Detection of events pertaining to security in the Science DMZ network

Technical Report

Mariecarmen A. Reynoso Toribio
mariecarmen.reynoso95@gmail.com
Computer Science Department
University of Puerto Rico - Rio Piedras

Advisor:
Humberto G. Ortiz Zuazaga, Ph.D
humberto.ortiz@upr.edu
Computer Science Department
University of Puerto Rico - Rio Piedras

Abstract:

The University of Puerto Rico use its network for different type of services like students services, administrative services, and research services. The Computer Science Department is building an independent network known as the Science DMZ (10GE) dedicated to science research and the transference of big data. This Science DMZ is a public network without a firewall, thus the analysis of the traffic in the Science DMZ network will help us to learn efficient ways to protect the network without affecting the data transfers.

Introduction:

Science DMZ is a portion of the network, built at or near the campus or laboratory's local network perimeter that is designed such that the equipment, configuration, and security policies are optimized for high-performance scientific applications rather than for general-purpose business systems or "enterprise" computing [1]. But Science DMZ doesn't have a firewall because the primary function of a firewall rule set is to permit or deny network traffic using packet header information in a process where each packet is typically matched against the firewall rule set [2]; but the data occupies a lot of memory and for be possible to transfer all the data the network cannot have a firewall.

Objetive:

Detect events on Science DMZ network using a packet sniffer called TCPDump and then use NIDS (Network Intrusion Detection System) called Snort[3][4], for see specifically the characteristics of the packets in the ip addresses connections.

Discussion: TCPDump and Snort

TCPDump is a packet sniffer and we used it in the last semester for see the characteristics of the two first packets in port 22, and we saw that the two ip address that connected on SSH had a connection in the first part of TCP. Then we worked inside the Science DMZ network and when we used the TCPDump commands on terminal we see that the ip address connection had a PUSH (Flag[P]) and that means that both ip's had a push of data in the port 22.

```
[mreynoso@hulk ~]$ sudo tcpdump -c 15  port 22
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
12:53:43.466460 IP 136.145.181.41.ssh > 136.145.181.172.54613: Flags [P.], seq 3145964318:3145964510, ack 3407664205, win 249, options [nop,nop,TS val
99285839 ecr 9579685], length 192
12:53:43.467859 IP 136.145.181.172.54613 > 136.145.181.41.ssh: Flags [.], ack 192, win 1444, options [nop,nop,TS val 9579816 ecr 99285839], length 0
12:53:43.469702 IP 136.145.181.41.ssh > 136.145.181.172.54613: Flags [P.], seq 192:576, ack 1, win 249, options [nop,nop,TS val 99285842 ecr 9579816],
length 384
12:53:43.470514 IP 136.145.181.41.ssh > 136.145.181.172.54613: Flags [P.], seq 576:784, ack 1, win 249, options [nop,nop,TS val 99285843 ecr 9579816],
length 208
12:53:43.471234 IP 136.145.181.172.54613 > 136.145.181.41.ssh: Flags [.], ack 576, win 1444, options [nop,nop,TS val 9579816 ecr 99285842], length 0
12:53:43.471529 IP 136.145.181.41.ssh > 136.145.181.172.54613: Flags [P.], seq 784:1136, ack 1, win 249, options [nop,nop,TS val 99285844 ecr 9579816],
 length 352
12:53:43.471533 IP 136.145.181.172.54613 > 136.145.181.41.ssh: Flags [.], ack 784, win 1444, options [nop,nop,TS val 9579816 ecr 99285843], length 0
12:53:43.472506 IP 136.145.181.41.ssh > 136.145.181.172.54613: Flags [P.], seq 1136:1488, ack 1, win 249, options [nop,nop,TS val 99285845 ecr 9579816]
, length 352
12:53:43.473498 IP 136.145.181.41.ssh > 136.145.181.172.54613: Flags [P.], seq 1488:1696, ack 1, win 249, options [nop,nop,TS val 99285846 ecr 9579816]
, length 208
12:53:43.474484 IP 136.145.181.172.54613 > 136.145.181.41.ssh: Flags [.], ack 1136, win 1444, options [nop,nop,TS val 9579817 ecr 99285844], length 0
12:53:43.474498 IP 136.145.181.41.ssh > 136.145.181.172.54613: Flags [P.], seq 1696:1904, ack 1, win 249, options [nop,nop,TS val 99285847 ecr 9579816]
, length 208
12:53:43.474751 IP 136.145.181.172.54613 > 136.145.181.41.ssh: Flags [.], ack 1488, win 1444, options [nop,nop,TS val 9579817 ecr 99285845], length 0
12:53:43.474956 IP 136.145.181.172.54613 > 136.145.181.41.ssh: Flags [.], ack 1696, win 1444, options [nop,nop,TS val 9579817 ecr 99285846], length 0
12:53:43.475583 IP 136.145.181.41.ssh > 136.145.181.172.54613: Flags [P.], seq 1904:2560, ack 1, win 249, options [nop,nop,TS val 99285848 ecr 9579817]
, length 656
12:53:43.476499 IP 136.145.181.41.ssh > 136.145.181.172.54613: Flags [P.], seq 2560:2768, ack 1, win 249, options [nop,nop,TS val 99285849 ecr 9579817]
, length 208
15 packets captured
17 packets received by filter
0 packets dropped by kernel
```

Then we change the method for see more information about the entries of the network and what happend about all the packets in any port. So, we decided to use an IDS (Intrusion Detection System). An IDS responsibility is to detect suspicious or unacceptable system and network activity and to alert a systems administrator to this activity[3]. This have two types: Network Intrusion Detection System (NIDS) and Host Intrusion Detection System (HIDS)[4].NIDS are intrusion detection systems that capture data packets traveling on the network media (cables, wireless) and match them to a database of signatures. Depending upon whether a packet is matched with an intruder signature, an alert is generated or the packet is logged to a file or database; and a HIDS analyze the traffic on a server or PC and this can d what happens in each host and detect the failed connections of them[4][5]. A one of the most popular NIDS is Snort. Snort is a open source network intrusion prevention system (IPS) capable of performing real-time traffic analysis and packet-logging on IP networks[6] and Snort is logically divided into multiple components. These components work together to detect particular attacks and to generate output in a required format from the detection system, and these are: Packet Decoder, Preprocessors, Detection Engine, Logging and Alerting System, Output Modules[5]. So, for our goal in this research is use Snort because this give us much information unlike TCPDump that the last one give us a general information of the packet.

Methodology:

We install snort by these commands in hulk[7].

```
wget https://www.snort.org/downloads/snort/daq-2.0.5.tar.gz
wget https://www.snort.org/downloads/snort/snort-2.9.7.3.tar.gz

tar xvfz daq-2.0.5.tar.gz
cd daq-2.0.5
./configure; make; sudo make install

tar xvfz snort-2.9.7.3.tar.gz
cd snort-2.9.7.3
./configure--enable-sourcefire; make; sudo make install

sudo /usr/local/bin/snort -v -i eth5 (run Snort by root)
```

```
WARNING: No preprocessors configured for policy 0.
05/26-07:24:42.573288 70.45.132.54:49666 -> 136.145.231.10:22
TCP TTL:43 TOS:0x0 ID:59056 IpLen:20 DgmLen:64 DF
***A**** Seq: 0xC10D76DD  Ack: 0x8C169D6E  Win: 0x4172  TcpLen: 44
TCP Options (6) => NOP NOP TS: 1583771 4119739296 NOP NOP Sack: 35862@40654
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

05/26-07:24:42.574131 136.145.231.10:22 -> 70.45.132.54:49666
TCP TTL:64 TOS:0x10 ID:19509 IpLen:20 DgmLen:1500 DF
***A**** Seq: 0x8C17261E  Ack: 0xC10D76DD  Win: 0xE3  TcpLen: 32
TCP Options (3) => NOP NOP TS: 4119739392 1583771
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

WARNING: No preprocessors configured for policy 0.
05/26-07:24:42.575498 70.45.132.54:49666 -> 136.145.231.10:22
TCP TTL:43 TOS:0x0 ID:59057 IpLen:20 DgmLen:64 DF
***A**** Seq: 0xC10D76DD  Ack: 0x8C169D6E  Win: 0x4172  TcpLen: 44
TCP Options (6) => NOP NOP TS: 1583771 4119739296 NOP NOP Sack: 35862@40654
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

05/26-07:24:42.575550 136.145.231.10:22 -> 70.45.132.54:49666
TCP TTL:64 TOS:0x10 ID:19510 IpLen:20 DgmLen:476 DF
***AP*** Seq: 0x8C172BC6  Ack: 0xC10D76DD  Win: 0xE3  TcpLen: 32
TCP Options (3) => NOP NOP TS: 4119739393 1583771
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

WARNING: No preprocessors configured for policy 0.
05/26-07:24:42.575559 70.45.132.54:49666 -> 136.145.231.10:22
TCP TTL:43 TOS:0x0 ID:59058 IpLen:20 DgmLen:64 DF
***A**** Seq: 0xC10D76DD  Ack: 0x8C169D6E  Win: 0x4172  TcpLen: 44
TCP Options (6) => NOP NOP TS: 1583772 4119739296 NOP NOP Sack: 35862@40654
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

WARNING: No preprocessors configured for policy 0.
05/26-07:24:42.575571 70.45.132.54:49666 -> 136.145.231.10:22
TCP TTL:43 TOS:0x0 ID:59059 IpLen:20 DgmLen:64 DF
***A**** Seq: 0xC10D76DD  Ack: 0x8C169D6E  Win: 0x4172  TcpLen: 44
TCP Options (6) => NOP NOP TS: 1583773 4119739296 NOP NOP Sack: 35862@40654
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Figure 1: Output of Ip Address connections on Snort.

```
05/26-07:24:42.870730 136.145.231.10:22 -> 70.45.132.54:49666
TCP TTL:64 TOS:0x10 ID:19532 IpLen:20 DgmLen:404 DF
***AP*** Seq: 0x8C177E8E  Ack: 0xC10D773D  Win: 0xE3  TcpLen: 32
TCP Options (3) => NOP NOP TS: 4119739689 1583848
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

05/26-07:24:42.870820 136.145.231.10:22 -> 70.45.132.54:49666
TCP TTL:64 TOS:0x10 ID:19533 IpLen:20 DgmLen:324 DF
***AP*** Seq: 0x8C177FEE  Ack: 0xC10D773D  Win: 0xE3  TcpLen: 32
TCP Options (3) => NOP NOP ^CTS: 4119739689 1583848
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

*** Caught Int-Signal
05/26-07:24:42.870959 136.145.231.10:22 -> 70.45.132.54:49666
TCP TTL:64 TOS:0x10 ID:19534 IpLen:20 DgmLen:484 DF
***AP*** Seq: 0x8C1780FE  Ack: 0xC10D773D  Win: 0xE3  TcpLen: 32
TCP Options (3) => NOP NOP TS: 4119739689 1583848
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

===============================================================================
Run time for packet processing was 10.455252 seconds
Snort processed 35153 packets.
Snort ran for 0 days 0 hours 0 minutes 10 seconds
   Pkts/sec:         3515
===============================================================================
Memory usage summary:
  Total non-mmapped bytes (arena):       782336
  Bytes in mapped regions (hblkhd):      12906496
  Total allocated space (uordblks):      668912
  Total free space (fordblks):           113424
  Topmost releasable block (keepcost):   97072
===============================================================================
Packet I/O Totals:
   Received:       552781
   Analyzed:        35153 (  6.359%)
    Dropped:       517651 ( 48.359%)
   Filtered:            0 (  0.000%)
Outstanding:       517628 ( 93.641%)
   Injected:            0
===============================================================================
```

```
===============================================================================
Breakdown by protocol (includes rebuilt packets):
        Eth:        35153 (100.000%)
       VLAN:            0 (  0.000%)
        IP4:        35151 ( 99.994%)
       Frag:            0 (  0.000%)
       ICMP:            0 (  0.000%)
        UDP:            2 (  0.006%)
        TCP:         6338 ( 18.030%)
        IP6:            0 (  0.000%)
    IP6 Ext:            0 (  0.000%)
   IP6 Opts:            0 (  0.000%)
      Frag6:            0 (  0.000%)
      ICMP6:            0 (  0.000%)
       UDP6:            0 (  0.000%)
       TCP6:            0 (  0.000%)
     Teredo:            0 (  0.000%)
    ICMP-IP:            0 (  0.000%)
    IP4/IP4:            0 (  0.000%)
    IP4/IP6:            0 (  0.000%)
    IP6/IP4:            0 (  0.000%)
    IP6/IP6:            0 (  0.000%)
        GRE:            0 (  0.000%)
    GRE Eth:            0 (  0.000%)
   GRE VLAN:            0 (  0.000%)
    GRE IP4:            0 (  0.000%)
    GRE IP6:            0 (  0.000%)
GRE IP6 Ext:            0 (  0.000%)
   GRE PPTP:            0 (  0.000%)
    GRE ARP:            0 (  0.000%)
    GRE IPX:            0 (  0.000%)
   GRE Loop:            0 (  0.000%)
       MPLS:            0 (  0.000%)
        ARP:            2 (  0.006%)
        IPX:            0 (  0.000%)
   Eth Loop:            0 (  0.000%)
   Eth Disc:            0 (  0.000%)
   IP4 Disc:        28811 ( 81.959%)
   IP6 Disc:            0 (  0.000%)
   TCP Disc:            0 (  0.000%)
   UDP Disc:            0 (  0.000%)
```

Figure 2: Packet Totals and Breakdown by protocol

Results:

In the last tech report we had 16851 failed connections on SSH using TCPDump. When when we used Snort we see the total of the packets received (552781 packets), packets analyzed (35153), packets dropped (517651), packets filtered (0), packets outstanding (517628) and packet injected (0).

Future Works:

We will create a program in python for save all information of the packets in a file.Then we will develop a program to monitor connections that does not interrupt the data transfer.

Conclusion:

The Science DMZ has the capacity to facilitate the data transfer of bid data, but since it doesn't have a firewall the user must be careful that their equipment is safe from network vulnerabilities. Using TCPdump we were able to analyze the packets on port 22 and recognize that the first two packets are part of the TCP connection and with Snort we can see all the information of these packets and detect the malicious packets. The analysis of such connections can help us to learn what are the vulnerabilities on the network and how to solve the problem efficiently.

Acknowledgments:

This work is supported by Academics and Training for the Advancement of Cybersecurity Knowledge in Puerto Rico (ATACK-PR), Scholarship for Cybersecurity Excellence.

Thanks to Humberto Ortiz Zuazaga for the opportunity for work with you.

References:

[1]General information about Science DMZ network: https://fasterdata.es.net/science-dmz/

[2] Architecture of Science DMZ network:
https://fasterdata.es.net/science-dmz/science-dmz-architecture/

[3] U Aickelin, J Twycross and T Hesketh-Robles, *Rule Generalisation using Snort,* School of Computer Science and IT, University of Nottingham, NG8 IBB UK.

[4] Daniel Sebastian Torres Vargas y el profesor Jean Polo Cequeda Olago, *INSTALACIÓN, CONFIGURACIÓN Y FUNCIONAMIENTO DEL IDS SNORT*, Universidad Francisco de Paula Santander, Facultad de Ingenieria, Ingenieria de Sistema, San Jose de Cucuta, 2012

[5] Rafeeq Ur Rehman, *Intrusion Detection Systems with Snort: Advanced IDS Techniques Using Snort, Apache, MySQL, PHP, and ACID* Chapter 1.

[6] What is Snort? Definition: http://www.webopedia.com/TERM/S/Snort.html

[7]Snort Official Page: https://www.snort.org/