Final Report of Network Security Research

M.S. INCS student Assawakomenkool, Nont

M.S. CSCI student Patel, Yash

Final Report of Network Security Research

The goal of this Research paper is to catch an adversary on the red hand, not just defense and stay quiet.

Exploitations

In the exploitation topic, we will pick up 10 examples of exploitation to work on this research paper which are Port scanning, DDoS/TCP SYN flood, Brute force, DRb remote code execution, Java RMI Server Insecure Default Configuration Java Code Execution, WordPress XMLRPC DoS, VSFTPD v2.3.4 Backdoor Command Execution, PHP Utility Belt – Remote Code Execution, Anonymous login (Samba Client) backdoor exploit, and Unrealircd 3.2.8.1 backdoor command execution. The example exploitations' testing result is on the separate file which is named "**exploitationManual_rev4.pdf**".

The idea of exploitation testing is shown as model in figure 1.

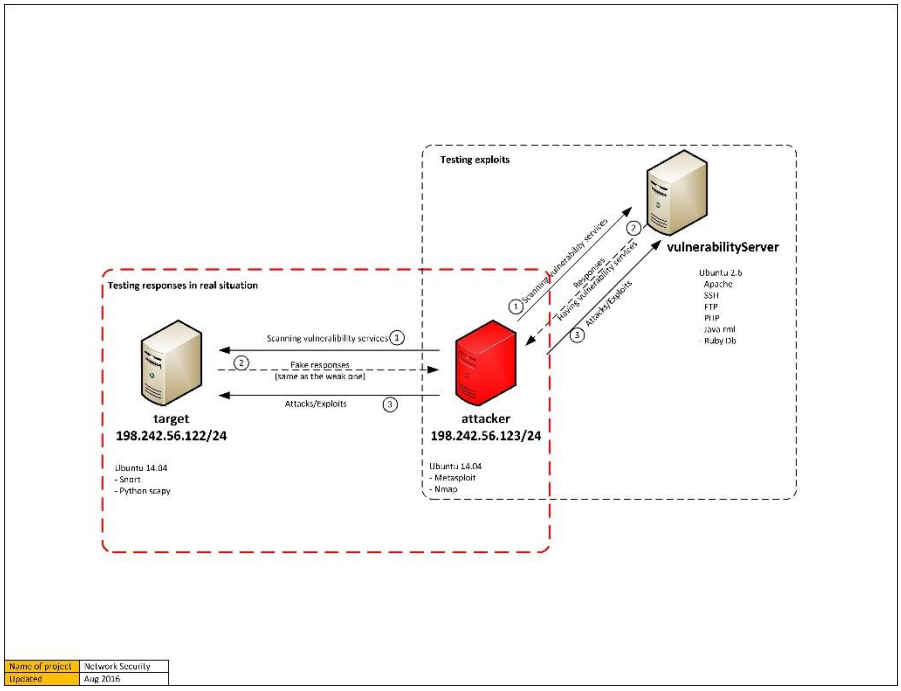

Figure 1 Model of exploitation testing

The adversary should not able to compromise a testing server which is maned "target", but the attacker will receive the same weak point information as receiving from the vulnerability Server that contain a number of weak services. For simplicity, finally we have scripted all the attack payload together and made it automated script file which can directly exploit any payload as per the choice taken.

Detecting and responding

The detecting and responding section, we use “Snort” to achieve our target. Snort service is run on the target server to detect, analyze, and respond network packets that send to/from the server. The model of how Snort service stand on the Target server is shown as figure 2 and the example of packets that we use for our case study is shown as figure 3.

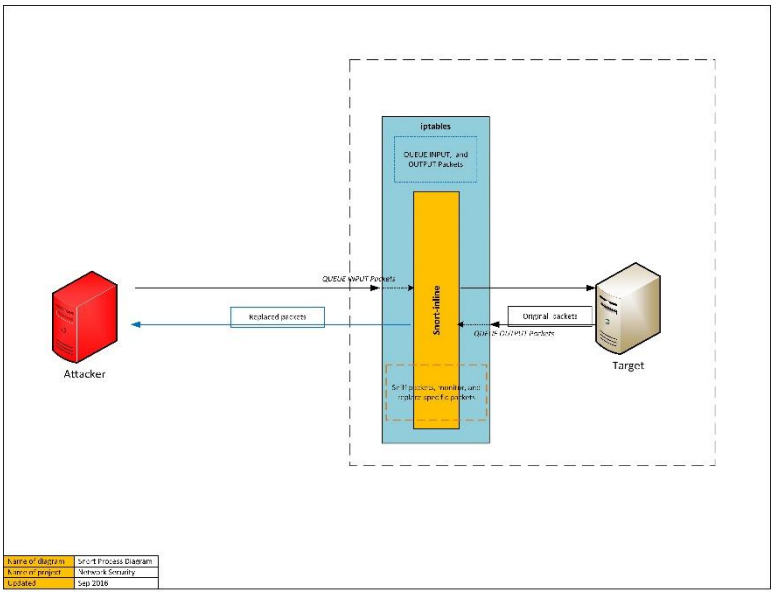

Figure 2 Working model of Snort service

Source	Destination	Protocol	Length	Info
198.242.56.123	198.242.56.122	TCP	60	60296→22 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
198.242.56.122	198.242.56.123	TCP	58	22→60296 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS
198.242.56.123	198.242.56.122	TCP	60	60296→22 [RST] Seq=1 Win=0 Len=0
198.242.56.123	198.242.56.122	TCP	60	60297→22 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
198.242.56.122	198.242.56.123	TCP	58	22→60297 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS
198.242.56.123	198.242.56.122	TCP	60	60297→22 [RST] Seq=1 Win=0 Len=0
198.242.56.123	198.242.56.122	TCP	74	51524→22 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_
198.242.56.122	198.242.56.123	TCP	74	22→51524 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS
198.242.56.123	198.242.56.122	TCP	66	51524→22 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=17
198.242.56.122	198.242.56.123	SSH	109	Server: Protocol (SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ul
198.242.56.123	198.242.56.122	TCP	66	51524→22 [ACK] Seq=1 Ack=44 Win=29312 Len=0 TSval=1
198.242.56.123	198.242.56.122	TCP	66	51524→22 [FIN, ACK] Seq=1 Ack=44 Win=29312 Len=0 TS
198.242.56.122	198.242.56.123	TCP	66	22→51524 [FIN, ACK] Seq=44 Ack=2 Win=29056 Len=0 TS
198.242.56.123	198.242.56.122	TCP	66	51524→22 [ACK] Seq=2 Ack=45 Win=29312 Len=0 TSval=1

Figure 3 Example packets

We analyzed a number of packets and tried many ways to get our goal and we end up with Snort’s rules that can modify contents in the payloads of the packets. The Snort’s rules are shown as below.

```
alert tcp any any -> any any (msg:"SYN FIN Scan"; flags: SF;sid:9000000;)
alert tcp any any -> any any (msg:"FIN Scan"; flags: F;sid:9000001;)
alert tcp any any -> any any (msg:"NULL Scan"; flags: 0;sid:9000002;)
alert tcp any any -> any any (msg:"XMAS Scan"; flags: FPU;sid:9000003;)
```

```
alert tcp any any -> any any (msg:"Full XMAS Scan"; flags:
SRAFPU;sid:9000004;)
alert tcp any any -> any any (msg:"URG Scan"; flags: U;sid:9000005;)
alert tcp any any -> any any (msg:"URG FIN Scan"; flags: FU;sid:9000006;)
alert tcp any any -> any any (msg:"PUSH FIN Scan"; flags: FP;sid:9000007;)
alert tcp any any -> any any (msg:"URG PUSH Scan"; flags: PU;sid:9000008;)
alert tcp any any -> any any (flags: A; ack: 0; msg:"NMAP TCP
ping!";sid:9000009;)
```

Snort's rule can also detect an abnormal connection by tracking number of packets per time period as the rule below.

```
alert tcp 198.242.56.123 any -> $HOME_NET any (msg:"SCAN Port Detected-
(connection limited)"; detection_filter: track by_src, count 30, seconds 60;
sid:1000006; rev:2;)
```

Before we end up with the Snort's rules, we tried to use Python and Scapy to modify the contents and it is success, but the modified packets get to the Attacker later than the original packet (It means the original packets is still sent out). So, the Attacker does not accept those modified packets. The code below is a Python code that can modify the content.

```
#! /usr/bin/env python
import logging
logging.getLogger("scapy").setLevel(1)

from scapy.all import *

packet = IP(src="198.242.56.122", dst="198.242.56.123",
ttl=64)/ICMP(type=0)/"Hello+This is response packet injected by testvm1@FOR
TESTING"
send(packet)
packet.show()

print "\nDOne\n"

-----------------------------
#!/usr/bin/env python
import os
import logging
logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
import scapy
from scapy.all import *

def pkt_callback(pkt):

        if TCP in pkt:
                if pkt[IP].src=="198.242.56.123" and
pkt[IP].dst=="198.242.56.122":
                    print"\nTCP-REQUEST"
                        sn = sr1(src="198.242.56.122",
dst="198.242.56.123")/TCP(sport=pkt[TCP].dport, dport=pkt[TCP].sport,
```

```
seq=pkt[TCP].ack, ack=pkt[TCP].seq, flags=pkt[TCP].flags,
window=pkt[TCP].window, options=pkt[TCP].options)/Raw(load=pkt[Raw].load))
                        sn.show()
                        send(IP(src="198.242.56.122",
dst="198.242.56.123")/TCP(sport=sn.sport, dport=sn.dport, seq=sn.seq,
ack=sn.ack, flags="PA", window=sn.window, options=sn.options)/Raw(load="SSH-
2.0-OpenSSH_4.7p1 Debian-8ubuntu1\r\n"))

        pkt.show()

while True:
        sniff(iface="eth0", prn=pkt_callback, filter="tcp and (port 22) and
tcp.flags.syn==0 and tcp.flags.ack==1" , count=1)
```

Feature extraction from datasets: Extracting Dataset

For the next step we have collected dataset which is network traffic for 30days of time period on development server. We applied our dataset on bellowed extraction program and get csv file with proper data table. We have also summarized and compare with recent research features on intrusion detection systems for anomaly method. We have also collected and documented list of features with can be useful to get feature which can more accurate for our research project.

Since we have our dataset in Binary format, we use Java to extract the features from the binary file to CSV format. The Java code is shown as below.

```
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.net.InetAddress;
import java.net.UnknownHostException;

import org.jnetpcap.Pcap;
import org.jnetpcap.nio.JMemory;
import org.jnetpcap.packet.JFlow;
import org.jnetpcap.packet.JFlowKey;
import org.jnetpcap.packet.JFlowMap;
import org.jnetpcap.packet.JHeader;
import org.jnetpcap.packet.JMemoryPacket;
import org.jnetpcap.packet.JPacket;
import org.jnetpcap.packet.JPacketHandler;
import org.jnetpcap.packet.JScanner;
import org.jnetpcap.packet.PcapPacket;
import org.jnetpcap.protocol.lan.Ethernet;
import org.jnetpcap.protocol.network.Ip4;
import org.jnetpcap.protocol.tcpip.Http;
import org.jnetpcap.protocol.tcpip.Http.Request;
import org.jnetpcap.protocol.tcpip.Http.Response;
import org.jnetpcap.protocol.tcpip.Tcp;
```

```
import org.jnetpcap.protocol.tcpip.Udp;
import org.jnetpcap.protocol.voip.Sip.ContentType;

import com.mysql.jdbc.UpdatableResultSet;

public class pcapParser {

	public static void main(String[] args) throws Exception {

		String name ="Previous_snort_tcpdump.log.1477131539";
		//String name ="DNS_missconfigure_snort_tcpdump.log.1478280656";
		//String name = "ftp-transfer.pcapng";

		String csvFile = "D:\\Extract_csv\\"+name+".csv";
		FileWriter cwriter = new FileWriter(csvFile);


		//PrintWriter writer = new PrintWriter("D:\\Extract_txt\\"+name+".txt",
"UTF-8");

		final String FILENAME = "D:\\log\\"+name;
		final StringBuilder errbuf = new StringBuilder();


		final Pcap pcap = Pcap.openOffline(FILENAME, errbuf);

		if (pcap == null) {
			System.err.println(errbuf); // Error is stored in errbuf if any
			return;
		}
		else{

		CSVUtils.writeLine(cwriter,
Arrays.asList("packet_type","frame","ether_dest","ether_source","ether_offset","ether_
type","","ip_dest","ip_source","offset","length","dst_port","source_port","seq","ack",
"hlen","reserved","flags","window","checksum","tcp_urgent","http_offset","http_length"
,"http_request","http_response","mss_offset","mss_length","mss_code","mss","win_offset
","win_length","win_code","win_scale","time_offset","time_code","time_length","time_ts
vl","time_tsecr"));

		pcap.loop(100, new JPacketHandler<StringBuilder>() {

			final Ip4 ip = new Ip4();
			final Tcp tcp = new Tcp();
			final Http http = new Http();
			final Udp udp = new Udp();
			final Ethernet  ether = new Ethernet();


			public void nextPacket(JPacket packet, StringBuilder errbuf) {

				System.out.printf("frame #%d ", packet.getFrameNumber());
			//	writer.println("frame #"+ packet.getFrameNumber());
				String packet_getFrame = packet.getFrameNumber()+"";

				if (packet.hasHeader(Tcp.ID)){
```

```
                                if(packet.hasHeader(ip) && packet.hasHeader(tcp) &&
packet.hasHeader(http)){

                                        packet.getHeader(http);
                                        packet.getHeader(ip);
                                        packet.getHeader(tcp);
                                        packet.getHeader(ether);

                            //          writer.println("http header::"+
http.toString());

                                        System.out.println("HTTP-PKT");

                                        String ether_offset = ether.getOffset()+"";
                                        String ether_dest =
etherEndPointStr((ether.destination()));
                                        String ether_source =
etherEndPointStr((ether.source()));
                                        String ether_type = ether.type()+"";

                                        //System.out.println(ether_time);

                                        String ip_dest = tcpEndPointStr(ip.source());
                                        String ip_source =
tcpEndPointStr(ip.destination());

                                        String tcp_offset = tcp.getOffset()+"";
                                        String tcp_length = tcp.getLength()+"";
                                        String tcp_destport = tcp.destination()+"";
                                        String tcp_sourceport = tcp.source()+"";
                                        String tcp_seq = tcp.seq()+"";
                                        String tcp_ack = tcp.ack()+"";
                                        String tcp_hlen = tcp.hlen()+"";
                                        String tcp_reserved = tcp.reserved()+"";
                                        String tcp_flag = tcp.flags()+"";
                                        String tcp_window = tcp.window()+"";
                                        String tcp_checksum = tcp.checksum()+ "";
                                        String tcp_urgent = tcp.urgent()+"";

                                        String http_offset = http.getOffset()+"";
                                        String http_length = http.getLength()+"";

                                        String req_method =
http.fieldValue(Request.RequestMethod);
                                String req_url = http.fieldValue(Request.RequestUrl);
                                String req_ver = http.fieldValue(Request.RequestVersion);
                                String req_host = http.fieldValue(Request.Host);
                                String req_user = http.fieldValue(Request.User_Agent);
                                String req_accept = http.fieldValue(Request.Accept);
                                String req_lan =
http.fieldValue(Request.Accept_Language);
                                String req_encode =
http.fieldValue(Request.Accept_Encoding);
                                String req_cookie = http.fieldValue(Request.Cookie);
                                String req_connection =
http.fieldValue(Request.Connection);
                                String req_cache =
http.fieldValue(Request.Cache_Control);
```

```
                                String req_date = http.fieldValue(Request.Date);

                                String res_code = http.fieldValue(Response.ResponseCode);
                                String res_msg =
http.fieldValue(Response.ResponseCodeMsg);
                                String res_server = http.fieldValue(Response.Server);
                                String res_acc = http.fieldValue(Response.Accept_Ranges);
                                String res_con =
http.fieldValue(Response.Content_Length);
                                String res_cache =
http.fieldValue(Response.Cache_Control);
                                String res_expire = http.fieldValue(Response.Expires);

                                String http_req =
(req_method+req_url+req_ver+req_host+req_user+req_accept+req_lan+req_encode+req_cookie
+req_connection+req_cache+req_date).replaceAll(",", "//");
                                String http_res =
(res_code+res_msg+res_server+res_acc+res_cache+res_expire).replaceAll(",", "//");

                                String mss_offset="",mss_length =
"",mss_code="",mss_m="",win_offset="",win_length="",win_code="",win_scale="",time_code
="",time_offset="",time_tsval =
"",time_tsecr="",time_length=""/*,http_req="",http_res=""*/;

                                        for (JHeader subheader : tcp.getSubHeaders())
{
                                                if (subheader instanceof Tcp.MSS) {
                                            Tcp.MSS mss = (Tcp.MSS) subheader;
                                            mss_offset = mss.getOffset()+"";
                                            mss_length = mss.length()+"";
                                            mss_code = mss.code()+"";
                                            mss_m = mss.mss()+"";
                                    }
                                                if (subheader instanceof
Tcp.WindowScale) {
                                            Tcp.WindowScale win = (Tcp.WindowScale)
subheader;
                                            win_offset = win.getOffset()+"";
                                            win_length = win.length()+"";
                                            win_code =  win.code()+"";
                                            win_scale = win.scale()+"";
                                    }
                                                if (subheader instanceof Tcp.Timestamp)
{
                                            Tcp.Timestamp time = (Tcp.Timestamp) subheader;
                                            time_code = time.code()+"";
                                            time_offset = time.getOffset()+"";
                                            time_length = time.length()+"";
                                            time_tsval = time.tsval()+"";
                                            time_tsecr = time.tsecr()+"";
                                    }
                                        }

                                        try {
                                                CSVUtils.writeLine(cwriter,
Arrays.asList("HTTP",packet_getFrame,ether_dest,ether_source,ether_offset,ether_type,i
p_dest,ip_source,tcp_offset,tcp_length,tcp_destport,tcp_sourceport,tcp_seq,tcp_ack,tcp
_hlen,tcp_reserved,tcp_flag,tcp_window,tcp_checksum,tcp_urgent,http_offset,http_length
```

```
,http_req,http_res,mss_offset,mss_length,mss_code,mss_m,win_offset,win_length,win_code
,win_scale,time_offset,time_code,time_length,time_tsval,time_tsecr));
                                        } catch (IOException e) {
                                               // TODO Auto-generated catch block
                                               e.printStackTrace();
                                        }

                                 }

                                 else if(packet.hasHeader(ip) && packet.hasHeader(tcp)
&& !packet.hasHeader(http)){

                                        packet.getHeader(tcp);
                                        packet.getHeader(ip);
                                        packet.getHeader(ether);

                                        //writer.println("tcp header::"+
tcp.toString());

                                        System.out.println("TCP-PKT");

                                        String ip_dest = tcpEndPointStr(ip.source());
                                        String ip_source =
tcpEndPointStr(ip.destination());
                                        String tcp_offset = tcp.getOffset()+"";
                                        String tcp_length = tcp.getLength()+"";
                                        String tcp_destport = tcp.destination()+"";
                                        String tcp_sourceport = tcp.source()+"";
                                        String tcp_seq = tcp.seq()+"";
                                        String tcp_ack = tcp.ack()+"";
                                        String tcp_hlen = tcp.hlen()+"";
                                        String tcp_reserved = tcp.reserved()+"";
                                        String tcp_flag = tcp.flags()+"";
                                        String tcp_window = tcp.window()+"";
                                        String tcp_checksum = tcp.checksum()+"";
                                        String tcp_urgent = tcp.urgent()+"";
                                        String mss_offset="",mss_length =
"",mss_code="",mss_m="",win_offset="",win_length="",win_code="",win_scale="",time_code
="",time_offset="",time_tsval = "",time_tsecr="",time_length="";

                                        String ether_offset = ether.getOffset()+"";
                                        String ether_dest =
etherEndPointStr((ether.destination()));
                                        String ether_source =
etherEndPointStr((ether.source()));
                                        String ether_type = ether.type()+"";

                                 //     System.out.println(ether_dest);

                                        for (JHeader subheader : tcp.getSubHeaders())
{
                                               if (subheader instanceof Tcp.MSS) {
                                    Tcp.MSS mss = (Tcp.MSS) subheader;
                                    mss_offset = mss.getOffset()+"";
                                    mss_length = mss.length()+"";
                                    mss_code = mss.code()+"";
                                    mss_m = mss.mss()+"";
                                 }
```

```
                                        if (subheader instanceof
Tcp.WindowScale) {
                                    Tcp.WindowScale win = (Tcp.WindowScale)
subheader;
                                    win_offset = win.getOffset()+"";
                                    win_length = win.length()+"";
                                    win_code =  win.code()+"";
                                    win_scale = win.scale()+"";
                                }
                                        if (subheader instanceof Tcp.Timestamp)
{
                                    Tcp.Timestamp time = (Tcp.Timestamp) subheader;
                                    time_code = time.code()+"";
                                    time_offset = time.getOffset()+"";
                                    time_length = time.length()+"";
                                    time_tsval = time.tsval()+"";
                                    time_tsecr = time.tsecr()+"";
                                }
                                   }

                                   try {

                                        //CSVUtils.writeLine(cwriter,
Arrays.asList("HTTP",packet_getFrame,ip_dest,ip_source,tcp_offset,tcp_length,tcp_destp
ort,tcp_sourceport,tcp_seq,tcp_ack,tcp_hlen,tcp_reserved,tcp_flag,tcp_window,tcp_check
sum,tcp_urgent,http_offset,http_length,http_request,http_response,mss_offset,mss_lengt
h,mss_code,mss_m,win_offset,win_length,win_code,win_scale,time_offset,time_code,time_l
ength,time_tsval,time_tsecr));
                                        CSVUtils.writeLine(cwriter,
Arrays.asList("TCP",packet_getFrame,ether_dest,ether_source,ether_offset,ether_type,ip
_dest,ip_source,tcp_offset,tcp_length,tcp_destport,tcp_sourceport,tcp_seq,tcp_ack,tcp_
hlen,tcp_reserved,tcp_flag,tcp_window,tcp_checksum,tcp_urgent,"","","","",mss_offset,m
ss_length,mss_code,mss_m,win_offset,win_length,win_code,win_scale,time_offset,time_cod
e,time_length,time_tsval,time_tsecr));
                                   } catch (IOException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                   }

                               }
                               else{
                                   System.out.println("NO-PKT");
                                   packet.getHeader(ip);
                                   packet.getHeader(tcp);
                                   packet.getHeader(udp);
                                   packet.getHeader(ether);

                                   String ether_offset = ether.getOffset()+"";
                                   String ether_dest =
etherEndPointStr((ether.destination()));
                                   String ether_source =
etherEndPointStr((ether.source()));
                                   String ether_type = ether.type()+"";

                                   String ip_dest = tcpEndPointStr(ip.source());
                                   String ip_source =
tcpEndPointStr(ip.destination());
                                   String tcp_offset = tcp.getOffset()+"";
```

```
                                        String tcp_length = tcp.getLength()+"";
                                        String tcp_destport = tcp.destination()+"";
                                        String tcp_sourceport = tcp.source()+"";
                                        String tcp_seq = tcp.seq()+"";
                                        String tcp_ack = tcp.ack()+"";
                                        String tcp_hlen = tcp.hlen()+"";
                                        String tcp_reserved = tcp.reserved()+"";
                                        String tcp_flag = tcp.flags()+"";
                                        String tcp_window = tcp.window()+"";
                                        String tcp_checksum = tcp.checksum()+"";
                                        String tcp_urgent = tcp.urgent()+"";
                                        String mss_offset="",mss_length =
"",mss_code="",mss_m="",win_offset="",win_length="",win_code="",win_scale="",time_code
="",time_offset="",time_tsval = "",time_tsecr="",time_length="";

                                        for (JHeader subheader : tcp.getSubHeaders())
{
                                                if (subheader instanceof Tcp.MSS) {
                                            Tcp.MSS mss = (Tcp.MSS) subheader;
                                            mss_offset = mss.getOffset()+"";
                                            mss_length = mss.length()+"";
                                            mss_code = mss.code()+"";
                                            mss_m = mss.mss()+"";
                                    }
                                                if (subheader instanceof
Tcp.WindowScale) {
                                            Tcp.WindowScale win = (Tcp.WindowScale)
subheader;
                                            win_offset = win.getOffset()+"";
                                            win_length = win.length()+"";
                                            win_code =  win.code()+"";
                                            win_scale = win.scale()+"";
                                    }
                                                if (subheader instanceof Tcp.Timestamp)
{
                                            Tcp.Timestamp time = (Tcp.Timestamp) subheader;
                                            time_code = time.code()+"";
                                            time_offset = time.getOffset()+"";
                                            time_length = time.length()+"";
                                            time_tsval = time.tsval()+"";
                                            time_tsecr = time.tsecr()+"";
                                    }
                                        }

                                        try {
                                                CSVUtils.writeLine(cwriter,
Arrays.asList("NO",packet_getFrame,ether_dest,ether_source,ether_offset,ether_type,ip_
dest,ip_source,tcp_offset,tcp_length,tcp_destport,tcp_sourceport,tcp_seq,tcp_ack,tcp_h
len,tcp_reserved,tcp_flag,tcp_window,tcp_checksum,tcp_urgent,"","","","",mss_offset,ms
s_length,mss_code,mss_m,win_offset,win_length,win_code,win_scale,time_offset,time_code
,time_length,time_tsval,time_tsecr));
                                        } catch (IOException e) {
                                                // TODO Auto-generated catch block
                                                e.printStackTrace();
                                        }


                                }
```

```
                    }
                           else{

                                  System.out.println("UDP-PKT");
                                  packet.getHeader(udp);
                                  packet.getHeader(ip);
                                  packet.getHeader(ether);
                           //     writer.println("udp header::"+ udp.toString());

                                  String ether_offset = ether.getOffset()+"";
                                  String ether_dest =
etherEndPointStr((ether.destination()));
                                  String ether_source =
etherEndPointStr((ether.source()));
                                  String ether_type = ether.type()+"";

                                  String ip_dest = tcpEndPointStr(ip.source());
                                  String ip_source = tcpEndPointStr(ip.destination());

                                  String udp_destport = udp.destination()+"";
                                  String udp_sourceport = udp.source()+"";
                                  String udp_offset = udp.getOffset()+"";
                                  String udp_length = udp.length()+"";
                                  String udp_checksum = udp.checksum()+"";
                                  String udp_hlen = udp.getHeaderLength()+"";

                                  try {
                                         CSVUtils.writeLine(cwriter,
Arrays.asList("UDP",packet_getFrame,ether_dest,ether_source,ether_offset,ether_type,ip
_dest,ip_source,udp_offset,udp_length,udp_destport,udp_sourceport,"","",udp_hlen,"","
","",udp_checksum,"","","","","","","","","","","","","","","","","","","","","",""));
                                  } catch (IOException e) {
                                         // TODO Auto-generated catch block
                                         e.printStackTrace();
                                  }

                           }
                    }
             }, errbuf);

             pcap.close();
             //writer.close();

             cwriter.flush();
          cwriter.close();



             }
      }

      private static String tcpEndPointStr(byte addrBytes[]) {
       String addr;
       try {
           addr = InetAddress.getByAddress(addrBytes).getHostAddress();
       } catch (UnknownHostException ex) {
           addr = "0.0.0.0";
       }
```

```
        return addr;
    }

      private static String etherEndPointStr(byte addrBytes[]) {
        String addr;
       // System.out.println(addrBytes);
        try {
            addr = InetAddress.getByAddress(addrBytes).getHostAddress();
        } catch (UnknownHostException ex) {
            addr = "0:0:0:0";
        }
        return addr;
    }
}
```

The example extracted feature CSV file from the binary file is shown as figure 4.

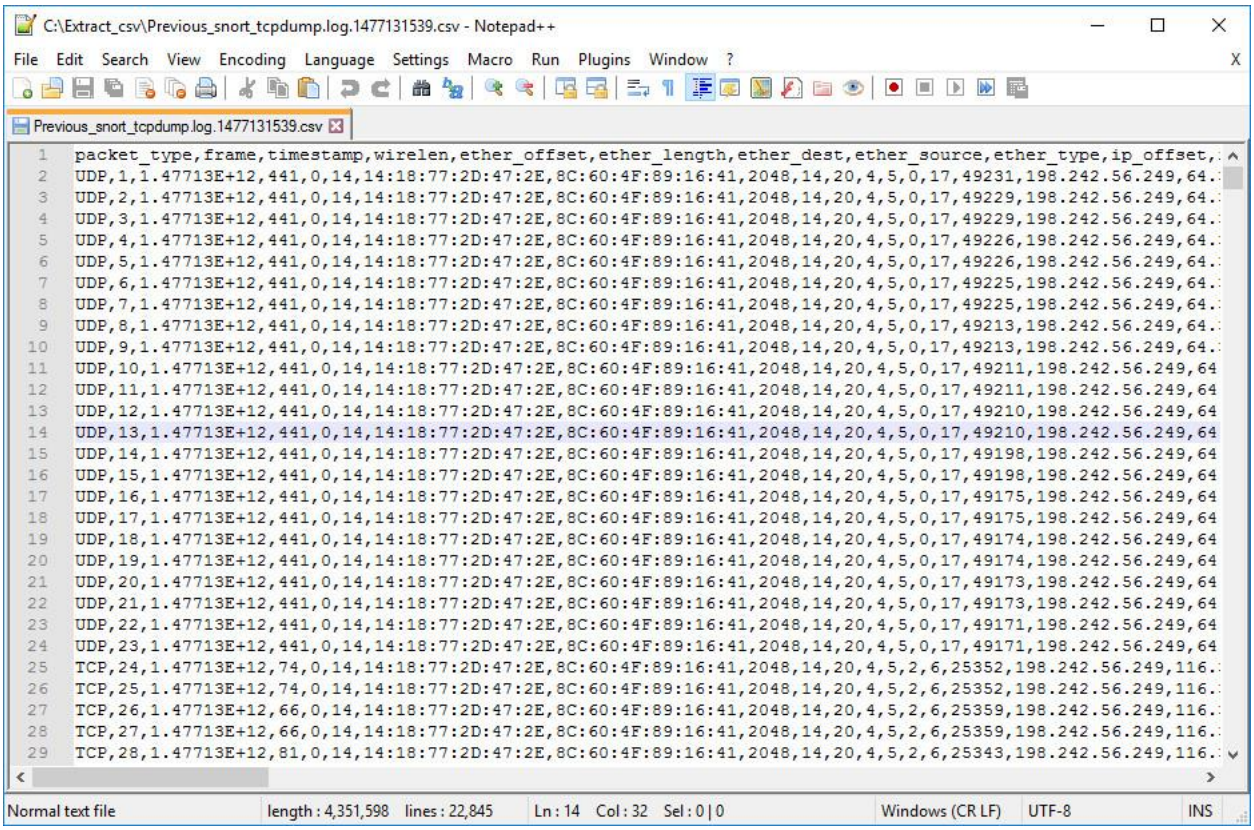

Figure 4 Extracted feature in CSV format from Dataset

References

https://www.safaribooksonline.com/library/view/snort-cookbook/0596007914/ch04s06.html

http://read.pudn.com/downloads111/sourcecode/windows/network/464726/snort-2.8.1/snort-2.8.1/doc/README.flow-portscan__.htm

http://books.gigatux.nl/mirror/snortids/0596006616/snortids-CHP-5-SECT-3.html

http://seclists.org/snort/2008/q4/109

https://www.snort.org/faq/readme-sfportscan

http://resources.infosecinstitute.com/snort-network-recon-techniques/#article

https://www.theseus.fi/bitstream/handle/10024/109245/Snort%20Thesis%20Bezborodov%20Sergey.pdf?sequence=1

http://commons.oreilly.com/wiki/index.php/Snort_Cookbook/Rules_and_Signatures

http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node24.html#reload:nonreloadable

http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node33.html

http://www.informit.com/articles/article.aspx?p=101171&seqNum=6

http://resources.infosecinstitute.com/snort-rule-writing-for-the-it-professional-part-2-2/

https://www.snort.org/faq/readme-stream5

https://pdfs.semanticscholar.org/c961/e76770c600e323e901989ac80d11690db676.pdf

http://www.netresec.com/?page=PcapFiles