Assignment 4: Vulnerability Scanning and Binary Exploitation [60 points]

# Part 1: Vulnerability Scanning [30 points]

Pick four “High” severity vulnerabilities and read OpenVAS’s summary / impact sections. How severe would you consider these vulnerabilities? Are there any you think should have been rated higher or lower? Next, pick another 4 “Medium” severity vulnerabilities. How severe would you consider these vulnerabilities? Are there any you think should have been rated higher or lower?

**“High” severity vulnerabilities**

1. rlogin Passwordless Login (10)

I consider this vulnerability very severe. A user should not be able to obtain root access without a password. A lot of damage can be done through the unwarranted access to root privileges. I see the severity ranking as accurate.

1. Test HTTP dangerous methods (7.5)

I consider this vulnerability very severe as well. A user should not be able to freely perform Put and DELETE methods on a server. The server could be flooded with garbage files and important files deleted freely. I would elevate the severity to an 8.

1. MySQL / MariaDB weak password (9)

I consider this vulnerability to be very severe. Logging into the database as a root user is possible with an empty password. This means that the database can be completely accessed, and all its contents stolen. I see the severity ranking as accurate.

1. VNC Brute Force Login (9)

I do not see this vulnerability as being that severe. It mentions that some VNC servers block IP addresses after 5 unsuccessful attempts. If that is in place, I would see this vulnerability as not being severe. Without the blacklisting in place, a brute force attack could easily be accomplished with a max password length of 8 characters. Taking both factors into account, I would downgrade the severity to an 8.

**“Medium” severity vulnerabilities**

1. FTP unencrypted Cleartext Login (4.8)

I would say that this vulnerability is not that severe. Its only fault the login and password can be detected if an attacker is sniffing the network. Due to packet sniffing not being the most common thing, I would say the severity of 4.8 is appropriate.

1. Twiki Cross-Site Request Forgery (6.8)

I see the vulnerability as being moderately severe. The rating of 6.8 is accurate.

1. SSL/TLS: Certificate Expired (5)

I do not see this vulnerability as being that severe. It can simply be mitigated by replacing the certificate. I would downgrade the severity to a 4.

1. Awiki Multiple Local File Include Vulnerabilities (5)

I see this vulnerability as being severe. There is no solution for it and gives an attacker mechanisms to perform attacks. I would elevate this vulnerability to a 7.

# Part 2: Source Code Analysis

Analyze the four provided C source files (ex1.c, ex2.c, ex3.c, ex4.c). For each file, answer the following: is this file vulnerable to buffer overflow? Why or why not? [12 points]

ex1.c:

Yes, buffer overflow will occur if the first command line argument exceeds the buffer size of 15 chars.

ex2.c:

Yes, buffer overflow will occur if 64 or more characters are entered for the password. Initially, the char array is set with one element so that only leaves room for 63 elements, the fgets allows 64 characters.

ex3.c:

Buffer overflow cannot occur. Fgets can store a max of 100 chars into the username array. The username array has room for 100 chars. The strcpy copies the username array to the buffer which has a size of 200.

ex4.c:

The code is vulnerable to buffer overflow. The gets function does not specify a max number of chars to be allowed.

Part 3: Binary Exploitation

You have been provided with a binary, simple\_echo, and its source code, simple\_echo.c. Copy this binary to the Metasploitable VM and disable ASLR using the following command: echo 0 | sudo tee /proc/sys/kernel/randomize\_va\_space

You can re-enable ASLR once you’ve finished the assignment using the following command, though it’s not essential (Metasploitable is intended to be vulnerable, after all):

echo 1 | sudo tee /proc/sys/kernel/randomize\_va\_space

Exploit the buffer overflow vulnerability in simple\_echo.c to run the pwn function. Provide a screenshot of the result and a brief explanation of how you accomplished it [18 points]

Could not get it to work, I tried to overflow the space allocated for the buffer, + 4 for the base pointer, and put the address of the pwn() function in the return address.

