International Journal of Computing and Business Research (IJCBR)

ISSN (Online) : 2229-6166

Volume 6 Issue 2 March 2015

EFFECTIVE IMPLEMENTATION OF DYNAMIC CLASSIFICATION FOR NETWORK FORENSIC AND TRAFFIC ANALYSIS

Manu Bansal

Assistant Professor

Department of IT

University Institute of Engineering & Technology

Panjab University, Sector 25, Chandigarh

ABSTRACT

Packet capturing is one of classical and most frequently used task performed by the network administrators. This is done to fetch the packets traveling in the network and finally detect any suspicious activity in the network. Finally, any out of the way activity or abnormal activity is analyzed by the intrusion detection system (IDS) tools for classification of attacks or type of the traffic. Enormous IDS tools are available including open source products which can classify the attacks or traffic from the PCAP (Packet Capture) Files fetched from honeypots or servers. This article explains about various aspects of PCAP and the detailed process of analyzing the PCAP using Snort IDS Tool for classification of traffic.

Keywords – Network Forensic using Data Mining, PCAP Analysis, Honeypot and Honeynets

International Journal of Computing and Business Research (IJCBR)

ISSN (Online) : 2229-6166

Volume 6 Issue 2 March 2015

INTRODUCTION

Network administrators generally face the regular issues of intrusion in their network by different media. To cope up with such issues, they make use of pcap (packet capture) that is having the application programming interface (API) for capturing the network traffic from different dimensions including ports, IP Addresses and associated parameters. In case of Unix-like systems, pcap is implemented in libpcap library. In case of Windows, it implements a port of libpcap that is known as WinPcap.

Whenever the honeypot is deployed for monitoring of network traffic, the monitoring or analyzing software can make use of libpcap or WinPcap for capturing of packets travelling over the specific network.

The base API of pcap is programmed in C Programming Language. For implementation of pcap in other programming languages such as Java, .NET Languages and other web based scripting languages they use a wrapper but we it should be noted that these wrappers are not provided by default by libpcap or WinPcap. In case of C++, the programs can link directly to the C API or makes use of an object-oriented wrapper.

The MIME type for the file format that is created and read by libpcap is *application/vnd.tcpdump.pcap*. The classical file extension for pcap is .pcap. In some tools,.cap and .dmp file extensions are also used.

libpcap and WinPcap are associated in terms of packet capturing as well as filtering engines with many open source and commercial network tools. These includes protocol analyzers (packet sniffers), network investigators, network IDES, traffic generators and network analyzers.

libpcap also support the feature so that the captured files can be exported and saved to a file. A captured file that is saved in the format that libpcap and WinPcap use can be easily analyzed by applications that understand that format, including tcpdump, Wireshark, NetworkMiner and many others.

TOOL FOR READING LIBPCAP

A number of tools are available in different regimes of network analysis and forensic investigation. Following is the list of tools touching different aspects of the network analysis which at the very base level makes use of libpcap

- **Tcpdump** : Tool for the capturing and dumping of packets for forensic and investigation
- **ngrep**, (Network Grep) : This tool shows packet data in user-friendly output scenario.
- **Wireshark** (Earlier it was named as Ethereal) : It is a GUI based packet capturing and protocol forensic tool.
- **Snort** : a network-intrusion-detection system.
- **Nmap**: a port-scanning and fingerprinting network utility
- **Bro IDS** : It is an IDS and network analysis tool
- **URL Snooper** : It is used to locate the addresses of audio and video files to allow recording them.
- **Iftop** : Used to display the usage of bandwidth in the network
- **EtherApe** : GUI Based tool for monitoring of network traffic and the usage of bandwidth in real time.
- **Bit-Twist** : Ethernet packet generator
- **Pirni** : A tool for network security tool used with jailbroken iOS devices.

- **Firesheep** : It is an extension for the Mozilla Firefox web browser. It intercepts the unencrypted cookies from different websites. It is also used for the session hijacking and network vulnerabilities.
- **Suricata** : Another network intrusion analysis and prevention platform.
- **WhatPulse** : A tool for statistical measurement (input, network, uptime) in the network.
- **Xplico** : Used as a NFAT (Network Forensics Analysis Tool)

SNORT

Snort is an open source tool written in C used as network intrusion prevention as well as the network intrusion detection system developed by Sourcefire. It is having excellent combination of the benefits of signature, protocol, and anomaly-based inspection. The statistics of Snort is huge with millions of downloads and nearly 400,000 registered users. Snort is not only the IDS but it is also used as an IPS (Intrusion Prevention System) for avoidance of any unwanted activity or unauthorized access of the resources. Snort can easily implement the protocol analysis and content investigation with number of other features. The excellent features of the tools includes detection of a variety of attacks and probes such as buffer overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts and many other.

The main configuration file is /etc/snort/snort.conf. In this configuration file, the actual information of the network or system is specified that is under investigation. All values and parameters are commented in the file so that the changes can be done very easily.

OTHER FREE INTRUSION DETECTION SYSTEMS (IDS)

- ACARM-ng
- AIDE
- Bro NIDS

- OSSEC HIDS
- Prelude Hybrid IDS
- Samhain
- Snort
- Suricata

DIFFERENT MODES OF SNORT FOR NETWORK FORENSIC

Snort can be configured in Three Different Modes :

1. **Sniffer**

   Sniffer mode is used to read the packets off of the network and display for the user in a continuous stream on the system.

2. **Packet Logger**

   Packet logger mode is used for the logging of the packets to the disk

3. **Network Intrusion Detection System**

   Network IDS mode is associated with configurable parameters which allow the tool to analyze network traffic for matching against a specific user defined rule set. Finally it performs several actions based upon what the tool investigates in the execution time.

SNIFFER MODE

To view only the TCP/IP packet headers to the screen, following command is used :

./snort -v

This command will execute Snort and show the IP and TCP/UDP/ICMP headers.

To display the application data in transit, the following is used :

./snort -vd

Using this command, Snort display the packet data as well as the headers.

To view a detailed analysis or description, showing the data link layer headers ::

./snort -vde

This command with different switched can also be executed as

./snort -d -v –e

PACKET LOGGER MODE

To record the packets to a disk, we need to specify the directory for logging. Snort will automatically execute into the packet logger mode:

./snort -dev -l ./log

Snort running in this mode will collect every data packet that it encounter and will places it in a directory hierarchy that is based upon the IP address of one of the hosts in the datagram.

If we specify a switch "-l", we will see that Snort uses the address of the remote computer as the directory in which it places packets. In some scenarios, it makes use of the local host address. For the logging of packets relative to the home network, we have to specify which network is the home network with the following command :

./snort -dev -l ./log -h 192.168.1.0/24

This command and rule will specify that we want to view the data link and TCP/IP headers as well as application data into the directory ./log, and you want to log the packets relative to the 192.168.1.0 class C network. All incoming data packets shall be recorded into the subdirectories of main directory log with the names of directory based on the address of the remote host. If we are working on a high speed network or if we want to log the packets into a compressed form for later analysis we consider logging in "binary mode".

In Binary mode, the logging of the packets is done in "tcpdump format" to a single binary file in the logging directory :

./snort -l ./log -b

We should note that we have not specified a home network in the command above because binary mode logs everything to a single file in which there is no need to specify how to format the output directory structure.

NETWORK INTRUSION DETECTION MODE

To run the tool in network intrusion detection (NIDS) mode, following command is executed :

./snort -dev -l ./log -h 192.168.1.0/24 -c snort.conf

snort.conf is the name of file having different rules. The rules set are applied from the snort.conf file to each packet to finally decide whether an action based upon the rule type in the file should

be taken. If we do not specify the output directory for the program, it will be /var/log/snort as default.

READING PCAP USING SNORT AND ALERT FILES

PCAP Files can be analyzed using Snort very easily. The PCAP file is passed with a Snort Command. Once the command is executed, Snort generates the alert file from that specific PCAP file.

To read a single pcap

```
$ snort -r mynetwork.pcap
$ snort --pcap-single= mynetwork.pcap
```

Read multiple pcap from a file

```
$ cat mypcaps.txt
pcap1.pcap
pcap2.pcap

$ snort --pcap-file=mypcaps.txt
```

This command will read pcap1.pcap, pcap2.pcap and all files under /home/mynetwork/pcaps. Snort will not attempt to check whether the files under that directory are really pcap files or not.

READ PCAPS FROM A COMMAND LINE LIST

```
$ snort --pcap-list="MyNetwork1.pcap MyNetwork2.pcap MyNetwork3.pcap"
```

This will read MyNetwork1.pcap, MyNetwork2.pcap and MyNetwork3.pcap.

READING PCAPS UNDER A DIRECTORY

```
$ snort --pcap-dir="/home/MyNetwork/pcaps"
```

This will include all of the files under /home/MyNetwork/pcaps.

USING FILTERS

```
$ cat MyNetwork.txt
MyNetwork1.pcap
MyNetwork2.pcap
```

Current Directory : /home/MyNetwork/pcaps

```
$ snort --pcap-filter="*.pcap" --pcap-file=MyNetwork.txt
$ snort --pcap-filter="*.pcap" --pcap-dir=/home/MyNetwork/pcaps
```

The above will only include files that match the shell pattern "*.pcap", in other words, any file ending in ".pcap".

```
$ snort --pcap-filter="*.pcap --pcap-file=MyNetwork.txt \
> --pcap-filter="*.cap" --pcap-dir=/home/MyNetwork/pcaps
```

In the above, the first filter "*.pcap" will only be applied to the pcaps in the file "MyNetwork.txt" (and any directories that are recursed in that file). The addition of the second filter "*.cap" will cause the first filter to be forgotten and then applied to the directory /home/MyNetwork/pcaps, so only files ending in ".cap" will be included from that directory.

```
$ snort --pcap-filter="*.pcap --pcap-file=MyNetwork.txt \
> --pcap-no-filter --pcap-dir=/home/MyNetwork/pcaps
```

In this example, the first filter will be applied to MyNetwork.txt, then no filter will be applied to the files found under /home/MyNetwork/pcaps, so all files found under /home/MyNetwork/pcaps will be included.

```
$ snort --pcap-filter="*.pcap --pcap-file=MyNetwork.txt \
> --pcap-no-filter --pcap-dir=/home/MyNetwork/pcaps \
> --pcap-filter="*.cap" --pcap-dir=/home/MyNetwork/pcaps2
```

In this example, the first filter will be applied to MyNetwork.txt, then no filter will be applied to the files found under /home/MyNetwork/pcaps, so all files found under /home/MyNetwork/pcaps will be included, then the filter "*.cap" will be applied to files found under /home/MyNetwork/pcaps2.

PRINTING THE PCAP

```
$ snort --pcap-dir=/home/MyNetwork/pcaps --pcap-show
```

The above example will read all of the files under /home/MyNetwork/pcaps and will print a line indicating which pcap is currently being read.

SNORT FULL ALERT FILE FORMAT

[**] [1:2010935:2] ET POLICY Suspicious inbound to MSSQL port 1433 [**]

[Classification: Potentially Bad Traffic] [Priority: 2]

07/25-20:31:31.817217 232.11.237.105:1000 -> 233.29.20.24:1433

TCP TTL:102 TOS:0x0 ID:256 IpLen:20 DgmLen:40

*******S* Seq: 0x43EE0000 Ack: 0x0 Win: 0x4000 TcpLen: 20*

[Xref => http://doc.emergingthreats.net/2010935]

[**] [1:2010935:2] ET POLICY Suspicious inbound to MSSQL port 1433 [**]

[Classification: Potentially Bad Traffic] [Priority: 2]

07/25-20:31:31.838622 17.1.27.05:6000 -> 23.12.20.17:1433

TCP TTL:103 TOS:0x0 ID:256 IpLen:20 DgmLen:40

*******S* Seq: 0x6D5F0000 Ack: 0x0 Win: 0x4000 TcpLen: 20*

[Xref => http://doc.emergingthreats.net/2010935]

[**] [1:2010935:2] ET POLICY Suspicious inbound to MSSQL port 1433 [**]

[Classification: Potentially Bad Traffic] [Priority: 2]

07/25-20:31:31.898603 17.1.27.5:6000 -> 23.19.20.22:1433

TCP TTL:103 TOS:0x0 ID:256 IpLen:20 DgmLen:40

*******S* Seq: 0x262F0000 Ack: 0x0 Win: 0x4000 TcpLen: 20*

[Xref => http://doc.emergingthreats.net/2010935]

From the above specified alert file that is fetched by Snort, we can plot the graphs or charts or any other pattern related to data mining.

EXTRACTION OF RELEVANT PATTERNS AND DATA FROM ALERT FILES USING NOTEPAD++

Once the alert file is generated, it can be opened in an Open Source Text Notepad++. In this tool, there is a unique feature of bookmarking of specific lines. To separate the lines having Classification, Priority, IPLen from Alert File, the feature of find and bookmark can be used.

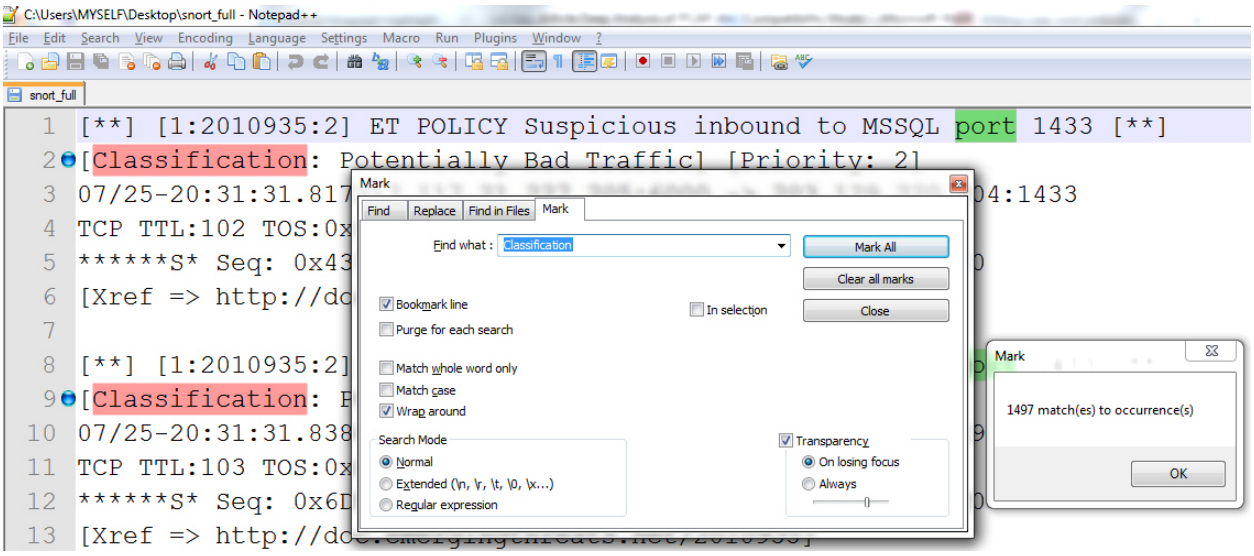

Figure 1 – Classification Process in Data Mining for Packet Analysis

Once the lines are marked with bookmarks, we can easily cut these selected lines

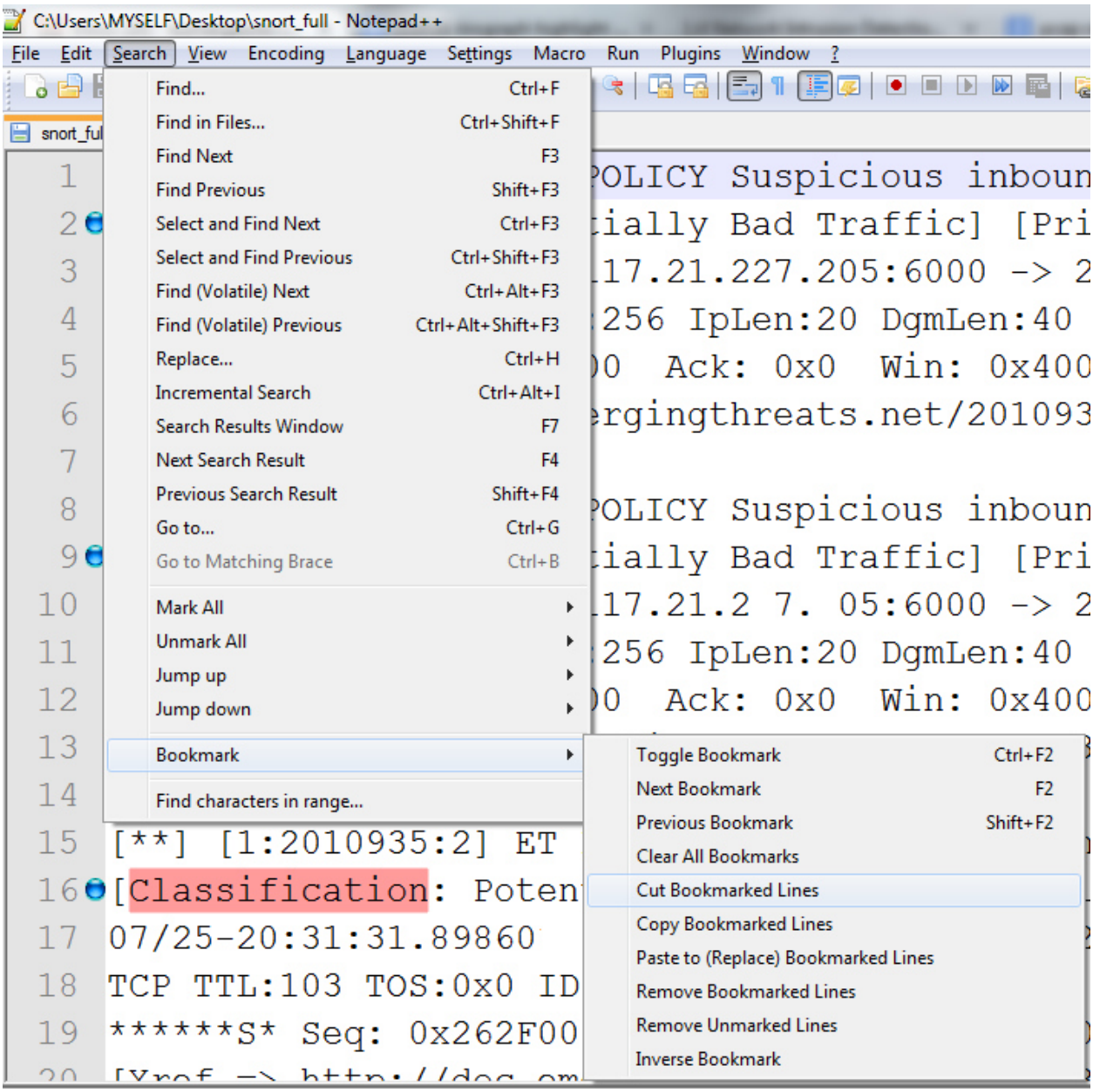

Figure 1 – Deep Investigation of the Keywords in PCAP Extracted Document

Finally, we will paste this lines to a separate file, and following file will be there :

[Classification: Potentially Bad Traffic] [Priority: 2] Dgmlen : 233

[Classification: Potentially Bad Traffic] [Priority: 2] Dgmlen : 233

[Classification: Potentially Bad Traffic] [Priority: 2] Dgmlen : 234

[Classification: Potentially Bad Traffic] [Priority: 2] Dgmlen : 232

[Classification: Potentially Bad Traffic] [Priority: 2] Dgmlen : 234

[Classification: Potentially Bad Traffic] [Priority: 2] Dgmlen : 231

[Classification: Potentially Bad Traffic] [Priority: 2] Dgmlen : 232

[Classification: Potentially Bad Traffic] [Priority: 2] Dgmlen : 231

[Classification: Potentially Bad Traffic] [Priority: 2] Dgmlen : 232

[Classification: Potentially Bad Traffic] [Priority: 2] Dgmlen : 230

Using this methodology, we can get the following type of sheet in the Spreadsheet Package :

Classification	**Priority**	**IPLen**	**dgmLen**
Attempted Administrator Privilege Gain	1	20	231
Potentially Bad Traffic	1	20	231

Attempted Administrator Privilege Gain	1	20	231
Potentially Bad Traffic	2	20	231
Attempted Administrator Privilege Gain	1	20	231
Attempted Administrator Privilege Gain	1	20	231
Potentially Bad Traffic	2	20	231
Attempted Administrator Privilege Gain	1	20	23
Attempted Administrator Privilege Gain	1	20	23
Potentially Bad Traffic	2	20	24

This method can be used to analyze any pcap captured with any packet capturing tool for detailed investigation of the packets and associated parameters.

CONCLUSION

The process of capturing and analyzing can be used by the research scholars as well as the practitioners in the stream of network administration and security. Additionally, the generation and detailed investigation of alert files fetched from pcap gives the signature patterns that are used by the intrusion detection systems in real world scenario.

REFERENCES

[1] Huang, X., Wang, J., Aluru, S., Yang, S. P., & Hillier, L. (2003). PCAP: a whole-genome assembly program. *Genome research*, *13*(9), 2164-2170.

[2] Lamping, U., & Warnicke, E. (2004). Wireshark User's Guide. *Interface*, *4*, 6.

[3] McRee, R. (2006). Security Analysis with Wireshark. *ISSA Journal*, 39-45.

[4] Mitra, A., Najjar, W., & Bhuyan, L. (2007, December). Compiling pcre to fpga for accelerating snort ids. In *Proceedings of the 3rd ACM/IEEE Symposium on Architecture for networking and communications systems* (pp. 127-136). ACM.

[5] Li, S., Torresen, J., & Soraasen, O. (2003, April). Exploiting reconfigurable hardware for network security. In *Field-Programmable Custom Computing Machines, 2003. FCCM 2003. 11th Annual IEEE Symposium on* (pp. 292-293). IEEE.

[6] Patton, S., Yurcik, W., & Doss, D. (2001). An Achilles' heel in signature-based IDS: Squealing false positives in SNORT. *Proceedings of RAID 2001*.

[7] Yang, G., Rong, C., & Dai, Y. (2004). A Distributed Honeypot System for Grid Security. In *Grid and Cooperative Computing* (pp. 1083-1086). Springer Berlin Heidelberg.