Buffer Overflow SLMail-5.5.0 Service and Gain Root Shell

Devin Trejo
devin.trejo@temple.edu

April 8, 2016

1 Summary

2 Introduction

2.1 Background SLMail5.5.0

SLMail is a message management tool that was advertised towards small to medium sized businesses published by SeatleLabs. The software was popular around the year 2001 for its ease of use and "security" of its email service [**?**]. The service was also scalable for an unlimited number of users to use. The software boast a number of security features including, "Limiting viruses by identifying specific files or types not permitted to enter/leave the server, rejecting emails containing unwanted words, avoiding external use of server as relay for spam, reduce flow of junk mail (anti-spam filter), and authenticate users before they send mail" [**?**]. The last "security" feature was instead a security flaw as the password authentication had a buffer overflow vulnerability. The service is no longer developed as is apparent if one were to search for SLMail on SeattleLabs' website today.

The SLMail service is an 3rd party program bought and downloaded direct from SeattleLabs's website and typically installed on a Windows 2k server. The default options after a succesfull installation of SLMail can be seen in figure 1. The specific version we concern ourselves for this project will by **SLMail5.5.0** which has a known buffer overflow exploit inside the user authentication prompt. When logging in over POP3, an application standard protocol for retrieving emails from a remote server, SLMail will prompt for a user-name and password combination associated with the desired email. If we write our user-name as any string combination and a password containing a shell program we can setup and execute the script on the remote mail server. The referenced shell script will be specially crafted to open a port on the remote server, that gives us access to a shell that contains administrative privileges.

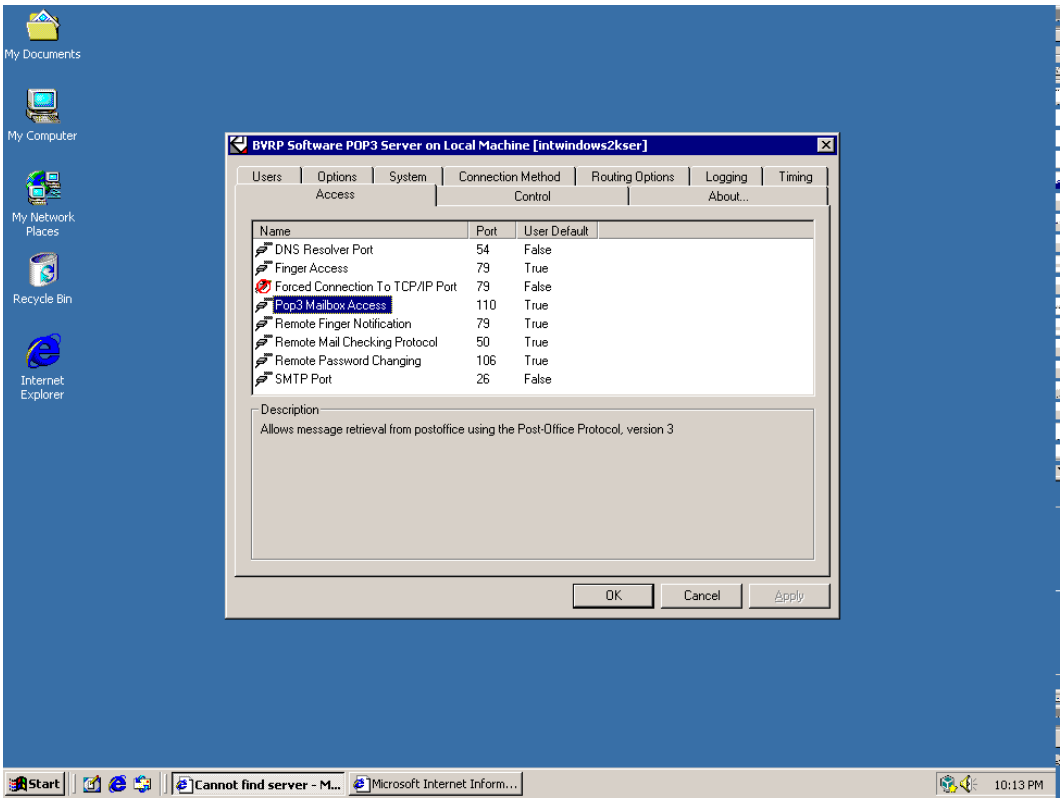

Figure 1: Default SLMAail Port Configuration

2.2 Attack Approach: Fuzzing Attack

The first step for this attack is to gain more information of the SLMail 5.5.0 service. We will implement a technique known as **fuzzing** which will allow us to discover information such as service versions, buffer sizes, and in general the coding implementation of the remote service. To begin the fuzzing process, we try to find the buffer size of the PASS field used by SLMail's POP3 protocol. The first step is to write a script that loops over an array of increasing buffer sizes trying to determine the full length of the input buffer size. Since we already know there is an buffer overflow exploit for these fields we can expect at some point our input to overflow the allocated buffer and crash the program. The idea and goal for this specific fuzzing processes is to overwrite the **EIP register** or the address location on the program stack containing the location in memory the program should return to after executing the USER and PASS input prompt function.

For this assignment we will examine the structure of the SLMail-5.5.0 program and gain insights on its construction. The POP3 interface seen on port 101 is not compatible with standard the standard http protocol as is demonstrated in figure 2. Instead we write a Python script that creates a socket connection to the POP3 service and interfaces with the server using POP3 protocol commands.

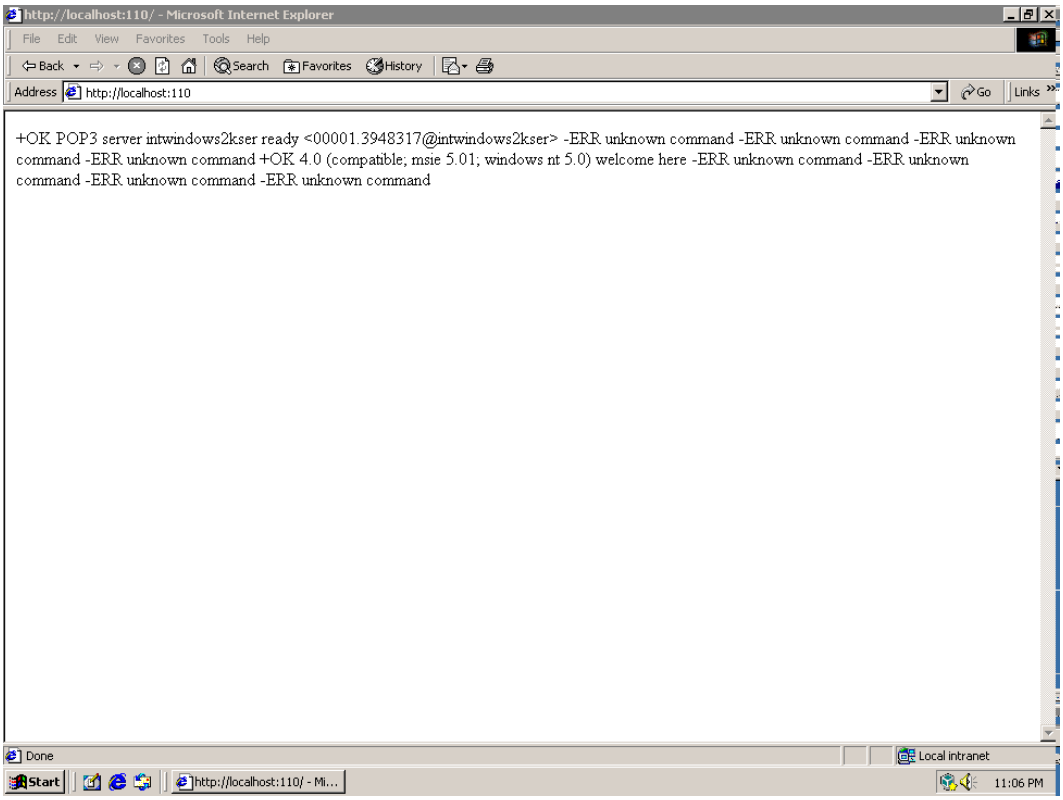

Figure 2: Trying to Connect to SLMail POP3 over HTTP

2.3 Test Environment

For this project we use our default test environment. We have a virtual private network consisting of our Windows 2000 SP4 Server, Kali Linux penetrating machine, and a host machine running through Oracle Virtual Box. The virtual network has a DHCP server running on the VM host machine. The IP/MAC addresses for each are provided in table 1.

Platform	MAC ADDR	Platform IPv4 Address
Kali Linux:	08:00:27:94:5b:ba	192.168.56.102
Windows 2k Server:	08:00:27:87:29:68	192.168.56.105
VM Host Machine:	08:00:27:7c:86:0d	192.168.56.100

Table 1: IP Configuration for SLMail Pen-test Virtual Network

3 Discussion

To begin the intrusion we first have to setup our SLMail server. For this test we used default parameters as seen in figure 1. Next we conducted a NMAP scan from our Kali Linux Machine using NMAP. The scan we performed was a full version scan using the parameters seen shown below.

```
$ nmap -nsV 192.168.56.105
```

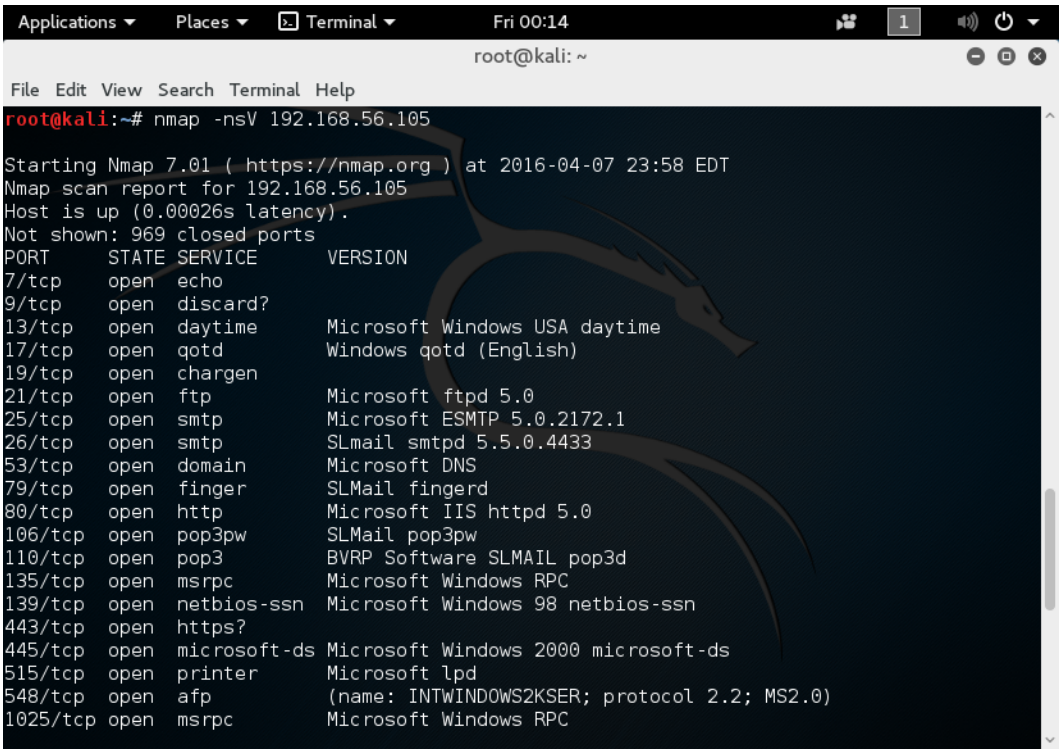

Figure 3: NMAP Scan of Windows Server 2k

From the scan results seen in figure 3 we can see a multitude of open ports and the services running behind the ports. What we are interested in is port 110 which is the standard port for POP3 operations. We know from our research that after installing SLMail an open POP3 port will open that contains the known buffer overflow vulnerability. The NMAP scan revealed a number of other services running on our Windows 2k Server instance but for this test we will focus on port 110.

4 Conclusion

References

Appendix

Listing 1: SLMail Password Fuzzing Script

```
#!/usr/bin/env python3
import socket, sys, os
```

```
def main():
    # Client machine IPv4 address
    #
    clientIP = 10.10.10.102
    clinetPORT = 25

    # Declare a acceptable buffer size that fits into TCP Packet
    #
    BUFFER_SIZE = 1024

    # Create a new socket to the server
    #
    serverSock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    # Try to connect to passed IP
    #
    try:
        s.connect((clientIP, clinetPORT))
    except Exception:
        s.close()
        return -1

    # Check to see if the a name exists on the remote machine
    #
    for userEmail in listEmails:
        msgCmd = "220 " + username + " ESMTP"

        s.send(bytes(msgCmd, 'UTF-8'))

        # Wait to receive a message
        data = s.recv(BUFFER_SIZE)
        data = bytes.decode(data, 'UTF-8')
        print("Received data: \n" + data)
```