# Global Information Assurance Certification Paper

# 60 Seconds on the Wire:
# A Look at Malicious Traffic

*GIAC (GCIA) Gold Certification*

Author: Kiel Wadner, wadnerk@gmail.com
Advisor: Antonios Atlasis

Accepted: August 7, 2013

Abstract

Despite advances in detection, malware remains an active and high-risk threat to organizations. Understanding the characteristics of malware traffic can be vital in detecting, as well as responding to an incident inside an organization. In this paper, over 20,000 PCAPS generated by known malware are explored to find these characteristics. The focus of the research is on HTTP traffic since this was the predominate communication protocol seen. Based on the findings, suggestions are offered towards effective detection of malware traffic. As it will be shown, despite attempts to hide or obfuscate itself, malicious traffic was detected with a high level of success.

# 1. Introduction

Malware depends on its communication network to receive commands, extract information and infect systems. Due to this reliance on networked resources, traffic analysis becomes a valuable and effective method for detecting malware on host machines. Despite the frequency of malware traffic, network administrators and incident responders may not be aware of what characteristics are common to malware. By looking at traffic generated while malicious samples are executed the characteristics of the traffic can be recorded and investigated.

In the 2013 *Data Breach Investigations Report* Verizon defined malware as "[…] any malicious software, script, or code added to an asset that alters its state or function without permission." (Verizon, 2013). This is a good working definition, but emphasis should be placed on "without permission". It is the intent of malware that makes its actions malicious. When observing network traffic generated by malware it is the intent that makes it malicious, not just the content. For example, an HTTP GET to Amazon's Web Services is not normally malicious – unless the intent is to communicate with part of a Command and Control (C&C) infrastructure. In a real environment the challenge is deciphering intent to understand what might be malicious.

The research for this paper started with a basic question: *What can be discovered in 60 seconds of watching malicious traffic?* The goal was to discover what investigating a relatively large set of PCAPs generated by known malicious software would reveal. What detections worked well? What network actions did the malware take when it was first executed? The hypothesis was while malware displays many normal network actions there would be actions that stand out as abnormal. This hypothesis held true in the end.

The "malicious intent" of a sample was determined with the help of Norman Shark's Malware Analyzer G2 (MAG2) product (Norman Shark, 2013)[1]. The MAG2 is a

[1] Disclaimer: At the time of writing, the author was an engineer employed by Norman Shark.

Kiel Wadner, wadnerk@gmail.com

behavioral analysis system that operates under sandboxing principles. Details of the Windows environment used are given in the next sections. Each sample's risk score, as determined by MAG2, and VirusTotal scan results were used to identify likely malicious samples.

To provide a survey of malicious communications, a significant amount of traffic is necessary. To this end, a set included just over 20,000 PCAP files was generated, each containing a maximum of 60 seconds of communication from a sample. Knowing that a malicious sample generated the traffic allows for assumptions that usually would not be safe to make in a live environment. An example is using the User-Agent strings from HTTP headers as a detection method. Since the execution environment was controlled, the User-Agent strings that should be used are limited. Requests with User-Agent strings that are known to be invalid are at a minimum interesting and potentially clear indicators of malicious traffic. Outside of lab environments such assumptions may not be safe to make. Since the environment was controlled, the assumptions can be made safely and what has been learned can be applied to a real environment.

Using the pre-described methodology, the final goal of this paper is to help us understand what the characteristics of malware traffic are and finally, to provide a starting point when trying to detect and analyze the traffic generated by malware.

## 2. Environment and Execution

At first, each sample was loaded into the MAG2 appliance via a web API and an analysis task queued. Samples were injected into a virtual Windows XP SP3 environment and allowed to run up to 60 seconds. During execution, 40 tasks ran concurrently but were isolated from each other. Windows XP was chosen over Windows 7 to increase the likelihood of successful execution. Other changes to the environment over a typical workstation included turning off the built-in firewall and automatic updates to make the system as vulnerable as possible. The MAG2 appliance also makes changes to avoid being detected by VM aware malware but these changes do not affect the results presented. The Windows XP profile can be customized with any third party software or versions desired. The table below shows the software that was present in the profile used.

A noticeable omission is Microsoft Office. The original sample set only had a small selection of documents, and the majority of them had minimal traffic generated or none at all. It was decided to omit them from the final results.

| Adobe Flash Player | Adobe Reader | Java | Internet Explorer 8 |
|---|---|---|---|
| 11.0.1.152 | 9.4.0 | 6.0.320 | 20090308.140743 |

Table 1: Installed software in analysis profile

When executed the sample had full access to the Internet through a "dirty-line" connection. So called dirty-lines are Internet connections separated from the normal corporate network and often use VPN or DSL providers to hide the originating organization. Due to a limitation of MAG2 only IPv4 traffic is allowed through and thus captured. At least one sample made a DNS request to Hurricane Electric, an IPv6 tunnel provider (Hurrican Electric, 2013), so it is likely IPv6 traffic would have existed if possible.

Full Internet access is not always the preferred way to analyze malware and caution should be used with this approach. Malware analysts who want isolation typically use INetSim, FakeNet or similar tool to mimic what resources are available. According to *Practical Malware Analysis*, (Sikorski, Honig, 2012) "INetSim is the best free tool for providing fake services, allowing you to analyze the network behavior of unknown malware samples by emulating services such as HTTP, HTTPS, FTP, IRC, DNS, SMTP, and others." This option was considered but was decided against so that actual two-way traffic could be examined. Mimicking a network is a very valid approach and can be the right approach when trying to understand the behavior of malware.

After the 60-second execution, the MAG2 appliance powered off the VM and reverted to a clean state before processing the next sample. The captured events were matched against generic and specific behavioral patterns to assign a risk score. Once processing was complete, the PCAPs were downloaded from the MAG2 appliance and analyzed with a mixture of a custom python framework, and well known tools, which will be described in the next section. While ideally only malicious traffic would exist in the PCAP, Windows itself tends to be chatty and introduces extra communication.

Examples of this will be seen throughout the paper. In some cases it will have affected the traffic statistics of a PCAP.

## 3. Tools Used and Analysis

A variety of tools, both existing and custom, were used to analyze the PCAPS. Much of the post-processing was coordinated with a custom toolset written by the paper's author dubbed *Malware Tracks*, which is covered below.

The analysis process included the following tasks:

- Determine the file type.
- Look-up the scan report via VirusTotal API.
- Run the PCAP through the Suricata IDS engine.
- Post-process the Suricata log files to get alerts, statistics, http connections, and TLS certificates.
- Parse PCAPS with Scapy to extract finer details.
- Summary review of the findings.
- Deep-Dives of specific PCAPS based on interesting summary items found.

A benefit of this approach was leveraging the Suricata results to guide the analysis process. Combined with the VirusTotal, a general idea of what could be in each PCAP was possible. Scapy, a Python tool for both parsing and creating network traffic (Scapy, 2013), was used to get specifics at the packet level. The PCAPS that did not trigger any alerts were looked at closer to determine why.

### 3.1 Malware Tracks

Written in Python, *Malware Tracks* automates as much of the work as possible and served as the glue between different bits of information. At the time of publication it is still very rough code with assumptions of how it will be used. Hence, it is provided "as-is" on GitHub (https://github.com/kwadner/malware-tracks), but future work is planned. Please see the GitHub repository for the latest information on this tool.

## 3.2 Suricata

In the world of open source Intrusion Detection Systems (IDS) there are two main products: Snort from SourceFire, and Suricata from the Open Information Security Foundation (OISF). Initially the plan was to use both Snort and Suricata and compare the results, but after the initial proof-of-concept with Suricata it was decided to use that tool exclusively. This was mostly due to the ease and speed of processing samples through a --*unix-sock*et option. This option starts the engine a single time, and then receives the PCAPS through a raw socket connection. This made bulk process the PCAPS relatively quick and simple. Details can be found on the OISF website (The Open Information Security Foundation, 2013).

Suricata can be configured to log several types of information. The options used were *fast-alert*, *http-log*, *tls-log*, and *stats*. *Fast* is a simple raw-text alert format as seen below in Figure 1 below. While not the most efficient method, it did allow easy parsing and the performance impact was negligible with the size of PCAPs used.

```
12/11/2012-03:22:29.614826 www.e-zeeinternet.com [**]
/count.php?page=952020&style=LED_g&nbdigits=9 [**] Opera/10 (Windows
NT 5.1; US; x86) [**] 10.74.26.100:1048 -> 209.68.32.176:80
```

Figure 1: Example http log entry from Suricata

The HTTP log contains the requests including the host URL, the URI with query string, the user-agent and the destination IP address. Since it is common for malware to use HTTP as a means of communication, the study of this paper has been focused on this protocol.

```
12/11/2012-03:22:21.348345  [**] [1:2803880:2] ETPRO TROJAN Win32/Sality.AT
Checkin [**] [Classification: A Network Trojan was detected] [Priority: 1] {TCP}
10.74.30.100:1030 -> 66.228.49.83:443
```

Figure 2: Example fast-alert log entry from Suricata

Suricata's TLS log records the certificate information used during an HTTPS request, which includes information about the subject (certificate owner), and the issuer of the certificate. An example of this is presented below.

```
12/11/2012-03:22:02.452162 10.74.33.100:1029 -> 149.12.66.231:443  TLS:
Subject='C=--, ST=SomeState, L=SomeCity, O=SomeOrganization,
OU=SomeOrganizationalUnit,
CN=localhost.localdomain/emailAddress=root@.localdomain' Issuerdn='C=--
, ST=SomeState, L=SomeCity, O=SomeOrganization,
OU=SomeOrganizationalUnit,
CN=localhost.localdomain/emailAddress=root@.localdomain'
SHA1='f5:1f:08:49:a9:4b:44:d4:f3:ce:a9:2b:e6:55:f3:3a:e3:10:cb:f4'
VERSION='TLSv1'
```

Figure 3: Example TLS log entry from Suricata

The stats log has information generated by the engine during processing. Examples include the number of packets and bytes by protocol, average packet size, and memory used during processing the PCAP. This information was not used except to identify cases where detection did not happen due to limited data being sent.

### 3.3 Emerging Threats Rule Set

Quality network signatures are required by an IDS to have success in identifying malicious traffic. Suricata supports the ability to use both Snort VRT signatures and the signatures from the Emerging Threats community and company. For the research an Emerging Threats Pro account was used. Pro was chosen over the community version since it includes additional signatures targeting malware. Although the testing and investigation occurred over several months and included different versions of the rule-set, the final data is based on rules downloaded on April 22, 2013, and the default configuration file used for rule activation.

## 3.4 VirusTotal

The VirusTotal (VT) website (VirusTotal, 2013) includes a wealth of information about malware, suspected malware, and clean files. During the vetting process, samples were selected that included multiple detections on VT. A few exceptions were made that were known to be malicious but had yet to show up on VirusTotal. Access to VirusTotal's data is free via a rate limited API or through their website. A Python script was used to download scan reports for each sample, and incorporated into Malware Tracks database for analysis.

## 3.5 Limitations of Analysis

There are three limitations to the analysis process that should be addressed up front. They don't affect the applicability of what is presented, but limit the scientific value of the findings. Two of the three limitations exist because of the nature of malware.

**Malware is non-deterministic:**

Anyone who has spent time looking at malware knows it does not always act the same. An example of this is VM-aware samples such as covered by Alien Vault Labs on their blog (Alien Vault Labs, 2012). Changing behavior is more prevalent when the sample has access to the Internet and external stimulus is not controlled. There is no way to guarantee a sample will always behave the same or even execute. This could be due to the environment it runs in, different commands from the C&C servers, bugs in the malware, availability of online resources, and other factors. For this research, that means the data presented is closely tied to the specific PCAPs generated during testing. If tested again the results would have variations.

**Malware may not act maliciously:**

An inherent flaw of behavioral analysis is the reliance on the sample acting malicious. It is possible that a sample is programmed to not act maliciously early on. However, just because a sample doesn't act maliciously in 60 seconds doesn't mean it isn't bad. In the sample set used, all files had malicious host-based events, or a high hit count on VirusTotal but that did not guarantee malicious network activity would take

place. Samples that did not exhibit any network traffic were not included in the final sample set.

**Sample set was not normalized:**

No attempt was made to ensure the same number of samples from malware families or classifications existed. Since the goal was to provide some understanding around malware traffic, not statistical analysis of traffic the lack of normalization is not a problem.

## 4. About the Malware Used

The malware used for the paper was gathered over several months from the backend feeds of the author's employer Norman Shark. The final sample set had 20,587 unique files and was limited to Windows executables and DLLs for scope reasons. All samples had to generate some traffic during their execution. For those wishing to do similar research, or look at malware in general, websites such as virusshare.com (VirusShare, 2013) and contagiodump (Contagio, 2013) are good places to acquire samples.

Based upon the VirusTotal scan results a wide collection of malware families are represented. Absolute classification of the malware by AV software is difficult due to differences in naming and classification styles. The paper *Automatic Classification and Analysis of Internet Malware*, (Bailey et al., 2007) concludes that AV alone for classification is not complete, consistent, and their conciseness varies greatly. The paper goes on to describe a method for clustering based on host events, which is worth reading for those interested. Clustering can also be done based on network traffic as illustrated in *Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces* (Perdisci, Lee, Feamster, 2010). Clearly clustering is a topic in itself and beyond the scope covered here.

| | |
|---|---|
| (GUI) Intel 80386, for MS Windows, WISE installer self-extracting archive | 1 |
| (DLL) (console) Intel 80386 (stripped to external PDB), for MS Windows, UPX compressed | 1 |
| (GUI) Intel 80386, for MS Windows, InstallShield self-extracting archive | 1 |
| (GUI) Intel 80386 (stripped to external PDB) system file, for MS Windows, UPX compressed | 1 |
| (DLL) (console) Intel 80386 Mono/.Net assembly, for MS Windows | 2 |
| (DLL) (GUI) Intel 80386 (stripped to external PDB), for MS Windows, UPX compressed | 2 |
| (DLL) (GUI) Intel 80386 Mono/.Net assembly, for MS Windows | 2 |
| (GUI) Intel 80386, for MS Windows, UPX compressed, Nullsoft Installer self-extracting archive | 2 |
| (DLL) Intel 80386, for MS Windows | 2 |
| (GUI) Intel 80386, for MS Windows, UPX compressed, RAR self-extracting archive | 3 |
| (DLL) (console) Intel 80386, for MS Windows, PECompact2 compressed | 4 |
| (DLL) (native) Intel 80386, for MS Windows | 5 |
| (console) Intel 80386, for MS Windows, UPX compressed | 5 |
| (console) Intel 80386 (stripped to external PDB), for MS Windows | 5 |
| (console) Intel 80386 Mono/.Net assembly, for MS Windows | 6 |
| (GUI) Intel 80386 Mono/.Net assembly, for MS Windows, COFF | 8 |
| Intel 80386 (stripped to external PDB), for MS Windows, PECompact2 compressed | 8 |
| (GUI) Intel 80386, for MS Windows, MS CAB-Installer self-extracting archive | 8 |
| (GUI) Intel 80386 (stripped to external PDB), for MS Windows, MS CAB-Installer self-extracting archive | 10 |
| (console) Intel 80386, for MS Windows, PECompact2 compressed | 10 |
| (DLL) (console) Intel 80386 (stripped to external PDB), for MS Windows | 11 |
| (GUI) Intel 80386, for MS Windows, RAR self-extracting archive | 12 |
| (GUI) Intel 80386 (stripped to external PDB), for MS Windows, UPX compressed | 13 |
| (GUI) Intel 80386, for MS Windows, Petite compressed | 15 |
| (GUI) Intel 80386 system file, for MS Windows | 25 |
| (DLL) (GUI) Intel 80386, for MS Windows, PECompact2 compressed | 25 |
| (DLL) (GUI) Intel 80386 (stripped to external PDB), for MS Windows | 34 |
| (DLL) (console) Intel 80386, for MS Windows | 85 |
| (console) Intel 80386, for MS Windows | 98 |
| (DLL) (GUI) Intel 80386, for MS Windows, UPX compressed | 117 |
| MS-DOS executable | 117 |
| (GUI) Intel 80386, for MS Windows, PECompact2 compressed | 155 |
| MS-DOS executable, MZ for MS-DOS | 186 |
| (GUI) Intel 80386, for MS Windows, Nullsoft Installer self-extracting archive | 257 |
| (GUI) Intel 80386 (stripped to external PDB), for MS Windows | 445 |
| (GUI) Intel 80386 Mono/.Net assembly, for MS Windows | 510 |
| (GUI) Intel 80386 (stripped to external PDB), for MS Windows, Nullsoft Installer self-extracting archive | 536 |
| (GUI) Intel 80386, for MS Windows, UPX compressed | 1555 |
| (DLL) (GUI) Intel 80386, for MS Windows | 3421 |
| (GUI) Intel 80386, for MS Windows | 12884 |

Figure 4: File types of sample set as determined by libmagic

Samples were selected at random from the thousands of samples processed each day by the Norman Shark backend feed. This resulted in an uneven distribution of families and behavior. This limits the statistical precision of the research, yet it does not negate the value in understanding the network behavior. Network defenders do not see a statistical norm of malware samples attacking their network in the real world.

Even though samples had multiple detections and a high behavioral risk score no vendor detected every sample. This should reinforce the need to observe network behavior of malware when trying to both detect and understand it.

Kiel Wadner, wadnerk@gmail.com

## 4.1 Common Specimens

The following two charts breakdown the samples to provide an understanding of the diversity. The AV vendors were selected based on high detection rates in the sample set, and having a descriptive naming convention to provide the specimen identification. The two used vendors were AntiVira and Kaspersky. Many other products could have been selected and provided results of equal quality.

According to the AntiVira (Avira, 2013) results, a total of 3398 unique identifiers were in the sample set. Twenty-six of those had a hundred or more occurrences. The most instances were for TR/Dropper.Gen that identifies generic piece of malware that drops a file. Without more detailed clustering it is not easily determined how unique those specimens truly are. There were 2784 identifiers that occurred only one or two

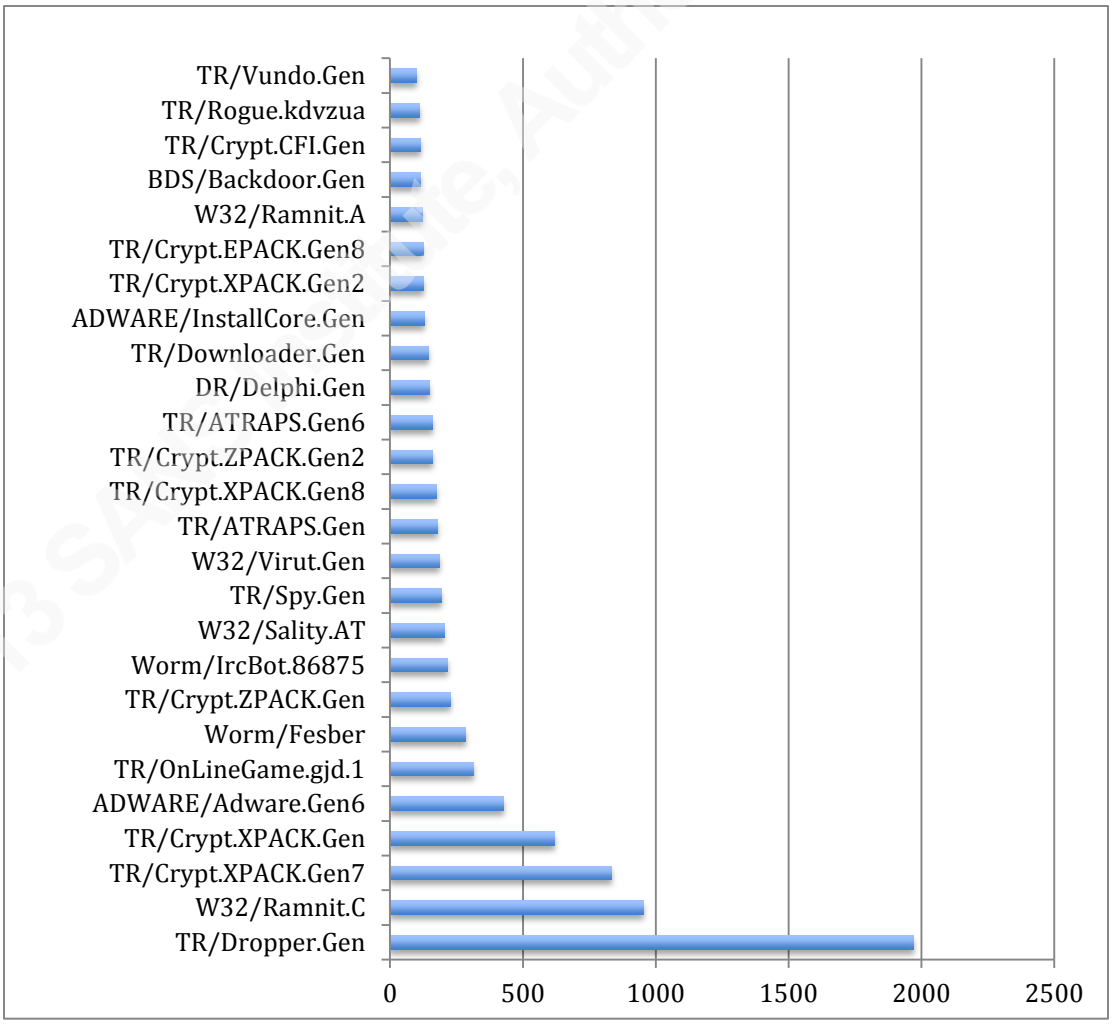


Figure 5: Antivira sample identifiers with 100 or more instances

Kiel Wadner, wadnerk@gmail.com

times. There are four named specimens that stood out as influencing the results given how many instances there were. They are: Ramnit, Sality and Virut. Ramnit is used to steal sensitive information, and connects to a random appearing URL (Microsoft, 2013a). Among other things Sality creates a peer-to-peer botnet over UDP, downloads additional components (Symantec, 2013). Virut will often communicate over IRC to command and control servers (Microsoft, 2013b).

The second AV product used for comparison was from Kaspersky (Kaspersky Lab US, 2013), which had 5528 unique identifiers. Sality and Virut, which were in the top list for AntiVira, also appeared in the Kaspersky results. The malware called Nimnul by Kaspersky is an alias for Ramnit on the AntiVira list. At 4983, the Kaspersky results had a higher number of identifiers that only occurring once or twice. The top identifier was again a generic heuristic.

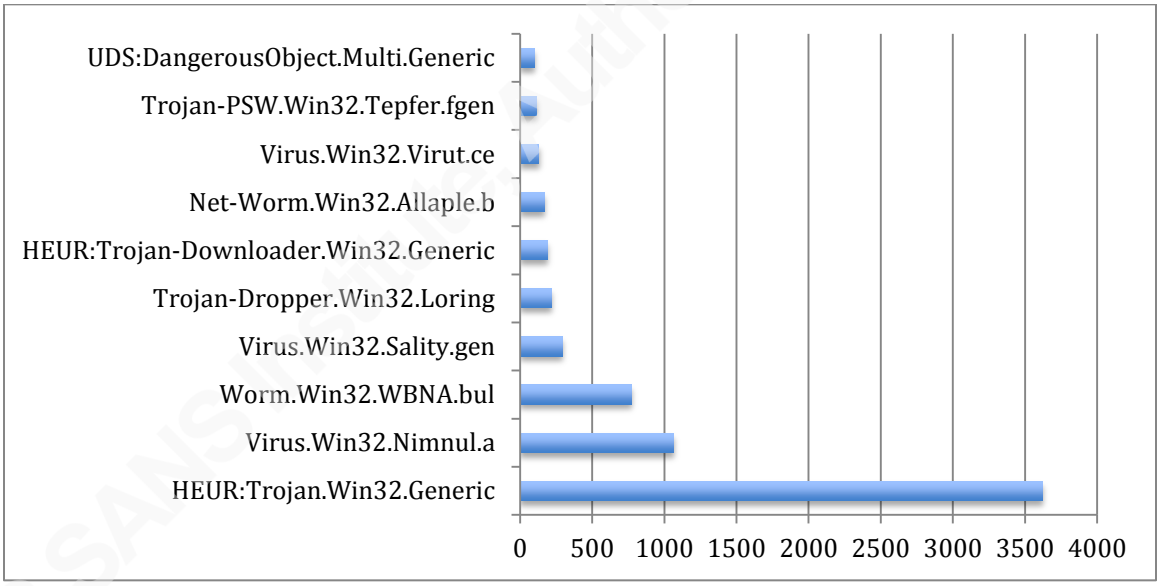


Figure 5: Kaspersky identifiers with 100 or more instances

## 5. High Level Overview of Traffic

This section provides a "10,000 feet" perspective of the traffic observed. The direct application might be minimal but they offer insight into malware communication methods.

Despite a history of using IRC and a gradual move to P2P, malware still utilizes HTTP heavily as a network protocol. This makes sense given the legitimate uses; it would not be feasible to out right block HTTP like IRC or some P2P protocols. During investigation, 49.77% of the samples utilized HTTP in the 60 seconds of execution. It is possible given a longer execution time more samples would have utilized HTTP. This number would also increase if samples that generated minimal amounts of traffic were excluded. Examples are samples that sent a ping request but didn't receive a reply, or those that initiate with a SYN and get a RST.

The following three figures show the number of samples that use each identified protocol type. The information was generated via tshark's protocol hierarchy statistics command: *tshark -qz io,phs -r capture.pcap*. This information was then flattened to the transport protocol. By using the sample count instead amount of data sent, the relative frequency of the protocols can be more easily seen instead of a single sample that generated a lot of traffic.

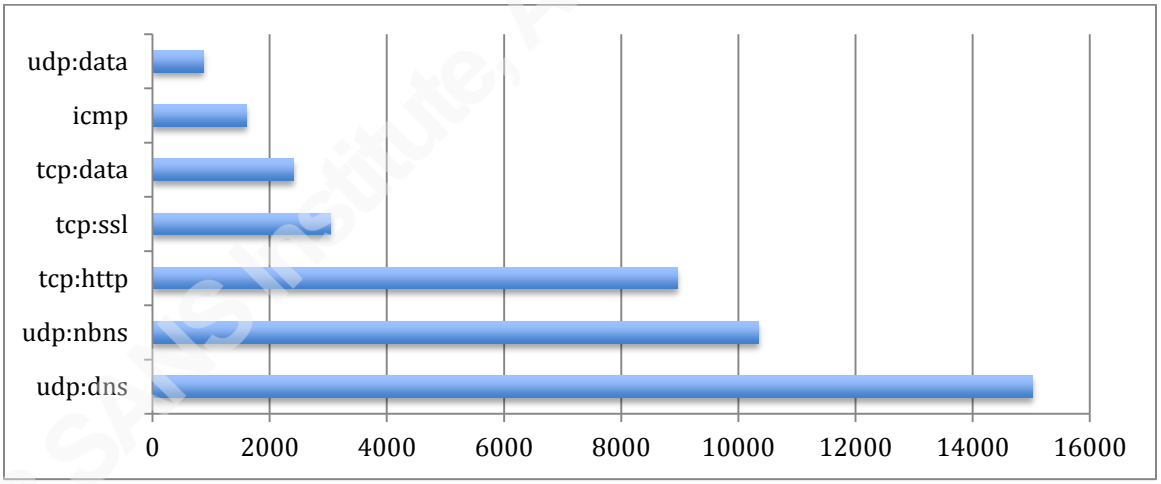


Figure 6: Protocols according to tshark that occur in 1000 samples or more

There are no surprises in the top protocols used. Just over 15,000 samples utilized DNS. It makes sense a majority of the samples utilize DNS to locate their network resources. This provides resiliency to their network and allows them to utilize techniques such as fast-flux networks (Salusky, Danford, 2007). It cannot be said that the remaining samples would not use DNS in other situations. Further analysis would have been needed to understand why DNS requests did not occur. The NetBios Name Service traffic (udp:nbns) is not unexpected, nor necessarily malicious. No strong conclusions

can be made for the tcp:data and udp:data results without closer inspection of the PCAP files. These appear to be cases where tshark was unable to determine the protocol used.

The next chart shows protocols that occurred at least 10 times, but wasn't one of the top values. Several of these standout as interesting. The first is tcp:reload-framing,

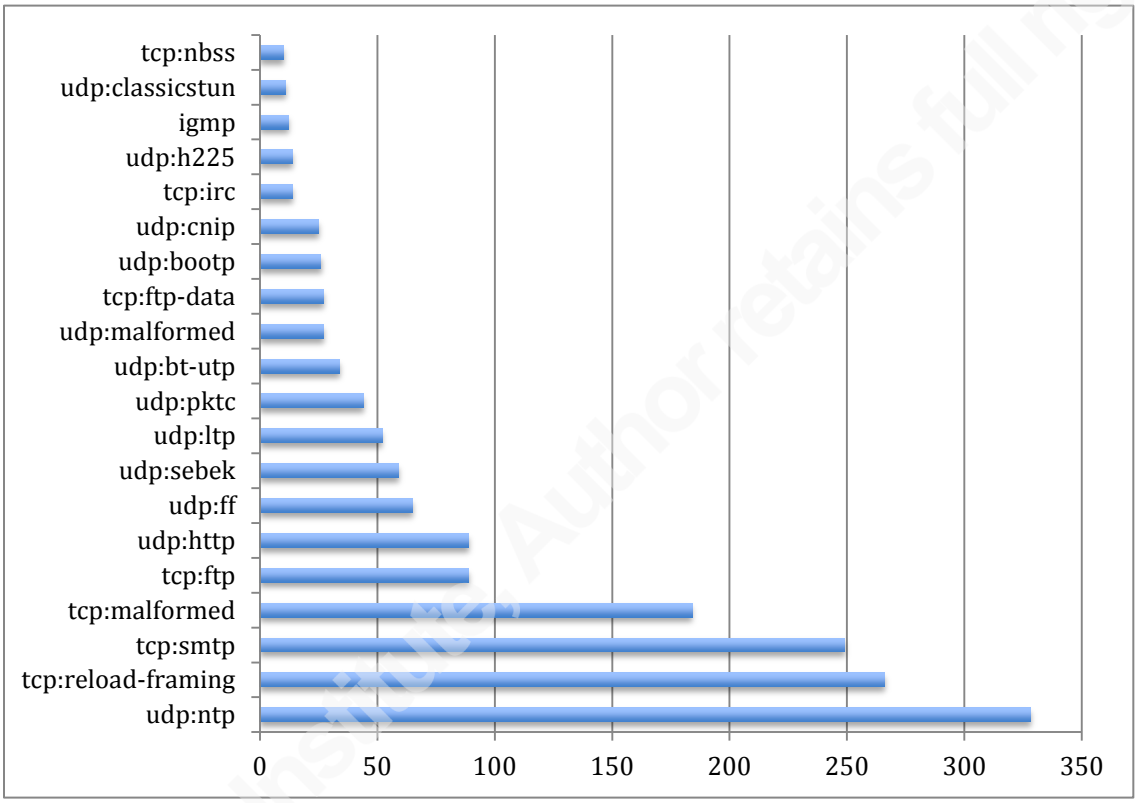


Figure 8: Protocols according to tshark that occur in 10 to 350 samples

which according to the RFC is, "A P2P signaling protocol provides its clients with an abstract storage and messaging service between a set of cooperating peers that form the overlay network" (Jennings et al., 2013). This combined with uTorrent Transport Protocol (udp:bt-utp) are strong indicates that samples were trying to utilize some type of P2P network.

NetBIOS (tcp:nbss) is a legacy protocol used prior to Windows Vista to communicate across a local network (Microsoft, 2010), and is known to be used by malware to spread (Aquilina, Casey, Malin, 2008).

**Unique Domains and DNS**

There were 106,418 individual DNS Query Requests for 25,464 unique hosts indicating multiple samples requested the same hosts. HTTP requests were only made to roughly half of those hosts (11,570). The responses were not investigated so it is not clear if those hosts were no longer resolvable, or if perhaps the IP result was used by other protocols. The MAG2 appliance was set to use 8.8.8.8, Google's public DNS server for all requests. The following chart shows the top requested hosts from the DNSQR.

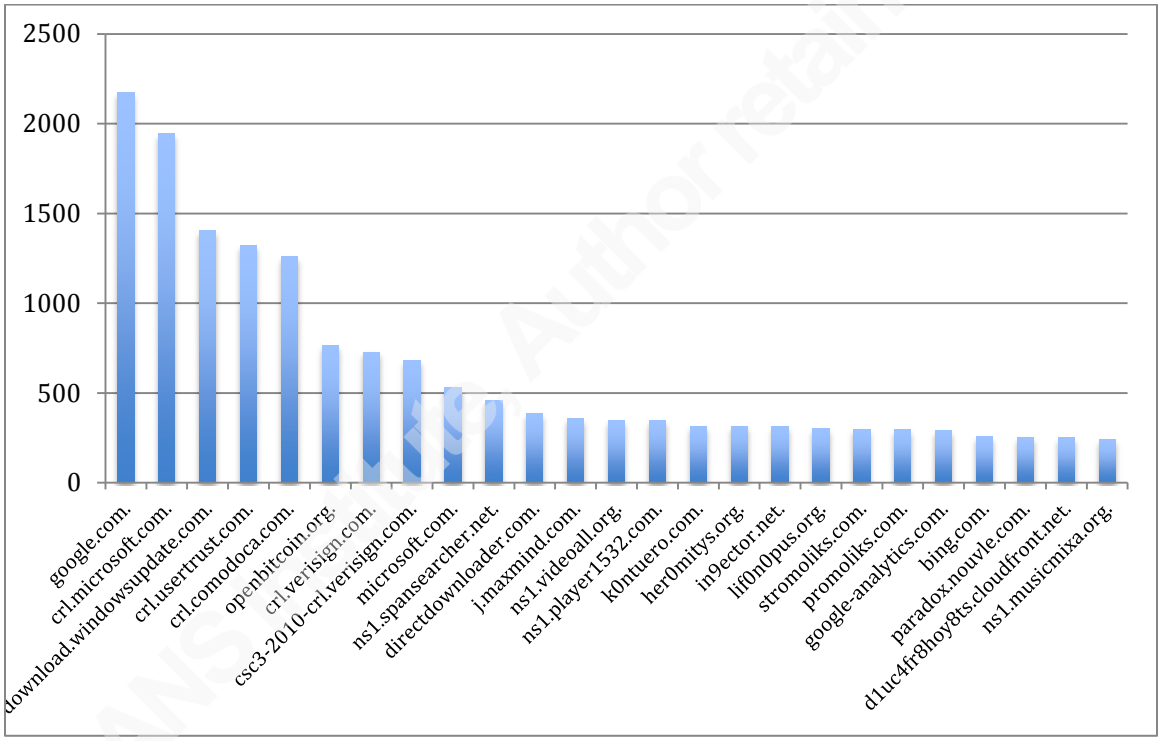


Figure 7: Top domains by number of DNS Query Requests

Several of the results are expected, like the high number of requests for Google.com, Verisign and Microsoft. The number of requests to openbitcoin.org would suggest some of the samples were interested in Bit Coin mining. CloudFront is Amazon's web service for content delivery. The trend of using cloud resources was evident with approximately 40 different hosts queried that were on the cloud-front, aws, or linode providers. If BitCoin mining is not allowed in your environment spotting a DNS request related to it is a clear signal something needs investigated.

Kiel Wadner, wadnerk@gmail.com

**IDS Alerts**

The Emerging Threats rule-set provides some useful information beyond just detection. Since the majority of the traffic was generated by malware even informational alerts help understand what occurred. There were 832 unique IDS rules triggered. Rules that flagged 100 or more PCAPS are shown in the table. Something that stood out immediately is the high number of PCAPs flagged with Vobus or Sality (over 1200 each). According to the AV results, less then 250 samples each were identified as these. This is a case where the behavior analysis can provide a clearer picture then the static analysis typically done by antivirus products.

| Rule Name | # of pcaps |
|---|---|
| ET TROJAN Vobfus/Changeup/Chinky Download Command | 1233 |
| ETPRO TROJAN Win32/Sality.AT Checkin | 1226 |
| ET MALWARE User-Agent (Mozilla/4.0 (compatible)) | 1078 |
| ETPRO TROJAN Worm.Win32.Vobfus Checkin 1 | 970 |
| ETPRO MALWARE ADWARE/InstallCore.Gen Checkin | 808 |
| ET POLICY Cnet App Download and Checkin | 696 |
| ET INFO DYNAMIC_DNS Query to a Suspicious no-ip Domain | 670 |
| ET CNC Zeus/Spyeye/Palevo Tracker Reported CnC Server (group 6) | 481 |
| ET TROJAN ZeroAccess udp traffic detected | 394 |
| ETPRO MALWARE AdWare.Win32.DirectDown.A Install | 383 |
| ETPRO MALWARE Toolbar Download | 379 |
| ET TROJAN ZeroAccess Outbound udp traffic detected | 365 |
| ET POLICY Maxmind geoip check to /app/geoip.js | 359 |
| ETPRO MALWARE Adware.DirectDownloader Checkin | 358 |
| ET DNS Reply Sinkhole - Zinkhole.org | 331 |
| ET POLICY Outdated Windows Flash Version IE | 305 |
| ET TROJAN Murlo Trojan Checkin | 267 |
| ETPRO USER_AGENTS Suspicious user agent (Google page) | 264 |
| ET DNS Reply Sinkhole - sinkhole.cert.pl 148.81.111.111 | 262 |
| ETPRO MALWARE Adware.iBryte.B Install | 257 |

Kiel Wadner, wadnerk@gmail.com

| | |
|---|---|
| ETPRO MALWARE Win32.AdWare.iBryte.C Install | 252 |
| ETPRO TROJAN Worm.Win32.Vobfus Checkin 3 | 243 |
| ET DNS Reply Sinkhole - Dr. Web | 238 |
| ET MALWARE Suspicious Mozilla User-Agent - Likely Fake (Mozilla/4.0) | 204 |
| ETPRO MALWARE Adware.Win32/Hotbar User-Agent (RPCriCheck) | 204 |
| ET DNS Non-DNS or Non-Compliant DNS traffic on DNS port Reserved Bit Set | 203 |
| ET POLICY DNS Query for .su TLD (Soviet Union) Often Malware Related | 203 |
| ET INFO DYNAMIC_DNS Query to 3322.org Domain | 201 |
| ET DNS Non-DNS or Non-Compliant DNS traffic on DNS port Opcode 8 through 15 set | 196 |
| ET MALWARE Adware.Gen5 Reporting | 177 |
| ET TROJAN System Progressive Detection FakeAV (INTEL) | 162 |
| ET TROJAN Simda.C Checkin | 152 |
| ET CNC Zeus/Spyeye/Palevo Tracker Reported CnC Server (group 25) | 127 |
| ET CNC Zeus/Spyeye/Palevo Tracker Reported CnC Server (group 24) | 127 |
| ET TROJAN WORM_VOBFUS Checkin Generic | 115 |
| ET CURRENT_EVENTS Known Hostile Domain ilo.brenz.pl Lookup | 103 |
| ETPRO TROJAN Backdoor.Win32/Simda.gen!A Checkin | 102 |
| ETPRO POLICY Suspicious User-Agent (LuaSocket) | 102 |
| ETPRO MALWARE TROJ_OPENCANDY_0000000.TOMA Install | 102 |
| ET POLICY External IP Lookup Attempt To Wipmania | 101 |
| ETPRO TROJAN Hotbar/Clickpotato.tv Checkin | 100 |
| ETPRO MALWARE Hotbar Spyware Reporting to vic.asp | 100 |

Figure 8: Emerging Threats rule names that flagged 100 or more PCAPS

The second alert of interest were 831 samples flagged with sinkhole traffic. Sinkholes work "by intercepting outbound DNS requests attempting to access known malicious domains, such as botnets, spyware, and fake antivirus, an organization can control the response and prevent organization computers from connecting to these domains" (Bruneau, 2010). This indicates at least some of the malware's network infrastructure has been taken down. It would also imply that using a sinkhole could be an

effect counter-measure against malware talking to specific domains. Bruneau's paper is a good starting for those wanting to know more.

Five other alerts are related to DNS traffic in the top list indicating this might be a good technique for identifying malware. Of interest is the 399 PCAPS that were flagged for non-compliant DNS traffic. Unfortunately time did not allow a thorough examination of these PCAPs, but it is unlikely these are a natural artifact of the analysis machine or process.

**User-Agents**

One of the most interesting items discovered during analysis was the number of unique User-Agents encountered and how many of them are clearly wrong. The User-Agent field is an HTTP header included with web requests that identifies the application making the request. The following table shows the top 25 agents based on the number of connections made. HTTP requests from the sample set included 868 unique user-agent values; of those 326 are incorrect browser identifications! Incorrect browser identification can be attributed to 7,487 unique samples or about 73% of all malware samples that used HTTP and 36% of all samples.

| User-Agent String | times |
|---|---|
| Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 2.0.50727; .NET CLR 3.0.04506.648; .NET CLR 3.5.21022) | 46429 |
| Mozilla/5.0 | 35131 |
| Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1) | 24901 |
| Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0; .NET CLR 2.0.50727; .NET CLR 3.0.04506.648; .NET CLR 3.5.21022) | 13428 |
| Microsoft-CryptoAPI/5.131.2600.5512 | 13184 |
| <useragent unknown> | 6806 |
| NSISDL/1.2 (Mozilla) | 5137 |

| | |
|---|---|
| Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; SV1) | 3372 |
| Mozilla/4.0 (compatible; MSIE 2.0; Windows NT 5.0; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0) | 2850 |
| Better Installer(Mozilla) | 2535 |
| Mozilla/5.0 (Windows; U; Windows NT 6.0; pt-PT; rv:1.9.2.17) Gecko/20110420 Firefox/3.6.17 | 2381 |
| fbi.gov | 2170 |
| Mozilla/4.0 (compatible) | 2060 |
| Mozilla/5.0 (Windows; U; Windows NT 6.1; sv-SE; rv:1.9.2.17) Gecko/20110420 Firefox/3.6.17 | 1918 |
| NSIS_ToolkitOffers (Mozilla) | 1562 |
| Mozilla/5.0 (Windows; U; Windows NT 5.1; sv-SE; rv:1.9.2.17) Gecko/20110420 Firefox/3.6.17 | 1068 |
| Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1) | 882 |
| Opera/10 (Windows NT 5.1; US; x86) | 834 |
| Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0) | 766 |
| Mozilla/5.0 (Windows; U; Windows NT 5.1; pt-PT; rv:1.9.2.17) Gecko/20110420 Firefox/3.6.17 | 611 |
| Mozilla/5.0 (Windows; U; Windows NT 6.0; en-GB; rv:1.9.2.17) Gecko/20110420 Firefox/3.6.17 | 610 |
| Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; SV1; .NET CLR 1.1.4322) | 551 |
| User-Agent: Opera/10.60 Presto/2.2.30 | 489 |
| Mozilla/4.0 | 477 |
| Mozila | 423 |

Figure 9: User-Agents with incorrect web-browser identification

An entry of <*useragent unknown*> indicates that Suricata failed to parse the value. When reviewing the chart, remember that these were generated on a Windows XP (English) machine, with only Internet Explorer 8 installed. In that controlled environment, it is easier to see the oddities such as Opera, old version of the CLR, non-US English versions, and even a User-Agent of "fbi.gov".

In production environments, blocking based on User-Agents may not be possible, but keeping a list of what has been seen may allow administrators to notice when something is wrong. Like the other suggestions above, tracking and automating an environment to look for such oddities can provide an additional layer of detection beyond traditional IDS systems.

**Common Ports**

The following two tables show the top 20 TCP and UDP ports that were in use. Unsurprisingly HTTP (80), HTTPS (443), SMTP (25) were the most used TCP ports. Ports 8000, and 8080 are also common for HTTP. The port 25 makes it likely that at least one of the samples was sending SPAM messages during its short-lived life.

| TCP Port | Packets | TCP Port | Packets |
|---|---|---|---|
| **80** | 7043303 | **1177** | 2791 |
| **443** | 186097 | **1935** | 2464 |
| **25** | 88301 | **1604** | 2062 |
| **16471** | 40088 | **447** | 1946 |
| **81** | 9587 | **139** | 1923 |
| **8080** | 9266 | **21** | 1732 |
| **8000** | 6076 | **6667** | 1440 |
| **34354** | 4248 | **25417** | 1401 |
| **16464** | 3305 | **8090** | 1281 |
| **16629** | 2920 | **53** | 1277 |

Figure 10: The top 20 TCP destination ports

According to a 2012 report by Kevin McNamee at Kindsight Security Labs (McNamee, 2012a), TCP ports 16471 and 16464 have been associated with ZeroAccess/ Sirefef. Another Kindsight report (McNamee, 2012b) links port 34354 to ZeroAccess as well.

The UDP Port usage tells a similar story. As expected DNS (port 53) was most common. UDP ports 16471 and 16464 are likely connected to ZeroAccess (based on the same Kindsight report). The Suricata results indicated 365 samples sent UDP traffic associated ZeroAccess, which matches up.

The UDP port 80 was initially surprising, but Johannes Ullrich posted an ISC Diary back in 2006 about malware using that port (Ullrich, 2006). This is not a new method of obfuscation or trying to by pass firewalls and continues to be used.

| UDP Port | Packets | UDP Port | Packets |
|---|---|---|---|
| **53** | 152292 | **9120** | 518 |
| **8010** | 22314 | **123** | 437 |
| **16471** | 19002 | **5041** | 432 |
| **80** | 7371 | **52371** | 411 |
| **16464** | 6457 | **1900** | 285 |
| **8090** | 5186 | **11937** | 189 |
| **8001** | 3054 | **3478** | 180 |
| **42113** | 2681 | **8000** | 154 |
| **62111** | 1250 | **25549** | 151 |
| **0** | 690 | **5100** | 136 |

Figure 11: Common UDP destination ports

In most environments it won't make sense to block ports known to be associated with malware via a firewall. However, watching for an unusual increase on those ports may be enough of a reason for a closer look.

## 6. Deep Dives

The next sections looks closer look at items that were found to be both interesting and have application in understanding malware network behavior.

Kiel Wadner, wadnerk@gmail.com

## 6.1 TLS Certificates

As the cat-and-mouse game continues between malware authors and the security industry, the malware authors continue to take action to secure their networks. Rossow, et al., noted in their researcher a mix of well-known techniques to customer encryption or obfuscation. (Rossow, et al., 2012).

During post-analysis of the sample set, the TLS certificates were reviewed. Transport Layer Security (TLS) is the successor of SSL and aims to "provide communications security over the Internet." (Dierks, Rescorla, , 2008). When a web resource is accessed over TLS (aka HTTPS), the web server presents its TLS certificate for verification by the client. Suricata kindly records some of this certificate information in the tls.log file that was post-processed. While this doesn't give access to the encrypted data, some oddities do stand out as useful (or at least interesting).

In the traffic set, 64 unique certificate requests occurred. The majority of these were for legitimate locations such as *.google.com, or *.g.doubleclick.net, but some were suspicious. One example is the 14 certificates where both the subscriber and issuer

| issuer cname | subscriber cname | SHA1 |
|---|---|---|
| www.sleu6xoy.com | www.3lwbeouf22kwfcgwh4.net | 1a:39:75:16 |
| www.xob3xic6xsmge.com | www.j6kzxwaiq4hto5ia.net | 48:24:94:73 |
| www.r3nogzlghdt57s.com | www.tts7imed2rbr.net | 2b:e1:5b:69 |
| www.gxbm6mgpw.com | www.ppwf62hyaale56d2.net | 04:39:9f:c1: |
| www.izs2gg4ibuqdck.com | www.zgybyityeptv.net | ae:31:55:2f: |
| www.xatmurynqmzwzn6l5v6h.com | www.dgv3auvosuuaupjza.net | 5b:e3:22:0f: |
| www.rpqkhr22f5msq.com | www.i2drxyroexbbmj.net | 9a:b6:98:09 |
| www.pnunsxy6x7tekvjyaq.com | www.5f54zuziiuiwouy7w7.net | 89:d0:e3:79 |
| www.3p7mb56cbeo4t3.com | www.y3y7sqi4mlouv2.net | 65:04:99:68 |
| www.zbpoqwi5kc4zpc4sul.com | www.4mtzvinkuap3ga4.net | 9c:3e:43:4e |
| www.wpvosfay34pyj7rw.com | www.gagbeht5w7q.net | f3:6a:b3:57: |
| www.sylmzsxcqjllu.com | www.yimisrrhrwdbhk42z5t.net | 3e:4d:97:5c |
| www.yaa4vezdyxbumhi7.com | www.t2g3viexnnu6kzg6hwgt.net | 6a:b4:55:3f: |

Figure 12: TLS certificate issuers with pseudo-random cnames

organization were empty, as were their country, state, and locality fields. Looking at those 14 items reveal some suspicious looking CNAMEs.

Looking at the alerts we can see a bit more going on. Each of the 14 samples triggered the same two alerts:

"[2014932] DynDNS CheckIp External IP Address Server Response"

"[2012758] DYNAMIC_DNS Query to *.dyndns. Domain"

Other oddies noticed include:

- "MyCompany Ltd." as a issuer organization
- "lolcat" for a cname
- Black issuer cnames
- Bad locations such as "Sometown US"

Based on observations during the research, TLS certificate anomalies could be a good artifact during incident response – assuming this information is available to you. The human eye and brain is good at picking out things that aren't quite right. Spotting suspect certificates from a host could indicate a compromised machine, and perhaps help identify who or what is behind it.

From an automated detection perspective, the issuer information would be a good place to focus. Issuers that do not contain at least one actual word are worth a closer look, as are ones with known bad such as 'lolcat', and 'MyCompany Ltd.'. Through organized automated bulk analysis as was done for this research, a blacklist of sorts could be generated and maintained to provide this information.

## 6.2 Emerging Threats Rule Hits

The Emerging Threats Pro rule set with Suricata flagged 10,743 PCAPs via 832 different rules. While not amazing it does mean 52% of total sample set was detected within the first minute of being active. A higher detection rate had been expected so the missed detections were looked at closer.

The byte and packet statistics for missed PCAPs were exported to Excel and sorted by packet count. What was noticed was a majority missed PCAPs had a minimal amount of traffic inbound or outbound. Although all PCAP files had some traffic, the initial filtering did not require a specific amount. A subsection of the missed were opened in Wireshark and a few patterns emerged. First was outgoing DNS requests with no reply. The second was SYN packets followed by a RST. Third was only traffic on

the internal network – NetBIOS for example. It is difficult to detect malicious intent without sufficient traffic, so filtering was applied.

It was decided that more then 3 packets needed to be both sent and received outside of the internal network to count towards the detection rate. This is still a low bar considering at least two packets are needed by the sender to complete a 3-way handshake. When the 7163 PCAPs that did not meet the minimum traffic criteria are not counted the detection rate jumps to 81%! This is a respectable amount and shows that existing IDS technology can do a decent job at detecting malware specific threats.

This is a good time to pause and look at these percentages differently. Nearly 35% of the malware sample set generated minimal traffic within one minute. Closer behavioral analysis would be required to fully understand why, or if a longer duration would have allowed detection.

## 6.3 HTTP Requests – An Adware Example

User-Agents are not the only telling piece of information from HTTP traffic. The requests themselves include valuable information in correlating samples and understanding what is occurring. One of the items that stood out in the HTTP requests were samples that talked to legitimate service providers such as YouTube, Amazon, or Azure.

There was one Amazon AWS host in particular that was contacted by 32 different malware samples. These will serve as the basis for the example. The VirusTotal scan reports indicate the samples are a form of adware and likely a variant of *iBryte*, which is used to serve ads, generate click revenue, and distribute malware. In Q3 of 2012 Kaspersky ranked *iBryte* as the 7th most malicious program on the Internet (Kaspersky Lab US, 2012). The main method of distribution for *iBryte* is to bundle its ad components with legitimate software (Sophos, 2013). The following picture is a subset of the requests to the AWS host from two different samples. It contains four requests from *b10a5d*[2], followed by a single request *b3b252*.

[2] The first 6 characters of the sample's MD5

| | |
|---|---|
| b10a5d | /impression.do/?user_id=83e83165-69e0-4ed3-92b3-f626efdc5f83&event=setup_run&spsource=myfreedownload_picasaalbumdown&traffic_source=myfreedownload&offer_id=myfreedownload_picasaalbumdown |
| b10a5d | /impression.do/?user_id=83e83165-69e0-4ed3-92b3-f626efdc5f83&event=dpi_1&spsource=myfreedownload_picasaalbumdown&traffic_source=myfreedownload&offer_id=myfreedownload_picasaalbumdown |
| b10a5d | /impression.do/?user_id=83e83165-69e0-4ed3-92b3-f626efdc5f83&event=json_installer_initialize_1081&spsource=myfreedownload_picasaalbumdown&traffic_source=myfreedownload&offer_id=myfreedownload_picasaalbumdown |
| b10a5d | /impression.do/?user_id=83e83165-69e0-4ed3-92b3-f626efdc5f83&event=installer_initialize_6629&spsource=myfreedownload_picasaalbumdown&traffic_source=myfreedownload&offer_id=myfreedownload_picasaalbumdown |
| b3b252 | /impression.do/?user_id=7943651e-9f9f-483e-accc-471de3c6f82d&event=setup_run&spsource=Adperi0_mplayer_us_direct&traffic_source=Adperio&offer_id=mplayer |

Figure 15: URI requests from adware samples

The first thing noticed was the *user_id* parameter. This changed with each sample, and likely is used as a way to track the install base. The second thing is the query parameters indicating a *source* include "matomy". Others not pictured included "DSNR". These likely reference two advertising and marketing groups. The *Matomy Group*, is an advertising and marketing company that according to their website (Matomy, 2013), "offer[s] publishers and advertisers the complete range of performance-based marketing solutions on web, mobile and social platforms." One of their companies, *adperio*, list Blockbuster, Discover, Experian, Netflix and others customers (Adperio, 2013). DSNR, which is part of DMG (DMG, 2013) is another legitimate advertising company. To be clear: the paper's author is not implying any malicious intent by Matomy or DSNR. The connection between the samples and the two advertising networks is pointed out because using advertising networks is a known way to both distribute malware and to generate revenue for malware authors through ad-clicks.

So what is happening with these similar looking requests? To determine that the focus will turn to an individual PCAP file and the conversations that occurred. Using Wireshark it was determined that four TCP flows occurred in the selected PCAP.

**Flow 1:**

Downloads four 1x1 pixel PNGs from "impressions" amazonaws.com site. Using 1x1 pixel images is a common technique for tracking and website navigation. These four PNGs were downloaded in four different requests. This in itself is not malicious.

Kiel Wadner, wadnerk@gmail.com

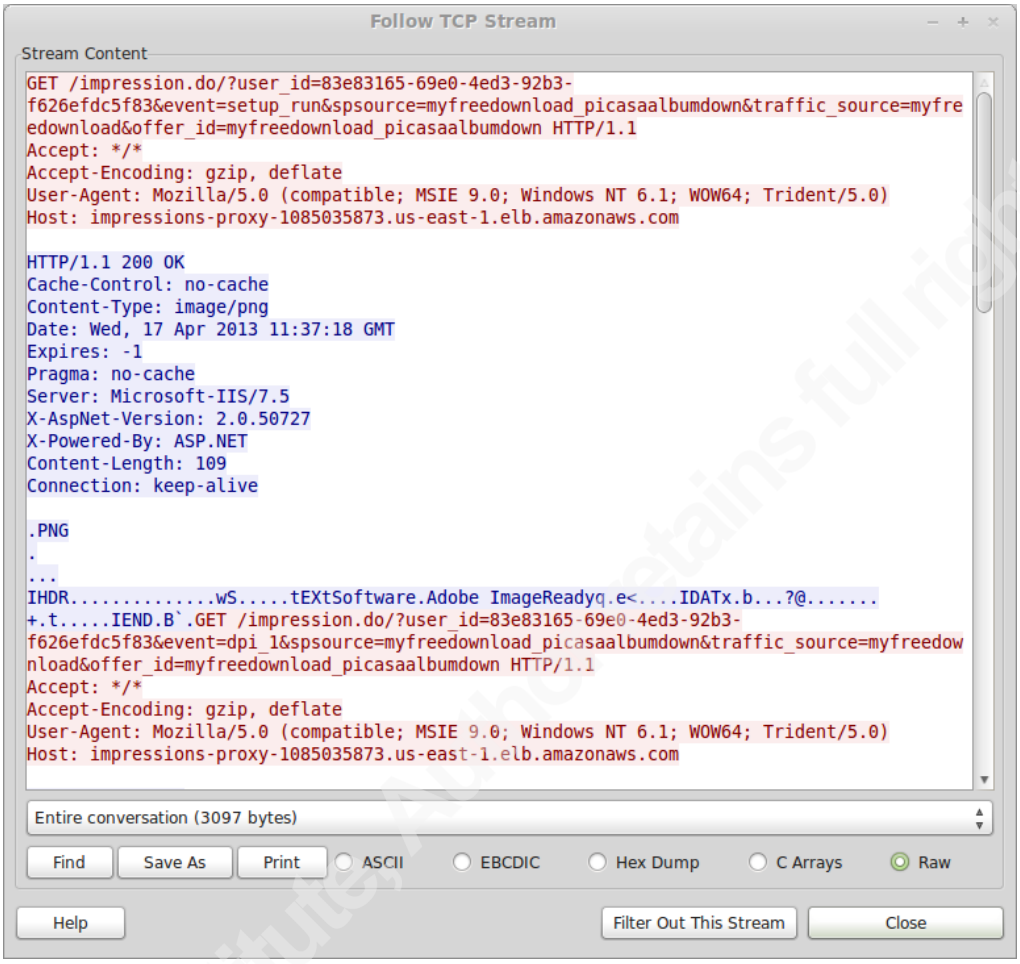


Figure 13: Flow 1's TCP stream from Wireshark

**Flow 2:**

Contacts optimum-installer.com and downloads a JSON file containing the current offers and links to 114 other software packages. The User-Agent used in this request turns out to be the *user_id*, found in Flow1 when downloading the original images and having the *event=setup_run*. This makes sense given they will want to tailor the offers by user.

Kiel Wadner, wadnerk@gmail.com

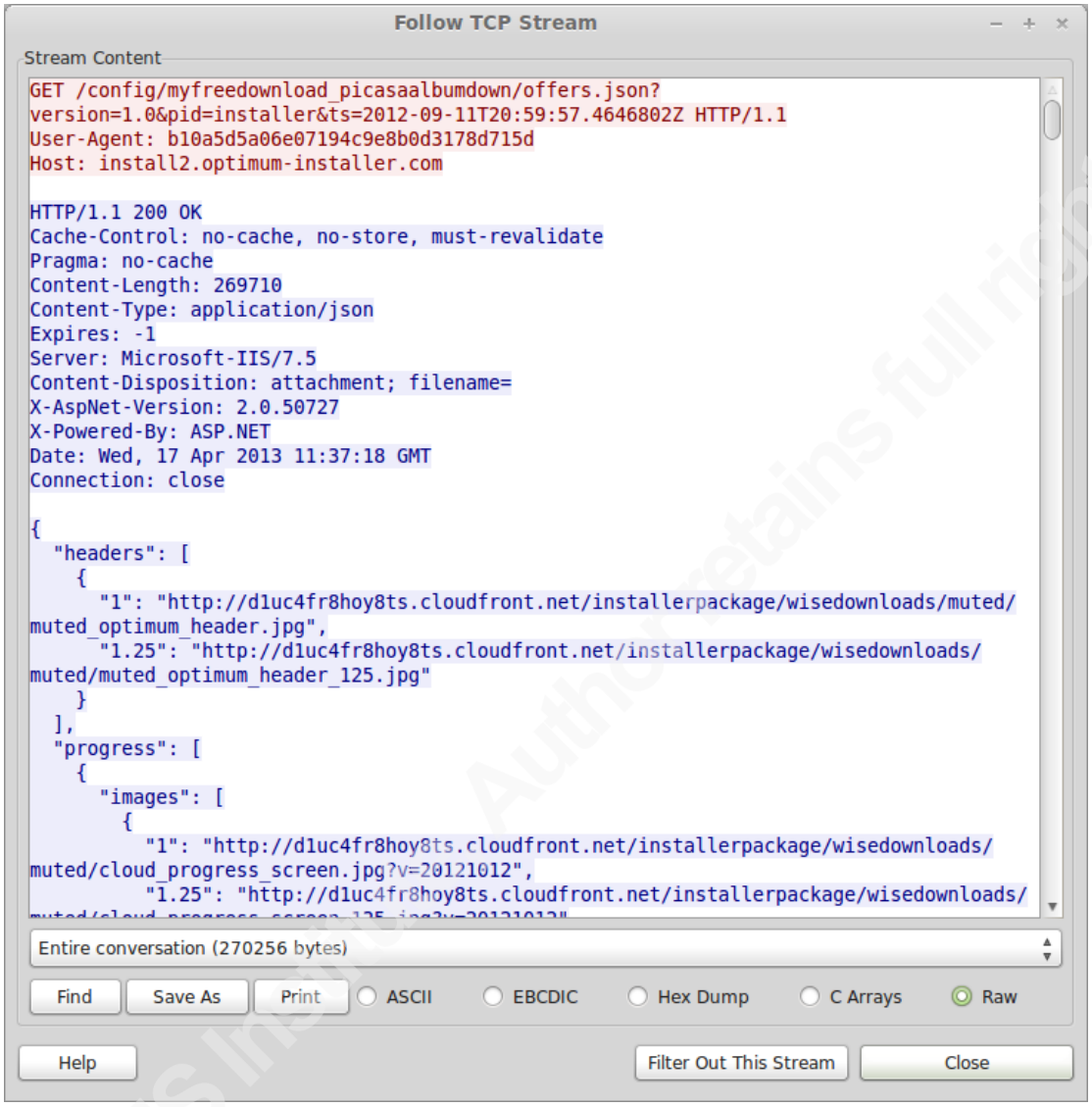


Figure 14: Flow 2's TCP stream from Wireshark

**Flow 3**:

This flow downloads a JPG image for each of the offers found in the above JSON. This contacted another Amazon domain, *cloud-front.net*, which is Amazon's marketing domain. An example of the images is seen to the right. Notice it presents itself as a legitimate installer for a well-known piece of software.

Kiel Wadner, wadnerk@gmail.com

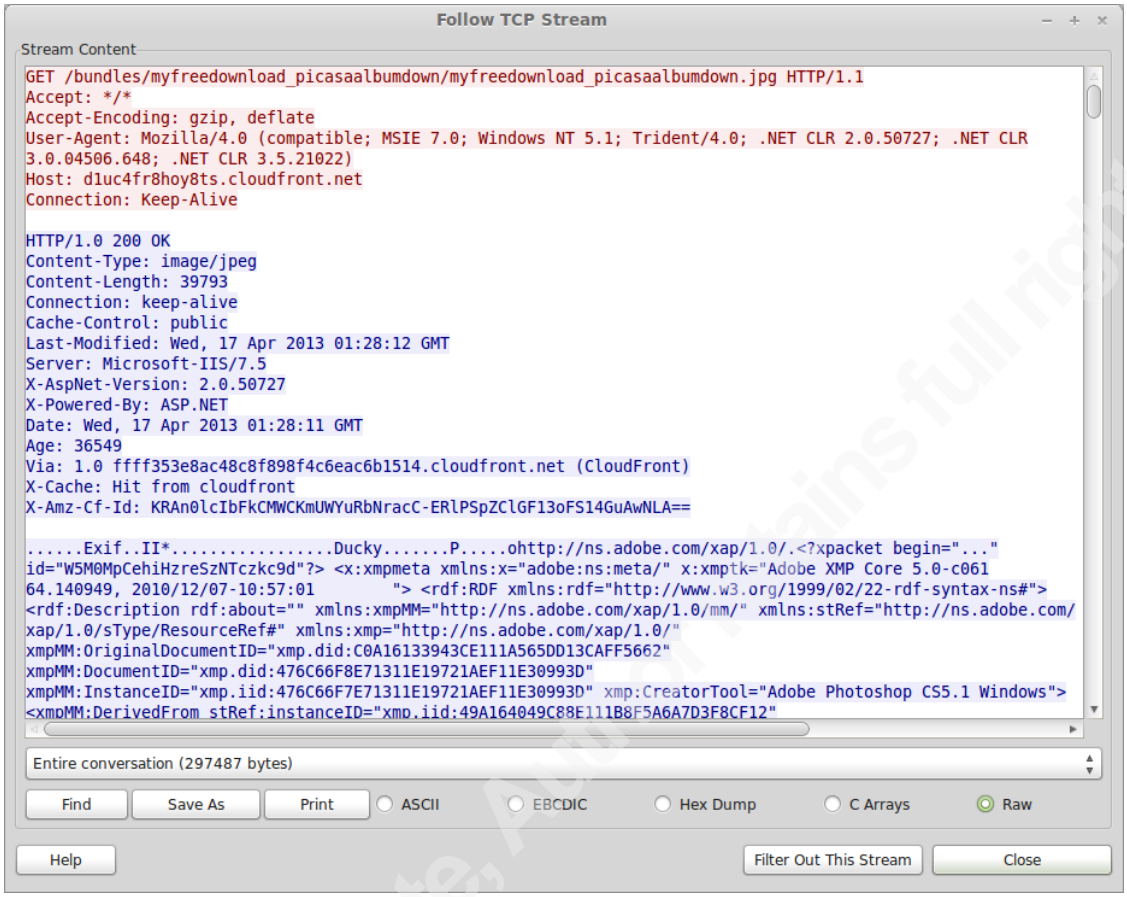


Figure 15: Flow 3's TCP stream from Wireshark

**Flow 4:**

Downloads a .NET executable from d2m.adk-mobile.com. The saved file (MD5: 5aff6686906f00cf8b530c45d675883f) is identified by 14 of 45 antivirus engines as of July 28, 2013 on VirusTotal. No vendors gave it a well-known name, but describe it as a BackDoor or Trojan. At this point, the sample has gone from annoying adware to a trojan. When the URI that download the trojan was searched for 180 unique files were found in the sample set.

What is interesting is of the 180 samples discovered via the D2M-Precheck.exe URI, only the 32 originally looked at talked to the original Amazon AWS server! The remaining samples made the initial call for campaigns and ultimately the trojan through different servers.

This very specific example is provided so that readers can understand the intelligence possible to gather via HTTP requests. When responding to an incident and

reversing a sample looking through web logs of known malware can help identify other samples or infected hosts.

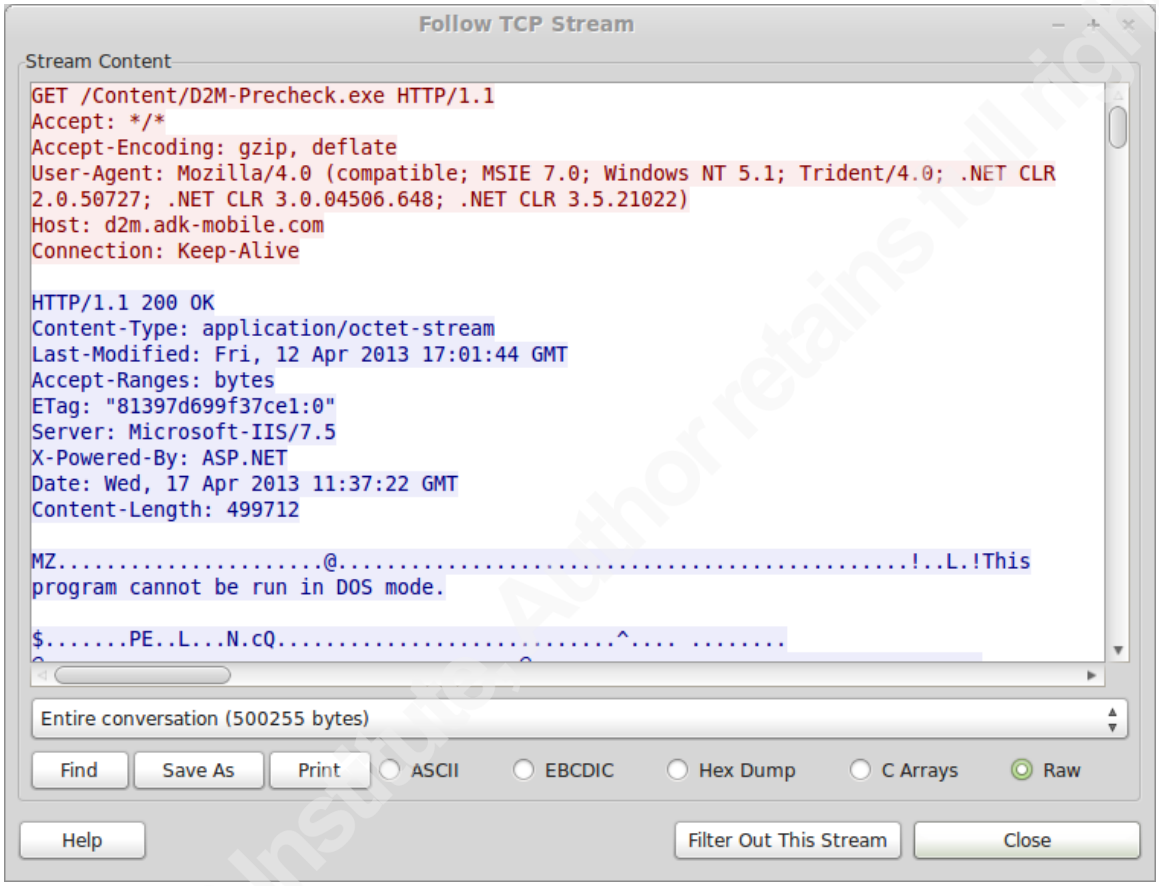

Figure 16: Flow 4's TCP stream from Wireshark

## 6. Early-warning signs of potentially malicious traffic

By capturing and trending User-Agents in your environment anomalies will be easier to spot. Seventy-three percent of malware using HTTP (36% of all samples) used improper User-Agents that could not exist in the environment. Many were visibly odd and can serve as an indicator something needs further investigations. By knowing your environment and what machines should exist on different subnets this process can be further refined. For example, seeing browsers for Mac's, Linux or non-supported browsers are an immediate red flag.

Many destination ports become associated with different types of malware. When responding to an incident, if you have traffic logs, review the ports used in the

compromise time frame and compare to published lists. This technique may not reveal a lot of malware, but it easy to check and can provide simple clues. Trending these for your entire network, or subnets can help identify anomalies as well.

Not many samples used TLS but it was in use. Many of the certificates created had missing values or clearly bogus values. Identifying these early can add to other indicators that something is wrong. The deep-dive on advertising networks highlights a case were potentially unwanted software can escalate to malicious payloads. While not practical for everyone, filtering out calls to advertising networks can limit a mode of distribution for malware.

A pattern should have emerged around the suggestions. Record, automate and look for anomalies. As much as malware tries to blend there are actions that are out of place or not quite right. Nothing is a mythical "silver bullet" so multiple techniques can be used to catch clues. Remember the classic sage advice of layered defenses, and to add protections around the "crown jewels" of the network.

## 7. Conclusions

The research for this paper started with the question "*What can be discovered in 60 seconds of watching malicious traffic?*" What seemed like a simple question to answer at the beginning turned into a tangled nest of additional questions. Some have been answered in this paper, but many are left for future work. Several suggestions were also made from the observations throughout for detecting potentially malicious traffic easily and promptly. In all 20,587 malicious samples were executed for up to 60 seconds. Forty-nine percent of samples utilized HTTP traffic so this ended up the focus of the research. A total of 106,418 DNS Query Requests were made to 25,464 unique hosts. Looking at the hosts and the malware types indicate several hosts were used as points of contact for different malware families.

The use of Suricata with the Emerging Threats Pro rule set proved to be fairly successful in detecting malicious traffic. When a minimum of 3 packets was required to be sent and received the detection rate was 81% of samples. While room for

improvement exists, it demonstrates that network detection of malware traffic can be an effective layer of defenses. Invalid User-Agents were one of the highest alert types in Suricata, and when the controlled environment is considered this was even higher. While TLS certificates did not reveal much it did show that malware using TLS often has certificate information that is suspect. Further research would be required to determine if legitimate software often uses poorly formed certificates.

Future work is expected in several areas of the research. First, the *Malware Tracks* framework will be revamped as a web application to expand the usefulness to others. This will hopefully allow other researchers to utilize the tools in their own lab. Second, the author plans to further investigate the use of public cloud resources in malware networks. Last the sequence and timing of events will be investigated to determine what specific actions are taken, and in what order.

Kiel Wadner, wadnerk@gmail.com

## 8. References

Adperio. (2013). *Adperio*. Retrieved May 6, 2013, from http://www.adperio.com/about.cfm

Alien Vault Labs (Nov 5, 2012) *Your malware shall not fool us with those anti analysis tricks*. Retrieved Aug 5, 2013 from http://labs.alienvault.com/labs/index.php/2012/your-malware-shall-not-fool-us-with-those-anti-analysis-tricks/

Aquilina, J. M., Casey, E., & Malin, C. H. (2008). Malware Incident Response. *Malware forensics investigating and analyzing malicious code* (p. 29). Burlington, MA: Syngress Publishing

Avira. (2013). *Avira Antivirus Software for home and for business*. Retrieved July 23, 2013, from http://www.avira.com/en/avira-free-antivirus

Bailey, M., Oberheide, J., Andersen, J., Mao, Z. M., Jahanian, F., & Nazario, J. (2007, January). Automated classification and analysis of internet malware. In *Recent Advances in Intrusion Detection* (pp. 178-197). Springer Berlin Heidelberg.

Bruneau, Guy (2010, August 7). *DNS Sinkhole*. Retrieved from www.sans.org/reading_room/whitepapers/dns/dns-sinkhole_33523

Contagio (2013) *contagio malware exchange*. Retrieved from http://contagioexchange.blogspot.com/

Dierks T., Rescorla, E. (August 2008) *The Transport Layer Security (TLS) Protocol*. Retrieved May 8 2013 from tools.ietf.org/html/rfc5246

DMG - DSNR Media Group (2013) *About DMG*. Retrieved from http://dsnrmg.com/about

Hurricane Electric (2013) *Hurricane Electric IPv6*. Retrieved from http://ipv6.he.net/

Jennings, C., Lowekamp, B., Rescorla, E., Schulzrinne H. (Feb 24, 2013). *REsource LOcation And Discovery (RELOAD) Base Protocol*. Retrieved August 4, 2013, from tools.ietf.org/html/draft-ietf-p2psip-base-26

Kiel Wadner, wadnerk@gmail.com

Kaspersky Lab US (2013). *Kaspersky Lab US | Antivirus & Internet Security Protection Software*. Retrieved from http://usa.kaspersky.com/

Kaspersky Lab US (2012). *Kaspersky Security Bulletin 2012*. *The overall statistics for 2012*. Retrieved July 23, 2013, from http://www.securelist.com/en/analysis/204792255/

Matomy. (2013). *About Matomy Group*. Retrieved July 23, 2013, from http://www.matomy.com/about-matomy-group

McNamee, Kevin. (2012a) *Botnet: ZeroAccess/ Sirefef*. Retrieved Feb. 12, 2012 from www.kindsight.net/sites/default/files/Kindsight_Malware_Analysis-ZeroAcess-Botnet-final.pdf

McNamee, Kevin. (2012b) *New C&C Protocol for ZeroAccess/ Sirefef*. Retrieved Feb 13, 2013 from www.kindsight.net/sites/default/files/Kindsight_Malware_Analysis-New_CC_protocol_ZeroAccess-final2.pdf

Microsoft. (July 2, 2010) *The NetBIOS Interface*. Retrieved August 4, 2013 from http://msdn.microsoft.com/en-us/library/bb870913(v=vs.85).aspx

Microsoft. (2013a) *Win32/Ramnit*. Retrieved July 23, 2013 from http://www.microsoft.com/security/portal/Threat/Encyclopedia/Entry.aspx?Name=Win32/Ramnit

Microsoft. (2013b) *Win32/Virut*. Retrieved July 23, 2013 from http://www.microsoft.com/security/portal/threat/encyclopedia/entry.aspx?Name=Win32%2fVirut#tab=1

Norman Shark (2013). *Malware Analysis (MAG2)*. Retrieved July 27, 2013 from http://normanshark.com/products-solutions/products/malware-analysis-mag2/

Perdisci, R., Lee, W., & Feamster, N. (2010, April). Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces. In *NSDI* (pp. 391-404).

Kiel Wadner, wadnerk@gmail.com

Rossow, C., Dietrich, C., & Bos, H. (2012). Large-scale analysis of malware downloaders. In *Detection of Intrusions and Malware, and Vulnerability Assessment* (pp. 42-61). Springer Berlin Heidelberg.

Salusky, W., & Danford, R. (2007, July 13). Know Your Enemy: Fast-Flux Service Networks. *Blogs | The Honeynet Project*. Retrieved July 20, 2013, from http://www.honeynet.org/book/export/html/130

Scapy. (2013). *About Scapy*. Retrieved August 5, 2013, from http://www.secdev.org/projects/scapy/

Sikorski, M., & Honig, A. (2012). Practical malware analysis: The hands-on guide to dissecting malicious software. San Francisco: No Starch Press.

Sophos (2013) *iBryte Optimum Installer Detailed Analysis*. Retrieved from http://www.sophos.com/en-us/threat-center/threat-analyses/adware-and-puas/iBryte%20Optimum%20Installer/detailed-analysis.aspx

Symantec. (2013) *W32.Sality* Retrieved July 23, 2013 from http://www.symantec.com/security_response/writeup.jsp?docid=2006-011714-3948-99

The Open Information Security Foundation (2013) *The Open Information Security Foundation*. Retrieved July 24, 2013, from http://www.openinfosecfoundation.org/

Ullrich, Johannes. (Dec 1, 2006) *Port 80 UDP Malware*. Retrieved Feb 13, 2013 from https://isc.sans.edu/diary/Port+80+UDP+M

VirusTotal. (2013) *VirusTotal*. Retrieved July 27, 2013 from www.virustotal.com

VirusShare. (2013) *VirusShare*. Retrieved Aug 5, 2013 from http://virusshare.com/

Verizon (May 2013) *Data Breach Investigations Report* Retrieved June 25, 2013 from http://www.verizonenterprise.com/DBIR/2013/sdfs

Kiel Wadner, wadnerk@gmail.com