Automated Net Flow Heuristic Signature Production

Andrew Gill
Glasgow Caledonian University
agill200@caledonian.ac.uk

Abstract. Network threat detection is a common event that occurs on networks daily. As part of this process packet inspection must take place, the process consists of manual inspection using PCAP files, identifying and analysing suspicious activity or potentially how a breach on a network has occurred. Even through the use of tools like Wireshark the process can still be a monotonous one, consisting of analysing hundreds of thousands, possibly even millions of data packets. As a result the rate of detection of false positives is high. Is paper investigates the challenges in combining heuristic packet analysis with Intrusion Detection Systems and how the process of mitigating false positives can be automated.

Keywords: Intrusion Detection Systems, Signature Heuristics, Threat Detection

1 Introduction

Every day more than 250,000 new instances of malware are created and distributed across the internet (Lyne, 2013).
As a result of this, each individual strain creates a unique potential threat to enterprise networks. One of the main challenges faced by network administrators is identifying malicious packets before they have a chance to spread across a network and cause severe damage. Currently, the standard defence against this is achieved by, implementing two methods of perimeter and internal security: firewalls and Intrusion Detection Systems (IDS).

Firewalls can be setup for monitoring of external attacks and prevent them from happening, Firewalls, by definition allow control of traffic on a network. They do this by restricting how ports can be accessed from data packets coming either from inbound and outbound traffic. In the case of malware, firewalls can block a strain of malware that has infected a system from contacting a command and control server (C2 Servers). By performing egress filtering, firewalls can control outbound traffic by allowing specified ports and hosts. (Brenton, 2006)

Intrusion Detection Systems are applied when a threat makes it through the firewall. They can be setup to analyse packet headers and signatures in order to identify potential threats before they pass across a network. Intrusion Detection Systems (IDS) also look for attacks that originate from an internal system.

An issue that Intrusion Detection Systems face is the growing occurrences and large volume of false positives which they generate (Victor et al, 2009). How can false positives be avoided and removed from a system? By integrating Deep Packet Inspection (DPI) and heuristics with Intrusion Detection Systems, an automated solution could be created to lower the rate of false positives. The benefit of lowering the rate of false positives is that it would free up Network Administrators' time, allowing them to focus on potentially anomalous or malicious traffic.

Deep Packet Inspection is a technology mainly used in large scale infrastructure. It uses a process of inspecting the contents of packets at a high level, looking for malicious code passing over a network in real time. This is done to improve performance, by looking at multiple strings within packets, a baseline of what is considered normal traffic can be reached and, therefore malicious code and activity has a higher chance of being identified.

'Heuristics' refers to the act or process of finding or discovering. It originates from the early 19th century and is derived from the Greek word 'heuriskein'. In Computer Science 'signature heuristics' refers to generating checksum information of data strings and packets on the network, then create identifying properties using this information to help an IDS determine whether a string of data is potentially a threat or not. Heuristics have been implemented into AntiVirus successfully but there is a lack of support for them in Intrusion Detection Systems.

This paper investigates the challenges in combining heuristic packet analysis with Intrusion Detection Systems and how the process of mitigating false positives can be automated.

2 Literature Review

To date literature covering the topic of automating false positive mitigation and signature production is sparse. This review will focus on material covering Intrusion Detection Systems, Deep Packet Inspection, signature heuristics and the reduction of false positives.

Intrusion Detection Systems are implemented in many large scale infrastructures, when traffic on a network is partly matched against known attack signatures, an intrusion detection system will output a false positive or false negative (Victor, 2009). It is not plausible to rely on an Intrusion Detection System to successfully, and correctly, classify events that flow on a network or on the system it has been deployed. Perfect and concise intrusion detection, like its counterpart perfect security, is simply not a current obtainable goal due to the complexity and rapid technological evolution of modern systems. However, with this in consideration, an IDS can strive to raise the bar and compete more thoroughly with attackers by reducing large scale attacks, in addition to increasing the steps required for an attacker to achieve complete system exploitation. As previous research on the classification of false positives and false negatives has shown, there are methods of identifying threats to an organisation's internal network by applying different methods and techniques for detection.

Deep Packet Inspection when used with an IDS can improve the accuracy of detection rates. By inspecting the packet contents it allows a system to become more efficient at classifying traffic as false positive, known good and false negative. One of the main challenges that DPI faces is when dealing with traffic that has encrypted content the system cannot read the contents and therefore is unable to decide if the packets are safe or not. This can be overcome by placing DPI behind a decryption device meaning packet data will be decrypted before it can inspect the information located inside (AbuHmed et al, 2008). Another issue DPI faces is the problem when signatures crossover. Some Intrusion Detection Systems categorize signatures based on the protocol that the packet has been transmitted on. Therefore there are times when a signature may match multiple categories and this therefore causes signature overlapping (AbuHmed et al, 2008).

Signature based detection is one of the most fundamental techniques for identifying malicious activity on your network (Atlasis, 2013). Signature heuristics create an effective solution in that it allows a system to look for specific patterns in net flow data and match it against known good and bad signatures, in doing this an intrusion detection system is able to potentially correct, protect and educate itself against unknown threats. Heuristics implementation into an IDS is important as it allows for large amounts traffic to be filtered and any packets with suspicious contents or headers matching will be flagged.

3 Hypothesis and Methodology

If all malicious code is known, hashes can be searched at packet level in real time and anomalous activity will be highlighted more accurately.

In order to simulate a network compromise or malicious activity on a network, a large scale network infrastructure packet capture file was built from samples to profile how a real attack would be carried out. Wireshark was used to complete this task. The PCAP file includes signatures that match known command and control servers (C2 Servers), simulated attacks across a network and transmissions of malicious packets. Using the same technology and similar techniques that IDS use this methodology proposes to simplify and create a solution. To eliminate false positives by utilising heuristics using predefined checksums. Thus this system would compare hashed known malicious code with a sample of traffic from a network similarly hashed at the data level of a packet.

By comparing both sets of hashes it can be derived whether a packet on the network contains known malicious code. Comparison of sets of hashes is computationally light compared to full packet inspection which is heavy, or header packet inspection can be used which generates false positives. One potential benefit of this system is that it could reduce the rate of false positives. By actively scanning all packets IDS identifies, packets containing malicious code would be almost immediately detected. This would increase the effectiveness of threat detection on a given network. After implementing this system two classifications of traffic are left: traffic that will still appear as false positive but is now can be whitelisted as good and anomalous traffic which could potentially be malicious

By extending this approach in real time across a network it is possible to scan for known malicious code at the data level of a packet. This would increase the speed of detection; allow for faster response time; making the system quicker and more efficient.

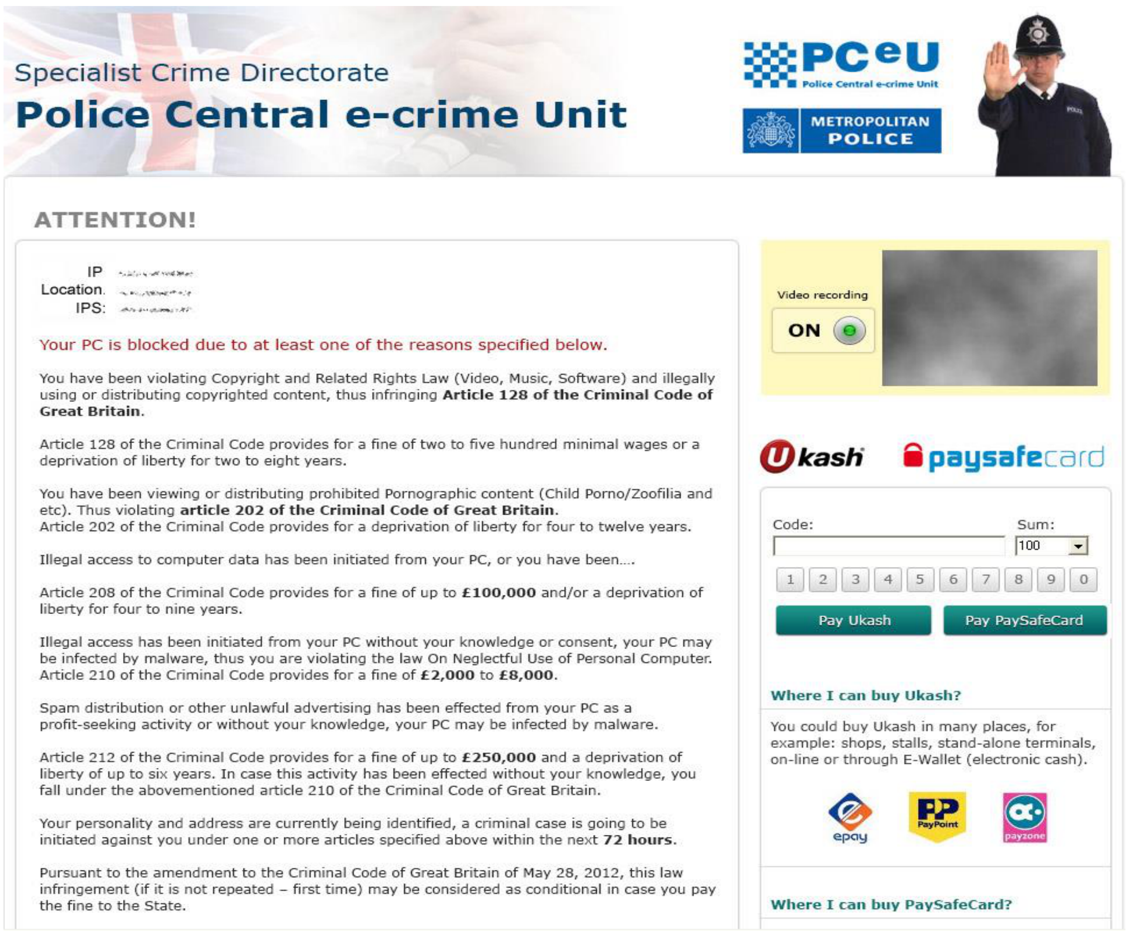
3.1 Testing

This was demonstrated by executing a couple strains of malware on a virtual machine running windows 7 x64, allowing each strand to run for five minutes causing it to run through its procedure of encryption and infection of the target machine. Then monitoring all the traffic generated by communication between the target machine and the host server using Wireshark. Sample packet capture files were also analysed to look for evidence of malicious code. From the analysis specific details were extracted from groups of packets and hashed using the following algorithms to keep integrity: MD5, SHA1 and SHA256. These hashes were then checked against the different strands to determine if any patterns could be identified.

Secondly network traffic was created by simulating communication between a server hosting malicious code and a victim machine, this was achieved by running two virtual machines on a network and running Wireshark on the host machine to monitor traffic. The machine used to simulate a server was running kali and used the apache2 service as a web server to host malicious code samples locally. The target machine running windows 7 was used to access the malicious webserver via the IP address of the server. The network traffic that contained the target machine (Windows 7 VM) accessing the malicious links on the server was captured using wireshark. The specific traffic containing the target connecting, accessing and proceeding to download was then hashed and added to a pool of known malicious and unusual traffic.

3.2 Malware Research

During research there were many strands of malware that were analysed but the cryptolocker variants were chosen for their high volume of network traffic produced. By looking at other malware strains such as the uKash virus or commonly known as the Metropolitan police virus but there are also many different forms and shapes the malware can take. Interpol Virus, FBI, Interpol Virus, MoneyPak, Ukash, yorkshire, UK, Ice, Department of Justice, White screen, jak usunąć, Ransom ware Reloaded, Winlock, Reveton, Zeus, Paysafecard, Metropolitan Police Service, Police National E-Crime Unit, and Votre ordinateur a ete pour violation de la loi Française are all different types of Ransom ware that does the same thing., it was apparent that initial infection was the only identifiable network traffic produced, where the virus dials home from a Trojan dropper exe file and downloads the ransom ware, from this the user is hit with a screen similar to the one shown in fig 1.



ATTENTION!

IP [redacted]
Location. [redacted]
IPS: [redacted]

Your PC is blocked due to at least one of the reasons specified below.

You have been violating Copyright and Related Rights Law (Video, Music, Software) and illegally using or distributing copyrighted content, thus infringing **Article 128 of the Criminal Code of Great Britain**.

Article 128 of the Criminal Code provides for a fine of two to five hundred minimal wages or a deprivation of liberty for two to eight years.

You have been viewing or distributing prohibited Pornographic content (Child Porno/Zoofilia and etc). Thus violating **article 202 of the Criminal Code of Great Britain**.
Article 202 of the Criminal Code provides for a deprivation of liberty for four to twelve years.

Illegal access to computer data has been initiated from your PC, or you have been....

Article 208 of the Criminal Code provides for a fine of up to **£100,000** and/or a deprivation of liberty for four to nine years.

Illegal access has been initiated from your PC without your knowledge or consent, your PC may be infected by malware, thus you are violating the law On Neglectful Use of Personal Computer. Article 210 of the Criminal Code provides for a fine of **£2,000** to **£8,000**.

Spam distribution or other unlawful advertising has been effected from your PC as a profit-seeking activity or without your knowledge, your PC may be infected by malware.

Article 212 of the Criminal Code provides for a fine of up to **£250,000** and a deprivation of liberty of up to six years. In case this activity has been effected without your knowledge, you fall under the abovementioned article 210 of the Criminal Code of Great Britain.

Your personality and address are currently being identified, a criminal case is going to be initiated against you under one or more articles specified above within the next **72 hours**.

Pursuant to the amendment to the Criminal Code of Great Britain of May 28, 2012, this law infringement (if it is not repeated – first time) may be considered as conditional in case you pay the fine to the State.

Where I can buy Ukash?

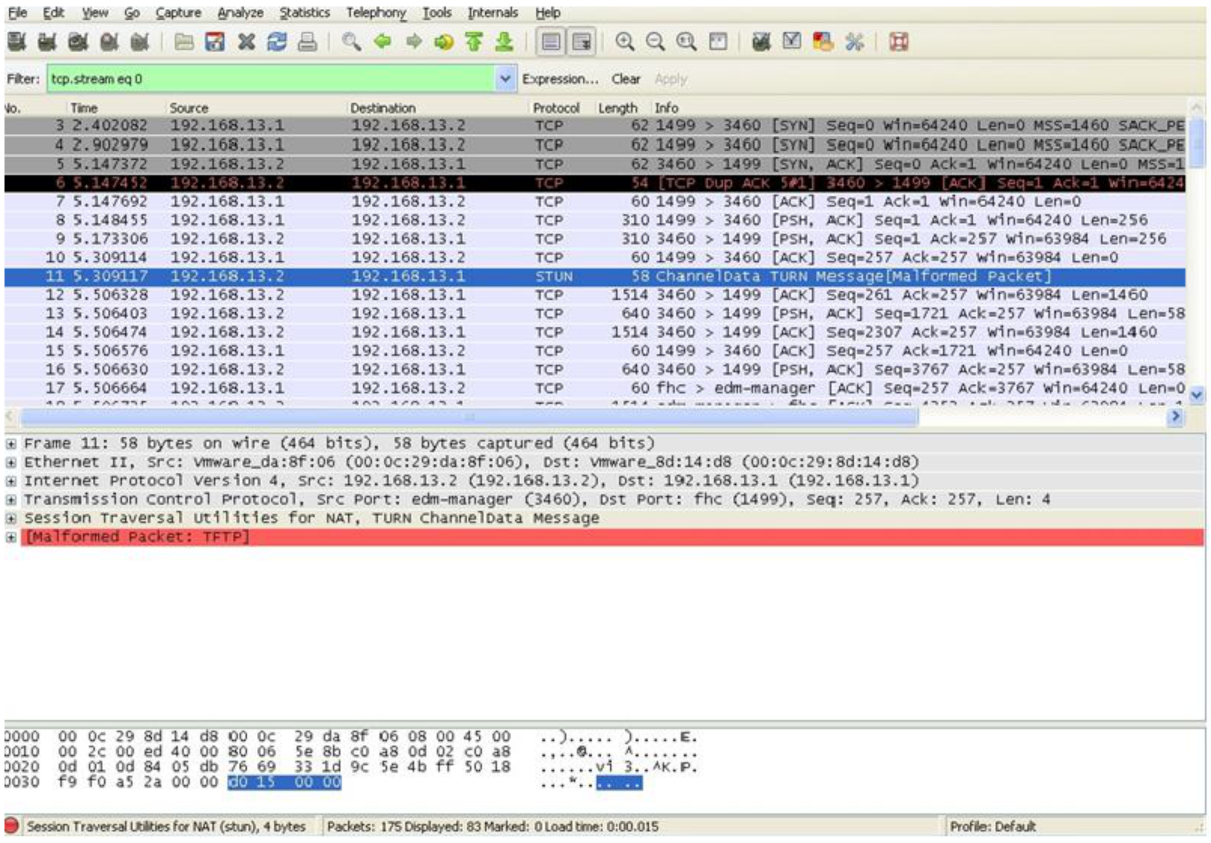
You could buy Ukash in many places, for example: shops, stalls, stand-alone terminals, on-line or through E-Wallet (electronic cash).

Where I can buy PaySafeCard?

figure 1

The malware then goes on to grab the location and IP data from the computer's registry, although this information is not always 100% accurate. As a result of reverse engineering the malware it is apparent that this particular strain that was analysed aims to scare the user as much as possible, it does this by tapping into the user's machine's hardware and accessing the webcam to show the user.

One strand of malware that was used for analysis also in this paper was the poison Ivy remote access Trojan (RAT) also commonly referred to as an Advanced Persistent Threat(APT), the way in which this strain works is that it is in two parts: the client program and the host. The client is usually sent via an e-mail or instant message attachment as a word document or picture slideshow, from this the infection only occurs when the user runs the infected document or picture file then the attached payload then runs. The network traffic generated by poison ivy when the program is initially run consists of communication between the host and victim machine. When the stream of messages stops, Wireshark was stopped and here is the output:


File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: tcp.stream eq 0
Expression... Clear Apply

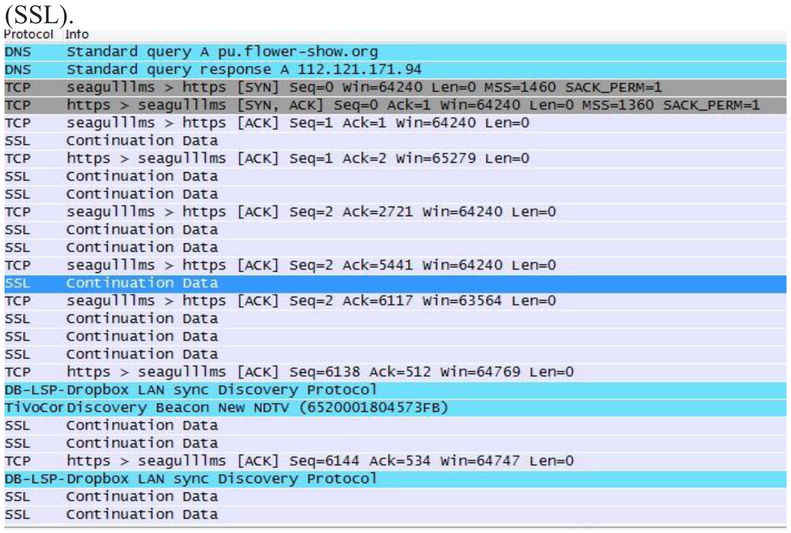
```
No.  Time      Source        Destination   Protocol Length Info
  3 2.402082  192.168.13.1  192.168.13.2  TCP        62 1499 > 3460 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PE
  4 2.902979  192.168.13.1  192.168.13.2  TCP        62 1499 > 3460 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PE
  5 5.147372  192.168.13.2  192.168.13.1  TCP        62 3460 > 1499 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1
  6 5.147452  192.168.13.2  192.168.13.1  TCP        54 [TCP Dup ACK 5#1] 3460 > 1499 [ACK] Seq=1 Ack=1 Win=6424
  7 5.147692  192.168.13.1  192.168.13.2  TCP        60 1499 > 3460 [ACK] Seq=1 Ack=1 Win=64240 Len=0
  8 5.148455  192.168.13.1  192.168.13.2  TCP       310 1499 > 3460 [PSH, ACK] Seq=1 Ack=1 Win=64240 Len=256
  9 5.173306  192.168.13.2  192.168.13.1  TCP       310 3460 > 1499 [PSH, ACK] Seq=1 Ack=257 Win=63984 Len=256
 10 5.309114  192.168.13.1  192.168.13.2  TCP        60 1499 > 3460 [ACK] Seq=257 Ack=257 Win=63984 Len=0
 11 5.309117  192.168.13.2  192.168.13.1  STUN       58 ChannelData TURN Message[Malformed Packet]
 12 5.506328  192.168.13.2  192.168.13.1  TCP      1514 3460 > 1499 [ACK] Seq=261 Ack=257 Win=63984 Len=1460
 13 5.506403  192.168.13.2  192.168.13.1  TCP       640 3460 > 1499 [PSH, ACK] Seq=1721 Ack=257 Win=63984 Len=58
 14 5.506474  192.168.13.2  192.168.13.1  TCP      1514 3460 > 1499 [ACK] Seq=2307 Ack=257 Win=63984 Len=1460
 15 5.506576  192.168.13.1  192.168.13.2  TCP        60 1499 > 3460 [ACK] Seq=257 Ack=1721 Win=64240 Len=0
 16 5.506630  192.168.13.2  192.168.13.1  TCP       640 3460 > 1499 [PSH, ACK] Seq=3767 Ack=257 Win=63984 Len=58
 17 5.506664  192.168.13.1  192.168.13.2  TCP        60 fhc > edm-manager [ACK] Seq=257 Ack=3767 Win=64240 Len=0

⊞ Frame 11: 58 bytes on wire (464 bits), 58 bytes captured (464 bits)
⊞ Ethernet II, Src: Vmware_da:8f:06 (00:0c:29:da:8f:06), Dst: Vmware_8d:14:d8 (00:0c:29:8d:14:d8)
⊞ Internet Protocol Version 4, Src: 192.168.13.2 (192.168.13.2), Dst: 192.168.13.1 (192.168.13.1)
⊞ Transmission Control Protocol, Src Port: edm-manager (3460), Dst Port: fhc (1499), Seq: 257, Ack: 257, Len: 4
⊞ Session Traversal Utilities for NAT, TURN ChannelData Message
⊞ [Malformed Packet: TFTP]

0000  00 0c 29 8d 14 d8 00 0c  29 da 8f 06 08 00 45 00   ..)..... ).....E.
0010  00 2c 00 ed 40 00 80 06  5e 8b c0 a8 0d 02 c0 a8   .,..@... ^.......
0020  0d 01 0d 84 05 db 76 69  33 1d 9c 5e 4b ff 50 18   ......vi 3..^K.P.
0030  f9 f0 a5 2a 00 00 d0 15  00 00                     ...*.. ....

Session Traversal Utilities for NAT (stun), 4 bytes | Packets: 175 Displayed: 83 Marked: 0 Load time: 0:00.015 | Profile: Default
```

As a result of monitoring network traffic generated by poison ivy for a period of five minutes, it was shown that all communication and commands between the host and the target was identified as using Secure Socket Layer (SSL).



```
Protocol Info
DNS      Standard query A pu.flower-show.org
DNS      Standard query response A 112.121.171.94
TCP      seagulllms > https [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
TCP      https > seagulllms [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1360 SACK_PERM=1
TCP      seagulllms > https [ACK] Seq=1 Ack=1 Win=64240 Len=0
SSL      Continuation Data
TCP      https > seagulllms [ACK] Seq=1 Ack=2 Win=65279 Len=0
SSL      Continuation Data
SSL      Continuation Data
TCP      seagulllms > https [ACK] Seq=2 Ack=2721 Win=64240 Len=0
SSL      Continuation Data
SSL      Continuation Data
TCP      seagulllms > https [ACK] Seq=2 Ack=5441 Win=64240 Len=0
SSL      Continuation Data
TCP      seagulllms > https [ACK] Seq=2 Ack=6117 Win=63564 Len=0
SSL      Continuation Data
SSL      Continuation Data
SSL      Continuation Data
TCP      https > seagulllms [ACK] Seq=6138 Ack=512 Win=64769 Len=0
DB-LSP-Dropbox LAN sync Discovery Protocol
TiVoCor Discovery Beacon New NDTV (6520001804573FB)
SSL      Continuation Data
SSL      Continuation Data
TCP      https > seagulllms [ACK] Seq=6144 Ack=534 Win=64747 Len=0
DB-LSP-Dropbox LAN sync Discovery Protocol
SSL      Continuation Data
SSL      Continuation Data
```

The capture file above shows communication between the Command and control server (pu.flower-show.org) also referred to as the host part of the program. When communication in the form of *continuous data* is shown the protocol it uses is SSL and from previous background knowledge on malware analysis this can be categorised as malicious commands from the host or attacker. From background research into the way poison ivy works it is apparent that the RAT uses the Camellia algorithm to encrypt its communications to the target machine. The Camellia algorithm is a symmetric-key block cipher, specifically with poison ivy it uses a 256-Bit key for maximum self-defence of the malware.

4 Results

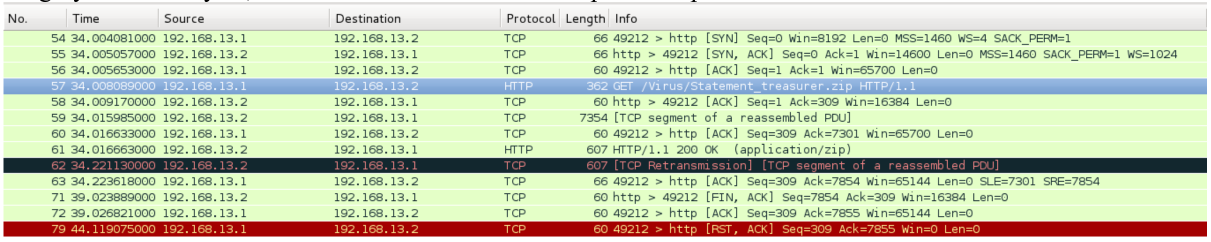
Firstly analysis of a piece of malware named crypto defender showed evidence that when the malware was first run it attempts to contact Microsoft's crypto service in order to generate the unique RSA-2048 key it uses to encrypt a user's files. It was observed that the first few attempts at communication are sent over plain http so the packet content was extracted and the Uniform Resource Identifier (URI) was hashed in order to create a list of potential malicious URI links that can be matched against other malware. The next step the malware takes is it sends a get request for a random string of text from what can be assumed is the command and control server (C2) as there are multiple communication channels after the get request is sent. After further analysis the string is a unique machine identifier that the malware uses to assign a unique payment page to that specific machine. After the connection is made all traffic to and from the machine are sent over HTTPS on port 443.

Secondly looking at PCAP sample files from a few sources, it was clear when certain types of malicious activity occur on a network, an example of this was observed when a piece of malware is unable to contact it's C2 server it begins to spam communication over HTTP to shopping sites and spam sites, this was observed as a second potential payload, not activated unless encryption was not possible. This activity was collectively hashed as well.

Thirdly a well-known Remote Access Trojan(RAT) named poison ivy was inspected, all of the traffic generated by this strand of malware was encrypted using Camellia as an algorithm and the traffic was sent over HTTPS with SSL(secure socket layer) therefore the packet contents could not be inspected, instead the characteristics of the packets were noted. It was shown that poison ivy was using the TCP protocol on port 3460, so from this a rule on an IDS can be created to look for traffic communicating on that port. Also from the description tab of a packet poison Ivy uses initially to communicate with its commanding server, "*pu.flower-show.org*" was identified as a known C2 server. Although communications were encrypted some defining fingerprints of communications being initiated and data flow beginning were collected, hashed and added to list of signatures.

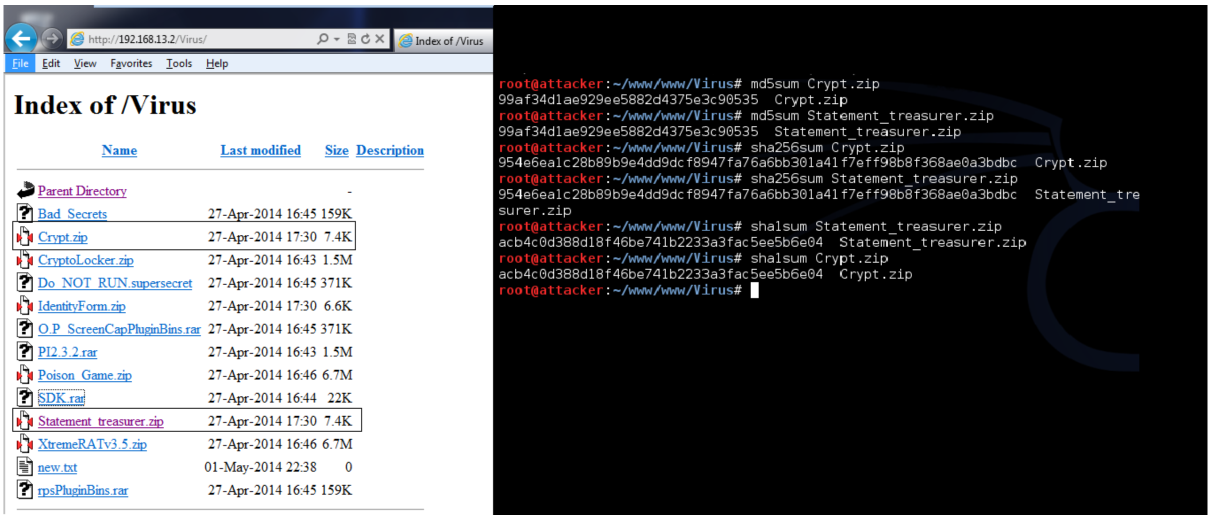
It was also evident when looking for BOTNET traffic dialling out of an internal network. Port 443(HTTPS) and SSL are commonly used along with independent encryption algorithms by certain malware strains when receiving master commands from command and control (C2) servers. The most common patterns that were spotted when identifying malicious traffic was data being transmitted across obscure ports (above 1023).

Finally when looking at the target machine accessing the attacking server in this testing scenario it was evident that certain characteristics can be searched for when monitoring traffic, specifically the malicious content being hosted by the kali machine in this test case was mainly Trojan droppers which have a file size of roughly 6-25 kilobytes, this file size can be seen in the packet capture shown below:



No.	Time	Source	Destination	Protocol	Length	Info
54	34.004081000	192.168.13.1	192.168.13.2	TCP	66	49212 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
55	34.005057000	192.168.13.2	192.168.13.1	TCP	66	http > 49212 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM=1 WS=1024
56	34.005653000	192.168.13.1	192.168.13.2	TCP	60	49212 > http [ACK] Seq=1 Ack=1 Win=65700 Len=0
57	34.008089000	192.168.13.1	192.168.13.2	HTTP	362	GET /Virus/Statement_treasurer.zip HTTP/1.1
58	34.009170000	192.168.13.2	192.168.13.1	TCP	60	http > 49212 [ACK] Seq=1 Ack=309 Win=16384 Len=0
59	34.015985000	192.168.13.2	192.168.13.1	TCP	7354	[TCP segment of a reassembled PDU]
60	34.016633000	192.168.13.1	192.168.13.2	TCP	60	49212 > http [ACK] Seq=309 Ack=7301 Win=65700 Len=0
61	34.016663000	192.168.13.2	192.168.13.1	HTTP	607	HTTP/1.1 200 OK (application/zip)
62	34.221130000	192.168.13.2	192.168.13.1	TCP	607	[TCP Retransmission] [TCP segment of a reassembled PDU]
63	34.223618000	192.168.13.1	192.168.13.2	TCP	66	49212 > http [ACK] Seq=309 Ack=7854 Win=65144 Len=0 SLE=7301 SRE=7854
71	39.023889000	192.168.13.2	192.168.13.1	TCP	60	http > 49212 [FIN, ACK] Seq=7854 Ack=309 Win=16384 Len=0
72	39.026821000	192.168.13.1	192.168.13.2	TCP	60	49212 > http [ACK] Seq=309 Ack=7855 Win=65144 Len=0
79	44.119075000	192.168.13.1	192.168.13.2	TCP	60	49212 > http [RST, ACK] Seq=309 Ack=7855 Win=0 Len=0

from the screen capture it can be seen the windows host (192.168.13.1) accessing the malicious server (192.168.13.2) and downloading the file titled Statement_treasurer.zip, the file downloaded can be identified as a potential Trojan dropper by looking at two factors, the file in this testing scenario shares the same hash value as a file containing cryptolocker also located on the server:



Index of /Virus

Name	Last modified	Size	Description
Parent Directory		-	
Bad_Secrets	27-Apr-2014 16:45	159K	
Crypt.zip	27-Apr-2014 17:30	7.4K	
CryptoLocker.zip	27-Apr-2014 16:43	1.5M	
Do_NOT_RUN.supersecret	27-Apr-2014 16:45	371K	
IdentityForm.zip	27-Apr-2014 17:30	6.6K	
O.P_ScreenCapPluginBins.rar	27-Apr-2014 16:45	371K	
PI2.3.2.rar	27-Apr-2014 16:43	1.5M	
Poison_Game.zip	27-Apr-2014 16:46	6.7M	
SDK.rar	27-Apr-2014 16:44	22K	
Statement_treasurer.zip	27-Apr-2014 17:30	7.4K	
XtremeRATv3.5.zip	27-Apr-2014 16:46	6.7M	
new.txt	01-May-2014 22:38	0	
rpsPluginBins.rar	27-Apr-2014 16:45	159K	

```
root@attacker:~/www/www/Virus# md5sum Crypt.zip
99af34d1ae929ee5882d4375e3c90535  Crypt.zip
root@attacker:~/www/www/Virus# md5sum Statement_treasurer.zip
99af34d1ae929ee5882d4375e3c90535  Statement_treasurer.zip
root@attacker:~/www/www/Virus# sha256sum Crypt.zip
954e6ea1c28b89b9e4dd9dcf8947fa76a6bb301a41f7eff98b8f368ae0a3bdbc  Crypt.zip
root@attacker:~/www/www/Virus# sha256sum Statement_treasurer.zip
954e6ea1c28b89b9e4dd9dcf8947fa76a6bb301a41f7eff98b8f368ae0a3bdbc  Statement_tre
surer.zip
root@attacker:~/www/www/Virus# sha1sum Statement_treasurer.zip
acb4c0d388d18f46be741b2233a3fac5ee5b6e04  Statement_treasurer.zip
root@attacker:~/www/www/Virus# sha1sum Crypt.zip
acb4c0d388d18f46be741b2233a3fac5ee5b6e04  Crypt.zip
root@attacker:~/www/www/Virus# 
```

The net flow information captured during this test was hashed and added to a list of known bad.

5 Conclusions

This paper has demonstrated that a certain degree of signature production automation is theoretically possible and that from an IDS design perspective it is an achievable goal with current technology. However this is ongoing research and there is still a significant amount of research that can be done into the subject matter. The factors that need to be considered though are that full automation will still require human interaction at some points to validate results. Developing automation from results collected would not be a difficult task. All that would be required is to write a python script that looks at the hashes gathered by the IDS then compare them with hashes of traffic that has previously been classed as a false positive then comparing these hashes with list of known good to detect whether a packet has malicious content or not. Also constantly scanning for traffic previously classed as false positives it then uses that information to work out whether that packet or string of packets is malicious or not.

6 Further Research

In future it would be of interest to look into any technologies that can be harnessed, are low cost and can be implemented easily. Possibly looking at the use of Raspberry Pi(RPi) or Arduino in a stacked environment to help with an automated system, as all that would be required is a couple of Pis to be run in a network stack and that stack running the python scripts. There are numerous projects that have already proven that the RPi can be used for a high performance supercomputing calculations. Transferring this power and performance onto a network monitoring scale would not be too much of a hard task. Also there is the added level of development with the expansion potential of the RPi. Also if either the RPi or Arduino were to be used this would keep the cost down of running a system for automation, the only issues with this that would need to be addressed is potentially cost versus performance and factoring in storage as well. Also looking at potential ways of developing more advanced searching algorithms that can be implemented to current systems through the use of rules. Also on-going research will be done into developing a proof of concept using python to analyse packet information and it will also look at comparison of traffic hashing based upon layer 3 packet information.

References

1. Lyne, J. (2013). *How to spy on hackers.* Available: http://blog.ted.com/2013/02/28/how-to-spy-on-hackers-james-lyne-at-ted2013/.
2. Brenton, C. (2006). *Egress Filtering FAQ.* Available: https://www.sans.org/reading-room/whitepapers/firewalls/egress-filtering-faq-1059.
3. Ganta Jacob Víctor, Sreenivasa Rao Meda, V CH Venkaiah. (2009). *False Positives in Intrusion Detection Systems.* Available: http://www.academia.edu/1431396/False_Positives_in_Intrusion_Detection_Systems.
4. AbuHmed, T, Mohaisen, A, Nyang, D. (2008). *A Survey on Deep Packet Inspection for Intrusion Detection Systems*. Available: http://arxiv.org/pdf/0803.0037.pdf
5. Atlasis, A. (2013). *HTTP header heuristics for malware detection.* Available: https://www.sans.org/reading-room/whitepapers/detection/http-header-heuristics-malware-detection-34460.

Acknowledgements
I would like to thank Ollie Whitehouse and NCC Group for setting the research topic of this paper.