**Penetration Test Report for Victim1**

**-Sairam Bokka**

**VX05251**

**CYBR 644**

**Contents**

1. **Executive Summary-------------------------------------------------------------------------------- 3**
2. **Host Discovery -------------------------------------------------------------------------------------- 4**
3. **Information Gathering ---------------------------------------------------------------------------- 5**
4. **Exploitation ------------------------------------------------------------------------------------------ 6**
5. **Privilege Escalation--------------------------------------------------------------------------------- 7**
6. **Post Exploitation------------------------------------------------------------------------------------ 8**
7. **Recommendations --------------------------------------------------------------------------------10**

**Executive Summary:**

The victim PC runs an older version of Ubuntu OS. We discover the victim PC using a simple nmap scan over the local network and discover the vulnerable machine. We do a thorough scan of the victim PC using nmap and discover all the open ports of the victim PC along with version information of all the applications running on different ports of the machine. We move forward with discovering vulnerabilities and exploiting them using metasploitable console. We search for available modules that can target and exploit open ports and outdated software running on them like distccd and samba. Some of the applications such as samba run with root privileges, which gives us root access to the victim machine. Using root access, we get all the information from the victim machine, such as passwords, databases, etc.

**Host Discovery:**

A computer screen shot of a computer screen

Description automatically generated

We discover the host by doing a simple nmap scan over the network. The scan shows three hosts active out of which we have windows machine, kali linux and victim machine.

**Information Gathering:**

A screenshot of a computer

Description automatically generated

We do a detailed scan on the victim machine using nmap. We use the following command:

Nmap -sV -p- 1-65535 192.168.17.132 which scans over all ports with version detection. We can see a lot of open ports with outdated software being used for hosting the web application.

**Exploitation:**

1. Distccd exploitation: The distcc program has a daemon running as a network service which is vulnerable to DistCC Daemon Command Execution and can be exploited using the Metasploit module.

A screenshot of a computer program

Description automatically generated

We use the distcc\_exec module available in metasploitable console and set the RHOST to the IP address of the victim and exploit. We get a remote shell with user daemon.

**Priviledge Escalation:**

1. Samba Exploitation: This version of Samba smbd 3.x – 4.x is vulnerable to one of the exploit modules in Metasploit called samba usermap\_script.

A screenshot of a computer

Description automatically generated

We set the RHOST as our victim PC IP address and run the exploit. We get a remote shell access on the victim PC. Now using python, we spawn an interactive shell, which by default for samba, runs with root privileges.

**Post Exploitation:**

1. Password Hashes: As root user, we can easily grab password hashes from the machine.

A screenshot of a computer

Description automatically generated

1. Mysql Database Compromise: The MySQL database present on the system uses the default credentials of root/root and the database can be compromised to disclose sensitive data.

A screen shot of a computer

Description automatically generated

1. PostgreSQL database Compromise: The PostgreSQL installation on the box also uses default credentials postgres/postgres and can be compromised to disclose sensitive data. To access this database we need to switch to the postgres user with password postgres.

A screenshot of a computer

Description automatically generated

**Recommendations**:

1. Update and patch:

* Ensure that the Ubuntu operating system is updated with the latest security patches and updates.
* Keep the web application framework and all its dependencies up to date with the latest stable versions.

1. Secure configuration:

* Review and harden the web server configuration (e.g., Apache or Nginx) to follow security best practices.
* Disable unnecessary modules, services, and features that are not required for the application's functionality.
* Configure proper file and directory permissions to prevent unauthorized access.

1. Input validation and sanitization:

* Implement strict input validation and sanitization mechanisms to prevent common web vulnerabilities such as SQL injection, cross-site scripting (XSS), and command injection.
* Validate and sanitize user inputs on both the client-side and server-side.
* Use parameterized queries or prepared statements when interacting with databases to mitigate SQL injection risks.

1. Authentication and authorization:

* Implement strong authentication mechanisms, such as multi-factor authentication (MFA), to protect user accounts.
* Enforce secure password policies, including minimum length, complexity, and regular password updates.
* Properly implement user roles and access controls to ensure that users can only access the resources and functionalities they are authorized for.

1. Secure communication:

* Enable HTTPS/SSL to encrypt all sensitive data transmitted between the client and the server.
* Use strong encryption algorithms and properly configure SSL/TLS certificates.
* Implement HTTP Strict Transport Security (HSTS) to enforce secure connections.

1. Error handling and logging:

* Implement proper error handling mechanisms to prevent sensitive information from being disclosed in error messages.
* Log security-related events, such as authentication attempts, access violations, and system errors, for monitoring and incident response.

1. Security headers: Implement security headers such as X-XSS-Protection, X-Frame-Options, X-Content-Type-Options, and Content-Security-Policy to protect against various web vulnerabilities.
2. Least privilege principle: Ensure that the web application runs with the least privileges necessary to perform its functions. Avoid running the application with root or administrative privileges.
3. Regular security testing: Conduct regular penetration testing and vulnerability assessments to identify and address any new security issues. Perform code reviews and security audits to identify and fix vulnerabilities in the application's codebase.
4. Security awareness and training: Provide security awareness training to developers, administrators, and users to educate them about secure coding practices, common vulnerabilities, and best practices for maintaining a secure environment.