**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.

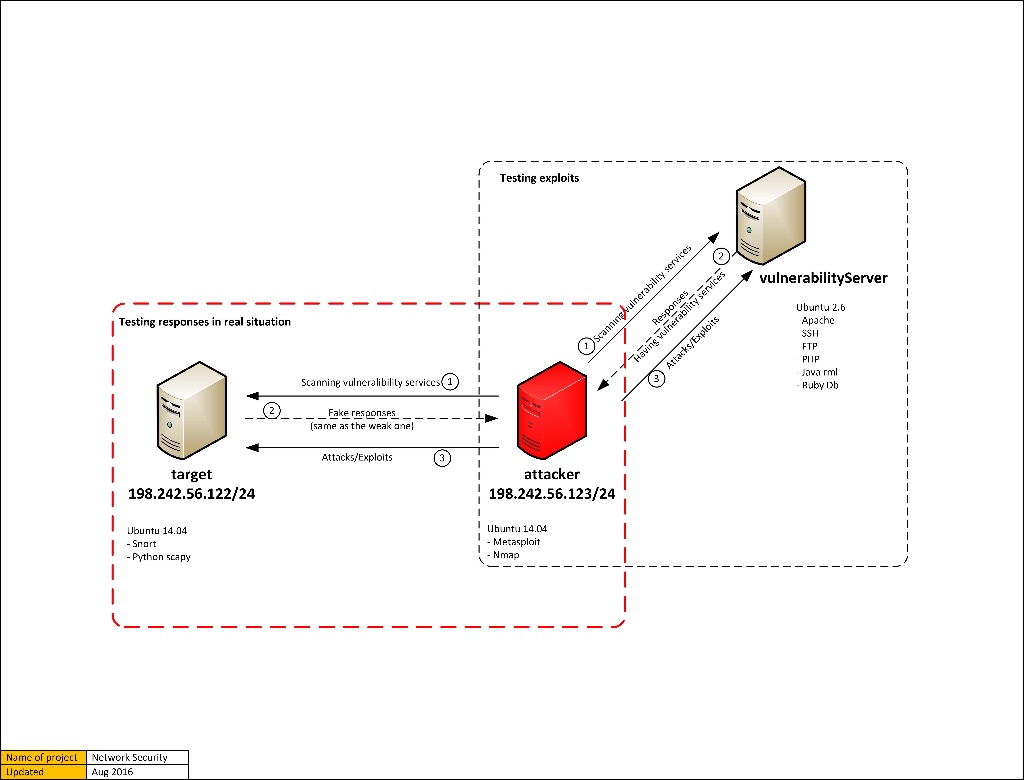


Figure 1 Model of exploitation testing

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.

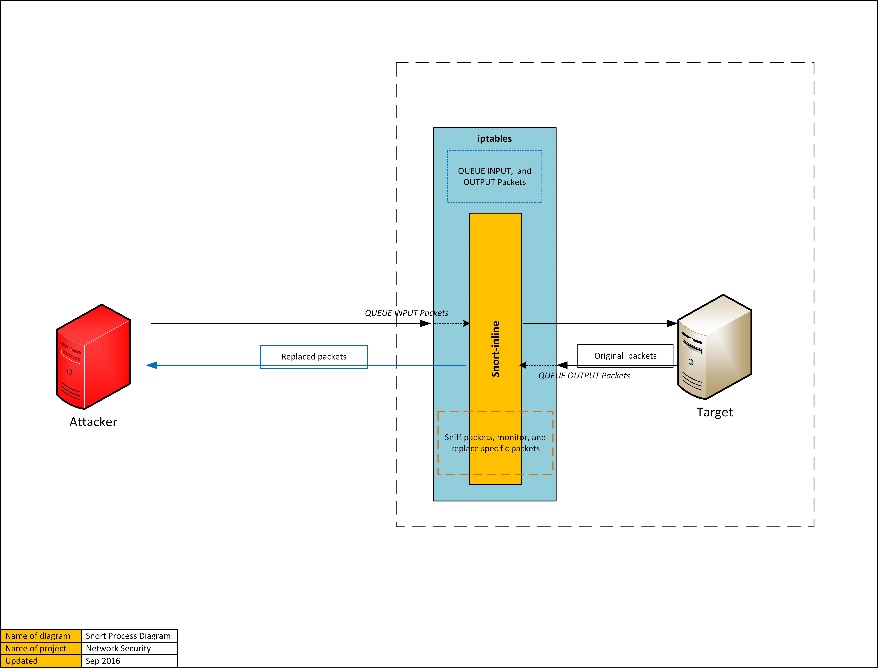


Figure 2 Working model of Snort service

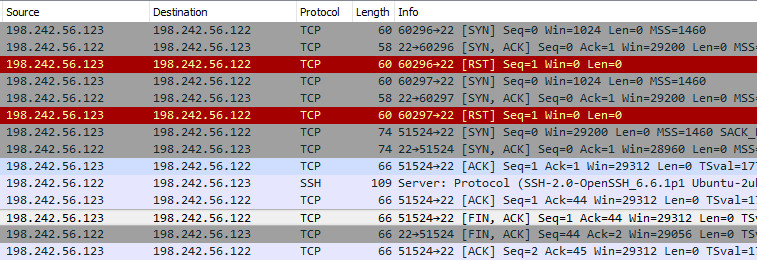


Figure 3 Example packets

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\_tsvl","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,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,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\_destport,tcp\_sourceport,tcp\_seq,tcp\_ack,tcp\_hlen,tcp\_reserved,tcp\_flag,tcp\_window,tcp\_checksum,tcp\_urgent,http\_offset,http\_length,http\_request,http\_response,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));

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,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{

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\_hlen,tcp\_reserved,tcp\_flag,tcp\_window,tcp\_checksum,tcp\_urgent,"","","","",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{

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.

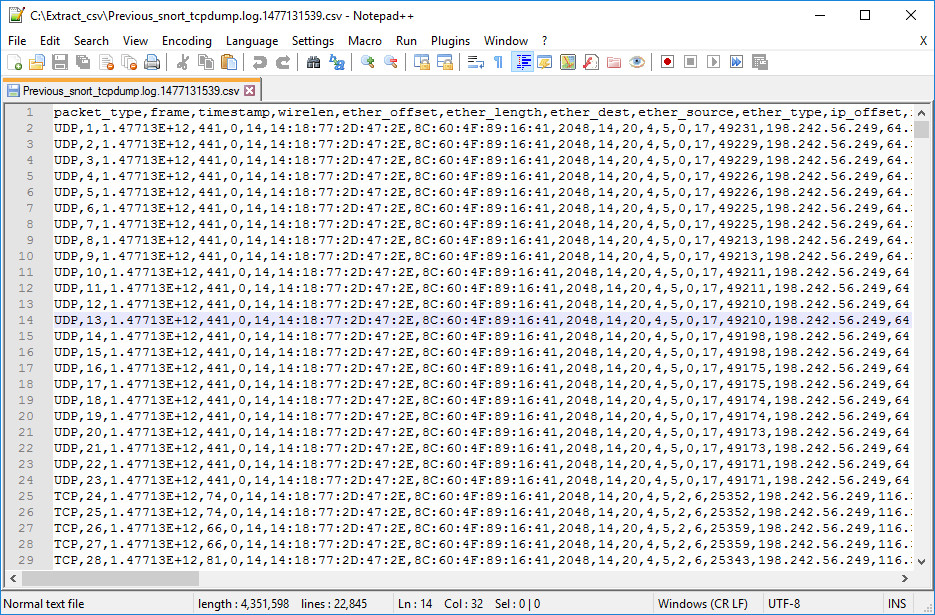


Figure 4 Extracted feature in CSV format from Dataset

**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>