). The testing was performed in a controlled environment with the full knowledge and cooperation of Kioptrix's management.

1.2 Risk Ratings

To provide a clear and concise risk scoring system, the following risk naming and colors were used throughout this report. It's important to note that quantifying the overall business risk posed by the identified issues is beyond our scope. Therefore, some risks may be classified as high from a technical perspective but could be considered acceptable by the business due to other controls or factors unknown to us.

RISK LEVEL	DESCRIPTION	CVSSV3 SCORE
Low Risk	Indicates a vulnerability or issue with minimal impact that can be easily mitigated	**0.0 – 3.9**
Medium Risk	Indicates a vulnerability or issue with a moderate impact that requires more effort to address and mitigate	**4.0 – 6.9**
High Risk	Indicates a vulnerability or issue with a significant threat to security that requires immediate attention and remediation	**7.0 – 8.9**
Critical Risk	Indicates a vulnerability or issue with the highest severity and imminent threat that requires urgent action	**9.0 – 10.0**

1.3 Findings Overview

Below is a high-level overview of findings identified during testing. These findings are covered in depth in the Technical Details section of this report.

Finding #	Description	Risk
1	Ubuntu OS Unsupported Version	CRITICAL
2	SQL Injection Vulnerability in Password Field	HIGH
3	Security Misconfiguration (Root Login without Password in MySQL)	HIGH
4	Samba Badlock Vulnerability	HIGH
5	Browsable Web Directories	MEDIUM
6	SMB Signing not Required	MEDIUM

2. TECHNICAL DETAILS

2.1 Vulnerability Descriptions

2.1.1 Ubuntu OS Unsupported Version

CRITICAL

CRITICAL Unix Operating System Unsupported Version Detection

Description

According to its self-reported version number, the Unix operating system running on the remote host is no longer supported.

Lack of support implies that no new security patches for the product will be released by the vendor. As a result, it is likely to contain security vulnerabilities.

Solution

Upgrade to a version of the Unix operating system that is currently supported.

Output

```
Ubuntu 8.04 support ended on 2011-05-12 (Desktop) / 2013-05-09 (Server).
Upgrade to Ubuntu 21.04 / LTS 20.04 / LTS 18.04.

For more information, see : https://wiki.ubuntu.com/Releases
```

The penetration testing identified that the operating system (OS) running on the remote host is Ubuntu 8.04, which is no longer supported. The lack of support for the OS implies that it is likely to contain security vulnerabilities.

Ubuntu 8.04 was released in April 2008 and reached its end of life (EOL) in May 2013. This means that it no longer receives security updates or bug fixes from the Ubuntu developers, leaving the system vulnerable to known security flaws and exploits.

Using an unsupported operating system is a significant security risk, as it leaves the system vulnerable to attacks that could compromise sensitive data, disrupt business operations, or allow attackers to gain unauthorized access to the system. Attackers can exploit known vulnerabilities in the operating system to launch attacks such as remote code execution or privilege escalation.

Recommendation: Upgrade the Ubuntu OS to a supported version (Ubuntu 23.x) to ensure that security updates are received and vulnerabilities are patched in a timely manner.

2.1.2 SQL Injection Vulnerability in Password Field

HIGH

The application has a vulnerability that allows an attacker to perform SQL injection attacks through the password field of the login form. SQL injection occurs when untrusted input is directly incorporated into SQL queries without proper validation or sanitization. In this case, the password field lacks adequate input validation, allowing an attacker to inject malicious SQL code.

By exploiting this vulnerability, an attacker can manipulate the SQL query executed by the application, potentially bypassing authentication mechanisms and gaining unauthorized access to the system. The injected SQL code can alter the intended behavior of the query, leading to various malicious activities such as extracting sensitive information, modifying or deleting data, or executing arbitrary commands on the underlying database server.

The lack of proper input validation and sanitization in the password field allows an attacker to craft malicious input that is interpreted as part of the SQL query instead of a plain text password. The application then executes the modified query, unintentionally exposing the system to potential exploitation.

Recommendations: To mitigate the SQL injection vulnerability in the password field, the following recommendations should be implemented:

- ✓ Input Validation and Sanitization: Implement robust input validation and sanitization techniques to ensure that user-supplied data is properly validated and sanitized before being incorporated into SQL queries. Use parameterized queries or prepared statements to prevent the execution of injected SQL code.
- ✓ Least Privilege Principle: Apply the principle of least privilege by using dedicated database accounts with restricted permissions. Avoid using highly privileged accounts, such as the root or administrator accounts, for routine application operations.
- ✓ Security Testing: Regularly conduct comprehensive security testing, including vulnerability assessments and penetration testing, to identify and remediate SQL injection vulnerabilities. Use automated tools and manual testing techniques to validate the effectiveness of input validation and sanitization mechanisms.
- ✓ Patch and Update: Keep the application and its underlying components up to date with the latest security patches and updates. This helps protect against known vulnerabilities and ensures that any identified SQL injection vulnerabilities are addressed promptly.

2.1.3 Security Misconfiguration (Root Login without Password in MySQL) HIGH

The application has a security misconfiguration that allows the root user to log in to the MySQL database server without providing a password. This misconfiguration poses a significant security risk as it grants unrestricted access to the database server's most privileged account.

By failing to enforce a password requirement for the root user, the application exposes the database server to potential unauthorized access. Any attacker who can connect to the database server with the root user credentials can perform a wide range of malicious activities, including unauthorized data manipulation, information disclosure, and even complete compromise of the database server.

The absence of a password requirement for the root user is a severe misconfiguration that violates the principle of least privilege. It allows anyone with knowledge of the root user credentials to gain complete control over the database server, compromising the confidentiality, integrity, and availability of the data stored within.

Recommendations: To address the security misconfiguration of root login without a password in MySQL, the following recommendations should be implemented:

- ✓ Set a Strong Password: Immediately assign a strong, complex password to the root user account in MySQL. The password should meet industry best practices for password security, such as using a combination of uppercase and lowercase letters, numbers, and special characters.
- ✓ Restrict Root Access: Limit the root user's access to essential administrative tasks only. Avoid using the root account for routine application operations. Instead, create separate user accounts with appropriate privileges for day-to-day database activities.
- ✓ Apply Principle of Least Privilege: Follow the principle of least privilege by granting only the necessary permissions to user accounts. Restrict access to sensitive data and operations, ensuring that each user account has the minimum privileges required to perform their intended tasks.
- ✓ Regular Security Audits: Conduct regular security audits and vulnerability assessments to identify and address security misconfigurations promptly. Implement robust monitoring and log analysis mechanisms to detect any unauthorized access attempts or suspicious activities.
- ✓ Update Database Software: Keep the MySQL database software up to date with the latest security patches and updates. This ensures that any known security vulnerabilities or misconfigurations are addressed promptly.

2.1.4 Samba Badlock Vulnerability

HIGH

Description

The version of Samba, a CIFS/SMB server for Linux and Unix, running on the remote host is affected by a flaw, known as Badlock, that exists in the Security Account Manager (SAM) and Local Security Authority (Domain Policy) (LSAD) protocols due to improper authentication level negotiation over Remote Procedure Call (RPC) channels. A man-in-the-middle attacker who is able to able to intercept the traffic between a client and a server hosting a SAM database can exploit this flaw to force a downgrade of the authentication level, which allows the execution of arbitrary Samba network calls in the context of the intercepted user, such as viewing or modifying sensitive security data in the Active Directory (AD) database or disabling critical services.

Solution

Upgrade to Samba version 4.2.11 / 4.3.8 / 4.4.2 or later.

See Also

http://badlock.org
https://www.samba.org/samba/security/CVE-2016-2118.html

The Samba Badlock vulnerability is a security flaw that affects the Samba software, an open-source implementation of the Server Message Block (SMB) protocol used for file sharing and printer services in Windows-based networks. This vulnerability allows an attacker to perform a man-in-the-middle (MitM) attack and potentially gain unauthorized access to sensitive information.

The Badlock vulnerability arises due to an improper authentication security bypass in the way Samba handles certain authentication requests. By exploiting this vulnerability, an attacker can intercept and manipulate network traffic between Samba servers and clients, compromising the integrity and confidentiality of the data being transmitted.

An attacker leveraging the Badlock vulnerability can launch various malicious activities, including:

- ✓ Credential Theft: By intercepting authentication requests, the attacker can capture user credentials, such as usernames and passwords. These stolen credentials can be used for further unauthorized access to systems or sensitive resources.
- ✓ Man-in-the-Middle Attacks: The attacker can position themselves between the Samba server and client, allowing them to intercept and modify network traffic. This enables the attacker to manipulate data, inject malicious content, or redirect connections to malicious servers.
- ✓ Data Tampering: With control over the network traffic, the attacker can modify the data being transmitted between Samba servers and clients. This can lead to data tampering, unauthorized modifications, or even the insertion of malicious code or malware into legitimate network communications.
- ✓ Information Disclosure: The Badlock vulnerability can potentially expose sensitive information during the interception of network traffic. This includes confidential business data, intellectual property, personally identifiable information (PII), or any other sensitive information transmitted over the compromised Samba connections.

Recommendations: To mitigate the risks associated with the Samba Badlock vulnerability, the following recommendations should be followed:

- ✓ Patch and Update: Apply the latest security patches and updates provided by the Samba project or the respective operating system vendor. These updates typically address the Badlock vulnerability and enhance the overall security of the Samba software.
- ✓ Network Segmentation: Implement network segmentation to isolate critical systems and services from potentially compromised Samba instances. This helps contain the impact of any successful exploitation and limits the attacker's lateral movement within the network.
- ✓ Encryption and Secure Protocols: Implement encryption mechanisms, such as Transport Layer Security (TLS) or Secure Sockets Layer (SSL), to protect Samba communications. Enforce the use of secure protocols to ensure the confidentiality and integrity of data transmitted between Samba servers and clients.
- ✓ Strong Authentication: Implement strong and secure authentication mechanisms for Samba, such as multifactor authentication (MFA) or Kerberos, to enhance the overall security of the authentication process and mitigate the risk of credential theft

2.1.5 Browsable Web Directories

MEDIUM

Browsable web directories refer to directories on a web server that are accessible and viewable by anyone with a web browser. These directories often contain sensitive information, such as configuration files, source code, backup files, or other confidential data that should not be publicly accessible. The vulnerability arises when web directories are not properly secured and configured, allowing unauthorized individuals to browse and access the directory contents.

Recommendations: To mitigate the risks associated with browsable web directories, the following recommendations should be followed:

- ✓ Disable Directory Browsing: Ensure that directory browsing is disabled on the web server. This can be achieved by configuring the server settings or using appropriate directives in the web server configuration files.
- ✓ Secure Directory Permissions: Review and adjust directory permissions to restrict access only to authorized users or system processes. Follow the principle of least privilege and ensure that sensitive directories and files have appropriate access controls in place.
- ✓ Hide Sensitive Files: Move sensitive files, such as configuration files or backup archives, outside the web root directory. This prevents direct access through the web server and reduces the risk of inadvertent exposure.

2.1.6 SMB Signing not Required

MEDIUM

SMB (Server Message Block) signing is a security feature in the Microsoft Windows operating system that provides data integrity and authenticity for communications between SMB clients and servers. When SMB signing is not required or not enabled, it allows for potential security vulnerabilities and exposes the system to various risks.

SMB signing ensures that all data transferred between the client and server is digitally signed, preventing tampering or unauthorized modifications. When SMB signing is not enforced, an attacker can launch various attacks, including:

- ✓ Man-in-the-Middle Attacks: Without SMB signing, an attacker can intercept SMB communications between the client and server. By intercepting and modifying the data packets, the attacker can inject malicious code, alter data, or impersonate the server, leading to unauthorized access or data manipulation.
- ✓ Data Tampering: Lack of SMB signing allows attackers to modify the content of the SMB packets, potentially leading to data tampering or corruption. This can result in unauthorized modifications to files, commands, or other critical data being transmitted over the network.
- ✓ Credential Theft: If SMB signing is not required, an attacker can capture the authentication credentials exchanged between the client and server. This can lead to unauthorized access to sensitive resources, privilege escalation, or further compromise of the system.
- ✓ SMB Relay Attacks: In the absence of SMB signing, an attacker can perform SMB relay attacks. This involves intercepting authentication requests from clients and relaying them to another server, tricking the client into authenticating against the attacker's server. This allows the attacker to gain unauthorized access to the client's resources or perform other malicious activities.

Recommendations: To mitigate the risks associated with SMB signing not being required, the following recommendations should be followed:

- ✓ Enable SMB Signing: Ensure that SMB signing is enabled and required for all SMB communications. This can be configured in the Group Policy settings or through registry modifications on Windows systems.
- ✓ Keep Systems Updated: Regularly apply security patches and updates to the operating system and SMB-related components. This ensures that any vulnerabilities or weaknesses related to SMB signing are addressed.
- ✓ Network Segmentation: Implement network segmentation to isolate critical systems and resources from potentially compromised SMB instances. This limits the impact of any successful exploitation and restricts the attacker's movement within the network.

2.2 Exploitation Techniques

During the penetration testing, I identified a SQL injection vulnerability in the web application running on port 80. By injecting the character (') into the username and password fields, I was able to exploit this vulnerability. Using the injected payload (1 'or '1' = '1') as the password for the user "john," I gained unauthorized access to the member's control panel, which revealed a username and password.

Warning: mysql_num_rows(): supplied argument is not a valid MySQL result resource in **/var/www/checklogin.php** on line **28**
Wrong Username or Password
Try Again

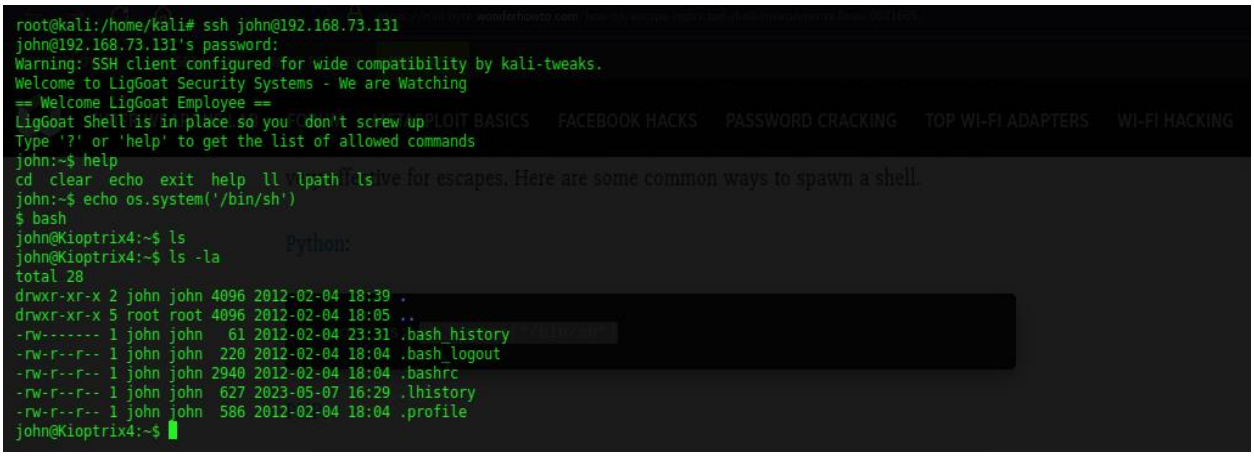
Member's Control Panel

Username : john

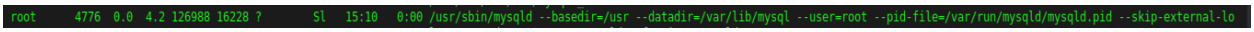
Password : MyNameIsJohn

Logout

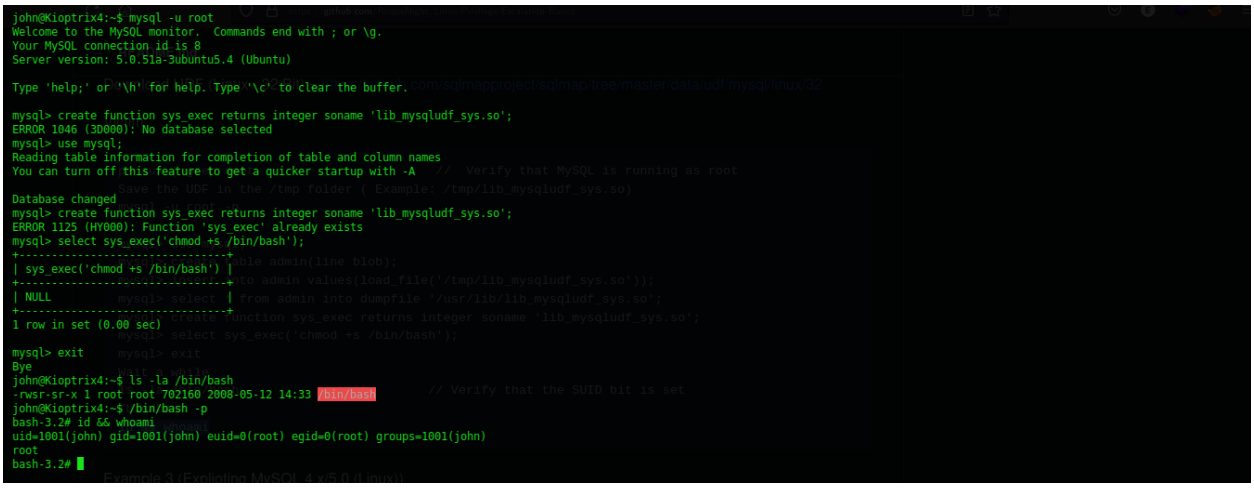
Subsequently, I attempted to access the system via SSH using the obtained credentials. However, I discovered that I was in a restricted shell environment with limited command execution capabilities. To bypass these restrictions, I leveraged the command "echo os.system('bin/sh')" to escape the restricted shell and gain access to a full-featured shell, granting me more control over the system.



With escalated privileges, I explored various methods to further elevate my access privileges to a root user. By running the command "ps aux | grep root," I identified that the MySQL service had root access on the system. Exploiting the lack of a password requirement for the MySQL root user, I successfully logged in as root.



To achieve complete control over the system, I utilized User-Defined Functions (UDF) within the MySQL database. UDFs are custom code snippets that extend the functionality of the MySQL server. I introduced a UDF into the database that loaded a library enabling the execution of commands through the sys_exec() function, providing me with the ability to execute arbitrary code.



2.3 Remediation Recommendations

Based on the vulnerabilities discovered in the penetration testing, the following remediation recommendations are provided to address each specific issue:

1. Ubuntu OS Unsupported Version:
   - ✓ Upgrade the Ubuntu operating system to a supported version that receives regular security updates.
   - ✓ Establish a process for regularly monitoring and applying security patches to keep the system up to date.
   - ✓ Consider implementing an automated patch management system to streamline the patching process.

2. SQL Injection Vulnerability in Password Field:
   - ✓ Implement parameterized queries or prepared statements to prevent SQL injection attacks.
   - ✓ Apply strict input validation and sanitization techniques to ensure user-supplied data is properly handled and doesn't execute as SQL statements.
   - ✓ Conduct a thorough code review to identify and fix any vulnerable areas prone to SQL injection.

3. Security Misconfiguration (Root Login without Password in MySQL):
   - ✓ Set a strong and unique password for the root user in MySQL.
   - ✓ Disable root login without a password and enforce authentication for all MySQL connections.
   - ✓ Regularly review and update user accounts and their privileges to adhere to the principle of least privilege.
   - ✓ Implement firewall rules to restrict access to MySQL from trusted sources only.

4. Samba Badlock Vulnerability:
   - ✓ Update the Samba software to the latest version, which includes patches for the Badlock vulnerability.
   - ✓ Regularly monitor and apply security updates for the Samba software.
   - ✓ Configure Samba with secure settings and follow best practices for securing file sharing services.

5. Browsable Web Directories:
   - ✓ Disable directory browsing in the web server configuration.
   - ✓ Implement access controls and permissions to restrict unauthorized access to directories.

- ✓ Regularly review and update file and directory permissions to ensure sensitive information is adequately protected.

6. SMB Signing not Required:
   - ✓ Enable SMB signing to ensure the integrity and authenticity of SMB communications.
   - ✓ Configure SMB server and client settings to require signing for all connections.
   - ✓ Regularly review and update the SMB configuration to enforce secure communication practices.

It is crucial to address these remediation recommendations promptly to mitigate the identified vulnerabilities and enhance the security of the system. Regular security assessments, vulnerability scanning, and penetration testing should also be conducted to identify any new vulnerabilities and ensure continuous protection against emerging threats.

3. CONCLUSION

The penetration testing conducted on the target system revealed several critical vulnerabilities and misconfigurations that could potentially expose sensitive data, compromise system integrity, and provide unauthorized access to malicious actors. The findings highlight the importance of regular security assessments to identify and address these vulnerabilities before they can be exploited.

The identified vulnerabilities include an unsupported version of the Ubuntu operating system, SQL injection vulnerability in the password field, security misconfiguration allowing root login without a password in MySQL, Samba Badlock vulnerability, browsable web directories, and SMB signing not required. These vulnerabilities pose significant risks to the confidentiality, integrity, and availability of the system and its associated data.

To mitigate these vulnerabilities, immediate action is required. Recommendations include upgrading the Ubuntu operating system to a supported version, implementing strong input validation and parameterized queries to prevent SQL injection attacks, securing the MySQL configuration with proper authentication and access controls, applying patches and updates to address the Samba Badlock vulnerability, disabling directory browsing in the web server, and enforcing SMB signing for secure communication.

Furthermore, it is crucial to establish a proactive approach to security, including regular patch management, vulnerability scanning, and penetration testing. Implementing robust security practices, such as least privilege access control, monitoring user accounts and privileges, and ensuring secure configurations, will greatly enhance the overall security posture of the system.

It is essential to understand that security is an ongoing process, and addressing these vulnerabilities is only the first step. Continuous monitoring, security awareness training, and regular assessments are vital to stay ahead of evolving threats and maintain a robust security posture.

4. APPENDICES

4.1 Appendix A: Detailed Methodology

1. Planning:
   a. Understand the scope of the engagement by identifying the target systems, applications, and networks that are to be tested.
   b. Define the goals and objectives of the testing, including what is in and out of scope.
   c. Establish a testing schedule that includes timelines, deadlines, and expected outcomes.
   d. Identify the team members who will be involved in the testing, their roles, and responsibilities.

2. Information Gathering:
   a. Use tools such as Nmap to discover open ports, services, and operating systems.
   b. Perform passive reconnaissance by analyzing the website structure and web server banner.
   c. Conduct active reconnaissance by performing vulnerability scanning using tools such as Nessus and Nikto.

3. Vulnerability Analysis:
   a. Analyze the results of vulnerability scanning to identify potential vulnerabilities.
   b. Conduct manual testing to validate the findings of the vulnerability scanning tool.
   c. Identify vulnerabilities that can be exploited to gain unauthorized access or to compromise the confidentiality, integrity, or availability of the system.

4. Exploitation:
   a. Exploit the identified vulnerabilities to gain unauthorized access to the system or network.
   b. Escalate privileges to gain administrative access to the system.

5. Reporting:
   a. Document the findings, including the vulnerabilities identified, the impact of these vulnerabilities, and recommendations for remediation.
   b. Provide a summary of the engagement, including the scope, methods, and outcomes.
   c. Present the findings to the client in a clear and concise manner.
   d. Provide guidance to the client on how to address the vulnerabilities identified during the testing.