Al-Najah National University
Intrusion Detection System

Analysis protection against VSFTPD exploit

Instructor:Dr.Othman Othman

Mohammed Adnan
Dima Bshara
Sineen Bazyan

1 Abstract

This experiment introduces a Wireshark analysis for the VERY SECURE FTP DAEMON (VSFTPD) exploit to make a SNORT rule to detect. In this experiment, there are four different solutions.

2 Introduction

VSFTPD is an FTP server that it can be found in Unix operating systems like Ubuntu, CentOS, Fedora, and Slackware. By default, this service is secure however a major incident happened in July 2011 when someone replaced the original version with a version that contained a backdoor. The backdoor exists in version 2.3.4 of VSFTPD and it can be exploited through Metasploit. In detail, the concept of the attack on VSFTPD 2.3.4 is to trigger the malicious vsf_sysutil_extra(); function by sending a sequence of specific bytes on port 21, which, on successful execution, results in opening the backdoor on port 6200 of the system.

3 Procedures

Required resources
We need 3 VMs and set up the environment as shown in Figure 1:

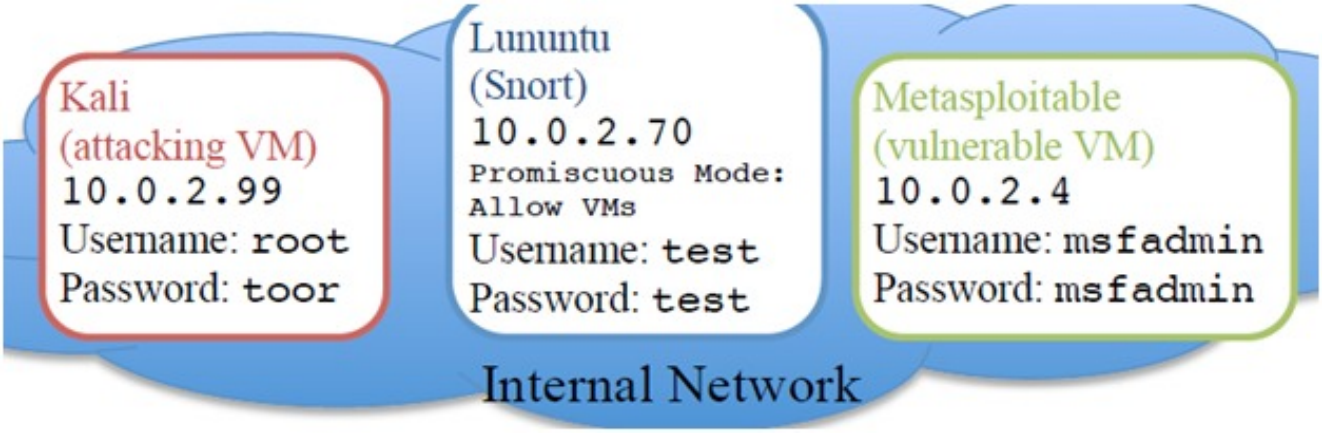


Figure 1: Set up the environment.

1. After Metasploit starts the following commands were used to prepare our attack and to charge it as shown in Figure 2.

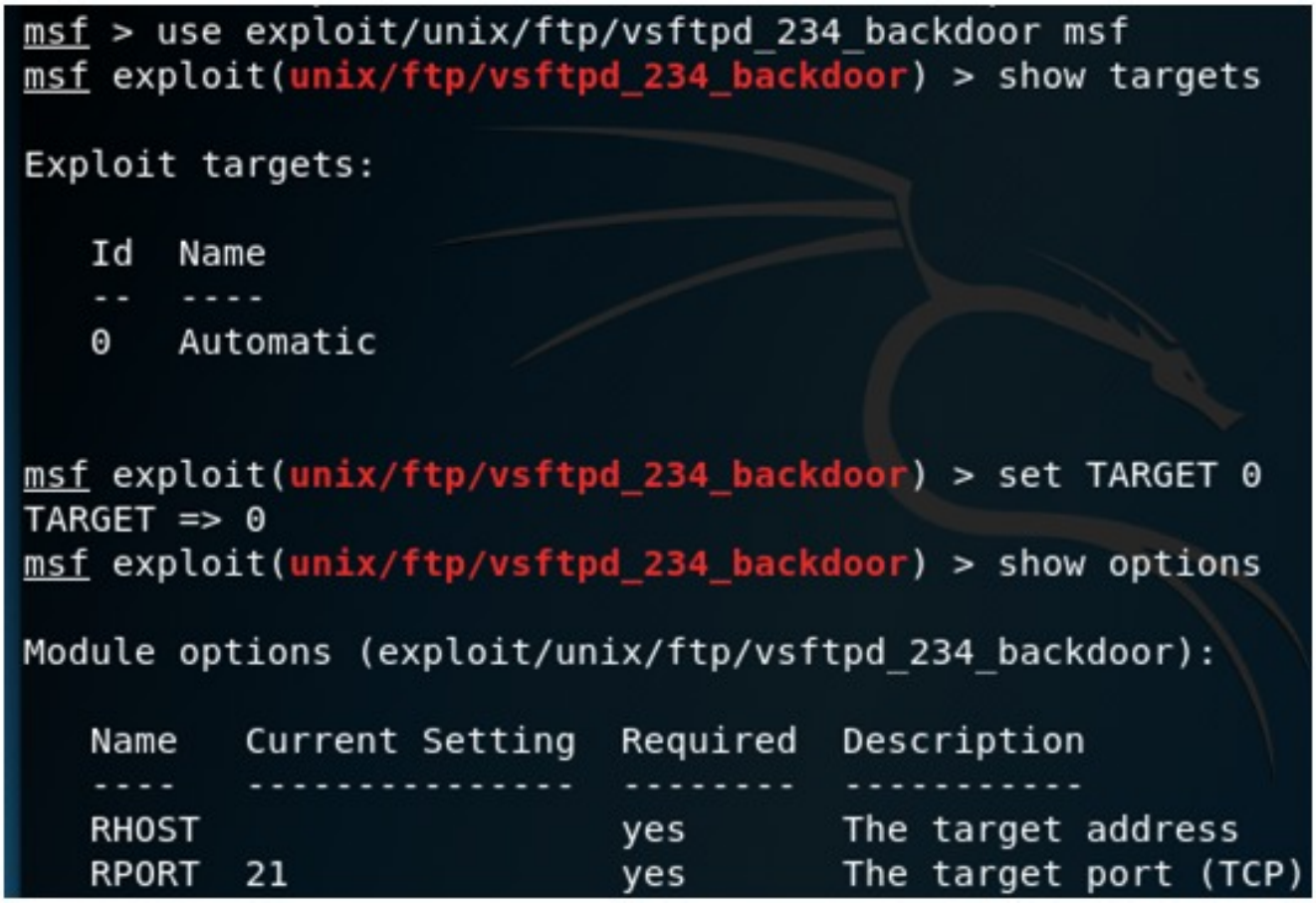


Figure 2: Load the module into Metasploit and display its options.

2. To run this exploit firstly there is a need to set the RHOST. By typing set RHOST followed by IP address of victim machine i.e. 10.0.2.4. Once it is entered and it has been exploited, the execution is successful. As shown in Figure 3:

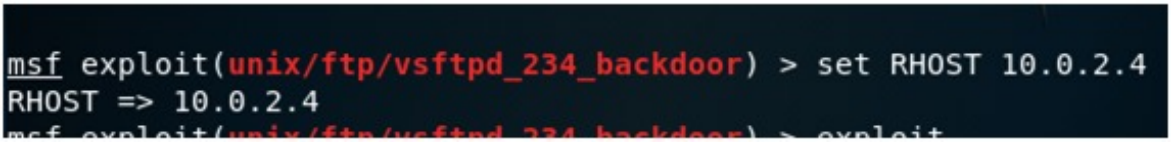


Figure 3: Configuring the vsftpd exploit.

3. Then we have performed some commands to ensure the exploitation is done completely on the victim machine, as shown in Figure 4

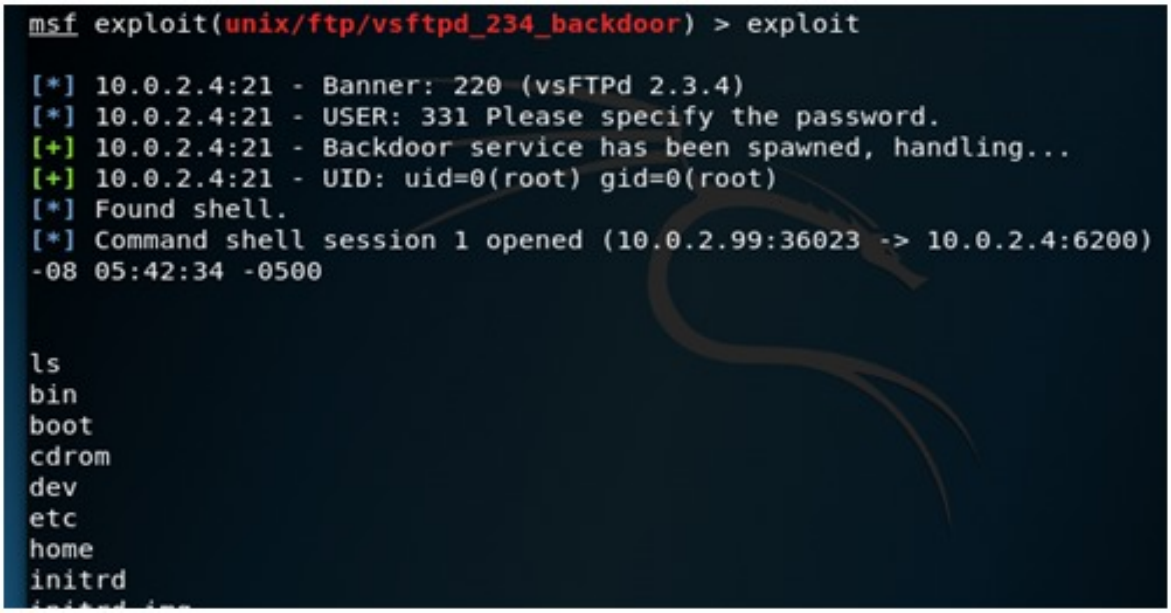


Figure 4: Using the backdoor to perform ls command.

3.1 Solution 1

1. The wireshark.pcap file was opened to analyze the traffic behavior, as shown in Figure 5

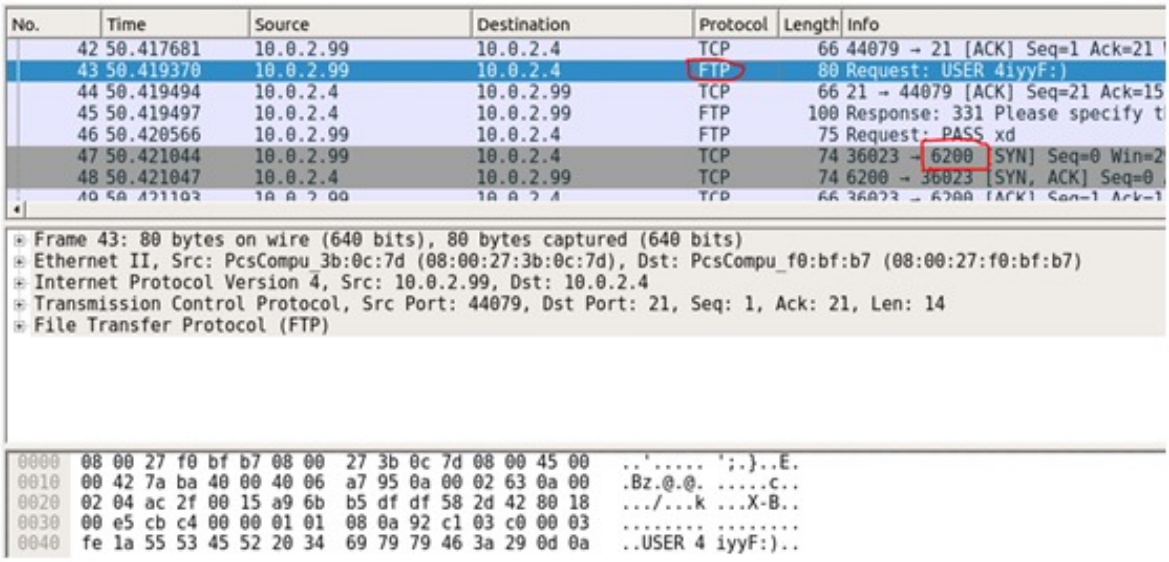


Figure 5: Analysis packets.

We obviously notice that the attacker machine is targeting the FTP protocol and using port 6200 as a backdoor.

2. While analyzing the payloads and headers of packets, we have noticed that the packet's payload contains access to the root privilege!! which refers to illegal access especially by elevating the privilege from a remote host, as shown in Figure 6

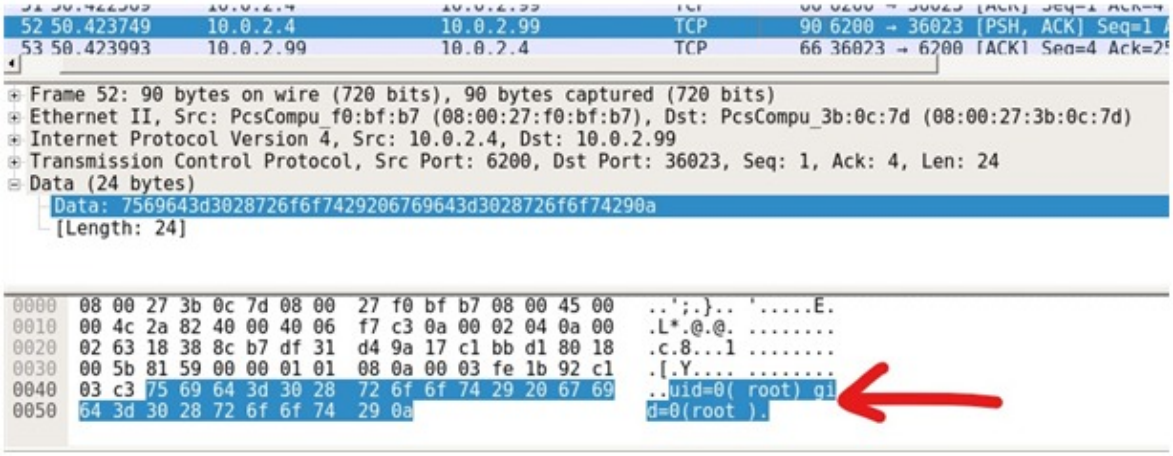


Figure 6: Figure out the signature of attack behavior.

3. To write a signature-based need to look for: (1)Directed at TCP session. (2)malicious packets contain the root privilege.
4. The rule will be as shown in figure 7. This is because we want to generate an alert of an attack with TCP protocol that might come from any network to any target so it's any to any, and the content must match user-id equals 0 with the root word between brackets. Note that sid value is used for testing purposes and rev value must follow it.
5. To test the snort we have performed the following command and the alert was displayed, as shown in Figure 8 .

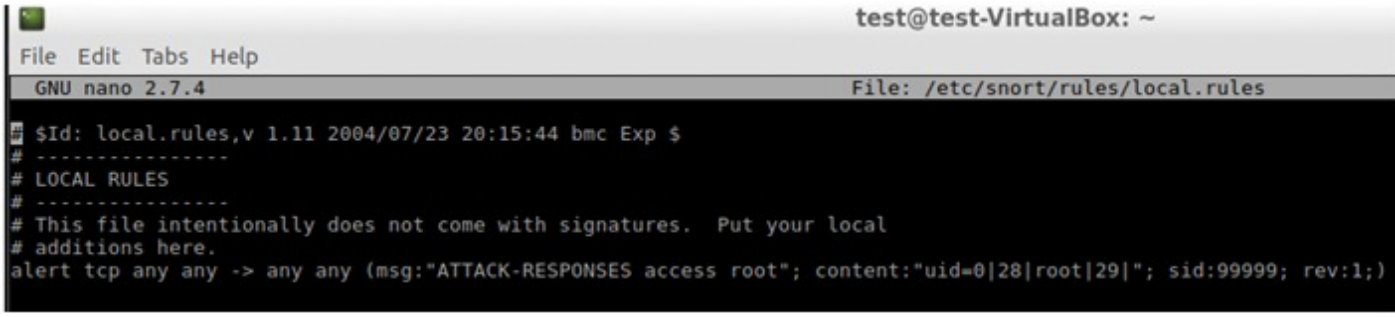


Figure 7: The snort rule.

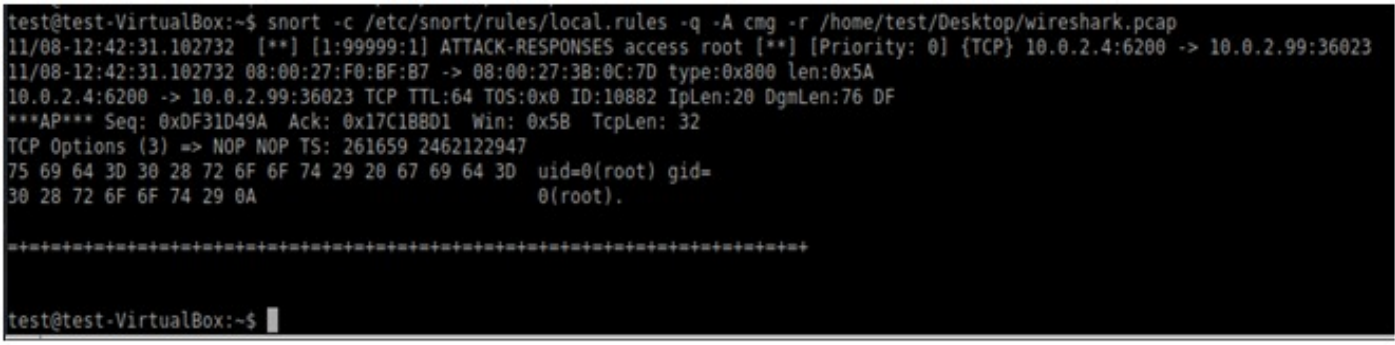


Figure 8: Snort alert.

3.2 Solution 2

1. what we are looking for is :
- A targeting at the victime's ip on port FTP
- Send a "USER" with smiley face:) . As shown in figure 9. **Why smiley face?** as shown in figure 10, this is the name of this attack.

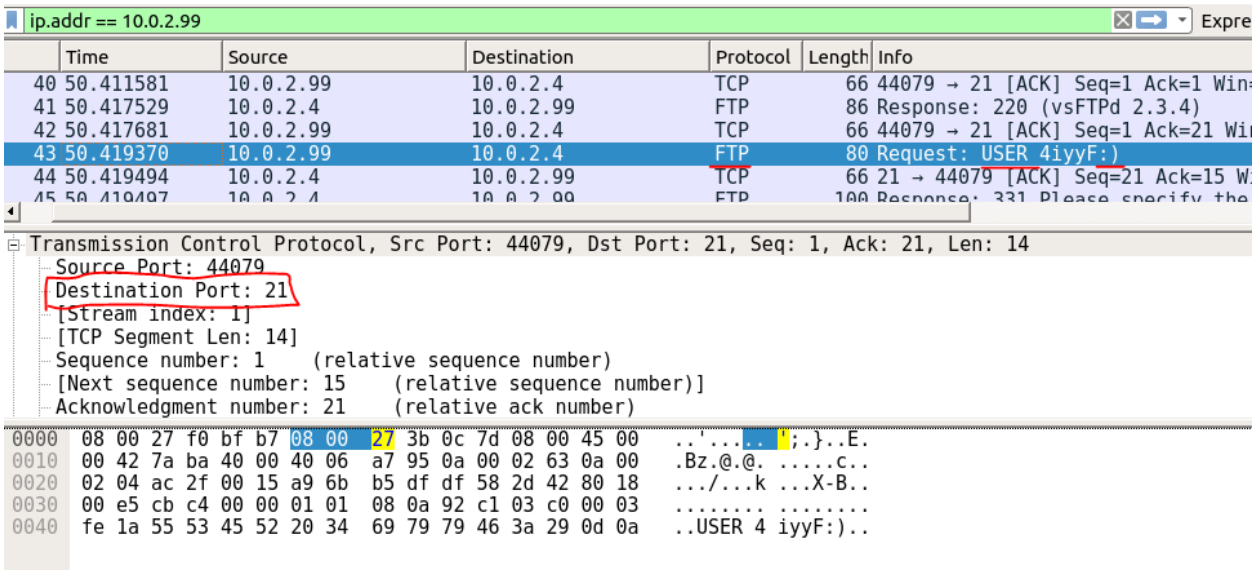


Figure 9: Signature of attack behaviour .

Smiley Face Attack:

The same version of FTP i.e. vsftpd 2.3.4 is vulnerable to another attack. Here while inputing the Name and Password, just put a smiley face i.e. :) at the end of name and password can be given as anything. The connection hangs up after password. and you can actually get the shell of the target using tool Netcat.

Figure 10: CVE .

2. The rule will be as shown in figure 11.

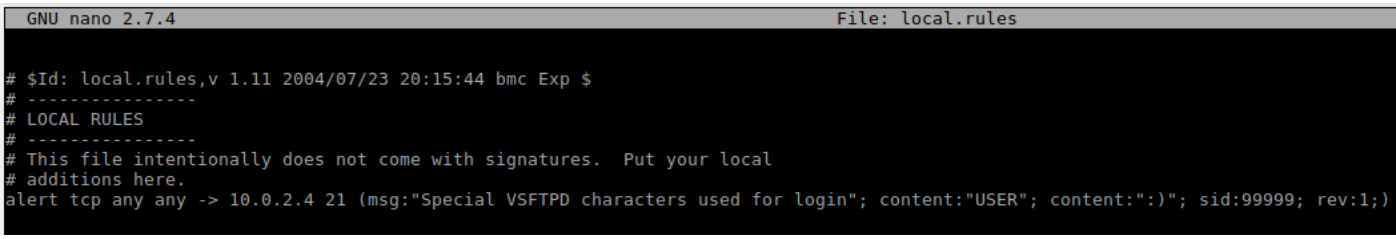


Figure 11: snort rule .

3. To test the snort we have performed the following command and the alert was displayed, as shown in Figure 12.

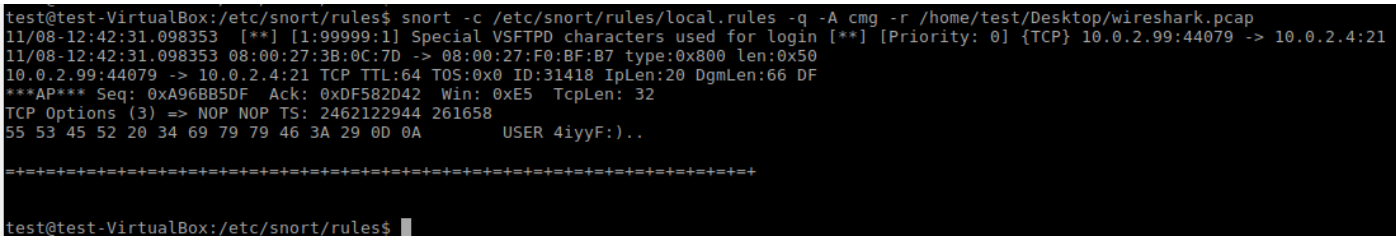
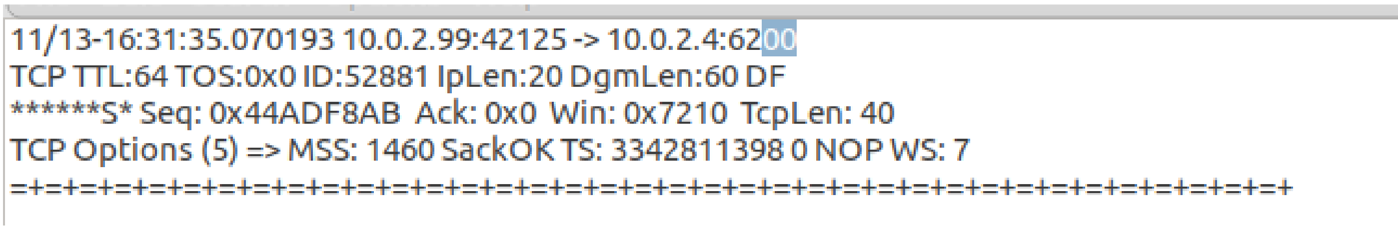


Figure 12: Test rule .

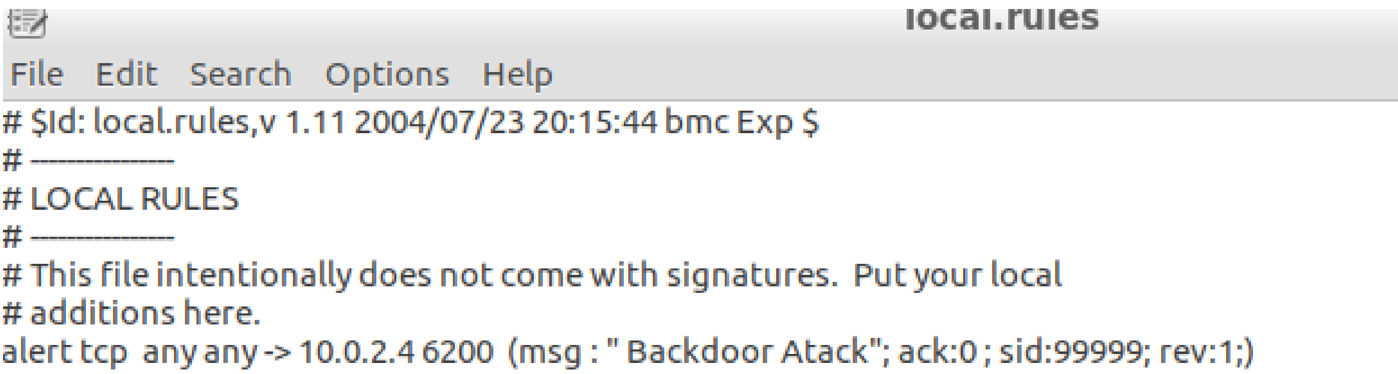
3.3 Solution 3

1. At first, we capture the first packet, which is an SYN packet that means the attacker will do listen on port 6200 to perform the backdoor attack as shown in figure 13. Then we write the rule as figure 14 to do an alert when coming from any IP and port to FTP server IP in port 6200, which is the most common backdoor port for this attack. After that, we put **ACK** to be equal to 0 which, means that the first packet of TCP connection is an SYN packet that will do listen for that port. In the end, we test the rule as shown in figure 15.



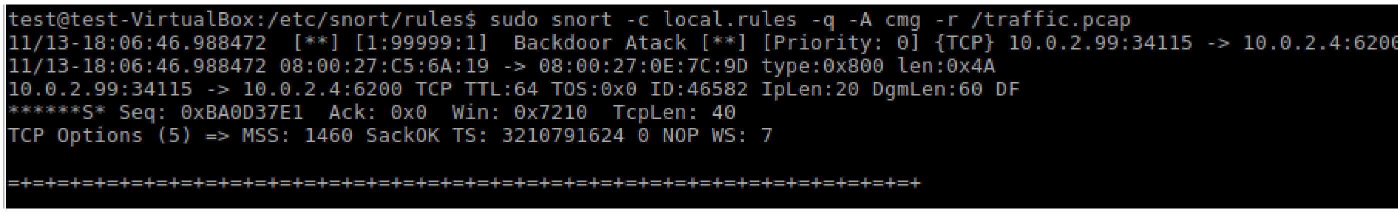
```
11/13-16:31:35.070193 10.0.2.99:42125 -> 10.0.2.4:6200
TCP TTL:64 TOS:0x0 ID:52881 IpLen:20 DgmLen:60 DF
******S* Seq: 0x44ADF8AB  Ack: 0x0  Win: 0x7210  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 3342811398 0 NOP WS: 7
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Figure 13: Capture the packet.



```
local.rules
File  Edit  Search  Options  Help
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# ----------------
# LOCAL RULES
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.
alert tcp  any any -> 10.0.2.4 6200  (msg : " Backdoor Atack"; ack:0 ; sid:99999; rev:1;)
```

Figure 14: Snort rule.



```
test@test-VirtualBox:/etc/snort/rules$ sudo snort -c local.rules -q -A cmg -r /traffic.pcap
11/13-18:06:46.988472  [**] [1:99999:1]  Backdoor Atack [**] [Priority: 0] {TCP} 10.0.2.99:34115 -> 10.0.2.4:6200
11/13-18:06:46.988472 08:00:27:C5:6A:19 -> 08:00:27:0E:7C:9D type:0x800 len:0x4A
10.0.2.99:34115 -> 10.0.2.4:6200 TCP TTL:64 TOS:0x0 ID:46582 IpLen:20 DgmLen:60 DF
******S* Seq: 0xBA0D37E1  Ack: 0x0  Win: 0x7210  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 3210791624 0 NOP WS: 7

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Figure 15: Snort alert.

3.4 Solution 4

1. At first, we capture the first packet of the TCP after the FTP connection establishing as shown in figure 16, as we see that packet contains (id), which lets us know that the attacker tries to get root permission. So in our rule as shown in figure 17 we do alert from any IP to FTP server in 6200 port which is the backdoor port that attacker listen to, then the content is "id" which is the content for this packet, and the TCP connection established, to the server. In the last, we test the rule as shown in figure 18.

```
00  08 00 27 0e 7c 9d 08 00  27 c5 6a 19 08 00 45 00   ..'.|... '.j...E.
10  00 37 dd f0 40 00 40 06  44 6a 0a 00 02 63 0a 00   .7..@.@. Dj...c..
20  02 04 b3 d7 18 38 e6 66  5f f5 42 3e 34 49 80 18   .....8.f _.B>4I..
30  00 e5 20 1d 00 00 01 01  08 0a bf 60 ca d7 00 17   .. ..... ...`....
40  b6 a2 69 64 0a                                     ..id.
```

Figure 16: Capture packet.

```
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# ----------------
# LOCAL RULES
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.
alert tcp  any any -> 10.0.2.4 6200  (msg : " Backdoor Atack is established in our server!!!!"; content:"id";flow:established,to_server; nocase; ; sid:99999; rev:1;)
```

Figure 17: Snort alert.

```
test@test-VirtualBox:/etc/snort/rules$ sudo snort -c local.rules -q -A cmg -r traffic.pcap
11/13-18:06:47.003459  [**] [1:99999:1]  Backdoor Atack is established in our server!!!! [**] [Priority: 0] {TCP} 10
.0.2.99:46039 -> 10.0.2.4:6200
11/13-18:06:47.003459 08:00:27:C5:6A:19 -> 08:00:27:0E:7C:9D type:0x800 len:0x45
10.0.2.99:46039 -> 10.0.2.4:6200 TCP TTL:64 TOS:0x0 ID:56816 IpLen:20 DgmLen:55 DF
***AP*** Seq: 0xE6665FF5  Ack: 0x423E3449  Win: 0xE5  TcpLen: 32
TCP Options (3) => NOP NOP TS: 3210791639 1554082
69 64 0A                                          id.

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Figure 18: test alert.

4 Conclusion

After persistence has been achieved, we are able to prevent this type of attack from an unauthenticated, remote attacker could exploit such this Metasploit tool to execute arbitrary code as root. So we tried and did the attack ourselves and write our own code in order to elevate ourselves beyond the realm of just relying on other security researchers' work, and maybe learn something for ourselves along the way.