2012

A Historical evaluation of C&C complexity

Conzetti Finocchiaro

A Historical Evaluation of C&C Complexity

By

Conzetti N. Finocchiaro

Committee Members
Dr. Peter Lutz
Dr. Jim Leone
Dr. Bo Yuan

Thesis submitted in partial fulfillment of the requirements for the degree
of
Master of Science in Networking and System Administration
Rochester Institute of Technology
B. Thomas Golisano College
of
Computing and Information Sciences

August 20, 2012

Rochester Institute of Technology
B. Thomas Golisano College
of
Computing and Information Sciences

Master of Science in
Networking and System Administration

Thesis Approval Form

Student Name: Conzetti Finocchiaro

Thesis Title: A Historical Evaluation of C&C Complexity

Thesis Committee

Name	Signature	Date

Peter Lutz
Chair

Jim Leone
Committee Member

Bo Yuan
Committee Member

Table of Contents

Table of Figures

Abstract

The actions of Malware are often controlled through uniform communications mechanisms, which are regularly changing to evade detection techniques and remain prolific. Though geographically dispersed, malware-infected nodes being controlled for a common purpose can be viewed as a logically joint network, now loosely referred to as a botnet. The evolution of the mechanisms or processes for controlling the networks of malware-infected nodes may be indicative of their sophistication relative to a point of inception or discovery (if inception time is unknown).

A sampling of botnet related malware at different points of inception or discovery can provide accurate representations of the sophistication variance of command and control processes. To accurately measure a sampling, a matrix of sophistication, deemed the Complexity Matrix (CM), was created to categorize the signifying characteristics of Command and Control (C&C) processes amongst a historically-diverse selection of bot binaries.

In this paper, a survey of botnets is conducted to identify C&C characteristics that accurately represent the level of sophistication being implemented within a specified time frame. The results of the survey are collected in a CM and used to generate a subsequent roadmap of C&C milestones.

1. Introduction

Network-based Covert Channels (NCCs) have been used as discrete end-to-end data transfer mediums for botnet communications. C&C operations rely on NCCs to remotely manage and update resources while evading detection mechanisms. An evolution of detection mechanisms directly impacts the evolution of NCCs used for such purposes, and as such, methods of C&C operations have vastly changed over time. The objective of the subsequent work is to identify and quantify the sophistication variance of C&C processes throughout a comparative analysis of multiple botnets created at different points in history. This

analysis will effectively prove or disprove that the sophistication or complexity of C&C methods are increasing over time.

Bots may rely on C&C communications, for the distribution of instructions, for routine updating purposes, and to remain effective and evasive. These communications serve as the lifeline to a *Botmaster*, or the entity managing the botnet, and as such are typically very important to the overall success of a botnet. Anti-malware solutions pose a significant threat to the longevity of C&C communications, so methods of evasion are regularly implemented to enable bots to operate unscathed.

The architecture of a botnet will vastly affect the methods of C&C that are implemented, and in some situations will directly affect the complexity of the C&C processes. A thorough understanding of the major botnet designs is thus a prerequisite to C&C evaluation. Botnets can be defined as either centralized, decentralized, or hybrid in design [22, 32], but only centralized and decentralized architectures will be explored in this research. The designs are briefly defined, below, and are explored in greater depth in section 2.2.

- Centralized: Receives or retrieves all C&C communications from a master node (or nodes) in a master/slave design (Master $\leftrightarrow$ Bot).
- Decentralized: Receives or retrieves some or all C&C communications from a series of peers in a peer-to-peer (P2P) design (Bot $\leftrightarrow$ Bot).

Historically, the centralized architecture has been a widely adopted design for successful botnet implementations. C&C master servers can be viewed as a single point of failure despite the potential to have intermediary nodes relaying communications [22]. In contrast, a decentralized design focuses on C&C methods being delivered through peers (other Bots), and effectively eliminates a central point of failure [3, 13]. A decentralized design must therefore rely upon a mechanism to discover or communicate with other peers, and may be much more difficult in practice to implement. The effectiveness of each design truly depends upon the methods of obscuring or securing the C&C traffic from detection.

In addition to being measured on architectural style, C&C communications will be measured by other distinguishable features. Distinguishable features will be categorized under one of the following high-level capability offerings: high availability and fault tolerance, authentication and non-repudiation, encryption methods, evasion tactics, transport protocols, and application protocols.

A broad analysis of C&C methods, adopted by well-known botnets throughout history, is required to interpret their overall complexity. This work seeks to do just that, by categorizing the characteristics of the C&C method into a complexity matrix (CM) and generating metrics from the collected data. As such, the analysis required two methods of evaluation to create complexity measurements:

(1) A logical deconstruction of C&C processes based upon prior research

(2) A physical evaluation of C&C processes on selected Bots, where prior research is missing or inadequate

The work herein produces a roadmap of measurable advancements in C&C methods through the evaluation of several botnet variants. The roadmap and supporting documentation will explicitly prove sophistication advancement as time progresses. Ideally, the outcomes of this project will enable a researcher to quickly and easily identify when a C&C method surfaced, relative to the malware that used it. In addition to being time-bound, the maturity of a specific C&C method will also be measurable.

2. Related Work &Background

In a sampling of prior research, major attention was paid towards innovative techniques for detecting C&C traffic or, alternatively, performing static analysis on particular C&C technologies to understand their functionality and potential applications. Functional advancements and varying levels of sophistication are documented in such efforts, which help identify a period of performance for a particular C&C method.

2.1 Related Work

Lashkari et al. introduced a broad overview of various prominent botnets [4], but the work lacks specifics on the capabilities and technologies leveraged by the botnet authors. The work presented in [4] is also lacking tangible evidence to prove or disprove advancements in C&C techniques. Norman introduced a similar approach at wide-scale botnet evaluation, but purely tailored his work towards the logical deconstruction and evaluation of P2P botnets [3]. Norman does, however, present a very detailed narrative on the history and direction of botnets in general.

This work improves upon a broad botnet survey and hones in on specifics of the C&C methods employed by Botmasters. To supplement the work performed in this survey, two other categories of prior research were leveraged for their specific vantage point on botnet functionality: botnet detection and botnet evaluation.

2.1.1 Botnet Detection

Botnets may elicit specific characteristics that can be used for detection purposes. Various detection methods are presented in prior research [5, 6, 24, 27, 29, 30, 32, 40], either as novel approaches or as sources of comparison for already available detection methods. This research methodology does not typically focus on detecting a single botnet family, but more generally provides a plausible basis for the detection of multiple botnets. Such research inevitably reveals many specifics of botnet C&C methods indirectly.

2.1.2 Botnet Evaluation

Evaluation tends to focus on the dissection of botnet families in an effort to understand the underlying technologies utilized by Botmasters. C&C methods are evaluated based upon specific protocol characteristics and design features in the chosen work [2-4, 7, 9-13, 21-23, 26, 28, 33, 37, 38, 39, 43, 44]. Such approaches directly reveal specifics of C&C methods.

2.2 Botnet C&C Architecture

Each botnet selected for evaluation in this research receives or retrieves C&C communications in either a centralized or decentralized fashion. Strictly speaking, a centralized architecture requires C&C communications to propagate from centralized masters (C&C server) to slaves (Bots). A decentralized architecture would thus purely rely upon peers (Bots) to receive and retrieve C&C communications. There are, however, designs that go beyond the traditional understanding of centralized or decentralized. Thus, we have several designs presented in the sampling of botnets chosen in this research.

Modern malware may introduce the separation of duties amongst different nodes in a hierarchical fashion. This hierarchical format exists in both Centralized and Decentralized architectures, and the naming conventions used to identify those roles are listed below and used throughout the document.

- Subnodes: The lowest tier of a hierarchical design; the bot or *zombie* computers
- Supernodes: The next intermediary tier of nodes after subnodes; bots equipped with more functionality or responsibilities
- Subcontrollers: The next intermediary tier of nodes after supernodes; services deployed to managed systems to hide C&C servers and perform basic C&C functions for lower layers
- C&C Operations Center: The highest tier of a hierarchical design; services deployed to manage and delegate responsibilities to the lower tiers, directly operated by a Botmaster

2.2.1 Centralized Architecture

A centralized architecture follows a master/slave topology, where a central authority (master) is responsible for the actions of its slaves. A legacy implementation (Fig. 1) of the centralized architecture follows a very simple one-to-many relationship. However, the addition of intermediary nodes to perform selective C&C functions represents a delegation of responsibilities and increases the difficulty in identifying the central authority for a botnet. The method of introducing layers of intermediary C&C nodes to create a more scalable, robust botnet is identified as the next-generation centralized design (Figure 2).

Legacy Centralized Design

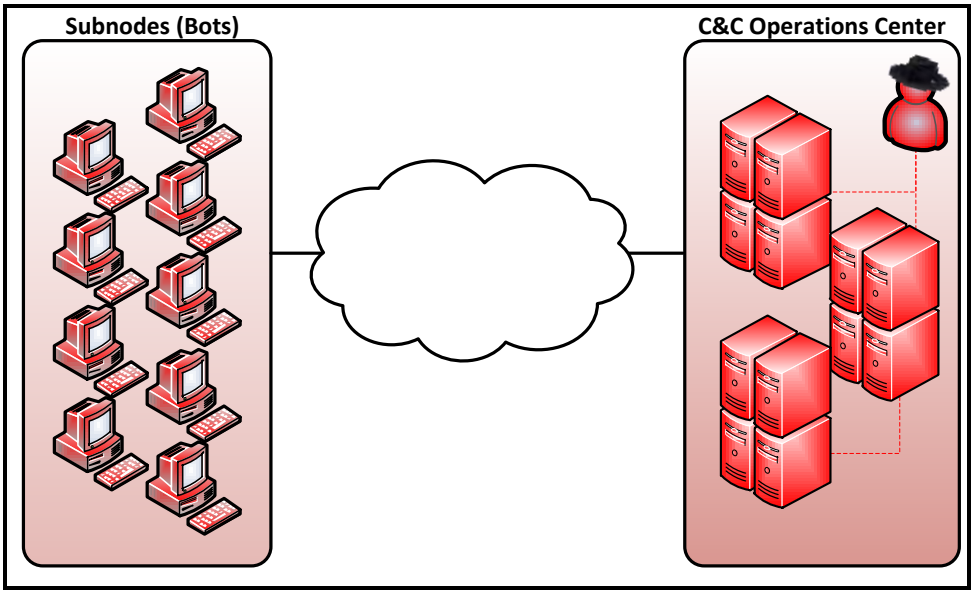

Figure 1: A Legacy Centralized Design with co-located C&C servers & a single Botmaster

Next-generation Design

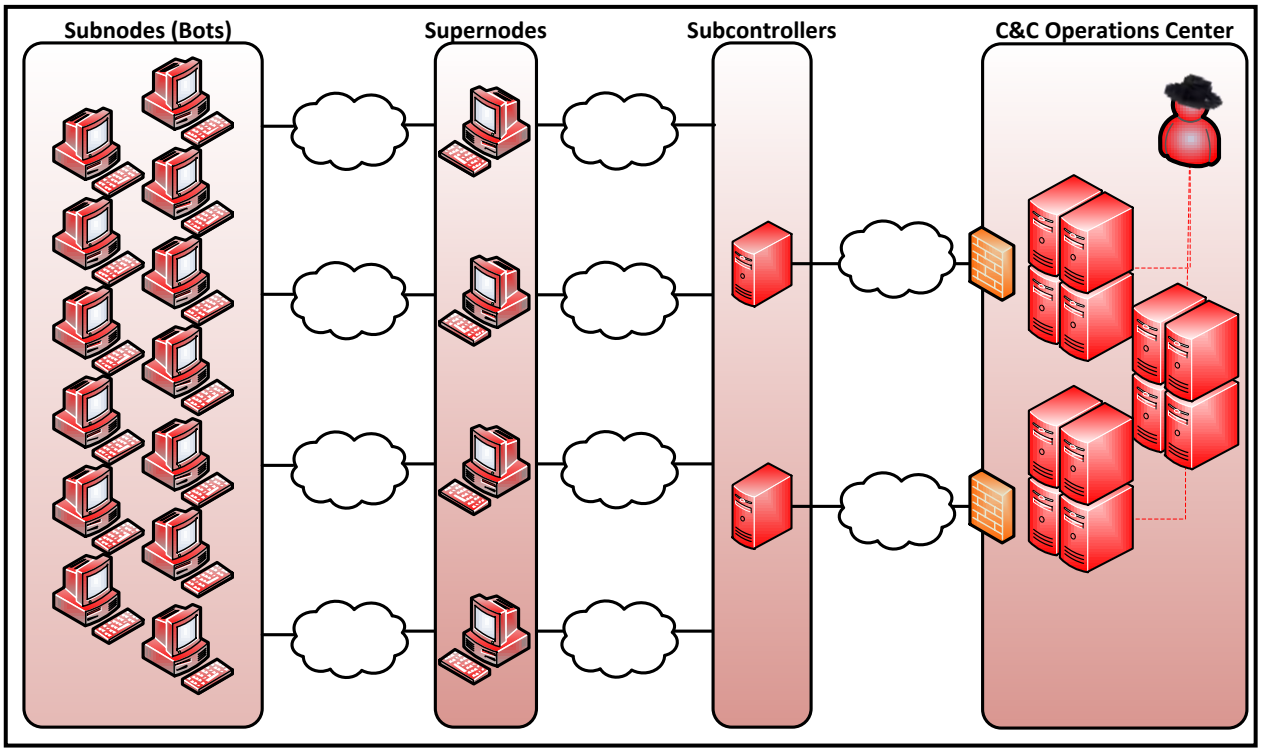

Figure 2: A Next-generation Centralized Design utilizing Supernodes and Subcontrollers for hierarchical C&C

2.2.2 Decentralized Architecture

A decentralized architecture follows a P2P topology, where no central authority is directly responsible for the actions of a peer. A basic implementation of the decentralized architecture follows a simple many-to-many relationship, where participating bots may have multiple connections to other participating bots at any given point in time. A full P2P (Fig. 3) design purely relies on its peers (Bots) for C&C instruction dissemination, but enables a Botmaster to distribute C&C instructions to his entire botnet through any given Bot. A hierarchical P2P design (Fig. 4) utilizes P2P communications for the use of specific C&C purposes, and introduces layers of hierarchical control points for extended C&C distribution and more robust functionality.

Full P2P Design

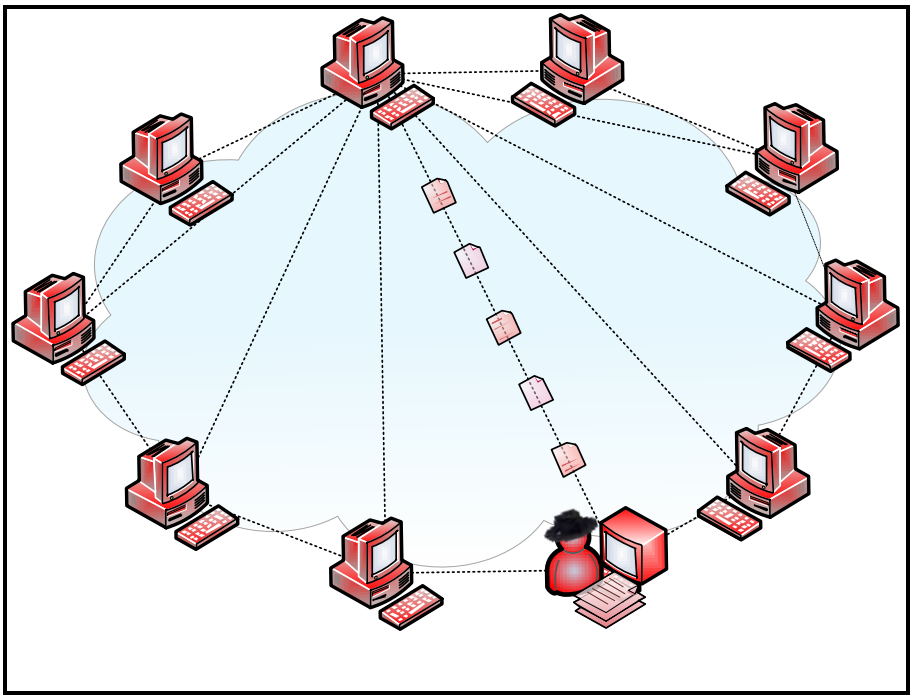

Figure 3: A Full P2P Decentralized Design, utilizing partial mesh P2P relationships & a single Botmaster

Hierarchical P2P Design

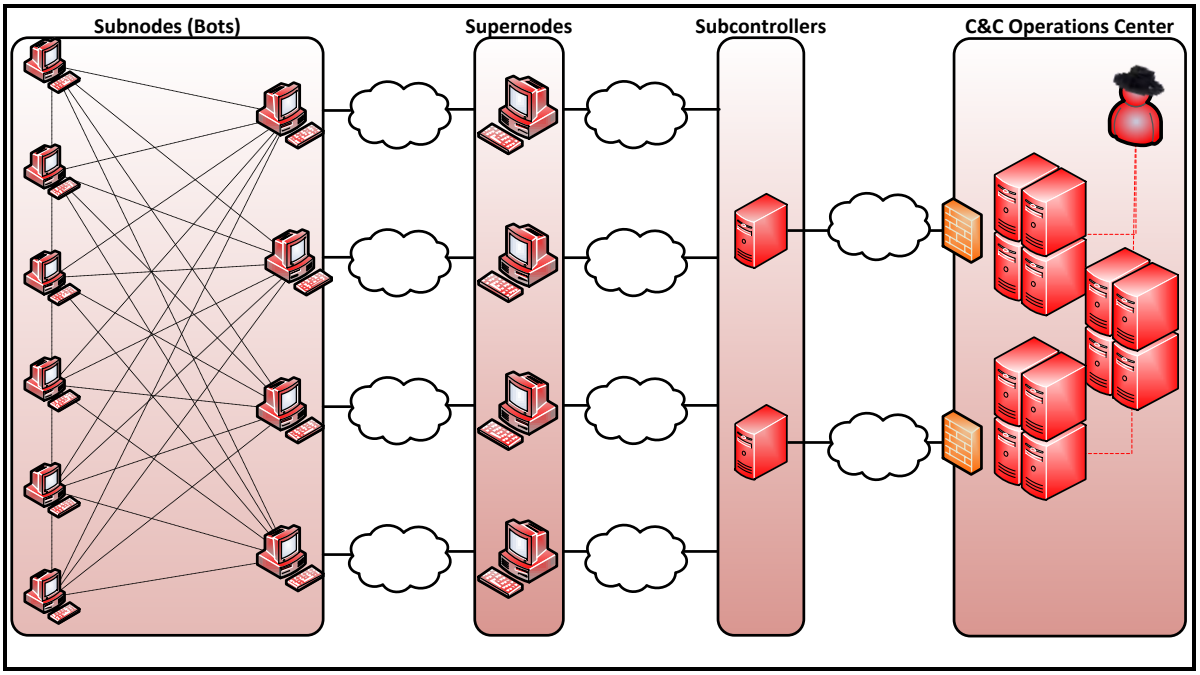

Figure 4: A Hierarchical P2P Decentralized Design, using partial mesh P2P relationships between Subnodes and utilizing Supernodes and Subcontrollers for hierarchical C&C

2.3 Botnet Selections & Overview

The growing number of botnet families presents a challenge when identifying an accurate subset of botnets that are representative of a given era. Thus, the botnets selected for evaluation in this project were chosen based upon the significance of one or more of the following properties:

(1) The proliferation or wide-scale effect on the Internet (and thus the users of)

(2) The precedence set by, or the novel use of, a C&C technique

(3) The period of performance of a specific botnet

By limiting the selection of botnet examples in this fashion, a manageable scope for botnet evaluation is created. However, the limited scope does not hinder the ability to accurately represent historical C&C sophistication variance.

To satisfy the requirement to represent a full spectrum of botnet C&C advancements, fourteen botnet examples were chosen after evaluating their respective C&C methods against the selection criteria. The following subsections will provide introductory details on the selected botnets.

2.3.1 Agobot

Agobot is a name given to a large family of botnet variants that were identified by Sophos [48] in October of 2002. The Agobot malware source code was released to the general public under the GNU General Public License, version 2 (GPL v2), spawning many subsequent botnets. Based on this open distribution, Agobot also may collectively refer to Gaobot, which Symantec first reported on January 13, 2004 [51].

2.3.2 Asprox

The Asprox botnet is comprised of *spambots*, used for the sole purpose of sending unsolicited email [42] and phishing attacks. The botnet was discovered and categorized as a Trojan by Symantec on June 8, 2007 [49].

2.3.3 Bagle

The Bagle botnet is comprised of spambots, used for the sole purpose of sending unsolicited email. According to M86 Security, Bagle surfaced in early 2004 [15, 16]. It is characterized by its ability to act as a proxy server for the relay of spam to its final destination, but can ultimately serve other purposes as required.

2.3.4 Bobax

Bobax is malware used primarily for spamming purposes. According to F-Secure [45], the botnet was discovered on May 16, 2004. Much like other spamming botnets, it is characterized by its ability to act as a proxy server for the relay of spam to its final destination.

2.3.5 Conficker

Conficker is a worm that exploits a vulnerability in Microsoft Windows *Server* service and provides several methods of propagation to other vulnerable hosts. The overall objective of the worm is not clearly understood, beyond the act of infecting other nodes. Microsoft reports that the first variant of Conficker was reported on November 21, 2008 [25].

2.3.6 iKee

iKee is a worm that propagates to vulnerable Apple iOS devices purportedly for the sake of phishing and information mining [11]. The malware was discovered on November 19, 2009 [11].

2.3.7 GTbot

GTbot, or Global Threat Bot, is the name given to a wide variety of malware with similar IRC-based botnet characteristics. Prior research by Canavan documents the introduction of GTbot variants in late 2000 [44].

2.3.8 Nugache

Nugache is malware utilized mainly for information mining [53], but has the capacity to be used for other nefarious purposes. Early variants of Nugache were discovered on April 30, 2006 [53].

2.3.9 Peacomm

Peacomm, otherwise known as *Storm*, is a multi-stage infection that is largely used for the generation of spam and DDoS attacks. Symantec identified the malware on January 19, 2007 [50].

2.3.10 Phatbot

Phatbot is an Agobot variant [51] that was detected by Symantec on November 21, 2003 as W32.HLLW.Gaobot.gen [52]. Phatbot inherits the capabilities of earlier variants, but is equipped with a larger exploit toolkit for compromising vulnerable machines.

2.3.11 SDbot

SDbot is remote control and administration malware used for information mining, DDoS attacks and infecting other nodes to perform similar functions. SDbot's existence became known by Symantec on April 30, 2002 [17].

2.3.12 Sinit

Sinit is remote control and administration malware purportedly used for the distribution of other malware [46]. Symantec first discovered the threat on October 9, 2003 [18].

2.3.13 Waledac

Waledac is multifunctional piece of malware that spreads and infects using various different attack vectors. The malware is utilized mainly for generation of spam, emerging in November 2008 [6, 43, 54].

2.3.14 Zeus

Zeus is the name given to a large *crimeware* suite used to create and control customized bot binaries. The name has become synonymous with the bots generated from the various versions of the Zeus suite, but this is not an accurate identifier for any single binary. The creation time of the crimeware is not known with absolute precision, but the kit is readily used as of this writing (August, 2012).

3. Evaluation Methodology

A sampling of botnet variants is evaluated for the existence of specific features to enable stealthy communications in their respective C&C topology. The analysis begins with a literature review of research performed on all of the identified botnet variants and completed with manual network traffic analysis as returned from a machine infected with a bot binary.

The manual analysis is utilized to enforce prior research and support statements made in this research. In the event that a bot binary was unattainable, the generated CM identifies this and utilizes the details of prior research.

3.1 Manual C&C Evaluation Tools & Processes

3.1.1 Infection Point

All malware was installed and executed on an unpatched version of the Microsoft Windows XP operating system (OS). A physical system was used to host the OS to avoid the possibility of the bot binary behaving differently if operated in a virtual environment. The only modification to the guest operating system was the installation of network and video device drivers.

A dedicated residential broadband connection was utilized as the connection medium for the infection point. A publicly routable IP address was assigned to the machine via DHCP.

3.1.2 Imaging

The Ghost 4 Linux (G4L) project was leveraged for the bare metal backup and restore of the system utilized as the infection point. The physical host was configured to boot using the Intel PXE protocol and download the unadulterated XP image using TFTP; these processes were executed for every malware review cycle.

3.1.3 Packet Analysis

Packet analysis of the chosen malware was performed using Tcpdump and Wireshark from a host running Debian Linux 6.0. The host shared a physical hub with the infection point, but was forced into promiscuous mode for raw network access to the collision domain.

3.1.4 Evaluation Criteria

Due to the number of malware variants selected for this research, limitations were placed on the evaluation to only relevant C&C details. The evaluation was limited in time and scope.

Each botnet evaluation lasted 4 hours, limiting the total time exhausted to a maximum of 56 hours. The inability to directly evaluate two of the botnet variants further reduced this value to 48 hours. This time does not include the time spent configuring the lab environment or reviewing the packet traces.

Several of the selected botnets implement a form of transport encryption or file-based encryption, but were not reverse engineered as part of this research. This restricts the evaluation to only glean the information that is human readable by means of direct packet analysis.

3.2 Lab Topology

Fig. 5 is a representation of the different topologies used in this work. The lab environment was separated for three different processes for the physical evaluation:

- **Imaging**: The bare-metal XP installation
- **Discovery**: Finding suitable malware for evaluation and performing signature-based identification
- **Infection**: Installing the malware on the infection point

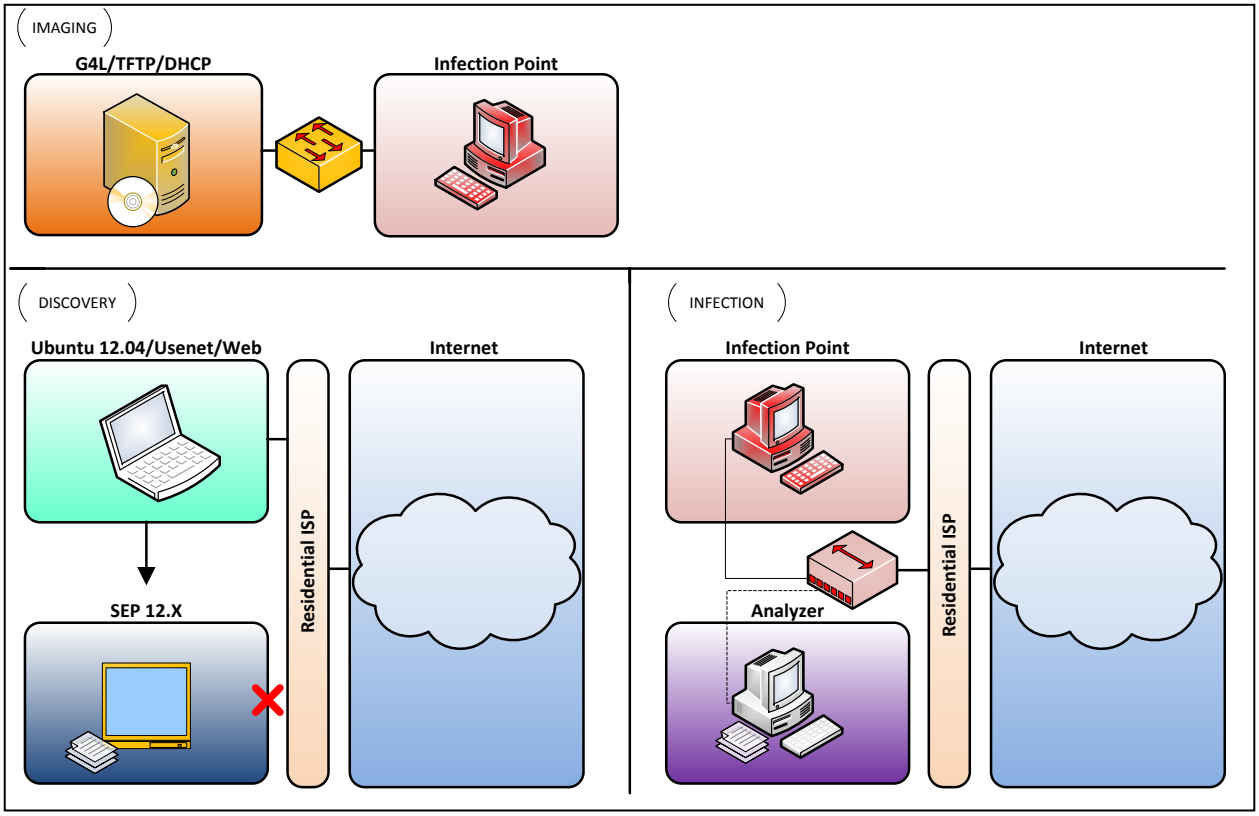

Figure 5: The three separate high-level processes and their respective topologies

3.3 Botnet C&C Evaluation

After conducting thorough research on fourteen botnet families, 1312 examples of botnet-related malware were sourced using Usenet and a variety of sites hosted on the .box.sk network. All malware samples were downloaded using a live distribution of Ubuntu Linux (12.04) and transported to a Windows 7 x64 virtual machine utilizing Symantec Endpoint Protection (SEP) for signature-based identification of the malware. The actual IP addresses of the externally hosted C&C nodes or botnet peers are not recorded as

part of the evaluation, as the research focuses purely on methods of C&C communications. The first three octets of the IP address associated with the infection point and the destination C&C master/peer are replaced with x's in the tabular data.

The observations of the malware are supplemented with references to prior research. The behaviors associated with a specific bot are verified through the research of others. In the case of the iKee iOS botnet, a working example could not be tested due to resource constraints, and thus the evaluation purely relies on the in depth research of iKee.B conducted by Porras et al. [11]. Additionally, only a number of the bots were capable of establishing C&C relationships with remote hosts presumably due to the removal or abandonment of key elements of their infrastructure.

Both the expected behaviors of the malware, dissected from prior research, and the actual observations of the malware C&C communications are generalized below. The CM presented in section 3.4 contains a complete list of features that identify C&C functionality for the chosen malware.

3.3.1 Agobot

Expected Behavior

Agobot implements a Legacy Centralized Design for C&C communications, based upon the descriptions of the malware in [12, 44]. Agobot variants attempt to utilize Internet Relay Chat (IRC) as the common method of C&C communications, as confirmed in [7, 27, 48] without implementing encryption or obfuscation of data at rest or data in motion. Agobot is expected to implement a custom syntax for the execution of actions as directed by a Botmaster [8].

Observed Behavior

The Agobot selection was identified as W32.Gaobot.BUU utilizing SEP. This Agobot variant attempted to utilize IRC as the common method of C&C communications over TCP/6667 (Table 1), but failed to complete a simple TCP three-way handshake to the remote host. After random intervals of time, the bot would attempt to connect to the same remote host, but the three-way handshake never completed.

Symptoms suggest that the downloaded binary is hardcoded with a C&C server IP address, and does not attempt to implement any forms of high availability.

No.	Time	Source	Prot.	Destination	Len.	Info
10	11.137147	*x.x.x.66*	TCP	*X.X.X.54*	58	54906 > 6667 [SYN] Seq=0

Table 1: Infected node attempting to initiate a connection on TCP/6667.

3.3.2 Asprox

Expected Behavior

Asprox utilizes the HTTP protocol for Next-generation Centralized C&C communications, without the advent of transport layer encryption. It does, however, retrieve C&C instructions in an encrypted template file which must be decrypted and interpreted by the recipient bot [10]. Outbound C&C communications are expected to occur on either TCP/80 or TCP/82 [15, 49]. Based upon the research by M86 Security Labs [42], Asprox bots are distributed with a list of domain names for initial C&C bootstrapping. The initial check-in process occurs with a HTTP/1.1 POST request to a remote host, followed by a HTTP/1.1 GET request to download an encrypted file with a .BIN file extension [10, 46]. The authoritative response for the given name lookup during the initial connection requests will return one of a series of IP addresses due to the use of double-flux DNS [42], which also indicates that responsible C&C servers are well protected from plain view.

Observed Behavior

The Asprox example was identified as Trojan.Asprox utilizing SEP. Seconds after installation, TCP SYN probes are issued against several well-known search engines (Table.2), which appear to be the connectivity checks identified in [15, 16, 46]. Within 60 seconds of running for the first time, the bot performed a name lookup for a seemingly random FQDN (Table.3).

No.	Time	Source	Prot.	Destination	Len.	Info
14	2.077903	*x.x.x.66*	TCP	74.125.228.97	58	43756 > http [SYN] Seq=0 Win=1024 Len=0 MSS=1460
15	2.082287	74.125.228.97	TCP	*x.x.x.66*	58	http > 43756 [SYN, ACK] Seq=0 Ack=1 Win=14300 Len=0 MSS=1430
16	4.212670	*x.x.x.66*	TCP	98.138.253.109	58	50799 > http [SYN] Seq=0 Win=1024 Len=0 MSS=1460
17	5.047804	98.138.253.109	TCP	*x.x.x.66*	58	http > 50799 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460

Table 2: TCP SYN scans initiated against Google.com and Yahoo.com.

No.	Time	Source	Prot.	Destination	Len.	Info
31	58.183426	*x.x.x.66*	DNS	*x.x.x.81*	83	Standard query 0x0004 A *xxxxx.ru*
32	59.656053	*x.x.x.81*	DNS	*x.x.x.66*	119	Standard query response 0x0004 A *x.x.x.19*

Table 3: Name lookup for C&C resource.

After the IP address of the remote host was resolved, the machine contacted the remote host utilizing TCP/80 and issued a standard HTTP/1.1 POST request for a file with a .PHP extension (Table 4). This behavior mimics the behavior identified in [10]. After a period of approximately thirty minutes, the malware requested the download of another file with a .PHP extension using a HTTP/1.1 GET request (Table 5). The extension of the retrieved file differed from the expected behavior. However, the contents of the file were still encrypted. Outbound traffic on TCP/25 became evident shortly after the download of the encrypted .PHP file, indicating that the bot was now equipped with a spamming template and instructions to perform the process of spamming.

No.	Time	Source	Prot.	Destination	Len.	Info
41	78.183426	*x.x.x.66*	HTTP	*X.X.X.121*	517	POST /ckl/stanje.php HTTP/1.1

Table 4: HTTP/1.1 POST check-in.

No.	Time	Source	Prot.	Destination	Len.	Info
1437	1837.783528	*x.x.x.66*	HTTP	*X.X.X.97*	540	GET /ckl/bar.php HTTP/1.1

Table 5: HTTP/1.1 GET request.

3.3.3 Bagle

Expected Behavior

Much like Asprox, Bagle utilizes the HTTP protocol for C&C communications, without the advent of transport layer encryption [19]. Initial check-in requests are performed against a series of hard-coded URLs with a HTTP/1.1 GET request on TCP/80 against a file with a .PHP file extension [1]. The malware then begins listening for connections on a series of TCP ports, as identified in [20]. Bagle does, however, retrieve C&C instructions in an encrypted file, which must be decrypted and interpreted by the recipient bot [19]. Unlike Asprox, it uses a Legacy Centralized Design for C&C communications.

Observed Behavior

The selected variant of Bagle is identified as W32.Beagle.AV@mm by SEP. The infected machine began performing name lookups on a series of FQDNs immediately after installation (Table 6). This name lookup behavior mimicked the expected behavior identified in prior research [1,19]. Many of the URLs were likely offline, as a connection wasn't built to a remote host until several lookups were performed. The machine fetched a .PHP file utilizing a standard HTTP/1.1 GET request (Table 7). The same standard procedure is purportedly performed when downloading an updated version of the bot binary [19], but was not witnessed in testing.

No.	Time	Source	Prot.	Destination	Len.	Info
22	8.691871	*x.x.x.66*	DNS	*x.x.x.81*	78	Standard query 0x0004 A *xxxxxxx.com*
24	9.141238	*x.x.x.81*	DNS	*x.x.x.66*	132	Standard query response 0xec47 No such name
27	9.470919	*x.x.x.66*	DNS	*x.x.x.81*	83	Standard query 0x0004 A *xxxxxx.in.cc*
28	9.480868	*x.x.x.81*	DNS	*x.x.x.66*	139	Standard query response 0x0004 A *x.x.x.122*

Table 6: A snippet of name lookup failures/successes.

No.	Time	Source	Prot.	Destination	Len.	Info
107	25.632279	*x.x.x.66*	HTTP	*X.X.X.230*	437	GET /brg.php HTTP/1.1

Table 7: HTTP/1.1 GET request (similar to Asprox).

Similar to Asprox, the infected node began opening outbound connections on TCP/25 in a typical fashion to spam. However, no further HTTP connections were discovered during the evaluation. The infected machine did, however, acknowledge TCP SYN scans from remote hosts that targeted TCP/33112, indicating that the malware began listening on TCP/33112 (Table 8). This was confirmed on the infection point by using the command-line application NETSTAT to view listening sockets (Table 9).

No.	Time	Source	Prot.	Destination	Len.	Info
914	1233.698020	*x.x.x.102*	TCP	*x.x.x.66*	58	46752 > 33112 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
917	1235.321910	*x.x.x.66*	TCP	*x.x.x.102*	60	33112 > 46752 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0

Table 8: TCP SYN against TCP/33112 with TCP AC from infection point.

Prot.	Local Addr.	Foreign Addr.	State	PID
TCP	*X.X.X.66*:33112	0.0.0.0:0	LISTENING	3412

Table 9: Using NETSTAT on the infection point to verify a listening state on TCP/33112.

3.3.4 Bobax

Expected Behavior

A Legacy Centralized Design is used by Bobax for C&C communications using HTTP or HTTPS as its transport medium. C&C traffic is sent encrypted or unencrypted to a C&C server over TCP/80 or TCP/447 respectively [38, 48]. Bobax utilizes port scanning to identify vulnerable hosts listening on TCP/5000 (UPnP) [45], and launches a remote exploit on the node to execute the Bobax malware loader. Bobax leverages a DGA to resolve an address of a C&C server and remain resilient.

Observed Behavior

The selected variant of Bobax is identified as W32.Bobax.B by SEP. After being infected with the Bobax malware, the machine began performing name lookups for a series of FQDNs. A series of TCP SYN requests were issued against the IP addresses returned from the name lookups. The remote hosts appear to be inactive, as TCP three-way handshake never completed. The malware remained dormant after these attempts to resolve inactive FQDNs.

3.3.5 Conficker

Expected Behavior

Conficker variants may utilize a decentralized or centralized architecture depending on the build of Conficker being evaluated and the circumstances that the bot is operating under [2]. Early variants of Conficker utilized only a DGA to resolve the addresses of C&C servers, whereas later variants were equipped with added capabilities to perform Internet-wide port scanning and various other methods to detect, exploit and build P2P relationships with other vulnerable nodes. Porras et al. discover that C&C communications are possible over UDP and TCP, and the payload in either situation is digitally signed and encrypted. Digital signage is performed using a 4096 byte RSA key and the MD6 hashing algorithm. Data encryption is performed with the RC4 stream cipher, and an encryption routine can be executed several times to add further layers of encryption [26]. When operating in a centralized architecture, Conficker will communicate on TCP/80, but will dynamically assign a communication port when operating in P2P mode. P2P port bindings are based upon the open ports discovered or requested during the port scan discovery phase [24].

Observed Behavior

The selected variant of Conficker is identified as W32.Downadup.E by SEP. Within ten minutes of the installation of the Conficker malware, the infected machine began scanning randomized IP addresses in the public address space using TCP SYN requests. Porras et al. describe similar behavior in their analysis of the Conficker C variant [26] as its method of building a P2P relationship with other Conficker peers in its discovery phase (Table 10). Unlike the prior research, however, a TCP connection was never built with a remote host. During the scanning process, TCP SYN packets were sent to several prominent websites, in what appears to be connectivity checks (Table 11). Beyond the constant port scanning cycle of the malware, no other established connections were observed.

Based upon the observed scanning behavior and the signature detection by SEP, the chosen binary of Conficker is ultimately a P2P variant. Prior research thus serves as the major source of input for the CM.

No.	Time	Source	Prot.	Destination	Len.	Info
1022	587.658121	*x.x.x.66*	TCP	*x.x.x.118*	58	51718 > 45597 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1023	591.189130	*x.x.x.118*	TCP	*x.x.x.66*	60	45597 > 51718[RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1024	592.665070	*x.x.x.66*	TCP	*x.x.x.118*	58	51718 > 18343 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1026	593.341012	*x.x.x.118*	TCP	*x.x.x.66*	60	18343 > 51718[RST, ACK] Seq=1 Ack=1 Win=0 Len=0
<< Truncated >>						
1057	643.172901	*x.x.x.66*	TCP	*x.x.x.43*	58	55670 > 50590 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1061	644.138873	*x.x.x.66*	TCP	*x.x.x.43*	58	55670 > 15851 [SYN] Seq=0 Win=1024 Len=0 MSS=1460

Table 10: TCP SYN probes launched from the infection point to random IP public IP addresses. In the first example, TCP RST responses are received, whereas the responses are being filtered for the second example.

No.	Time	Source	Prot.	Destination	Len.	Info
1022	1422.497360	*x.x.x.66*	TCP	65.55.206.228	58	53990 > http [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1023	1424.176123	65.55.206.228	TCP	*x.x.x.66*	58	http > 53990 [SYN, ACK] Seq=0 Ack=1 Win=8190 Len=0 MSS=1460

Table 11: TCP SYN scan initiated against MSN.com.

3.3.6 iKee

Expected Behavior

IKee.B utilizes a Legacy Centralized Design for C&C communications and, unlike other botnets evaluated in this work, specifically targets a mobile platform (Apple iOS) [11]. Utilizing HTTP as a transport medium, iKee.B sends a specially crafted GET request to a hard-coded IP address of a C&C server over TCP/80 to begin its bootstrap process [28]. The initial bootstrap process is comprised of downloading and executing various UNIX shell scripts on the infected handheld device. Due to the use of a hard-coded C&C server address, removing the C&C servers behind the IP address will cripple the botnet.

Observed Behavior

The resources to perform static packet analysis on the iKee.B botnet were not available. Furthermore, the research led by Porras et al. [11] identifies that the C&C servers for this malware are no longer active.

3.3.7 GTbot

Expected Behavior

Per the research presented by Canavan in [44], the Aladinz family of infections is a derivative of GTbot, and was amongst the assortment of malware collected prior to the manual evaluation. Such variants utilize a Legacy Centralized Design for IRC-based C&C communications, typically over TCP/6667 [12], but can vary based upon the infection. A mIRC client is bundled with the malware to retrieve customized C&C instructions [44] from the remote host due to the added functionality and powerful scripting capabilities [35, 36] available to a Botmaster.

Observed Behavior

The selected variant of GTbot is identified as Backdoor.IRC.Aladinz.B by SEP. Despite its age, the malware was capable of connecting to a remote IRC server using TCP/6667 and performing a simple login routine. The login routine utilized a randomized string of alphanumeric characters as the NICK and USER variables (Table 12) for the authentication phase, and responded to the standard PING message from the remote host with a PONG message (Table 13). Shortly thereafter, the bot appeared to join a channel named #MRZ by issuing the command /join #MRZ (Table 14). The IRC connection remained in an idle mode for the duration of evaluation, with only PING and PONG messages exchanged to keep the connection alive at 100 second intervals.

No.	Time	Source	Prot.	Destination	Len.	Info
15	3.020929	*x.x.x.66*	IRC	*x.x.x.4*	106	Request (NICK) (USER)
						NICK :2373abxd USER :2373abxd

Table 12: IRC NICK/USER authentication to remote C&C server.

No.	Time	Source	Prot.	Destination	Len.	Info
104	39.691632	*x.x.x.66*	IRC	*x.x.x.4*	86	Request (PING)
105	39.780873	*x.x.x.4*	IRC	*x.x.x.66*	131	Response (PONG)

Table 13: Initial IRC PING/PONG exchange.

No.	Time	Source	Prot.	Destination	Len.	Info
33	10.234800	*x.x.x.66*	IRC	*x.x.x.4*	114	Request (JOIN)
						Request: JOIN #MRZ :
42	16.049790	*x.x.x.4*	IRC	*x.x.x.66*	148	Response (JOIN)
						Response: 2343abxd@*x.x.x.66.x.cox.net* JOIN #MRZ

Table 14: IRC channel join on channel #MRZ.

3.3.8 Nugache

Expected Behavior

Dittrich and Dietrich [37, 39] detail two C&C architectures present in different revisions of Nugache – centralized and decentralized. Nugache variants that operate with a centralized architecture rely on the traditional use of IRC for a C&C mechanism. The focus for this research, however, will be on Nugache variants that employ P2P C&C communications.

Nugache infected peers rely upon a seed list of 22 other infected nodes [9] for its P2P bootstrap process. C&C communications utilize random, high-numbered TCP ports for outbound communications and encrypt their payload using 256-bit AES (Rijndael) session keys [13]. Session keys are exchanged between other peers referenced in the seed list [13] using ephemeral (short lifetime) RSA keys ranging in size from 64 to 128 bytes [39]. Dittrich and Dietrich further identify the AES block cipher mode as Output Feedback (OFB) [39], which effectively changes a block cipher into a stream cipher, making it ideal for networked communications. Upon a successful P2P connection, an updated list of peers is purportedly downloaded from a peer or series of peers [13, 32, 39].

Observed Behavior

The selected variant of Nugache is identified as W32.Nugache.A@mm.B by SEP, but did not illicit the characteristics as identified by Symantec [53]. Namely, Symantec identifies this variant as utilizing IRC over TCP/8 for C&C communications. This, however, was not the case. The only activities observed during the evaluation were TCP SYN requests against 9 unique IP addresses on randomized ports, which were never retried after the timeout period had lapsed (Table 15). The limited information gathered from the observation indicates a P2P variant of Nugache.

No.	Time	Source	Prot.	Destination	Len.	Info
911	86.138892	*x.x.x.66*	TCP	*x.x.x.113*	58	56133 > 53126 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
912	86.605351	*x.x.x.66*	TCP	*x.x.x.84*	58	56133 > 47813 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
913	86.694362	*x.x.x.66*	TCP	*x.x.x.97*	58	56133 > 34521 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
<< Truncated >>						

Table 15: A snippet of TCP SYN probes observed by the Nugache variant.

3.3.9 Peacomm

Expected Behavior

Peacomm resembles a decentralized architecture, due to its custom implementation of the Overnet P2P protocol [2], but utilizes hierarchy in C&C communications. Thus, this is categorized as a Hierarchical P2P Design.

Stewart identifies that the Overnet protocol is purely used for locational data; used by subnodes to identify the location of supernodes and used by subcontrollers to identify the location of subnodes [33]. The Overnet protocol utilizes the Kademlia [34] algorithm, which functions as a Distributed Hash Table (DHT) for file sharing purposes over UDP or TCP. A brief description of a DHT is presented in Appendix A.1.1. Likewise, Peacomm too utilizes Kademlia as its method of searching, but manipulates it to allow supernodes to publish its C&C socket to subnodes, and allow subnodes to function as search engines to other subnodes and subcontrollers [6, 33, 47]. Each participant of the Overnet network used by Peacomm stores other peer MD4 hashes (a unique peer identifier). The current date and a checksum value are encoded into the peer MD4 hash for subnodes. The hash is published to the other subnodes, whose peer MD4 hashes are close matches (as identified with a simple XOR function) to enable them to resolve the location for that specific subnode. If a peer MD4 hash is unknown by a peer, a search will yield the peer MD4 hash value of other subnodes that contain a close match (again, using an XOR function), and thus will quickly locate the subnode.

The actual C&C communications are performed over HTTP, using several tiers of nodes to direct and proxy communications [33]. As previously discussed, the socket of a supernode is made known to subnodes using P2P relationships. C&C traffic to supernodes is encoded using Base64, compressed using the zlib compression library, and is encapsulated in HTTP/1.1 POST requests. A challenge/response authentication method is also invoked prior to HTTP communications ensuing between supernodes and subnodes. Communications between the subcontroller layer and the C&C server vary slightly from these

methods, as traffic may also be encrypted using the RSA cipher suite to secure the transmission of locational data associated with active subcontrollers.

Peacomm leverages the use of double-flux DNS, at the supernode level of the hierarchy, to perform name resolutions for subnodes. It also introduces a reverse proxy at the supernode level, which delivers HTTP C&C communications to the subcontrollers [33] without revealing their location.

Observed Behavior

The variant of Peacomm that was downloaded was identified by SEP as Trojan.Peacomm.E. In what appear to be Overnet-style connection requests, the malware begins generating outbound UDP traffic on UDP/14507 to a series of remote hosts on random UDP ports immediately after being executed for the first time (Table 16). With no responses to the UDP traffic, this variant provided very little additional data to the prior information within the period of evaluation.

No.	Time	Source	Prot.	Destination	Len.	Info
140	39.128984	*x.x.x.66*	UDP	*x.x.x.25*	42	Source port: 14507 Destination port: 40125 [Malformed Packet]
141	39.131857	*x.x.x.66*	UDP	*x.x.x.2*	42	Source port: 14507 Destination port: 40125 [Malformed Packet]
142	39.133289	*x.x.x.66*	UDP	*x.x.x.67*	42	Source port: 14507 Destination port: 40125 [Malformed Packet]
<< Truncated >>						

Table 16: A snippet of malformed UDP traffic directed at external nodes.

3.3.10 Phatbot

Expected Behavior

Phatbot initiates and accepts connections to/from other Phatbot peers to efficiently route C&C communications. The WASTE protocol can be leveraged by Phatbot peers to participate in a partial mesh network [37], using link-level encryption to secure the communications between only the participants of a given WASTE session [14]. RSA is utilized as the method of session key exchange between participants for encrypted communications, in addition to being the technology used for public key authentication. WASTE encrypts the link using the Blowfish cipher in Propagating Cipher-block Chaining (PCBC) mode to secure the data in transit. Internet-routable and private (RFC 1918) addresses are both capable of

participating in the WASTE partial mesh topology [14], but will communicate in either an active or passive capacity (respectively) for C&C communications.

Observed Behavior

The variant of Phatbot that was downloaded was identified by SEP as W32.HLLW.Gaobot.gen, which is a signature description that covers Polybot variants as well [52]. During static analysis, TCP SYN requests were initiated against remote IP addresses on TCP/4387 (Table 17). For the duration of the evaluation, the malware attempted to connect to five different IP addresses over TCP/4387, but a three-way handshake never succeeded. The age of the malware and abandonment of WASTE is estimated to have played a large factor in such limited results.

No.	Time	Source	Prot.	Destination	Len.	Info
252	79.909342	*x.x.x.66*	TCP	*x.x.x.34*	58	39487 > 4387 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
253	79.910353	*x.x.x.66*	TCP	*x.x.x.92*	58	39487 > 4387 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
254	79.914608	*x.x.x.66*	TCP	*x.x.x.185*	60	4387 > 39487 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
<< Truncated >>						

Table 17: TCP SYN probes against remote nodes using the native WASTE TCP port designation.

3.3.11 SDbot

Expected Behavior

Gu identifies SDbot as the first standalone and open-source botnet utilizing IRC for C&C purposes [5]. SDbot is bundled with its own custom IRC client, which is required to send and receive IRC C&C communications over its Legacy Centralized architecture. During an initial bootstrapping process, SDbot initiates an IRC connection to the IP address of a C&C server, which is hard-coded into the application, and proceeds to download updated instructions. As described by Barford in [12], a bot performs an automated connection process, joins a channel, and awaits announcements from the Botmaster. The channel announcements honored by bots are limited to the KICK, NICK and PART/QUIT commands, and the remainder of commands is delivered to bots through PRIVMSG, NOTICE or TOPIC messages [12]. The extensive evaluation Gu et al. perform on IRC command obscurity [5] identifies that the IRC channel utilized for SDbot is not encrypted, nor are the communications difficult to discern.

Observed Behavior

The variant of SDbot that was downloaded was identified by SEP as Backdoor.IRC.SDbot, which is a very broad categorization for a large family of malware, making it difficult to know the specifics of the variant prior to infecting the victim (infection point). Upon infection, the malware immediately attempted to contact a remote host to establish a socket on TCP/6667. Within five minutes of the initial TCP connection request occurring, another request was initiated against a second remote host, again over TCP/6667. Neither connection was established. To verify that this variant of SDbot was capable of launching the IRC connection sequence to a remote host, a router was placed in front of the infected node to perform inbound and outbound address translations to a known active and anonymous IRC server. The two IP addresses observed from the initial (failed) connection requests were translated to a single IP of a live IRC server, resulting in the bot attempting to perform its login routine; a NICK of {malade}-4343844 and USER of m4343844 were used in the connection request (Table 18). An expected IRC PING message, as defined in RFC2812 [36] and outlined by Barford [12], was never received by the Bot, which resulted in the connection attempt failing permanently. At a very minimum, this process revealed a similar NICK naming convention that was identified by Goebel and Holz in [40] and confirmed the major C&C transport protocols.

No.	Time	Source	Prot.	Destination	Len.	Info
1022	1230.291838	*x.x.x.66*	IRC	*x.x.x.86*	114	Request (NICK) (USER)
						NICK : {malade}-4343844 USER : m4343844

Table 18: The USER/NICK revealed when translating the original destination address to a known/active IRC server.

3.3.12 Sinit

Expected Behavior

Sinit utilizes the Full P2P Decentralized design, building relationships with other bots through internet-wide port scans. Port scans are conducted through UDP/53 and check for the existence of other active bots by probing open ports [22]. Bots utilize HTTP as the C&C transport medium, and fetch updated versions of the Sinit client or C&C data using HTTP/1.1 GET requests over TCP/53 or a random, high-numbered TCP port. Sinit utilizes public key cryptography for authentication and encryption of data

between peers [9, 23] prior to the download of updated information. Infected machines perform connectivity checks against the random, high-numbered TCP ports to determine if the machine is reachable from the Internet [22].

Observed Behavior
A Sinit binary could not be accurately identified by means of signature-based detection for the purpose of research, limiting the research to a logical evaluation process.

3.3.13 Waledac

Expected Behavior
Waledac's architecture is largely decentralized, but described by Nunnery as hierarchical due to the role-based separation of duties amongst the infected hosts and the Botmaster-owned infrastructure [6]. This malware is being classified in this work as utilizing a Next-generation Centralized design. C&C communications require HTTP as the application protocol. Encoding of the HTTP traffic is performed using the Base64 encoding scheme, and the payload is compressed and encrypted. HTTP traffic is sent and received over TCP/80 and uses double-flux name resolution to retrieve C&C instructions from a subcontroller [6]. Calvet identifies that Waledac binaries are hardcoded with a list of 100-500 unique FQDNs or IP addresses for known supernodes (repeaters) [43]. Multiple supernodes are contacted for C&C instructions and to gather an updated list of other supernodes.

The takedown efforts by Microsoft [55, 56] for the infrastructure of Waledac in 2010 give reason to believe that limited results will be achieved through manual evaluation. Due to the nature of malware, and the time that has passed manual evaluation was still performed to collect further data about the malware.

Observed Behavior
The variant of malware was identified as W32.Waledac.C by SEP. Initial signs of C&C behavior were evident in the packet trace through a series of TCP three-way handshakes to seemingly random FQDNs on TCP/80. This behavior differs from the initial behavior described by Symantec [54], as it established a connection to several remote hosts rather than waiting for commands. The machine did, however, begin listening on TCP/80 for incoming connections. Within the period of evaluation a single download

occurred using a HTTP/1.1 GET request against a file with a .EXE extension, but the machine remained dormant following this download (Table 19).

No.	Time	Source	Prot.	Destination	Len.	Info
1007	1079.909342	*x.x.x.66*	HTTP	*x.x.x.31*	534	GET /runes123.exe HTTP/1.1
1008	1079.910353	*x.x.x.31*	TCP	*x.x.x.66*	1514	[TCP segment of a reassembled PDU]
<< Truncated >>						
1518	1079.914608	*x.x.x.31*	HTTP	*x.x.x.66*	421	HTTP/1.1 200 OK (application/x-msdos-program)

Table 19: HTTP/1.1 GET request and subsequent download of executable file.

3.3.14 Zeus

Expected Behavior

The bots created from various distributions of Zeus have different feature sets, so expectations were minimal. The Zeus variant was expected to communicate in a Next-generation Centralized design, using HTTP as its C&C transport mechanism, based upon the information from the Zeus Tracker project [57].

Observed Behavior

The variant of malware was identified as Trojan.Zbot by SEP. It initiated a TCP connection against a remote host using TCP/80 and successfully completed a three-way handshake (Table 20). During this process, no name lookups occurred, and only a single remote host was communicated with. Immediately following the handshake, a HTTP/1.1 GET request was issued to download a file with a .BIN file extension (Table 21) and then proceeded to download a file with a .EXE file extension (Table 22). The downloaded .BIN file was presumably an updated instruction file, but its contents were encrypted with an unknown cipher. The .EXE file was an installation package for the AV Security 2012 rogue-security software, which was revealed when the package was executed and launched on the infection point. Further analysis of the HTTP payload associated with the download of AV Security 2012 revealed several commands being appended to the payload as data padding (Table 23). The only further C&C communications observed during the evaluation were periodic check-ins to the remote host, with HTTP/1.1 POST requests against a file with a .PHP file extension. Due to the very simplistic nature of this variant, it is suspected that the malware is utilizing a Legacy Centralized Design for C&C communications as opposed to the expectations.

No.	Time	Source	Prot.	Destination	Len.	Info
12	11.854251	*x.x.x.66*	TCP	*x.x.x.20*	62	1052 > http [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
13	12.219794	*x.x.x.20*	TCP	*x.x.x.66*	62	http > 1052 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1372 SACK_PERM=1
14	12.221962	*x.x.x.66*	TCP	*x.x.x.20*	60	1052 > http [ACK] Seq=1 Ack=1 Win=64484 Len=0

Table 20: Three-way TCP handshake with remote C&C host

No.	Time	Source	Prot.	Destination	Len.	Info
15	12.223935	*x.x.x.66*	HTTP	x.x.x.20	220	GET /nbren.bin HTTP/1.1
16	12.444535	*x.x.x.20*	TCP	*x.x.x.66*	54	http > 1052 [ACK] Seq=1 Ack=166 Win=6432 Len=0
17	12.449296	*x.x.x.66*	TCP	*x.x.x.66*	1426	[TCP segment of a reassembled PDU]
<< Truncated >>						
56	13.535691	*x.x.x.20*	HTTP/DL	*x.x.x.66*	1265	unknown (0xa7) [Message: HTTP/1.1 200 OK\r\n]

Table 21: HTTP/1.1 GET request for a file with a .BIN extension

No.	Time	Source	Prot.	Destination	Len.	Info
173	130.885970	*x.x.x.66*	HTTP	*x.x.x.20*	219	GET /loader.exe HTTP/1.1
174	131.105633	*x.x.x.20*	TCP	*x.x.x.66*	54	http > 1054 [ACK] Seq=1 Ack=165 Win=6432 Len=0
175	131.111460	*x.x.x.20*	TCP	*x.x.x.66*	1426	[TCP segment of a reassembled PDU]
<< Truncated >>						
197	131.925387	*x.x.x.20*	HTTP/DL	*x.x.x.66*	1162	unknown (0x4d) [Message: HTTP/1.1 200 OK\r\n]

Table 22: Executable file download from remote host, presumably containing AV Security 2012.

```
KERNEL32.DLL...GetLastError...GetModuleHandleA... LoadLibraryA.2.FreeLibrary.D.VirtualProtect.

.ExitProcess....USER32.DLL...CloseWindow...SwitchToThisWindow.............................

.....................................................................................|K...P........

.......(...@...8...D.......k..............Z...g...s.......

vhgqn.dll.EndHyaessfre.ReadRqonhxb.InitYsyliubvg.Petugd
```

Table 23: Functions padded into the payload of the executable file download from Table 22.

3.4 Complexity Matrix

	C&C Characteristics	AgoBot	AsProx	Bagle	Bobax	Conficker	iKee	GTbot	Nugache	Peacomm	Phatbot	Sdbot	Sinit	Waledac	Zeus
Architecture	Centralized	1	1	1	1	1	1	1	0	0	0	1	0	1	1
	Decentralized	0	0	0	0	1	0	0	1	1	1	0	1	0	0
	Role Based Separation of Duties	0	1	0	0	1	0	0	1	1	0	0	0	1	0
	Uses Distributed Computing	0	0	0	0	0	0	0	1	1	1	0	1	0	0
	At least 1 Intermediary 0de	0	1	0	0	1	0	0	1	1	0	0	0	1	0
	More than 1 Intermediary 0des	0	1	0	0	1	0	0	1	1	0	0	0	1	0
High Availability & Fault Tolerance	Uses a Domain Generation Algorithm	0	1	0	1	1	0	0	0	0	0	0	0	0	0
	Uses Single-flux DNS	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	Uses Double-flux DNS	0	0	0	0	1	0	0	0	1	0	0	0	1	0
	Uses List of 0des	0	1	1	1	0	0	1	1	0	0	0	1	1	1
	Uses Hash Table of 0de Locations	0	0	0	0	0	0	0	0	1	1	0	0	0	0
	Dynamically Updates List of 0des	0	1	1	1	0	0	1	1	0	0	0	1	1	1
	Dynamically Updates List of Hashes	0	0	0	0	0	0	0	0	1	1	0	0	0	
	Performs Connectivity Checks	0	1	0	0	1	0	0	0	0	0	0	0	1	1
	Automatically Builds Relationships	0	0	0	0	1	0	0	1	1	1	0	1	1	0
Authentication & On-Repudiation	Requires Authentication	1	1	0	1	1	0	1	1	1	1	1	1	1	1
	Digitally Signs Messages	0	0	0	0	1	0	0	1	1	1	0	1	1	0
	Uses Challenge/Response Authentication	0	0	0	0	0	0	0	0	1	0	0	0	0	0
Encryption Methods	Payload Encryption	0	0	0	0	1	0	0	1	1	1	0	0	1	0
	Instruction File Encryption	0	1	1	1	0	0	0	0	0	0	0	0	1	1
	End-to-End Encryption	0	0	0	1	0	0	0	0	0	0	0	1	0	0
	Link-Layer Encryption	0	0	0	0	0	0	0	0	0	1	0	0	0	0
Evasion Tactics	Generates Arbitrary Traffic	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	Pads Instructions into Protocol Payload	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	Obscures Commands	1	0	0	0	0	1	0	0	0	0	1	0	0	0
	Hides Instructions in Flat File	0	1	1	1	0	0	0	0	1	1	0	1	1	1
	Encodes Payload	0	0	0	0	0	0	1	0	1	0	0	0	1	0
	Compresses Payload	0	1	1	1	1	1	0	0	1	0	0	0	1	1
Transport Protocols	Uses TCP	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Uses UDP	0	0	0	0	1	0	0	0	1	0	0	0	0	0
Application Protocols	Uses HTTP (includes HTTPS)	0	1	1	1	1	1	0	0	0	0	0	1	1	1
	Uses IRC	1	0	0	0	0	0	1	0	0	0	1	0	0	0
	Uses WASTE	0	0	0	0	0	0	0	0	0	1	0	0	0	0
	Uses Overnet	0	0	0	0	0	0	0	0	1	0	0	0	0	
	Uses Custom Protocol	0	0	0	0	0	0	0	1	0	0	0	0	0	0
	Complexity Score	5	14	8	11	16	5	7	13	19	12	5	11	19	12
	Dates of Inception or Discovery	2002	2007	2004	2004	2008	2009	2000	2006	2007	2003	2002	2003	2008	2012

3.4.1 C&C Characteristic Definitions

	C&C Characteristics	Definition
Architecture	Centralized	Botnet utilizes a centralized design.
	Decentralized	Botnet utilizes a decentralized design.
	Role Based Separation of Duties	Infected nodes are capable of serving different purposes in a botnet hierarchy.
	Uses Distributed Computing	Harnesses the computing power of other infected nodes to collectively perform a process or processes (e.g. searching).
	At least 1 Intermediary Node	At least 1 intermediary node is performing C&C functions between a subnode (bot) and a C&C master.
	More than 1 Intermediary Nodes	More than 1 intermediary nodes are performing C&C functions between a subnode (bot) and a C&C master.
High Availability & Fault Tolerance	Uses a Domain Generation Algorithm	Utilizes an algorithm for generating domain names, specifically those domain names used for C&C communications.
	Uses Single-flux DNS	Utilizes single-flux DNS to resolve C&C infrastructure elements (refer to A.1.3).
	Uses Double-flux DNS	Utilizes double-flux DNS to resolve C&C infrastructure elements (refer to A.1.4).
	Uses List of Nodes	Bots retain a list of nodes used in C&C communications, in the form of peers or masters (depending on design type).
	Uses Hash Table of Node Locations	Bots retain a hash table of node locations for identifying other nodes used in C&C communications (refer to A.1.1).
	Dynamically Updates List of Nodes	Bots are capable of updating node lists (used for C&C communications) on their own.
	Dynamically Updates List of Hashes	Bots are capable of updating hash tables (used for C&C location information) on their own.
	Performs Connectivity Checks	Performs checks for connectivity to identify: • Outbound connectivity to the Internet • Outbound/inbound communication ports • Access to C&C resources
	Automatically Builds Relationships	Automatically identifies relationships with peers or masters (depending on design) for C&C communications.
Authentication & Non-Repudiation	Requires Authentication	C&C communications only occur after authentication between botnet members occurs.
	Digitally Signs Messages	Messages used in C&C communications are digitally signed to verify the sender's identity.
	Uses Challenge/Response Authentication	C&C communications only occur after a challenge/response authentication routine succeeds.

Encryption Methods	Payload Encryption	The payload of the C&C communications is encrypted and requires a decryption routine to be run when the communications are processed by the recipient. The entire transport protocol is not encrypted, however.
	Instruction File Encryption	C&C instructions are provided in an instruction file that must be decrypted prior to reading from it.
	End-to-End Encryption	Encryption is used to secure a C&C communications channel between sender and receiver. Only the sender and receiver are intended to decrypt the communications.
	Link-Layer Encryption	Encryption is used to secure an entire domain of nodes. C&C communications can be encrypted and decrypted by all members of the encryption domain (e.g. all participants). Non-participants cannot communicate in the encryption domain.
Evasion Tactics	Generates Arbitrary Traffic	Botnet members generate meaningless network traffic to confuse a researcher.
	Pads Instructions into Protocol Payload	Instructions for carrying out a given task are padded in the protocol payload in addition to other content (e.g. updated bot binary). The task is carried out after the recipient processes the communications.
	Obscures Commands	C&C commands do not follow a typical or standard format, by means of manipulating a protocol message format.
	Hides Instructions in Flat File	C&C instructions are provided in an instruction file.
	Encodes Payload	The payload of C&C transactions are encoded prior to transmission and must be decoded by the recipient.
	Compresses Payload	The payload of C&C transactions are compressed prior to transmission and must be inflated by the recipient.
Transport Protocols	Uses TCP	Utilizes the Transmission Control Protocol (TCP) as the primary transport protocol for C&C communications.
	Uses UDP	Utilizes the User Datagram Protocol (UDP) as the primary transport protocol for C&C communications.
Application Protocols	Uses HTTP (includes HTTPS)	Utilizes the Hypertext Transfer Protocol (HTTP) or HTTP Secure (HTTPS) as the primary application protocols for C&C communications.
	Uses IRC	Utilizes Internet Relay Chat (IRC) as the primary application protocol for C&C communications.
	Uses WASTE	Utilizes the P2P protocol WASTE as the primary application protocol for C&C communications.
	Uses Overnet	Utilizes the P2P protocol Overnet as the primary application protocol for C&C communications.
	Uses Custom Protocol	Utilizes a custom protocol (e.g. non-standard) for C&C communications.

4. Historical Data

In this thesis, logical elements of C&C communications are identified and used to characterize various botnets. A complexity matrix was then created for the chosen botnets, and the hypothesis that botnets are becoming more complex over time is verified.

Historical data is used to quickly illustrate trends in malware complexity. A complexity score will be used to define the C&C complexity associated with a given botnet variant.

4.1 Complexity Score

By graphing the complexity *scores* created from the CM, it is possible to illustrate the changes in botnet complexity relative to an inception or discovery date (Fig. 6). The average of multiple complexity scores is used in Fig. 6 when multiple botnet variants exist for a given year.

Due to the infancy of mobile botnets, a drop in complexity is identified in 2009 with the logical evaluation of the iKee botnet. Also, another unexpected value is seen in 2012, which represents the unsophisticated variant of Zbot (Zeus) that was used in the evaluation.

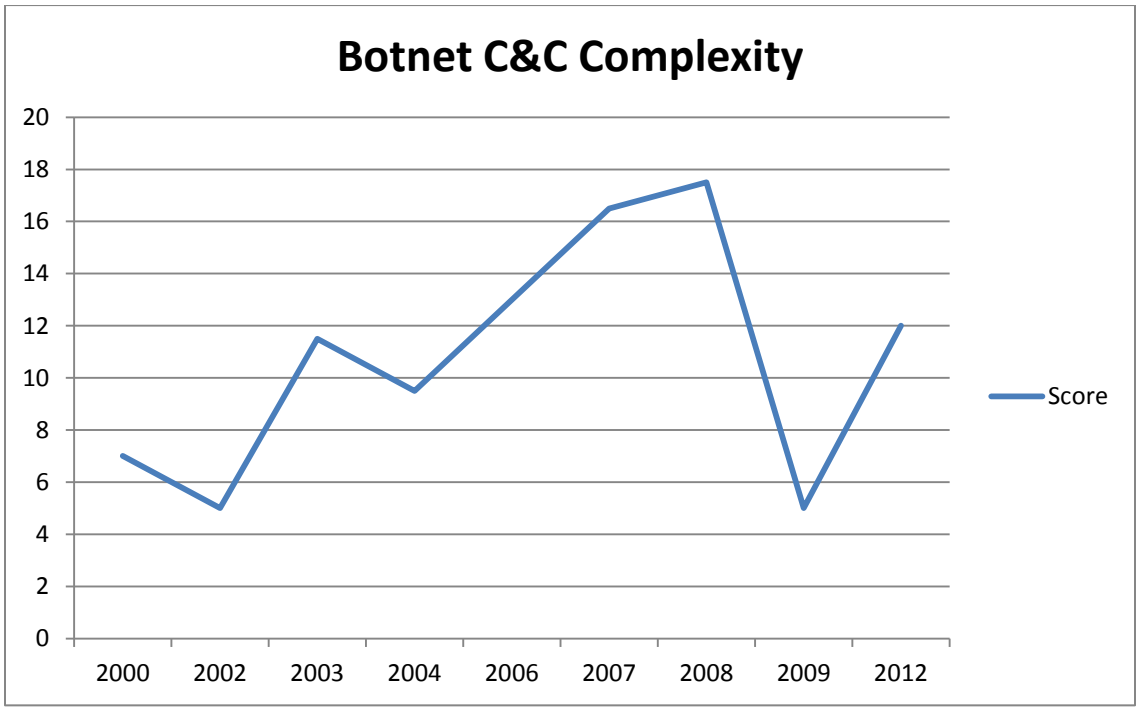

Figure 6: Botnet C&C complexity scores relative to date of inception or discovery of the particular variant of malware

5. Limitations & Future Work

This thesis evaluates a broad range of C&C methods employed by malware, but specific limitations exist in this research. The work presented is based upon static network traffic analysis and prior malware case studies; it does not utilize the advanced methods of malware deconstruction presented in prior research. Additionally, ideal conditions for manually evaluating a majority of the botnet examples were not present due to the degradation or removal of core C&C infrastructure elements.

Encrypting data at rest and in transit presents a challenge in identifying the core components of C&C communications. A significant number of the botnet selections utilized encryption for hiding C&C components, making static analysis limited in scope. Reverse engineering and real-time memory analysis make it possible to deconstruct encrypted C&C components. These methods were not employed in the evaluation phase of the work, and as such it relies on the credibility of prior research.

The proliferation of mobile devices has fueled the creation of malware for mobile platforms, which was not heavily discussed in this survey and subsequent evaluation. The single example, iKee, is an example of botnet behavior in its infancy. The topic of mobile platforms creating a larger canvas for botnets is thus left for future work.

The selected malware had 42.8% success rate of connecting to C&C resources during the manual evaluations. The limited results are caused by the age of the malware, environmental factors affecting key C&C infrastructure components, and the inability for the malware to recover from failure.

Due to the evaluation of some antiquated botnet variants, a manual approach of gathering the malware is used over an automated Honeypot/Honeynet design. A process for automation in this regard is left to future work as well.

6. Conclusion

Methods of C&C communications have vastly changed over time. Every aspect of C&C communications, from the architecture to advanced features of high-availability, is changing to evade detection. On a large scale, C&C communications are becoming more complex over time. The re-use of aging technologies or C&C approaches indicates that limited success is still possible without evolving, but it doesn't reap the benefits that more sophisticated approaches will for a Botmaster.

In testing, limited results were acquired with a manual evaluation of the selected variants of malware. Thus, it becomes difficult to evaluate malware of yesterday purely through observation. A collaborative effort is better leveraged to identify the idiosyncrasies that comprise a given botnet.

An approach at categorizing C&C features into a matrix proved useful, as the existence of specific features can accurately measure its complexity relative to prior malware and its creation date. Though legacy approaches at C&C still remain effective under specific circumstances, they ultimately will not achieve the same level of utility for affecting a large audience. Complexity in C&C techniques and the infrastructure that supports a botnet will ultimately impact its longevity, and an upward trend in complexity throughout history is clearly visible with the advent of a Complexity Matrix.

7. Acknowledgements

I would like to extend my thorough appreciation to my thesis committee for their sage guidance and assistance. I'd also like to extend a warm thank you to Susan Herzberg, whom gave me the final push to complete the requirements for the VNSM MS degree. Most importantly, I'd like to thank Gulmira Zhavgasheva, RIT VNSM 2010, whom has been my driving force for completion.

8. References

[1] “A Little Spam With Your Bagle?: M86 Security.” [Online]. Available: http://www.m86security.com/labs/i/A-Little-Spam-With-Your-Bagle-,trace.999~.asp. [Accessed: 01-Jul-2012].

[2] P. Porras, H. Saidi, and V. Yegneswaran, “A multi-perspective analysis of the storm (peacomm) worm,” *Computer Science Laboratory, SRI International, Tech. Rep*, 2007.

[3] B. J. Norman, “A Study of Peer-to-Peer Botnets,” M.S., Utah State University, United States -- Utah, 2008.

[4] A. H. Lashkari, S. G. Ghalebandi, and M. Reza Moradhaseli, “A Wide Survey on Botnet,” *Digital Information and Communication Technology and Its Applications*, pp. 445–454, 2011.

[5] G. Gu, V. Yegneswaran, P. Porras, J. Stoll, and W. Lee, “Active Botnet Probing to Identify Obscure Command and Control Channels,” in *Computer Security Applications Conference, 2009. ACSAC ’09. Annual*, 2009, pp. 241 –253.

[6] C. E. Nunnery, “Advances in modern botnet understanding and the accurate enumeration of infected hosts,” Ph.D., The University of North Carolina at Charlotte, United States -- North Carolina, 2011.

[7] “Agobot and the “Kit-chen Sink,” *infectionvectors.com - Agobot*, Jul-2004. [Online]. Available: http://www.infectionvectors.com/vectors/kitchensink.htm. [Accessed: 15-Jul-2012].

[8] “agobot3 command reference.” [Online]. Available: http://www.stanford.edu/~stinson/paper_notes/bots/bot_refs/agobot3_commandref.html. [Accessed: 04-Aug-2012].

[9] Ping Wang, S. Sparks, and C. C. Zou, “An Advanced Hybrid Peer-to-Peer Botnet,” *IEEE Transactions on Dependable and Secure Computing*, vol. 7, no. 2, pp. 113–127, Apr. 2010.

[10] R. Borgaonkar, “An Analysis of the Asprox Botnet,” in *Emerging Security Information Systems and Technologies (SECURWARE), 2010 Fourth International Conference on*, 2010, pp. 148 –153.

[11] P. Porras, H. Saïdi, and V. Yegneswaran, “An Analysis of the iKee. B iPhone Botnet,” *Security and Privacy in Mobile Information and Communication Systems*, pp. 141–152, 2010.

[12] P. Barford and V. Yegneswaran, “An inside look at botnets,” *Malware Detection*, pp. 171–191, 2007.

[13] S. Stover, D. Dittrich, J. Hernandez, and S. Dietrich, “Analysis of the Storm and Nugache Trojans: P2P is here,” *USENIX; login*, vol. 32, no. 6, pp. 2007–12, 2007.

[14] “Anonymous-P2P.org: WASTE.” [Online]. Available: http://anonymous-p2p.org/waste.html. [Accessed: 05-Aug-2012].

[15] “Asprox « M86 Security Labs Blog.” [Online]. Available: http://labs.m86security.com/tag/asprox/. [Accessed: 04-Aug-2012].

[16] “Asprox: M86 Security.” [Online]. Available: http://webcache.googleusercontent.com/search?q=cache:yJfP8261TWwJ:www.m86security.com/labs/spambotitem.asp%3Farticle%3D935+&cd=6&hl=en&ct=clnk&gl=us. [Accessed: 16-Jul-2012].

[17] “Backdoor.Sdbot | Symantec.” [Online]. Available: http://www.symantec.com/security_response/writeup.jsp?docid=2002-051312-3628-99. [Accessed: 22-Jul-2012].

[18] “Backdoor.Sinit | Symantec.” [Online]. Available: http://www.symantec.com/security_response/writeup.jsp?docid=2003-100910-5701-99. [Accessed: 28-Jul-2012].

[19] “Bagle: M86 Security.” [Online]. Available: http://www.m86security.com/labs/spambotitem.asp?article=938. [Accessed: 03-Jul-2012].

[20] “Bagle: Port Count.” [Online]. Available: http://www.m86security.com/newsImages/TRACE/Bagle_PortCount.txt. [Accessed: 04-Aug-2012].

[21] H. R. Zeidanloo and A. A. Manaf, “Botnet Command and Control Mechanisms,” in *Computer and Electrical Engineering, 2009. ICCEE ’09. Second International Conference on*, 2009, vol. 1, pp. 564 – 568.

[22] S. Heron, “Botnet command and control techniques,” *Network Security*, vol. 2007, no. 4, pp. 13–16, Apr. 2007.

[23] B. Sharmila, H. Bsubashini, and M. Hkalaiselvi, “Botnet Construction of Peer-to-Peer Network Schemes.” [Online]. Available: http://www.scribd.com/doc/57984998/final-doc. [Accessed: 06-Aug-2012].

[24] Q. Wang, “Characterizing Internet Worm Spatial-Temporal Infection Structures,” Ph.D., Florida International University, United States -- Florida, 2010.

[25] “Conficker | Conficker Worm | Conficker Virus.” [Online]. Available: http://www.microsoft.com/security/pc-security/conficker.aspx. [Accessed: 22-Jul-2012].

[26] P. Porras, H. Saidi, and V. Yegneswaran, “Conficker C P2P Protocol and Implementation,” 21-Sep-2009. [Online]. Available: http://mtc.sri.com/Conficker/P2P/. [Accessed: 22-Jul-2012].

[27] G. Gu, “Correlation-Based Botnet Detection in Enterprise Networks,” Ph.D., Georgia Institute of Technology, United States -- Georgia, 2008.

[28] Y. Zeng, K. G. Shin, and X. Hu, “Design of SMS commanded-and-controlled and P2P-structured mobile botnets,” in *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*, New York, NY, USA, 2012, pp. 137–148.

[29] M. Neugschwandtner, P. M. Comparetti, and C. Platzer, “Detecting malware’s failover C&C strategies with squeeze,” in *Proceedings of the 27th Annual Computer Security Applications Conference*, New York, NY, USA, 2011, pp. 21–30.

[30] M. Steggink and I. Idziejczak, “Detection of peer-to-peer botnets,” *University of Amsterdam, Netherlands*, 2007.

[31] A. Ghodsi, “Distributed k-ary system: Algorithms for distributed hash tables,” KTH-Royal Institute of Technology, 2006.

[32] B. Coskun, S. Dietrich, and N. Memon, “Friends of an enemy: identifying local members of peer-to-peer botnets using mutual contacts,” in *Proceedings of the 26th Annual Computer Security Applications Conference*, New York, NY, USA, 2010, pp. 131–140.

[33] J. Stewart, “Inside the Storm: Protocols and Encryption of the Storm Botnet,” 2008.

[34] “Kademlia: A Design Specification.” [Online]. Available: http://xlattice.sourceforge.net/components/protocol/kademlia/specs.html#intro. [Accessed: 05-Aug-2012].

[35] “mIRC: About mIRC.” [Online]. Available: http://www.mirc.com/about.html. [Accessed: 26-Jul-2012].

[36] “mIRC: RFC2812.” [Online]. Available: http://www.mirc.com/rfc2812.html. [Accessed: 04-Aug-2012].

[37] D. Dittrich and S. Dietrich, “New Directions in Peer-to-Peer Malware,” in *Sarnoff Symposium, 2008 IEEE*, 2008, pp. 1 –5.

[38] P. Royal, “On the kraken and bobax botnets,” *Whitepaper, Damball, Apr*, 2008.

[39] D. Dittrich and S. Dietrich, “P2P as botnet command and control: A deeper insight,” in *Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on*, 2008, pp. 41 –48.

[40] J. Goebel and T. Holz, “Rishi: Identify bot contaminated hosts by IRC nickname evaluation,” in *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, 2007, pp. 8–8.

[41] “SSAC Advisory on Fast Flux Hosting and DNS.” ICANN Security and Stability Advisory Committee, Mar-2008.

[42] “The Asprox Spambot Resurrects: M86 Security.” [Online]. Available: http://www.m86security.com/labs/i/the-asprox-spambot-resurrects,trace.1345~.asp. [Accessed: 01-Jul-2012].

[43] J. Calvet, C. R. Davis, J. M. Fernandez, J.-Y. Marion, P.-L. St-Onge, W. Guizani, P.-M. Bureau, and A. Somayaji, “The case for in-the-lab botnet experimentation: creating and taking down a 3000-node botnet,” in *Proceedings of the 26th Annual Computer Security Applications Conference*, New York, NY, USA, 2010, pp. 141–150.

[44] J. Canavan, “The Evolution of Malicious IRC Bots,” in *Proceedings of the VB2005 Conference*, 2005.

[45] “Threat Description: Bobax.” [Online]. Available: http://www.f-secure.com/v-descs/bobax.shtml. [Accessed: 22-Jul-2012].

[46] “Threat Description: Sinit.” [Online]. Available: http://www.f-secure.com/v-descs/sinit.shtml. [Accessed: 28-Jul-2012].

[47] J. Stewart, “Top Spam Botnets Exposed | Dell SecureWorks,” *Top Spam Botnets Exposed*, 08-Apr-2008. [Online]. Available: http://www.secureworks.com/research/threats/topbotnets/. [Accessed: 22-Jul-2012].

[48] “Troj/Agobot-A - Viruses and Spyware - Threat Analyses - Threat Center - Sophos.” [Online]. Available: http://www.sophos.com/en-us/threat-center/threat-analyses/viruses-and-spyware/Troj~Agobot-A.aspx. [Accessed: 15-Jul-2012].

[49] “Trojan.Asprox | Symantec.” [Online]. Available: http://www.symantec.com/security_response/writeup.jsp?docid=2007-060812-4603-99. [Accessed: 16-Jul-2012].

[50] “Trojan.Peacomm Technical Details | Symantec.” [Online]. Available: http://www.symantec.com/security_response/writeup.jsp?docid=2007-011917-1403-99&tabid=2. [Accessed: 28-Jul-2012].

[51] “W32.Gaobot.DX Technical Details | Symantec.” [Online]. Available: http://www.symantec.com/security_response/writeup.jsp?docid=2004-051816-5418-99&tabid=2. [Accessed: 15-Jul-2012].

[52] “W32.HLLW.Gaobot.gen | Symantec.” [Online]. Available: http://www.symantec.com/security_response/writeup.jsp?docid=2003-112112-1102-99. [Accessed: 28-Jul-2012].

[53] “W32.Nugache.A@mm | Symantec.” [Online]. Available: http://www.symantec.com/security_response/writeup.jsp?docid=2006-043016-0900-99. [Accessed: 26-Jul-2012].

[54] “W32.Waledac.C | Symantec.” [Online]. Available: http://www.symantec.com/security_response/writeup.jsp?docid=2012-020814-3639-99. [Accessed: 05-Aug-2012].

[55] “Waledac Technical Action Plan.” [Online]. Available: http://www.microsoft.com/security/sir/story/default.aspx#!waledac_technical. [Accessed: 22-Jul-2012].

[56] “What we know (and learned) from the Waledac takedown - Microsoft Malware Protection Center - Site Home - TechNet Blogs.” .

[57] “ZeuS Tracker :: ZeuS statistic.” [Online]. Available: https://zeustracker.abuse.ch/statistic.php. [Accessed: 05-Aug-2012].

A. Appendix

A.1 Definitions

Specific feature definitions are provided here due to their significance. The features have been clearly identified in prior work and are being summarized below for reference purposes. The presence or absence of these features is also used to categorize the sophistication of a botnet in the CM.

A.1.1 Distributed Hash Table

Ghodsi defines a Distributed Hash Table (DHT) as hash table distributed amongst a set of cooperating computers [31]. The hash table is used to store key/value pairs, to identify the location of a given resource. Searching across a set of cooperating computers is conducted to locate resources. Searching for a specific key will yield a given value, and the participating node that possesses the information on the searched resource will return the value or a pointer to another node with the value. This approach has been leveraged for file sharing services, such as BitTorrent, and has become useful in P2P malware as well.

A.1.2 Domain Generation Algorithm

A domain generation algorithm (DGA) is a computational method of generating FQDNs [31]. In terms of C&C communications, malware may utilize a DGA mechanism to create the FQDNs of C&C peers or masters (depending on the C&C topology). The DGA must be crafted in a manner that allows it to produce the same FQDNs for the individual responsible for registering the domain names. A DGA may not generate FQDNs in the same order when run, but it will be predictable to a certain degree. A malware author may perform the work of registering the domain names created by the DGA well in advance of the malware being distributed to enable it to quickly communicate with remote hosts behind seemingly random FQDNs. Malware can thus attempt connections to the FQDNs generated by an internal DGA until it discovers a domain that resolves to an IP address. The predictable FQDN generation techniques can also be utilized by researchers to identify domain names that will be used by a particular variant of malware prior to its registration with a registrar.

A.1.3 Single-flux DNS

Single-flux DNS refers to a method of overloading an Address (A) record in DNS with multiple IP addresses and very short time to live (TTL) values [10, 41]. The combination of these two features enables multiple IP addresses to be resolved from a single FQDN. Authoritative responses to name lookups can return a multitude of different IP addresses due to the round-robin behavior that occurs as a result of short TTL values. Malware authors utilize this functionality within botnets to build resiliency and redundancy into their C&C design and overcome the connectivity issues associated with utilizing single points of failure.

Additional advents to this technology could include overloading A records by means of dynamic registration with a rogue name server. This process allows other infected nodes to perform the duties of A record updates.

A.1.4 Double-flux DNS

Double-flux DNS builds upon the original idea of Single-flux DNS, but overloads the A records that map to NS records for a particular DNS zone [10, 41]. The result of such resource record overloading produces a large number of name servers for a zone. The remote host corresponding to a NS record at any point in time may purely be used to proxy a DNS lookup request to a true authoritative name server. By restricting the visibility of authoritative name servers to the remote hosts that proxy the DNS lookups, an authoritative name server can remain hidden from others.