Sandnet++ - A Framework for Analysing and Visualising Network Traffic from Malware[1]

Authors

Anthony Nelson, MSc (Royal Holloway, 2015)
Lorenzo Cavallaro, ISG, Royal Holloway

Abstract

One important step in combating malware is to understand how it communicates over a computer network. Most malware has to communicate remotely, whether to infect further victims, exfiltrate stolen information or receive instructions. Examining the network traffic generated by malware provides an opportunity to identify the unique features found only in malware traffic, and use these to distinguish it from benign traffic. Only if malware traffic is identifiable can it be blocked or otherwise disrupted. This article presents Sandnet++, a framework for analysing and visualising network traffic from malware. We also present several case studies showing how the Sandnet++ framework can be used to extract malware traffic features, allowing better malware detection.

Malware Analysis and Network Traffic Capture

Dynamic malware analysis is the process of analysing malware by executing it, often in a container or sandbox. The malware is usually allowed to perform its actions such as initial exploitation of a vulnerable computer program, installation then subsequent communication with command and control infrastructure. Such activities are monitored and recorded. Analysis can then be conducted on the data collected.

A variety of tools to run malware exist, such as Joe Sandbox, Cuckoo Sandbox and Anubis. Here we present Sandnet++. The Sandnet++ framework is split into two parts: the *capture framework* and the *analysis framework*. The capture framework runs a malware sample within a virtual machine to capture network traffic. The analysis framework uses the output of the capture framework, or other previously captured network traffic, to perform protocol feature extraction. This data can then be analysed and visualised directly, or clustering can be performed to group common network communications for analysis. A web interface and Application Programming Interface (API) is provided for all interactions with the system.

In the Sandnet++ system a virtual machine running the Windows XP (32 bit) operating system, Service Pack 3, is set up and configured to be used by Cuckoo Sandbox. A malware sample is submitted to run within the virtual machine and Cuckoo Sandbox takes care of executing the malware for a configurable time period, capturing the network traffic and resetting the virtual machine to a clean state. This forms the Sandnet++ *capture framework*, the output of which is a pcap file (a packet capture of network traffic).

[1] This article is to be published online by Computer Weekly as part of the 2016 Royal Holloway information security thesis series. It is based on an MSc dissertation written as part of the MSc in Information Security at the ISG, Royal Holloway, University of London. The full MSc thesis is published on the ISG's website.

The design of the capture framework is shown in Figure 1.

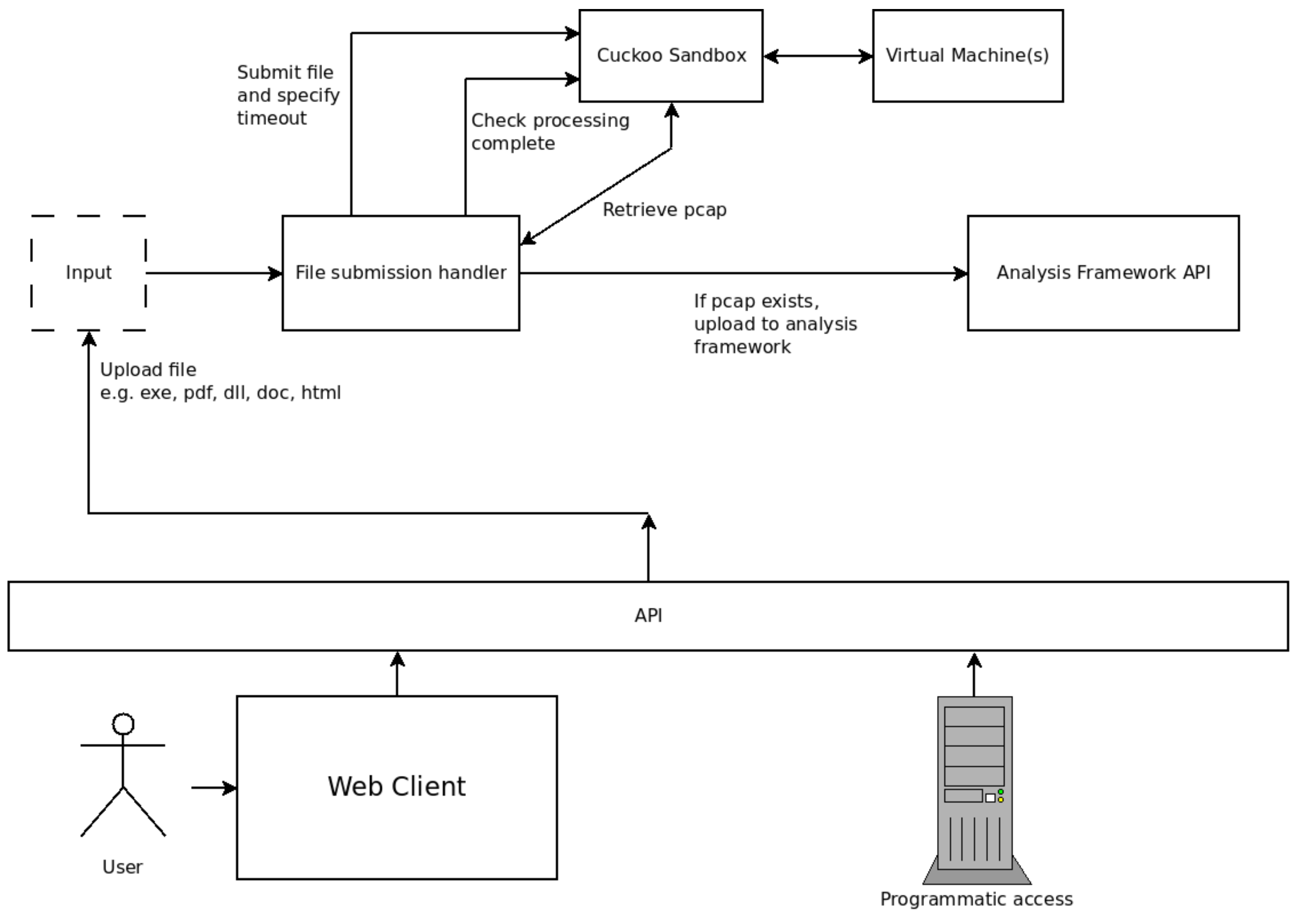

Cuckoo Sandbox
Virtual Machine(s)
Submit file and specify timeout
Check processing complete
Retrieve pcap
Input
File submission handler
Analysis Framework API
If pcap exists, upload to analysis framework
Upload file e.g. exe, pdf, dll, doc, html
API
Web Client
User
Programmatic access

Figure 1: Capture framework design

Protocol Feature Extraction

Many network Intrusion Detection System (IDS) signatures are written to look for matches in specific protocol fields, such as an Hypertext Transfer Protocol (HTTP) method, Uniform Resource Indicator (URI) or User-Agent. For example, take the signatures presented in *Firma: Malware clustering and network signature generation with mixed network behaviors*. The majority of these look for a specific value of at least one protocol field. Three of these are listed below:

- content: “POST”; http method; - requires the HTTP method to be “POST”
- content: “/picture.php”; http uri; - requires an HTTP URI of “/picture.php”
- dsize:13; content: “|04000001050000000007000100|”; - requires the packet payload size to be 13 bytes and contain the binary data “04000001050000000007000100” (represented as hexadecimal). The payload size is itself a protocol field.

Thus, malware may be identified through properties of its network traffic.

The approach taken by the Sandnet++ *analysis framework* is to extract protocol features from network traffic and to make these available for analysis. The raw extracted data may be viewed directly or may be further processed, for example by calculating the distribution of values for a particular feature or clustering traffic based on the different features it possesses.

The analysis framework design is shown in Figure 2. There are two feature extraction engines within the analysis framework. One operates on packets, the other on Transmission Control Protocol (TCP) sessions. The feature extraction engines produce the feature data used for analysis and visualisation.

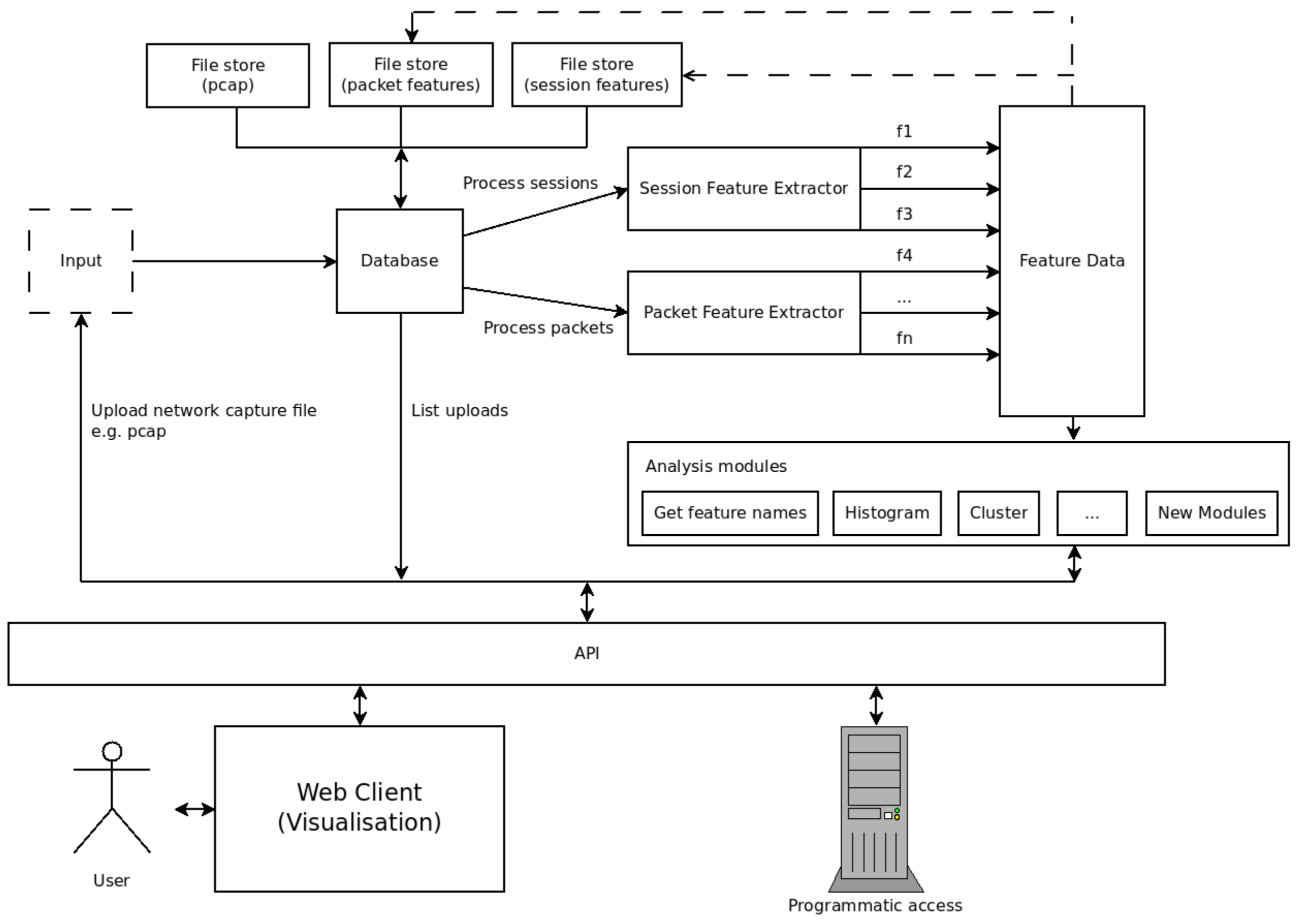


Figure 2: Analysis framework design

Packets Vs Sessions

Packets vs Sessions:

- **Packet** - protocol data unit at network layer, typically an IP version 4 packet.
- **Session** -reconstruction of one or more packets that form a TCP session, typically identified by a four tuple (source IP, source port, destination IP, destination port). A session can have a request and/or a response which represent the payload data contained within the TCP segments.

Packet Feature Extraction

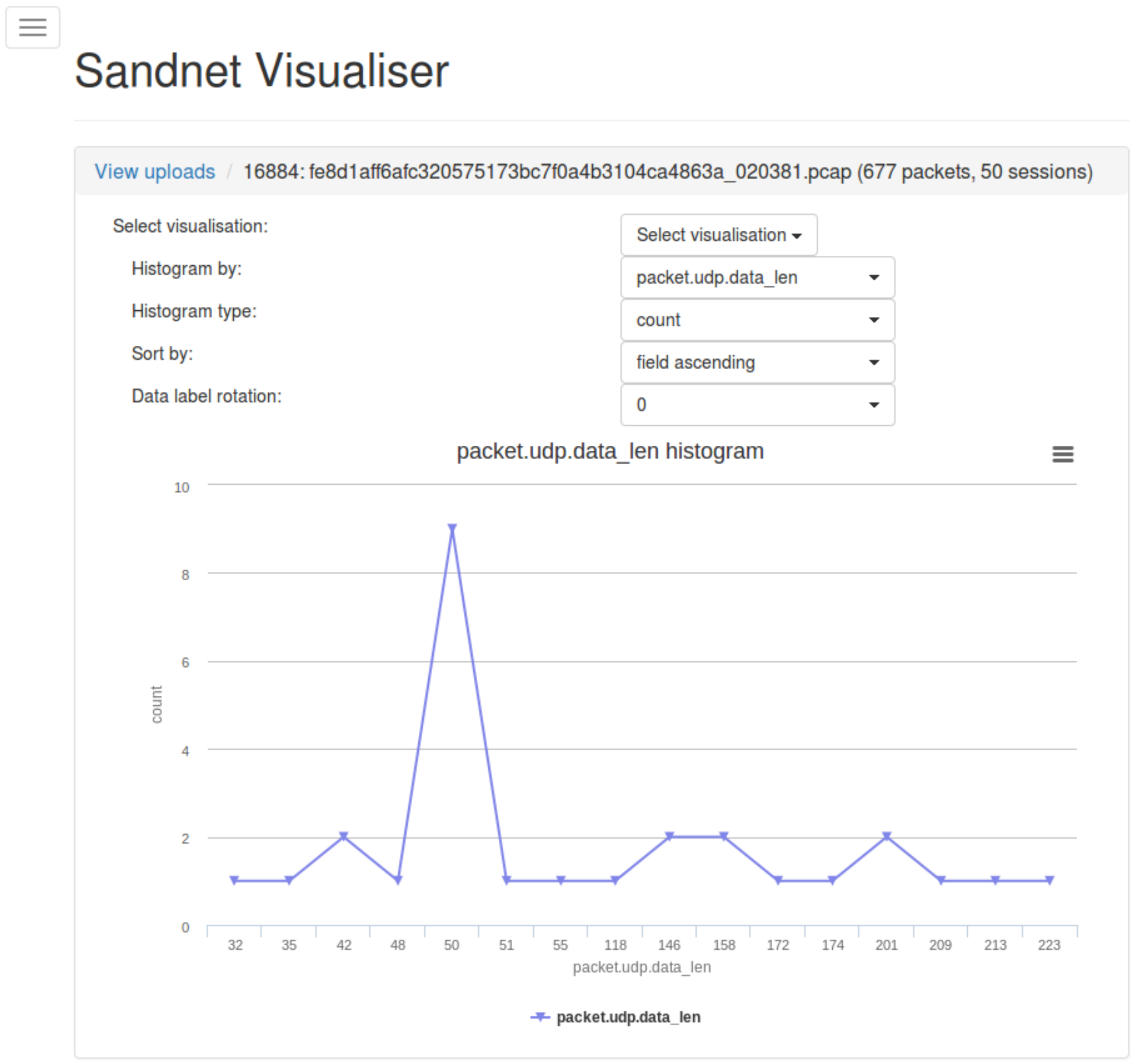
Packet features include information such as the packet timestamp, packet number within the pcap and the protocols/layers within the packet. The order of protocols is maintained (e.g. Ethernet, followed by IP, followed by TCP). Each protocol has a variety of fields extracted. For example, the TCP protocol has a “seq” field to represent the TCP sequence number and a “dport” field to represent the destination port number, and the Domain Name System (DNS) protocol has a “qname” field to represent the domain name queried.

Session Feature Extraction

Session features include basic TCP information such as source and destination IP addresses and port numbers, as well as request/response data length and number of packets in the session. The request and response protocols within the session are identified with each protocol having a variety of fields extracted. For example, the HTTP Request protocol has fields such as “method”, “uri”, “uri_param_names” and “headers”. Each HTTP header is also broken out as its own field, e.g. a field of “Host” with a value of “example.com”.

Visualisation

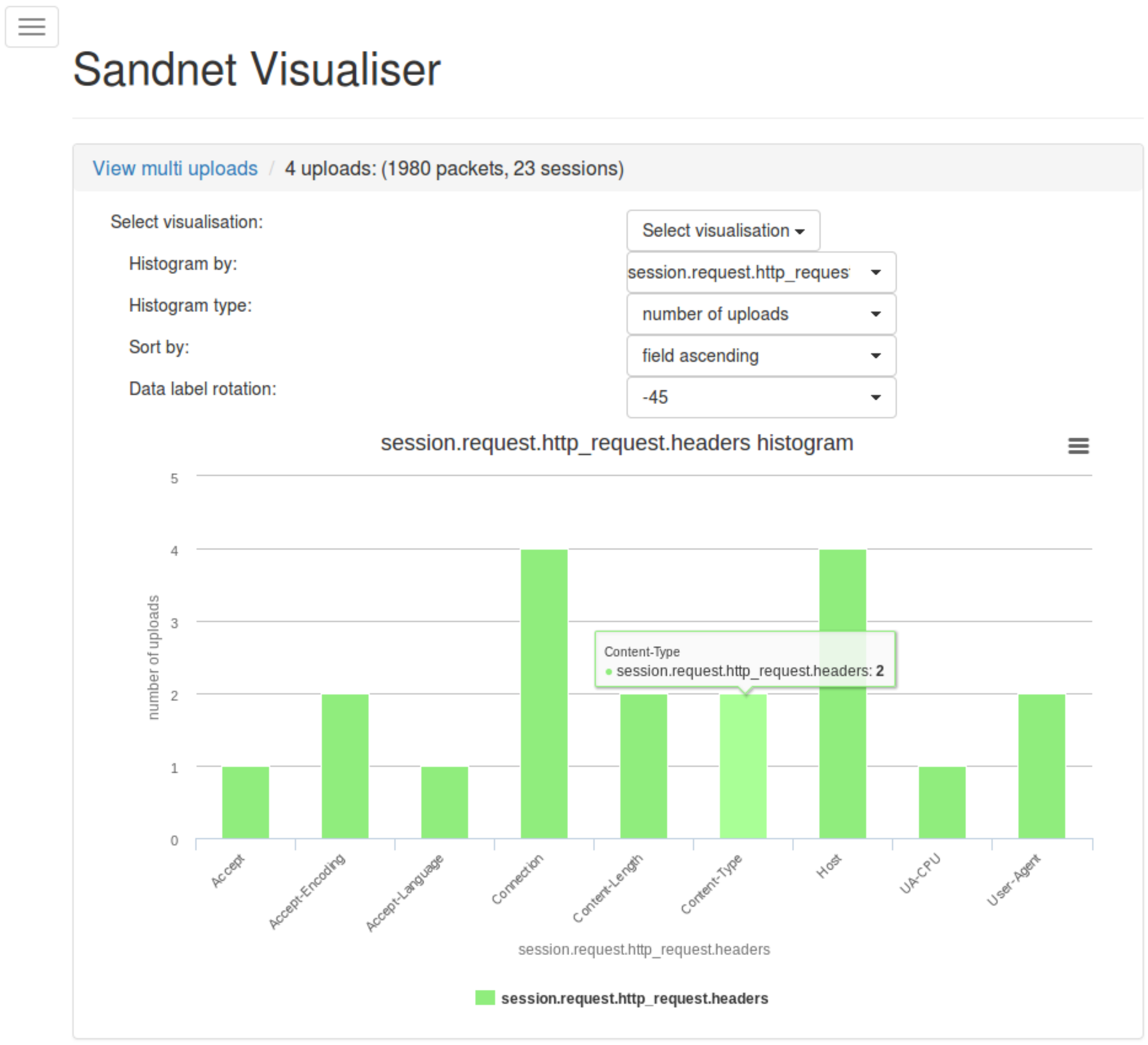
All interaction with Sandnet++ takes place via its API and uses Representational State Transfer (REST) principles. The API is a RESTful web service so any web client can be used, however, a web interface has been developed for interaction with the API to provide visualisation of the data. Example visualisations are shown in Figures 3, 4 and 5.


Sandnet Visualiser

View uploads / 16884: fe8d1aff6afc320575173bc7f0a4b3104ca4863a_020381.pcap (677 packets, 50 sessions)

Select visualisation: Select visualisation
Histogram by: packet.udp.data_len
Histogram type: count
Sort by: field ascending
Data label rotation: 0

Figure 3: Histogram results of the UDP data length, displayed as a line chart



Sandnet Visualiser

View multi uploads / 4 uploads: (1980 packets, 23 sessions)

Select visualisation: Select visualisation

Histogram by: session.request.http_reques

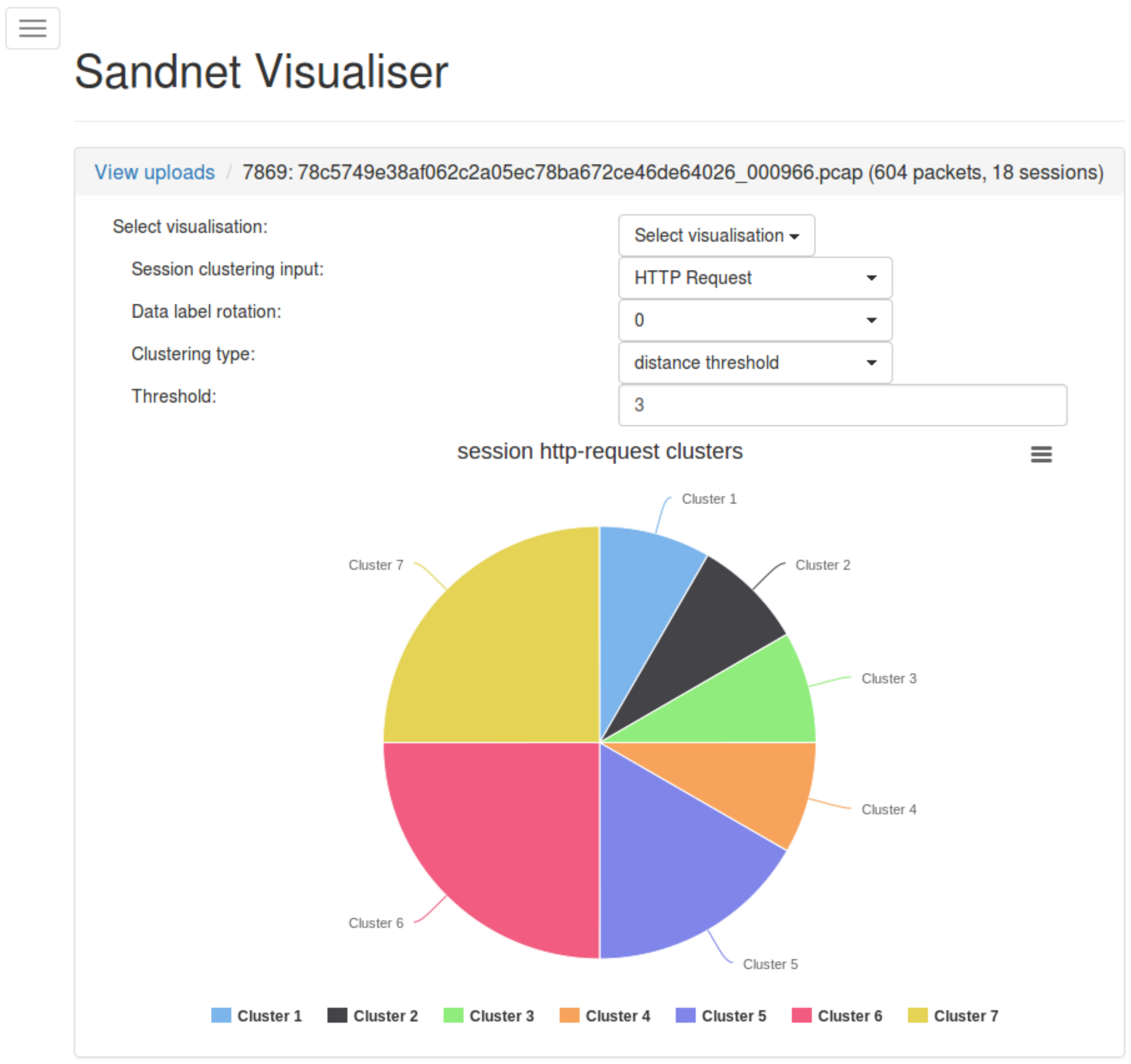
Histogram type: number of uploads

Sort by: field ascending

Data label rotation: -45

session.request.http_request.headers histogram

Figure 4: Histogram results of the number of pcaps containing HTTP request headers, displayed as a column chart


Sandnet Visualiser

View uploads / 7869: 78c5749e38af062c2a05ec78ba672ce46de64026_000966.pcap (604 packets, 18 sessions)

Select visualisation: Select visualisation

Session clustering input: HTTP Request

Data label rotation: 0

Clustering type: distance threshold

Threshold: 3

Figure 5: Session (HTTP request) clustering results displayed as a pie chart

Clustering

Why might we want to perform clustering of network traffic and how can we do this?

- *within a pcap from one malware sample* – this is useful to understand how the malware operates. It can highlight features that may form network IDS signatures to identify specific malware.
- *between pcaps from the same malware family* – this allows examination of malware samples to see how communications differ between different instances of the malware. For example, this might highlight how URI parameters change between specific installations.
- *between pcaps from related malware families* – this can highlight any similarities between families, possibly hinting at shared origins
- *between pcaps from unrelated malware families* – this allows identification of communication techniques that may be prevalent across a wide number of malware families
- *across the entire dataset* – this provides overall statistics and trending for all data examined

Two clustering algorithms are implemented in the analysis framework. Each uses specific packet or session features to perform the clustering, although new algorithms can easily be added using any of the extracted packet/session features available. The clustering available in Sandnet++ does not attempt to classify network traffic as malicious or not, but to partition the data into related communications.

The packet clustering used is for the User Datagram Protocol (UDP) and TCP transport layer protocols. This takes as input all packets (in one or more pcaps) and groups them into clusters containing similar transport layer features.

The session clustering used focuses on HTTP requests. This takes as input all HTTP requests (in one or more pcaps) and groups them into clusters containing similar HTTP request features. These features are:

- HTTP request method
- HTTP Uniform Resource Locater (URL) path
- HTTP parameter names
- HTTP parameter values

The precise details of how the clustering is performed can be found in the project thesis, available from the Royal Holloway Information Security Group.

Implementation

The Sandnet++ framework described in this article consists of approximately 6000 lines of source code. The majority of the framework is written using Python 3, JavaScript and HTML. For access to the source code please contact the authors.

Case Studies

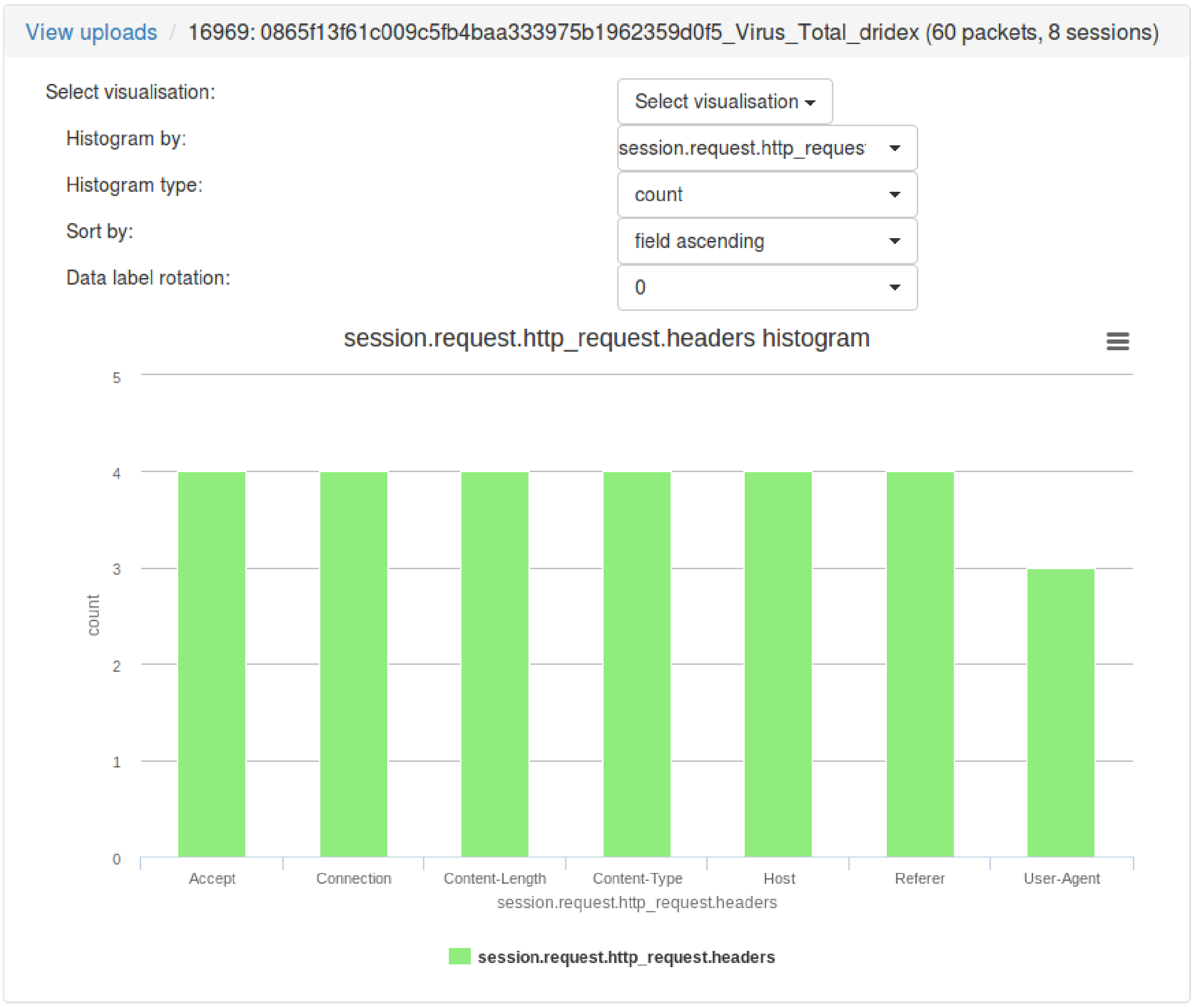
We now present our usage of Sandnet++ using several case studies. The full analysis is available in the project thesis available from the Royal Holloway Information Security Group.

Case Study 1 – Dridex Malware

Dridex is an evolution of Cridex malware. It is a relatively new threat first observed in November 2014 and is used to steal personal and financial information. The sample we obtained was first submitted to VirusTotal in February 2015. It has a Secure Hash Algorithm 1 (SHA1) value of 0865f13f61c009c5fb4baa333975b1962359d0f5. It was uploaded to the *capture framework* and run for 10 minutes to produce the required pcap file for the *analysis framework*.

There are 60 packets and eight TCP sessions in the pcap. No DNS is present. Four public IP addresses are contacted: “202.44.54.5”, “5.196.241.196”, “66.110.179.66” and “92.63.87.13”. As no DNS lookups are performed these IP addresses must be hardcoded into the malware.

Examining the session data reveals that there are four HTTP requests and four HTTP responses. These are evenly distributed between the four public IP addresses previously mentioned. The URI for every request is “/”, i.e. there are no parameter names or values. The histogram of the HTTP request headers shown in Figure 6 reveals an interesting characteristic of the traffic. The same HTTP request headers are present in all four of the requests, except for one which has no User-Agent header. If a network IDS signature was based on the User-Agent value it would miss the session that contained that request.


View uploads / 16969: 0865f13f61c009c5fb4baa333975b1962359d0f5_Virus_Total_dridex (60 packets, 8 sessions)

Select visualisation: Select visualisation

Histogram by: session.request.http_reques

Histogram type: count

Sort by: field ascending

Data label rotation: 0

session.request.http_request.headers histogram
count
5
4
3
2
1
0
Accept
Connection
Content-Length
Content-Type
Host
Referer
User-Agent
session.request.http_request.headers
🟩 session.request.http_request.headers

Figure 6: HTTP request header distribution for the Dridex malware sample

Inspection of the User-Agent values reveals two are in use: “Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.143 Safari/537.36” and “Mozilla/5.0 (Windows NT 6.3; WOW64; rv:34.0) Gecko/20100101 Firefox/34.0”. This appears to be an attempt by the malware to camouflage itself as either the Chrome or Firefox web browser.

With the exception of the Content-Length header, the remaining headers prove extremely interesting to study. The Accept, Content-Type, Host and Referer header values are all selected at random from a preconfigured list.

An HTTP request uses a Referer header to specify the previous web page that linked to the newly requested one. This is used for purposes such as checking search terms that directed a user to a web site or identifying affiliate sites that send a lot of traffic to partner web site.

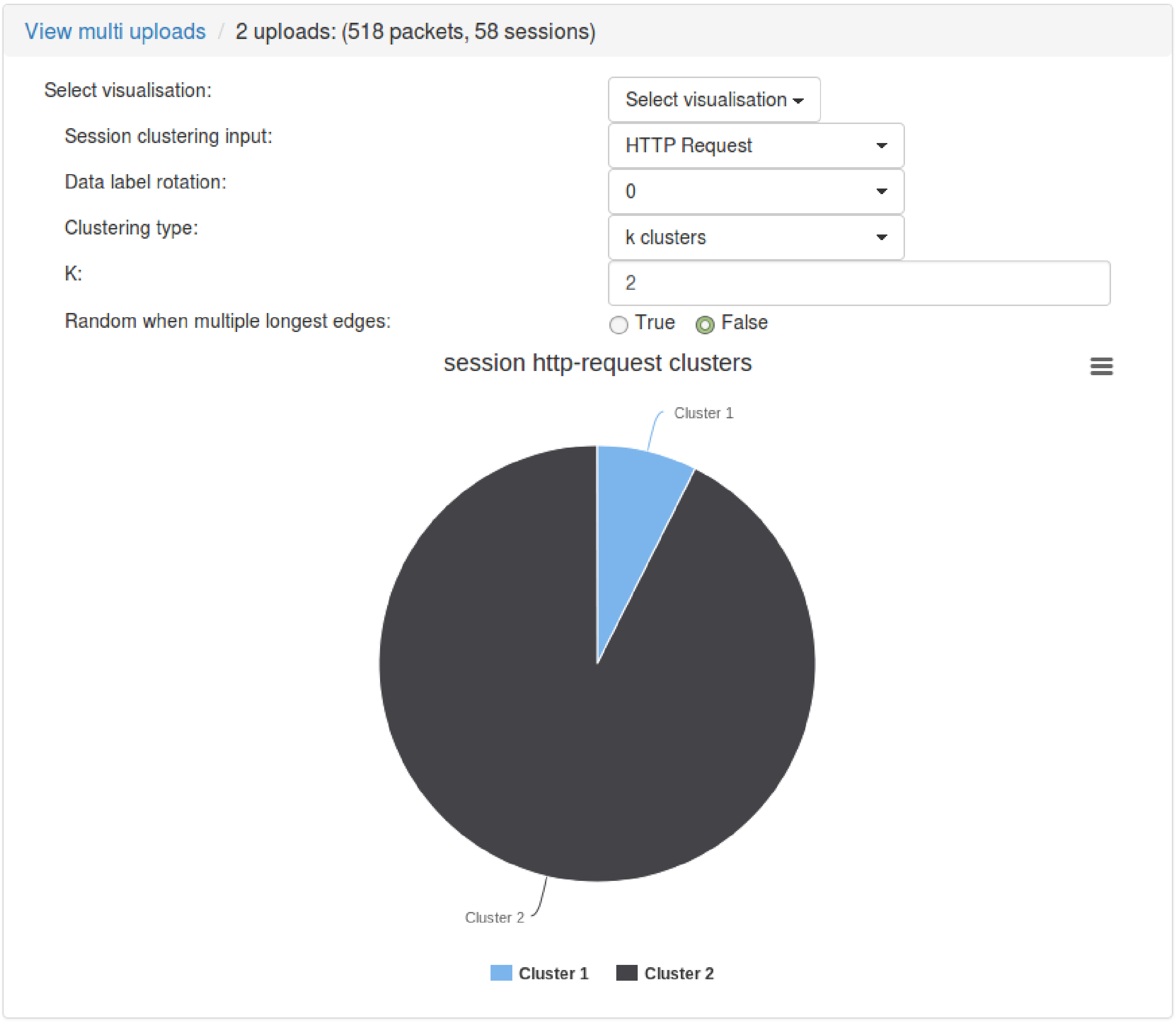
The Referer headers in this pcap are “https://aol.com/” and “https://facebook.com/”. There are two problems with this. Firstly the malware is setting a Referer even though there is none to legitimately set. Secondly, the HTTPS (rather than HTTP) protocol is specified in the Referer. Legitimate web browsers are unlikely to do this as the risk of information disclosure is too great to transmit an HTTPS URL over an HTTP channel – it may contain sensitive information such as that relating to a user login. This could form the basis of a heuristic network IDS signature that may work across different malware families that fail to observe this.

Case Study 2 – Clustering Zeus and Dridex Network Traffic

Zeus is a crimeware toolkit, designed to operate as a large-scale botnet. It is typically used to steal personal information and financial details and may be used to download other malware.

A Zeus malware sample was obtained from VirusTotal. It has a SHA1 hash value of cdae6fee42288a8cbf8de7ed683b6f325b4a71ef. It was uploaded to the *capture framework* and run for 10 minutes to produce the required pcap file for the *analysis framework*.

As Zeus and Dridex are different malware it would be useful to separate the traffic. Using the Sandnet++ session HTTP request clustering we can do this. As can be seen in Figure 7, clustering into 2 clusters, this accurately separates the network traffic from the two malware samples. The four Dridex requests are in cluster 1 whilst 50 Zeus requests are in cluster 2. This split is achieved as the features measured (HTTP request method, URL, parameter names and parameter values) are sufficiently different between the two malware samples.



View multi uploads / 2 uploads: (518 packets, 58 sessions)

Select visualisation: Select visualisation

Session clustering input: HTTP Request

Data label rotation: 0

Clustering type: k clusters

K: 2

Random when multiple longest edges: True [✓ False]

Figure 7:Zeus and Dridex HTTP request clusters with a k parameter value of 2

This approach can be used across multiple malware families to separate their traffic. A logical next step would be to generate network IDS signatures to identify traffic within a cluster, however this is out of scope for the current Sandnet++ framework.

Case Study 3 – Protocol Feature Extraction

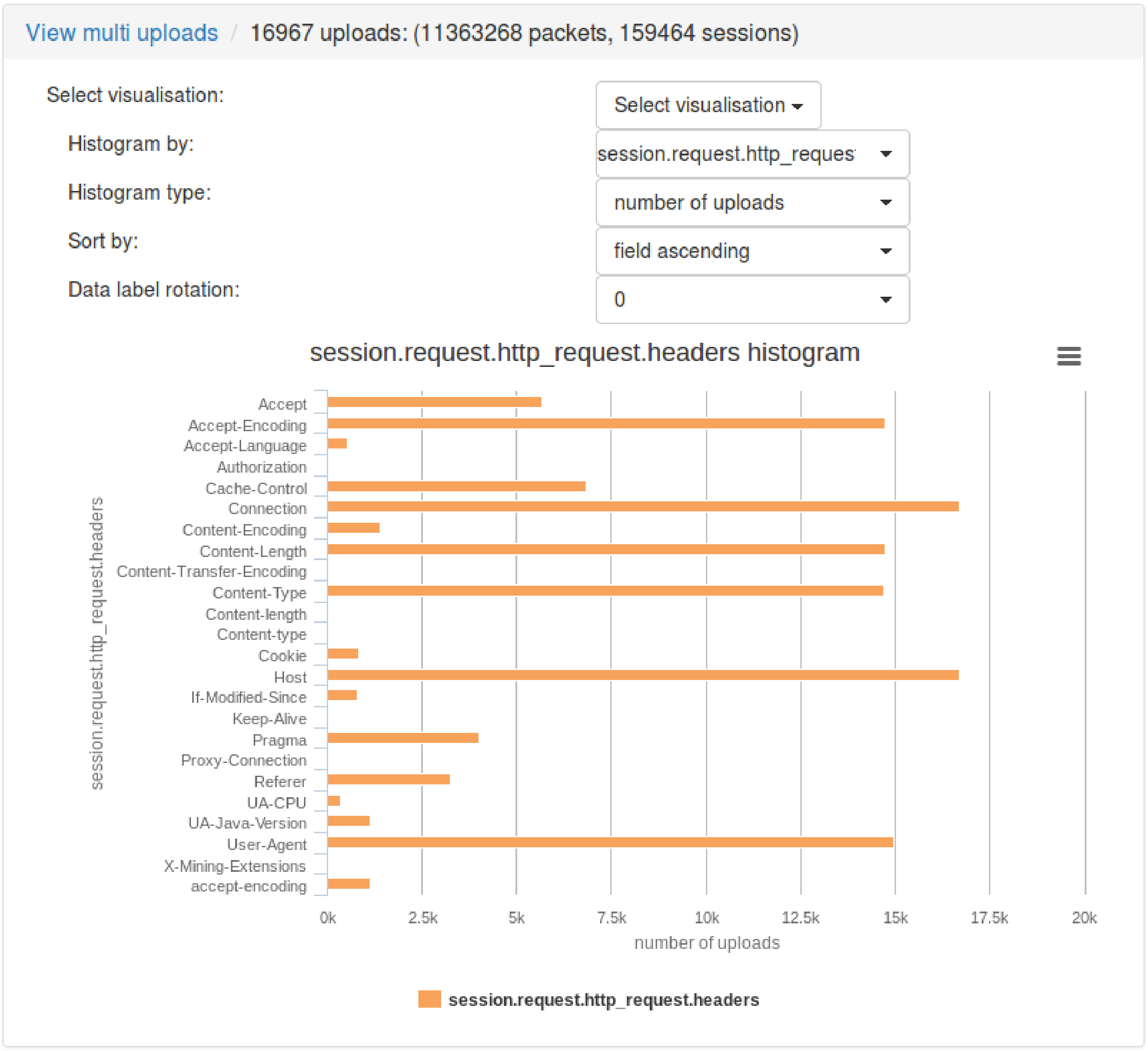
We conducted analysis across the entire MALICIA dataset (a collection of more than 17,000 pcaps generated from 11,000 malware binaries) to give an overview of the networking protocols used by malware. A subset of this is presented below, focussing on the HTTP request protocol.

86.7% of network traffic captures utilise HTTP GET requests, whilst 86.8% use HTTP POSTs. GET requests are typically used to retrieve resources (e.g. further malware to run) and POSTs to send information to a server (e.g. information stolen from the infected computer). The most popular URI path is the empty path ("/") which appears in 14677 (86.5% of) pcaps. The most popular URI path specific to a malware family is "/pony/gate.php" which appears in 1198 (7.1% of) pcaps.

The distribution of HTTP URI parameter values is much greater. There are 13429 different parameter values used across the dataset, many of which appear to be randomly generated. This is to be expected, especially if malware uses the parameter to indicate a unique victim ID, timestamp or some other data with high variability. However, the structure of a parameter value can be exceedingly useful. For instance, some values appear to be hashes 40 characters in length. Creating a regular expression to identify these is possible. Additionally some URI parameters appear obfuscated or URL encoded which indicates that some malware attempts to restrict access to its data without employing proper encryption.

There is a wealth of information that can be mined from HTTP headers. In total there are 24 different HTTP request headers present - Figure 8 shows the distribution in the MALICIA dataset. Sorting by name reveals the letter case errors in a number of different headers: e.g. accept-encoding" instead of "Accept-Encoding" and "Content-type" instead of "Content-Type".

As such, case sensitivity of HTTP header names can be used as a detection technique for malicious network traffic.


View multi uploads / 16967 uploads: (11363268 packets, 159464 sessions)

Select visualisation: Select visualisation

Histogram by: session.request.http_reques

Histogram type: number of uploads

Sort by: field ascending

Data label rotation: 0

Figure 8: HTTP request header names and the number of MALICIA traffic captures they appear in

The User-Agent HTTP request header is typically used to identify the software that makes an HTTP request. A fresh Windows XP install will use Internet Explorer as the default web browser. However, several of the User-Agents we observed claimed to be other browsers ("Opera/9 (Windows NT 5.1; ; x86)"), got the Internet Explorer User-Agent header wrong completely (just "Internet Explorer") or appeared to just leak information ("WINXPSP3 7875768F8B00CED2"). Malware usage of HTTP User-Agents has been widely studied - it is not the intention of this article to further advance this, but to highlight how the Sandnet++ framework can be used to study the User-Agents and quickly hone in on the values that might be of interest.

There are three different Accept-Language headers in the MALICIA dataset. One of these, "en-us" is present in 526 samples, the others ("en" and "en-US") occur in only two. As such they are a good identification of the malware traffic. Additionally, some of the "en-us" header values may be hardcoded into malware. As such, a dynamic malware analysis system, such as the *capture framework* designed for this project, could benefit if its localisation settings were set to a less popular value than English. This would make such hardcoded header values clear in any network captures.

Every other HTTP header can be mined for information. The *analysis framework* is capable of displaying details about the HTTP protocol, including the header names and values. This aids the identification of unique header names and values, and can help malware analysts to understand the behaviour and capabilities of malware, and of malware families.

Many features are useful to study when examining malware traffic. This is typically because malware uses an incorrect value of a protocol feature, uses a unique value, uses a fixed value or uses a value which does not make sense in the context of the host/network within which the malware resides. Examples of these from the traffic studied include:

- *incorrect feature value* – HTTP Host names observed with a space character instead of a "." or HTTP header names with the wrong letter case, e.g. "accept-encoding" instead of "Accept-Encoding"
- *unique feature value* – domain names used by malware, such as "risparmioassicurativo.net", or URIs such as "/0Pvo9Hnu/EpJbWNWD.exe"
- *fixed feature value* – repeated requests to the URI "/pony/gate.php" or the fixed User-Agent "Shareaza"
- *feature value inappropriate for a host/network* – the User-Agent "Mozilla/4.0 (compatible; MSIE 5.0; Windows 98)" observed from a Windows XP operating system

Conclusions

In this article we present Sandnet++, a new framework for analysing and visualising network traffic from malware. The case studies show how Sandnet++ has been used to analyse specific malware samples in detail, as well as perform cross-family analysis. Anomalies in the network traffic generated by malware are pointed out – this is all achieved by studying the protocol features that are extracted by Sandnet++ and aided by the clustering and visualisation capabilities.

Sandnet++ is fully extensible should other researchers wish to build upon it. Automation is an important feature of the framework due to increasing numbers of new malware. Having an API allows automated access to functionality, opens up the possibility for integration with other systems and enables analysis to be conducted quickly and efficiently.

There are many directions future work could take, for example, studying time based behaviour, calculating statistics for protocol fields (e.g. the length of a URI), performing network IDS signature generation for clustered traffic or implementing new clustering algorithms. Ultimately, all of these may be required to keep up with advances in malware technology.

Biographies

Anthony Nelson graduated from Royal Holloway, University of London in 2015, obtaining an MSc in Information Security with Distinction. He has substantial experience in network forensics, malware analysis and software development. He aims to combine all three to improve the lives of network defenders.

Lorenzo Cavallaro is a Senior Lecturer of Information Security in the Information Security Group (ISG) at Royal Holloway University of London. His research focuses largely on systems security. He has founded and is leading the recently-established Systems Security Research Lab (S2Lab) within the ISG, which focuses on devising novel techniques to protect systems from a broad range of threats, including those perpetrated by malicious software. In particular, Lorenzo's lab aims ultimately at building practical tools and provide security services to the community at large. In the past, Lorenzo was lucky enough to have the chance to work with a number of well-established groups (e.g. G. Vigna and C. Kruegel at UCSB, A. S. Tanenbaum and H. Bos at Vrije Universiteit, R. Sekar at Stony Brook University) during his PostDocs and visiting PhD periods. He is currently working on a number of research projects funded by EPSRC and EU FP7, publishing in top and well-known venues and serving as program committee member for well-known conferences and workshops.