Project: Detecting Intrusions using Snort

Clay Jones

Objective:

The objective of this Lab was to learn how to properly set up an intrusion detection tool "Snort." The objective was to detect traffic being sent from a linux machine to a windows machine. The linux machine will run different commands such as nmap and ping, along with running metasploit.

Set up:

First, I had to locate where my Snort application was and look at the files associated with it.

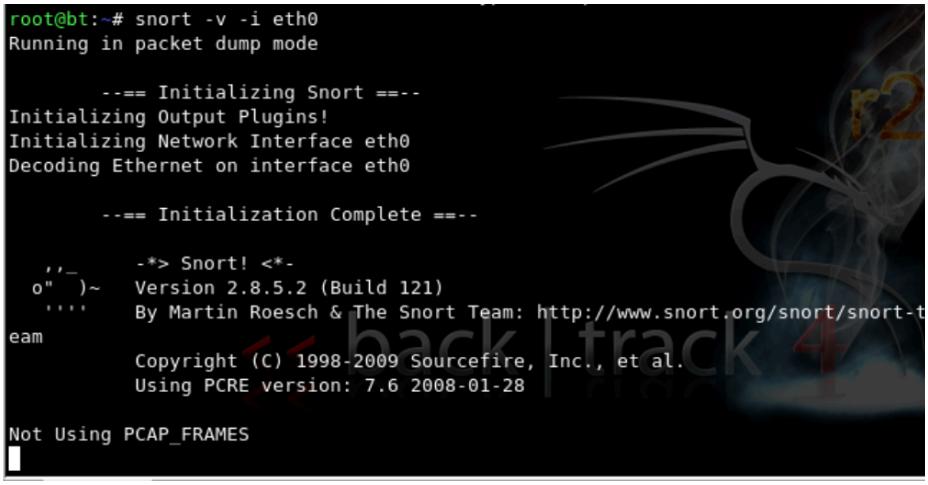
Snort Rules:

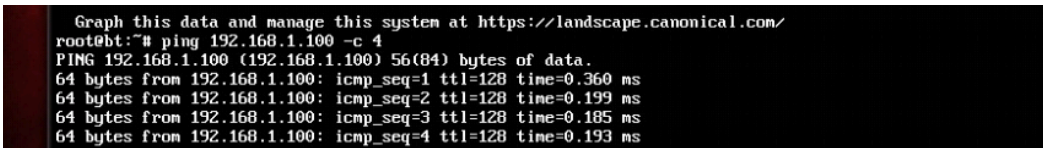
```
root@bt:~# ls /etc/snort/rules
Makefile.am                    community-web-cgi.rules     other-ids.rules
VRT-License.txt                community-web-client.rules  p2p.rules
attack-responses.rules         community-web-dos.rules     policy.rules
backdoor.rules                 community-web-iis.rules     pop2.rules
bad-traffic.rules              community-web-misc.rules    pop3.rules
bleeding-all.rules             community-web-php.rules     porn.rules
cgi-bin.list                   content-replace.rules       rpc.rules
chat.rules                     ddos.rules                  rservices.rules
community-bot.rules            deleted.rules               scan.rules
community-deleted.rules        dns.rules                   shellcode.rules
community-dos.rules            dos.rules                   smtp.rules
community-exploit.rules        experimental.rules          snmp.rules
community-ftp.rules            exploit.rules               spyware-put.rules
community-game.rules           finger.rules                sql.rules
community-icmp.rules           ftp.rules                   telnet.rules
community-imap.rules           icmp-info.rules             tftp.rules
community-inappropriate.rules  icmp.rules                  virus.rules
community-mail-client.rules    imap.rules                  voip.rules
community-misc.rules           info.rules                  web-attacks.rules
community-nntp.rules           local.rules                 web-cgi.rules
```

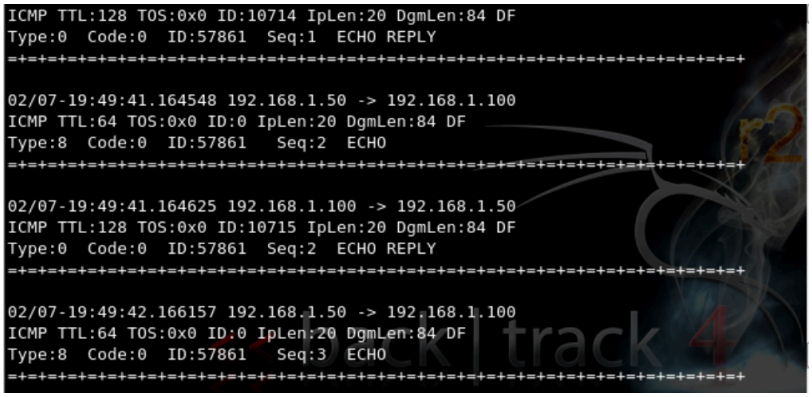
```
root@bt:~# ls /etc/snort/rules/telnet.rules
/etc/snort/rules/telnet.rules
root@bt:~# cat /etc/snort/rules/telnet.rules
# Copyright 2001-2005 Sourcefire, Inc. All Rights Reserved
#
# This file may contain proprietary rules that were created, tested and
# certified by Sourcefire, Inc. (the "VRT Certified Rules") as well as
# rules that were created by Sourcefire and other third parties and
# distributed under the GNU General Public License (the "GPL Rules").  The
# VRT Certified Rules contained in this file are the property of
# Sourcefire, Inc. Copyright 2005 Sourcefire, Inc. All Rights Reserved.
# The GPL Rules created by Sourcefire, Inc. are the property of
# Sourcefire, Inc. Copyright 2002-2005 Sourcefire, Inc. All Rights
# Reserved.  All other GPL Rules are owned and copyrighted by their
# respective owners (please see www.snort.org/contributors for a list of
# owners and their respective copyrights).  In order to determine what
# rules are VRT Certified Rules or GPL Rules, please refer to the VRT
# Certified Rules License Agreement.
#
#
# $Id: telnet.rules,v 1.56 2007/11/06 17:13:05 vrtbuild Exp $
```

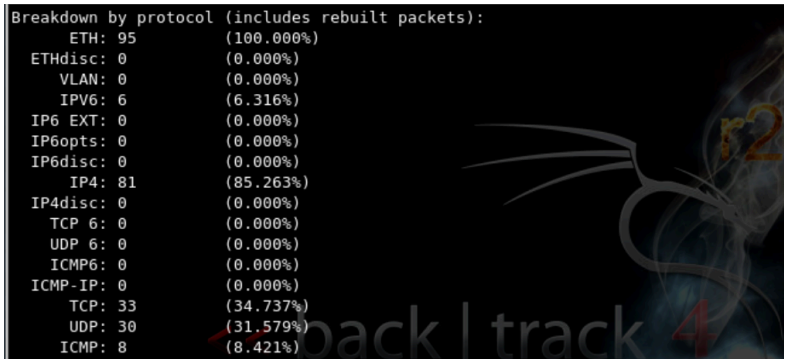
Afterwards, I took a look at the preconfigure rules created for each protocol. I opened Telnet specifically just to see the rules for one protocol.

Capturing Ping Scans



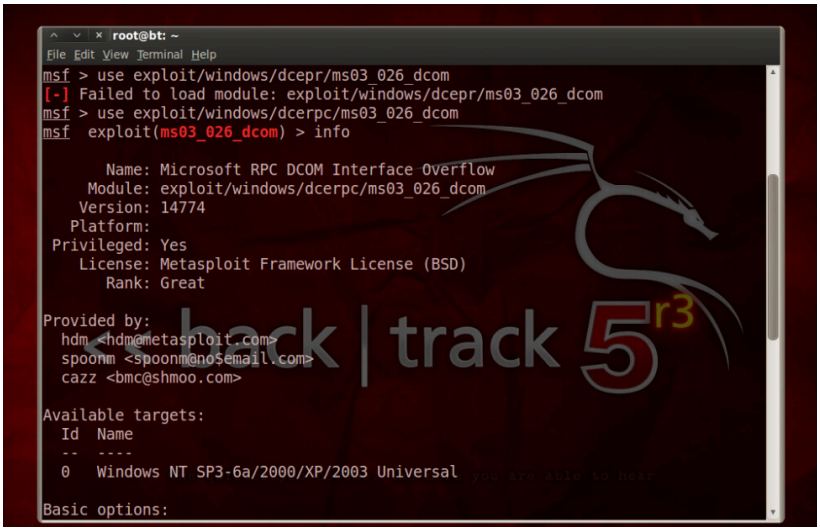


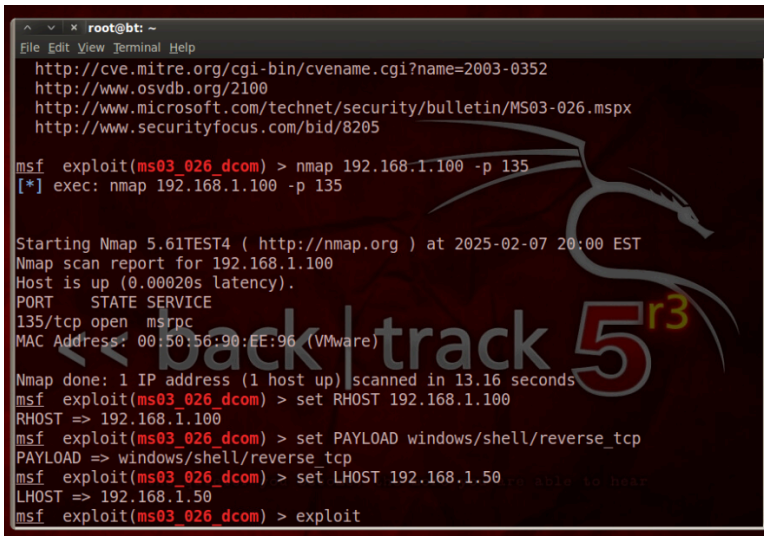




As you can see, I started the snort capture and then, on the linux machine, I ran 4 ping commands to generate network traffic. Snort detected the pings as you can see where the echo command is shown. Since there were 4 ping commands, you see that there were 8 ICMP packets captured.

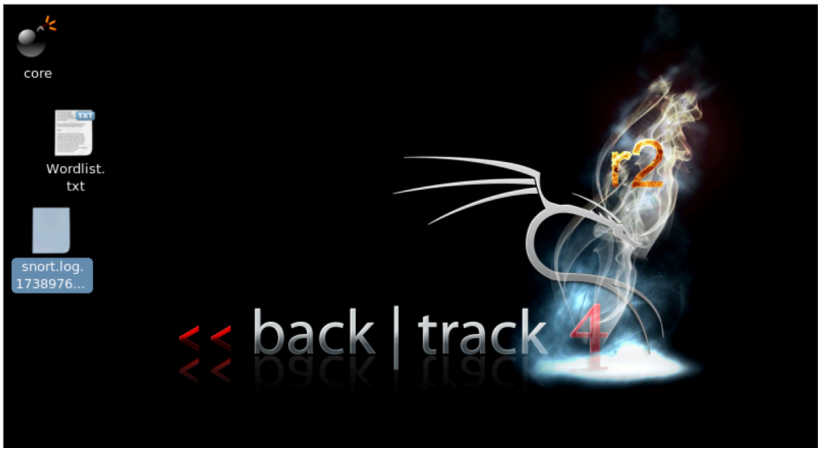
Sending a payload with Metasploit to generate traffic





I activated the listener for snort and set up the DCOM Remote Procedure Call exploit in Metasploit over port 135. I ran an nmap command to show port 135 was open. This module exploits a stack buffer overflow in the RPCSS service. This is a vulnerability in older windows systems. We set the windows machine as the host and sent a reverse shell payload to gain system access.

Detection Report:




```
root@bt:~# snort -l . -c /etc/snort/snort.conf - r 'root/snort.log.1738976189'
Running in IDS mode

        --== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file "/etc/snort/snort.conf"
PortVar 'HTTP_PORTS' defined :  [ 80 ]
PortVar 'SHELLCODE_PORTS' defined :  [ 0:79 81:65535 ]
PortVar 'ORACLE_PORTS' defined :  [ 1521 ]
PortVar 'FTP_PORTS' defined :  [ 21 ]
Tagged Packet Limit: 256
Snort BPF option: - r root/snort.log.1738976189
```

```
DCE/RPC
  Connection oriented
    Packet stats
      PDUs: 6
      Bind: 3
      Bind Ack: 3
      Request fragments: 0
      Response fragments: 0
      Client seg reassembled:
      Server seg reassembled:
===============================
===============================
Snort exiting
root@bt:~# kwrite alert
```

Shell

```
[**] [1:8694:2] NETBIOS DCERPC NCACN-IP-TCP IActivation remoteactivation overflow attempt [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1].
02/10-22:59:11.300964 192.168.1.50:50570 -> 192.168.1.100:135
TCP TTL:240 TOS:0x10 ID:0 IpLen:20 DgmLen:640
***AP*** Seq: 0xE4CE37CB  Ack: 0xD66B757  Win: 0xFDA7  TcpLen: 20
[Xref => http://www.microsoft.com/technet/security/bulletin/MS03-039.mspx][Xref =>
http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx][Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=2003-0715][Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=2003-0352][Xref =>
http://www.securityfocus.com/bid/8205]

[**] [1:1000002:1] Windows CMD.EXE banner [**]
[Priority: 0].
02/10-22:59:11.549628 192.168.1.100:1732 -> 192.168.1.50:4444
TCP TTL:128 TOS:0x0 ID:33931 IpLen:20 DgmLen:76 DF
***AP*** Seq: 0xB186A77E  Ack: 0x2B0521E5  Win: 0xFF0B  TcpLen: 20

[**] [1:1292:9] ATTACK-RESPONSES directory listing [**]
[Classification: Potentially Bad Traffic] [Priority: 2].
02/10-23:00:30.375078 192.168.1.100:1732 -> 192.168.1.50:4444
TCP TTL:128 TOS:0x0 ID:34385 IpLen:20 DgmLen:1219 DF
***AP*** Seq: 0xB186A809  Ack: 0x2B0521E9  Win: 0xFF07  TcpLen: 20
```

Lasty, I stopped my Snort capture. A snort log file appeared on my desktop and I copied that file in the command above and then ran a kwrite alert command to store all the detections in a text file. As I parsed through this file I discovered that the attack was by 192.168.1.50 over port 135 and the directory listing command was done on the victim's machine. All this was done by our linux machine and snort was able to detect and log the malicious traffic.

Lessons Learned:

- **IDS is better when paired with IPS**
- **Snort can be used for continuous monitoring**
- **Snort offers real time traffic monitoring**
- **Snort can have false positives**
- **Snort is heavily rule based**