NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

A TEST BED FOR DETECTION OF BOTNET INFECTIONS IN LOW DATA RATE TACTICAL NETWORKS

by

Russell W. Becker

September 2009

Thesis Advisor:	John McEachen
Co-Advisor:	Murali Tummala
Second Reader:	Vicente Garcia

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2009	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE A Test Bed for Detection of Botnet Infections in Low Data Rate Tactical Networks	**5. FUNDING NUMBERS**
6. AUTHOR(S) Russell W. Becker	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

The propagation of bots into a botnet, and the various malicious activities that could be carried out from within a tactical network, poses a significant threat to network security and tactical operations. This thesis presents a network architecture with the objective of near real-time detection of malicious activity and its propagation within a data rate (bandwidth) limited environment with periodic losses of connectivity without adding significant burden to the network.

A test bed is constructed that makes use of an intrusion detection system driven correlation tool, BotHunter, focused on outbound and inbound connections, rather than solely on inbound connections and a honeynet located in a high data rate area of a tactical network. The ability of the proposed architecture to identify malicious activities is validated when both BotHunter and the Honeynet successfully detect a bot infection.

14. SUBJECT TERMS Botnet, Tactical Network, BotHunter, Honeynet, Honeypot, Low Data Rate, Network Security	**15. NUMBER OF PAGES** 79
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 | Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

A TEST BED FOR DETECTION OF BOTNET INFECTIONS IN LOW DATA RATE TACTICAL NETWORKS

Russell W. Becker
Major, United States Marine Corps
B.S., University of Houston, 1992

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2009

Author: Russell W. Becker

Approved by: Professor John McEachen
Thesis Advisor

Professor Murali Tummala
Co-Advisor

Vicente Garcia
Second Reader

Jeffrey B. Knorr
Chairman, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The propagation of bots into a botnet, and the various malicious activities that could be carried out from within a tactical network, poses a significant threat to network security and tactical operations. This thesis presents a network architecture with the objective of near real-time detection of malicious activity and its propagation within a data rate (bandwidth) limited environment with periodic losses of connectivity without adding significant burden to the network.

A test bed is constructed that makes use of an intrusion detection system driven correlation tool, BotHunter, focused on outbound and inbound connections, rather than solely on inbound connections and a honeynet located in a high data rate area of a tactical network. The ability of the proposed architecture to identify malicious activities is validated when both BotHunter and the Honeynet successfully detect a bot infection.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

The propagation of bots, drone computers (processes) characterized by a command and control architecture and controlled by a bot controller, into an army of computers, a botnet, and the various malicious activities that could be carried out from within a tactical network poses a significant threat to network security and tactical operations.

The objectives of this work are near real-time detection of malicious activity and its propagation within a data rate (bandwidth) limited environment with periodic losses of connectivity without adding significant burden to the network.

This thesis assembles a test bed to emulate part of a tactical network architecture with low data rate and periodic losses of connectivity. The architecture makes use of an intrusion detection system driven correlation tool, BotHunter, focused on outbound rather than inbound connections and a honeynet to validate the BotHunter. BotHunter is placed in a position to experience losses in connectivity, as it will observe all inbound network traffic, but will be limited in its visibility of outbound traffic by the honeynet's connection limiting rules. The honeynet's honeywall is in a position to observe and record all network traffic.

The test bed architecture as a means of detecting botnet infections in low data rate tactical networks is validated. BotHunter continued to detect the bot infection after the periodic loss of connections caused by the

honeynet. The BotHunter detected bot infection that initially attempts to propagate prior to establishing command and control, a behavior that makes it particularly hazardous to tactical networks. The origination of the bot secondary infection and its malicious action of performing a Class B network scan were detected within a matter of minutes of the infection on a network limited to a bandwidth of 180 kbps.

Traffic analysis of all packets captured by the honeywall allowed determination of the network cost in terms of malicious traffic generated in the test bed by the single bot infection. The traffic cost for the bot infection captured in this work was measured to be 112 Bytes per second.

The requirement for positioning of a honeynet or honeynets within the test bed network architecture is validated by BotHunter's failure to capture all of the bot behavior, such as the attacking IP address.

While the concept was validated, three modifications are recommended for future work. A malware collection tool should be implemented in conjunction with BotHunter to enable collection, reporting and analysis. Placement of an instance of BotHunter between separate honeynets, without direct connectivity outside of the network, will test its effectiveness in internal bot propagation detection. Implementation of a honeynet that includes multiple instances of exactly the same operating system version will enable better observation of botnet propagation as it is likely to occur in a tactical network.

ACKNOWLEDGMENTS

I thank Professor’s McEachen and Tummala for allowing me infinite latitude in exploring. Many thanks to Donna Miller for her support early on and for being a sounding board. Robert “Bob” Broadston cannot be thanked enough for all of his technical assistance and words of wisdom and for all of the operating systems he allowed me to corrupt in the process.

To Robert McMillen of the Honeynet Organization, fellow NPS graduate and Marine, I extend my deepest appreciation for all your help steering a Noob out of the woods.

I cannot say enough about the support of my parents, Albert and Sandra Becker, who have supported me throughout my life, career and especially during my time at NPS. I am also grateful for the support of my siblings, Monte, Ronald, Kayse, and Karen, as well as their spouses and families. The whole family has supported me so that I might enjoy the luxury of continuing to serve.

I want to thank my wife, Hana, for her support through all the craziness that is school and the five deployments prior. And for my beautiful little daughters, Veronika and Mikaela, I am especially thankful for their love and affection…whatever time daddy got home.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Commercial, governmental and personal reliance on the Internet has reached a point that a vast majority in the developed world, and much of the rest of the world, is interconnected. Almost every long distance call, banking transaction, as well as national infrastructure, can be adversely affected by losses of Internet connectivity or by the malicious acts of individuals, organizations, or states that use the Internet. One kind of network exploitation is a “botnet.” The use of botnets allow for an individual or organization to harness and mass the computing power and bandwidth of large numbers of computers.

A botnet, a network of robot or drone computers (processes) characterized by the presence of a Command and Control (C2) channel, is a form of malicious software (malware) that is capable of self-propagation and can be controlled by a botmaster [1,2,3,4], unbeknownst to the computer’s owner. Frequently the botmaster will use the botnet for such purposes as conducting distributed denial of service (DDOS) attacks, collecting confidential information or for financial scams.

The botnet C2 architecture is the primary means by which it is classified [4] and distinguishes it from other malware such as viruses or worms. The most prevalent type of botnet C2 uses Internet relay chat (IRC). This is a centralized C2 architecture with the bots logging into a central IRC channel to receive commands and updates from the botmaster; however, this architecture has a significant weakness. It presents a single point of failure. If the

IRC server is taken off line, there is no longer a botnet but a number of individual bots without direction. In order to increase the survivability and hide the size of their botnets, botmasters have used other C2 architectures such as peer-to-peer (P2P), Hypertext Transfer Protocol (HTTP) and fast-flux networks [4]. P2P botnets come at a cost of increased latency in net response to commands, loss of definitive message acknowledgment, and increased complexity [5]. In recent research, Holz et al. met with some success in disrupting the communications of a P2P botnet [6]. Fast-flux is a more sophisticated approach to HTTP as a C2 architecture in an effort to increase the survivability of the network. In fast-flux networks, the botmaster uses a fully qualified domain name but rapidly changes the IP address the name resolves to by changing the DNS A records, and in some cases the authoritative name service records as well [7]. All of the above mentioned methods of C2 are further complicated by the use of encryption.

Bots are further characterized in [8] as Type I or Type II. A Type I bot first attempts to self-propagate prior to establishing communications with the C2, whereas the Type II does the opposite.

A. MOTIVATION

The protection of tactical networks that support the warfighter is predominantly reactive in nature, using definition driven IDS and IPS focused on preventing known attacks and located at the network perimeter. As well, they rely heavily on antivirus network scans, definition updates and propagation. The results are numerous alerts and alarms for information assurance and network administrators to dig through and analyze along with network traffic for signs of malicious activity. This leaves the subordinate or remote locations that may have rate limited transmission paths and limited training and without tools to detect and self diagnose a previously undefined botnet infection. An infection can propagate throughout the local area network and beyond its perimeter to a trusted adjacent unit or higher before it ever trips a perimeter security defense mechanism.

Due to most network defense systems being primarily focused on intrusion prevention and detection, there is a real question as to the ability to detect an infection that has bypassed those perimeter defenses and is either calling outbound or propagating within in order to mount some sort of attack from within, such as a DDOS. As of this writing, the author is not aware of any framework or architecture fielded that is aimed at supporting the signals or communications officer at the ship, regiment/brigade or lower level with detection and collection of malware.

B. OBJECTIVES

The overarching goal of this thesis is to assemble an architecture that increases the security of tactically deployed networks, enabling the detection of botnets that may have evaded the firewall, and to a lesser extent to capture copies of the malware code for reporting and analysis.

The primary objectives are near real-time detection of malicious activity and its propagation within a data rate (bandwidth) limited environment with periodic losses of connectivity without adding significant burden to the network. An additional topic to be explored in preparation for follow on research is the capture of malware for reporting and analysis.

The preponderance of intrusion detection system (IDS) and antivirus solutions are heavily definition driven with some heuristic components [9]. However, they are still reactive in nature, meaning that new threats may not be discovered unless they match a definition of a previously known threat. This presents the obvious question of how to respond to and stop the next generation of attacks. The first step is detection.

Tactical networks, or shipborne networks, are data rate limited. While most researchers using honeypots have implemented some sort of data rate limiting to mitigate the ability of their research computers being used by a botmaster to propagate an attack, this is a specific requirement that must be strongly considered in developing any network architecture for the study of tactical networks.

A test bed—defined here as a monitored and controlled environment designed to emulate a part or all of an actual network—can be used to establish conditions similar to those seen in tactical networks in order to facilitate understanding.

If a previously unidentified Type I bot infects the network, such as via a USB device, it may propagate within the local network and not be seen by the firewall or antivirus until it has spread extensively through the network and attempts a call home. This presents a significant problem.

In order to effectively combat a new bot variant, it is essential to capture the malware code for further analysis. While this is not a primary objective of this thesis, it is explored in this work in preparation for follow on research where it may be incorporated.

C. RELATED WORK

The body of work is rapidly growing as are the threats. The Honeynet Project has developed numerous tools and architectures designed for research in botnet tracking [1] and collection [9, 10], with Nepenthes [11] capable of collecting unknown exploits of old vulnerabilities. BotHunter [3] appears to be the first effort to use IDS type technology looking at outward traffic, rather than the customary inward, and using correlation to detect bots. BotHunter also allows for collaboration, with automated reporting back to SRI International. From a different perspective, [12] discusses a botnet communication model to evade detection by one of the tools used in this thesis, BotHunter, by grouping bots into local networks and subordinating them to a single controller bot within a switched network. The controller bot would then be tasked to manage the other bots and be the sole point of contact in and out of the compromised network.

All of the above were tested on large data rate networks or small stable, reliable networks. This thesis differs from previous research by focusing on bot/botnet detection in tactical networks, characterisized by low data rate connections and intermittent connectivity.

D. ORGANIZATION

Chapter II describes the process of creating and maintaining a botnet and covers a range of methods for detecting and capturing bots and combating botnets. Chapter III addresses the test bed design and how it fits into a low bandwidth tactical network. The results and analysis follow in Chapter IV. Chapter V provides conclusions, addresses the effectiveness of the test bed network architecture and offers suggestions for future research.

II. OVERVIEW OF BOTNETS

This chapter begins with the phases of creating and maintaining a botnet and follows with an introduction to different tools used to detect, capture and combat botnets.

A. CREATING AND MAINTAINING A BOTNET

There are generally four phases of creating and maintaining a bot: 1) initial infection, 2) secondary injection, 3) malicious activities, and 4) maintenance and upgrade [4]. It is quite common for the phases to be spread among different bots or groups of bots.

The initial infection can be accomplished in any of a number of methods [4]:

a) It could be an active exploit, initiated with some level of network enumeration and or vulnerability scanning. Upon locating a computer that meets the profile of a known vulnerability, the attacker will attempt an exploit tailored to the vulnerability.

b) While surfing the web, malware could be automatically downloaded. This may or may not require the user to actively click on links to initiate the malware download.

c) SPAM/email attachments are cleverly social engineered to deceive the user to open an attachment, resulting in automatic download and execution of the binary.

d) USB autorun. Conficker [13] uses the USB autorun for the malicious activities phase to further propagate.

All windows operating systems have specific ports for resource sharing—Ports 445/TCP, 139/TCP, 137/UDP and 135/TCP are discussed in detail in [1]—and these ports are credited for being a major mechanism through which bots are spread. These resource-sharing ports are trusted ports. Many computers are still left exposed to the Internet without a firewall, giving attackers easy access. Even with a firewall, there are other methods by which the attacker can gain a foothold in a network. Then, the bots can propagate within the network using the same resource sharing vulnerabilities. In addition to these, several other common vulnerabilities and ports are further discussed in [1] and will be observed and discussed in later sections.

The secondary injection, also known as the egg download, occurs when the full binary for the bot is downloaded. It will include all the tools required for the bot to continue exploitation and propagation. The egg download can come from the same IP address as the initial infection or a different IP address. In some cases, the egg download may occur at the same time as the initial attack/infection. One example of this would be Conficker, when introduced via a USB device.

Malicious activities can include further propagation, searching the machine for passwords, personal information, etc.

Maintenance and upgrade includes activities such as improving or adding to the bot’s toolkits, establishing a backdoor, patching the same vulnerability that allowed the original infection in an effort to prevent others from gaining control of the bot and changing the communications channel.

The botnets have become modular [1], with plug-and-play attributes, and show signs of collaborative effort in their design. Conficker was observed in [14] using a Metasploit module to spread itself. The SDBot possesses source code commenting that indicates multiple authors [5].

Bot creators are able to plug in modules to reduce the chances of detection with scripts to detect VMware based honeypots [1, 4]. Botmasters may have bots use low numbers of interactions reaching out to the controller, separate the infection phase from the other phase by time, or use long sleep or dormant cycles to avoid detection by traffic analysis.

Having described how bots and botnets are developed, it is time to discuss methods for learning more about them and combat them.

B. METHODS OF DETECTING, CAPTURING MALICIOUS CODE AND COMBATING BOTNETS

Methods of detecting and preventing infection from botnets have included use of antivirus software, firewalls, and Intrusion Detection Systems (IDS). In order to capture malicious code and improve the combat of botnets, researchers and security experts have progressed to the use or incorporation of honeypots and/or correlation tools.

1. Use of Antivirus and Firewalls

Antivirus software and firewalls rely heavily on *a priori* knowledge of the threats. Antivirus solutions scan the host for malicious software and remove patch or delete it. Firewalls can be preventative—blocking ports, IP addresses, and previously known attack patterns—but essentially they use filters that operate on rule sets, making them reactive in nature. If some knowledge of a specific threat is not known, then they will likely not be effective. Another issue with firewalls is that they are designed for perimeter security. Once an attacker has gotten past the perimeter, their effectiveness in preventing further proliferation within a LAN is almost non-existent. Conficker disables antivirus and firewall software and prevents the host computer from accessing security update Web sites [15].

2. Intrusion Detection Systems

IDS alone generally operate on signature or anomaly recognition; however, IDS predominantly look at inbound packet flows for signs of attacks. The IDS may detect and signal numerous attacks, but do not do well at discriminating between successful and unsuccessful intrusions [3]. The obvious gap in this approach comes from infections initiated by user actions or from an infection that begins internally and initiates outbound connections [3, 15].

3. Forensic Analysis of Network Traffic

Forensic analysis of network traffic can be a lengthy process and generally necessitates knowledge of an infection to justify the investment. In addition, it can include the use of tools to analyze router statistics.

4. Honeypots and Honeynets

A honeypot is defined by Spitzner as “an information system resource whose value lies in unauthorized or illicit use of that resource” [16]. A honeynet is defined as a type of honeypot, made up of a network of computers deliberately designed to be attacked and closely monitored behind a honeywall to capture all network activity. The honeywall is a type of firewall that performs a bridging function and is designed to permit intrusion, but limit outbound connections. The bridging function decreases the likelihood of detection by an attacker by allowing the honeywall to receive, record and drop or forward packets based upon a specified rule set without changing the packets [11]. If researchers want to tailor the honeynet to receive a particular type of attack, the honeywall can be configured to drop or ignore inbound packets that do not meet the established criteria. The outbound connection limiting is performed by simply having the honeywall drop the packets after the recording step. A honeypot or honeynet complements a network security plan, because it provides information in one or all of the following scenarios:

a) A previously unknown attack penetrates the firewall and propagates.

b) Either a Type I or Type II bot is introduced behind the firewall.

The critical piece in any of these cases is the bots propagating at least behind the firewall. Because any activity on a honeypot is malicious by nature, the honeypot provides a means of alarm and information collection to analyze the infecting malware [1]. The honeynet used in this thesis also includes a rootkit, Sebek, which is installed as a client on the individual honeypots. The rootkit hides in the kernel of the honeypot operating system (OS), acting as a keystroke logger and reports activity via UDP packets to the Sebek server that resides on the honeywall [10, 11]. Sebek bypasses the TCP/IP stack, going directly to the Ethernet card, to generate the UDP packets. This prevents the OS from being aware of the packets being sent. As long as every honeypot has Sebek running on it, it is blind to the Sebek packets sent by another Sebek client. The Sebek packets are pushed out onto the local area network (LAN) and are picked up and recorded by the Sebek server at the honeywall [10].

Yet another type of honeypot is a low interaction honeypot called Nepenthes that is specifically designed to acquire executable malware binary for further investigation [9].

5. Correlation Tools

The introduction of correlation engines into botnet research brought a powerful tool to the effort of detection of individual bots and botnets. The challenge for commercial and DoD networks is discovering, retarding or stopping and ultimately preventing a previously unknown method of attack. Several research tools including BotHunter, Botsniffer and Botminer attempt the discovery of previously known and unknown bots and botnets with the use of correlation algorithms [3, 17, 18]. BotHunter uses a combination of botnet behaviors but appears to be the first that includes outbound traffic in its correlation algorithm [3]. Specifically, BotHunter establishes a correlation model that attempts to identify a relationship between intrusion alarm log entries based upon detection of bot infection characteristics, including: inbound scanning, inbound exploit, outbound connections for secondary infection download, outbound attempts to establish command and control, and outbound infection scanning [3]. A correlation score base upon detection and sequencing together of bot infection behavior is generated in order to yield a relative confidence level. The confidence level of a bot detection increases as the correlation engine is able to sequence together more bot intrusion related events. However, bot detection does not require sequencing together or observation of all the bot infection characteristics [3].

While the BotHunter correlation model includes outbound communications patterns, it stands to reason the model should meet with comparable success if the application is run at the gateways of subnets of a larger architecture. For example, tactical networks frequently have a firewall performing perimeter defense, but not between trusted adjacent units. Infection of a Type I bot, attempting to propagate throughout all IP addresses beginning with the same 16 bits (/16 network), could potentially thoroughly infect the network and adjacent units and may even initiate a DDoS before it ever attempts to communicate through one of the firewalls. This leads to the need to consider placement of correlation tools like BotHunter between subnets of a LAN in order to discover bot activity early.

This chapter covered the four phases of creating and maintaining a bot, as well as methods of propagating and controlling a group of bots to form a botnet. It then addressed common approaches to discovery of bots and botnets. This leads to the next step of outlining the model design for this thesis.

III. TEST BED

This chapter will present a tactical network architecture and explain how the proposed test bed is intended to represent parts of the tactical network. The second section of the chapter will give specific details of the test bed implementation.

A. RELATIONSHIP OF TEST BED TO TACTICAL NETWORK ARCHITECTURE

The test bed in this thesis is built to address each of the requirements outlined in Section B, Chapter I. In some cases, a requirement cannot be fully achieved, so risk is accepted and discussed. The overarching objective is to establish an architecture to look at vulnerabilities of what tends to be a homogeneous network. The honeypots built for the test bed network are comprised of Windows operating systems (OS), albeit different versions.

Figure 1 presents a typical network diagram with an addition of the honeynet shown in the dashed box. Ideally honeypots would be dedicated to IP addresses across the architecture and all data redirected and tunneled to a central honeynet below the honeywall, as shown within the dashed box in Figure 1. Honeypots called Collapsar and Potemkin have been developed by the Honeynet Project to implement these types of architectures [9]. Both appear unreasonable to implement on a tactical network either in terms of a significant bandwidth cost or a coordination challenge due to the increased routing complexity.

For Collapsar, packets are sent across a link twice before arriving at or leaving a honeypot. In addition, there is an increase to the packet size as the packet has headers affixed to it in order to effect tunneling. The storm of packets that could follow an infection multiplied by two because of the tunneling architecture could unintentionally assist the malware in creating a denial of service attack.

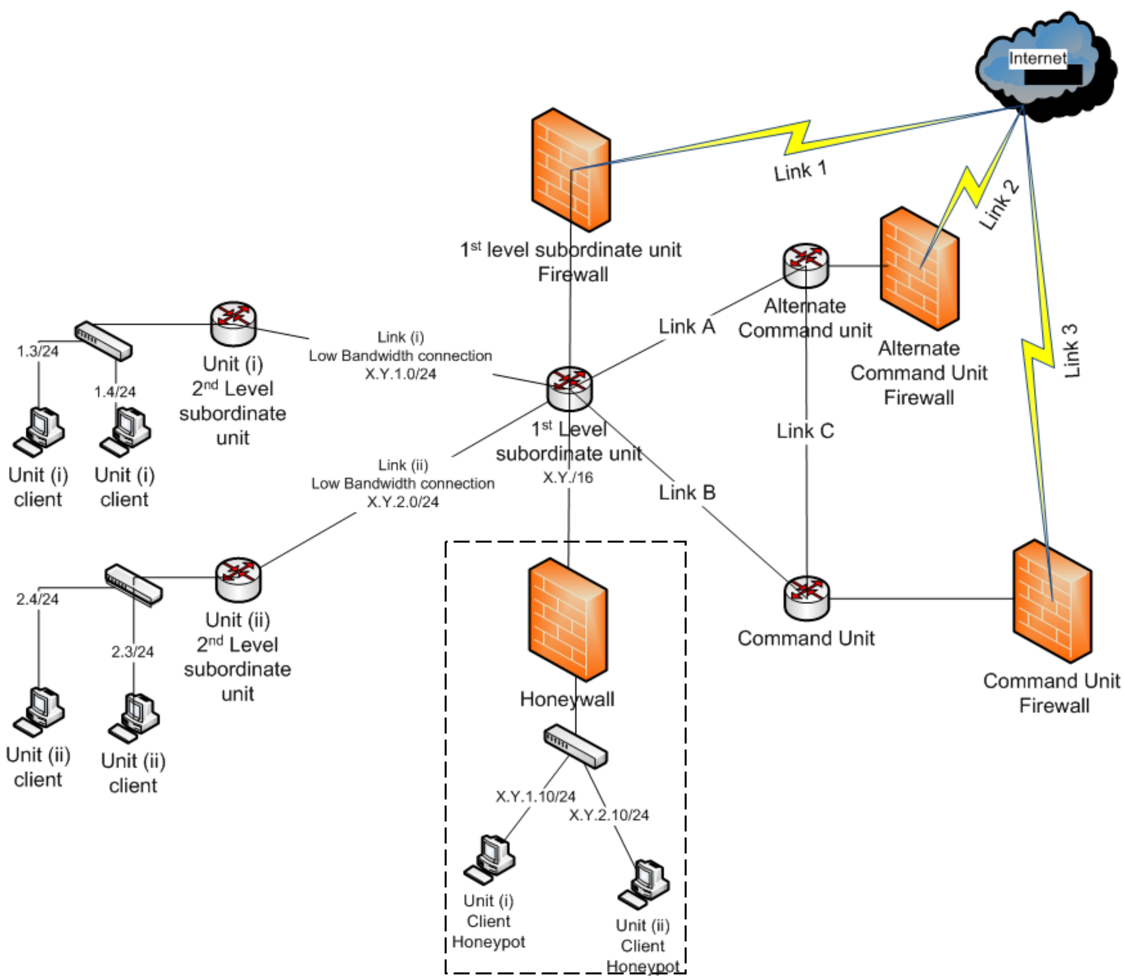

Figure 1. Typical Tactical Network Architecture With Addition of a Honeynet in the Dashed Box

With Potemkin, each router would require routing leaks to be created in order to make the packets destined for IP addresses that should be out at the far left of the architecture directed to behind the honeywall before they ever pass down Link (i) or Link (ii). While this may be detected easily by a traceroute, the real issue that makes this choice undesirable is the complexity of implementation and maintenance on a dynamically changing tactical network.

The test bed network architecture built closely follows that of [11] and is depicted in Figure 2. The test bed is intended to closely resemble a trusted low bandwidth link much like Link (i) or Link (ii) in Figure 1. BotHunter is run on what would appear as a production computer in [11] and not behind a firewall; however, this is actually intended to allow BotHunter to sniff traffic that might be transmitted between trusted subnets. Multiple instances of BotHunter could be run on almost any link, at either end or both ends of a long haul transmission path. Likewise, a single instance could be monitored by administrators at either or both ends.

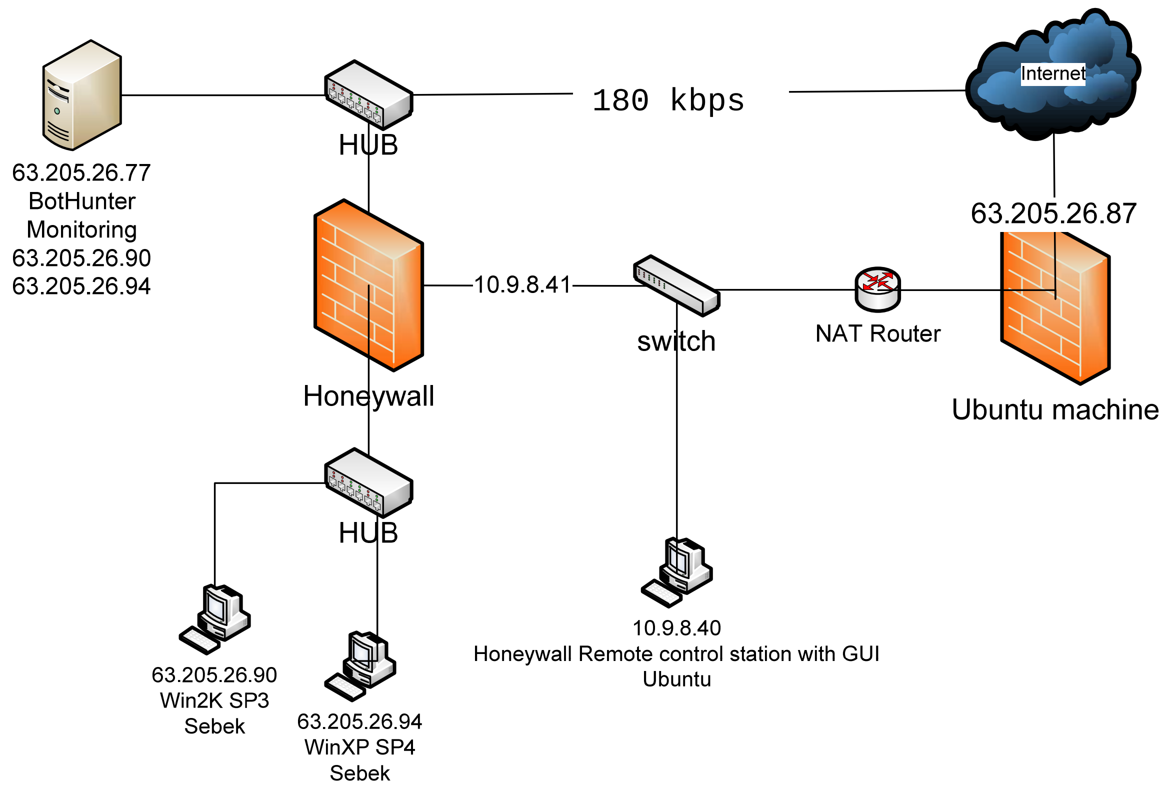

Figure 2. Test Bed Developed to Emulate the Low Data Rate Side of a Tactical Network and Detect bot infections (After [11])

The test bed architecture built also attempts to address the intermittent connectivity by the use of connection rate limiting in the Honeywall rules. As it is common for a tactical network to have losses of connectivity do to relocation or technical problems, the connection limiting performed by the Honeywall will force any bot to perform whatever routines it would do in such an instance. It will also test whether or not BotHunter is able to determine if a bot is still active upon reestablishment of network connectivity. Placement of BotHunter along the low bandwidth links gives an easily administered system that combines the advantages of an IDS with the correlation capability to give a higher probability of detecting and alerting to infection by a previously undefined bot. There is almost no cost in bandwidth efficiency with the exception of Snort updates, which can be scheduled for off-peak times [19]. On the high bandwidth end, placement of a honeynet within the parts of the network with high bandwidth between trusted connections gives an advantage of advanced detection within more stable areas of the network and allows for more complex honeynet design. In these areas of the network, redirecting and tunneling is an option.

The Internet access from a local Internet service provider is rate limited to no more than 180 kbps and not firewalled, as shown in Figure 2. This resembles a tactical network in data rate, but does not resemble a comparable load due to the lack of machines, services and users. Its raw exposure to the Internet is an attempt to accommodate this lack of load and to resemble the internal activity behind a firewall in a homogenous network. If any machine succumbs to an attack and is turned into a Type I bot, almost all machines in the network are equally likely to be compromised and become a Type I botnet before detection. This concern holds regardless of the method of initial infection.

Now that test bed's relationship to a tactical network architecture has been explained, the specifics of the test bed implementation will be detailed next.

B. TEST BED IMPLEMENTATION

The hardware and software to support the test bed pictured in Figure 2, and as described in the previous section, are outlined in this section. Specific OS and software settings are detailed in the Appendix.

A Linksys Etherfast 10/100 MBps hub (model EFAH08W version 2.0) is used to allow monitoring of all traffic in and out of the honeynet by both the honeywall and BotHunter. The choice of a hub with data collision control was made to reduce the loss of packets with a lower cost than a managed switch.

The BotHunter is run on a Dell desktop machine with 2 Gigbytes of RAM. An instance of BotHunter was installed and run on an Ubuntu 8.04 LTS Desktop operating system. The BotHunter version 1.0.2 software download [20], User’s manual [21], and Graphical User Interface manual [22], were all obtained from SRI International’s www.bothunter.net Web site. All BotHunter settings are located in the Appendix. BotHunter was placed outside of the honeywall in order to allow it to get Snort updates and to report to SRI International without affecting the honeywall’s connection limiting functionality. If placed behind the honeywall, the Snort updates and SRI International connections would have counted against the total allowed in and out of the honeynet.

The Honeywall is run on a Dell 2650 with 4 Gigabytes of RAM, 2.4 GHz processor, and three network interface cards (NICs). The Honeywall CDROM, Edition roo, was downloaded from the Honeynet Project and installed as described in [23]. As specified in [23], Ethernet Port 0 faces the Internet and Ethernet Port 1 faces the honeypots in Figure 2. Ethernet Port 2 is assigned an IP address in order to enable it to get Snort updates and to send alert messages and is positioned behind a firewalled Network Address Translation (NAT) router in order to add a level of protection. A remote control station was also used on a Gateway laptop running Ubuntu and connected behind the firewalled NAT router. Specific settings are shown in the Appendix.

The Honeypots consist of two Dell desktop systems with operating systems (OSs) and Sebek client installed on each. The first system is a Dell Dimension 8100 with 512 MB RAM, and 40 GB hard drive. The operating system installed is Windows 2000 Professional with service pack 1 (Win2K SP1). The second honeypot is a Dell Dimension 8400 with 512 MB RAM, and 70 GB hard drive. The operating system installed is Windows XP with service pack 2. In each case, the systems OSs were installed and network addresses assigned prior to Sebek being installed. Sebek was run on each machine, but never saved on the machine to make it more difficult for a bot or bot controller to discover it. All specific settings for the honeypots are in the Appendix but are modeled closely after [10].

A Netgear 6 port 10/100 Mbps dual speed hub, Model DS106, is used to allow monitoring of all traffic between honeypots by the Honeywall. There is a risk of collisions; however, the likelihood of a bot resending or sending additional packets and still being detected is good. An oversight in the network design is the possibility of collisions of Sebek packets with other packets while in route to the Sebek server in the honeywall. The Sebek packets are UDP based packets and are therefore unreliable.

This necessitates a managed switch or hub that can manage data collisions in order to avoid losing packets.

This chapter covered the test bed built for this thesis, the relationship of the test bed to a tactical network architecture and the specific software, operating systems and software settings utilized. The test bed design was tied to a portion of a tactical network as it transitions from high to low bandwidth connections on trusted links behind the network firewalls. The risks associated with the test bed design were also discussed. This leads to the next chapter, which will cover the results.

IV. TEST BED IMPLEMENTAION RESULTS

The intent of this chapter is to provide a proof of concept rather than present experimental results. The results demonstrate the effectiveness of the model design in a live environment. A botnet attack was detected by both the Honeynet and the BotHunter, proving the effectiveness of the design.

This chapter presents the results in four sections. Some overall traffic statistics are presented in Section A. Section B discusses the results from the honeynet, specifically data as captured by the Honeywall. Given the Honeywall captured all packets passing through it, the Honeywall data is used to provide forensic analysis. Section C presents the results as seen from BotHunter. Section D brings together and discusses the two previous sections. Although BotHunter and the Honeywall did not use the same time reference, all times are presented in or will include conversions to Coordinated Universal Time (UTC).

A. TRAFFIC STATISTICS

This section is intended to give some gross traffic statistics collected from the test bed (see Figure 2). It will separate the Sebek packets, as well as point out the common ports and connection types used.

Figure 3 is a screen capture from Ethereal showing the time frame of data collection results with the total amount of data collected at the top. The bottom half gives traffic statistics with Sebek packets and with Sebek packets filtered out in order to give a perspective of the traffic without the artificial inflation of traffic generated by Sebek.

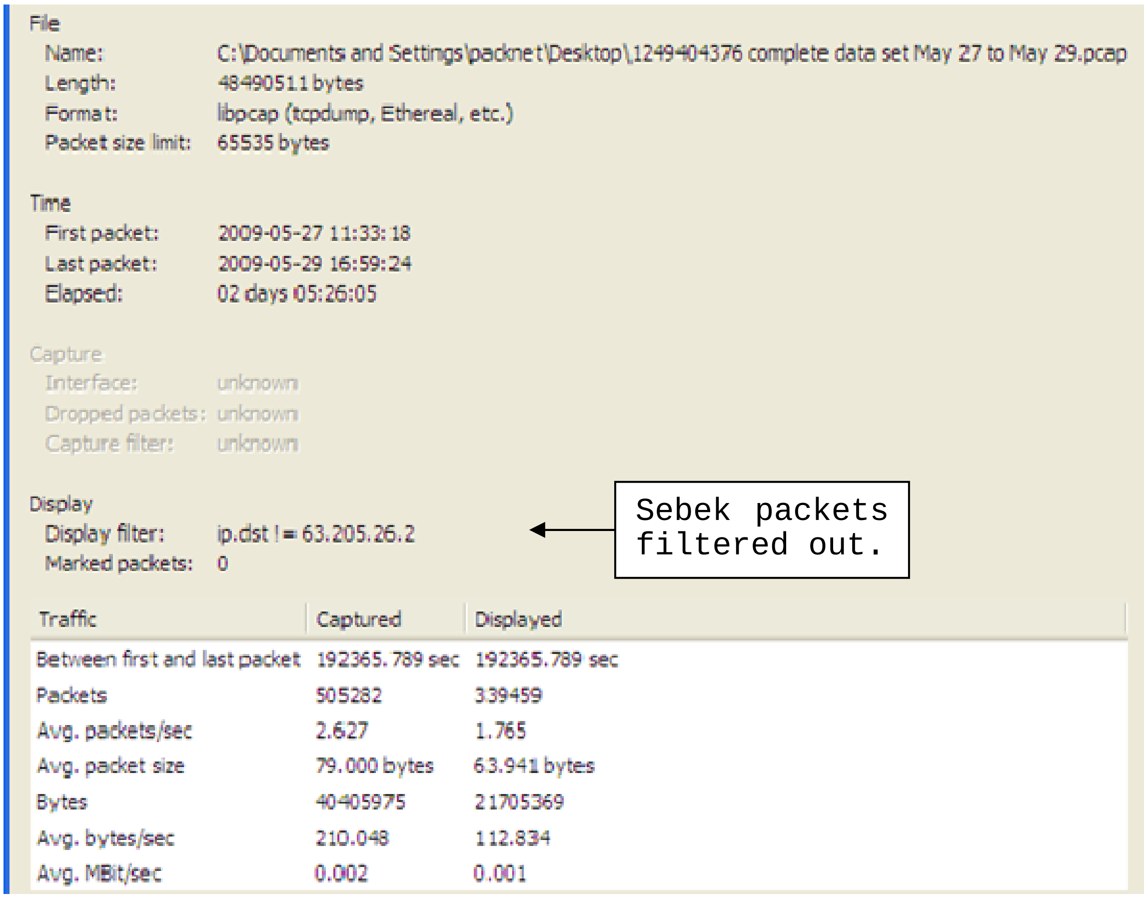

Figure 3. Gross Traffic Statistics With Sebek Packets Included on the Left Side and Filtered Out on the Right

The data collection period was over 2 days, 5 hours, 26 minutes, with the first packet being collected at 11:33 on May 27, 2009 (19:33 UTC) and the last at 16:59 on May 29 (00:59 on May 30 UTC), 2009. A total sent or received in or through the honeynet with Sebek packets filtered, as indicated under the column labeled Displayed in the bottom of Figure 3 was 339,459 packets or 21.7 megabytes.

Figure 4 is another Ethereal screen capture showing the destination ports for packets coming primarily from the honeypots with the Sebek packets filtered out. The exception is 63.205.26.67, which is not in the honeynet, but is later noted as an IP address of interest. The boxes direct attention to the more significant ports used.

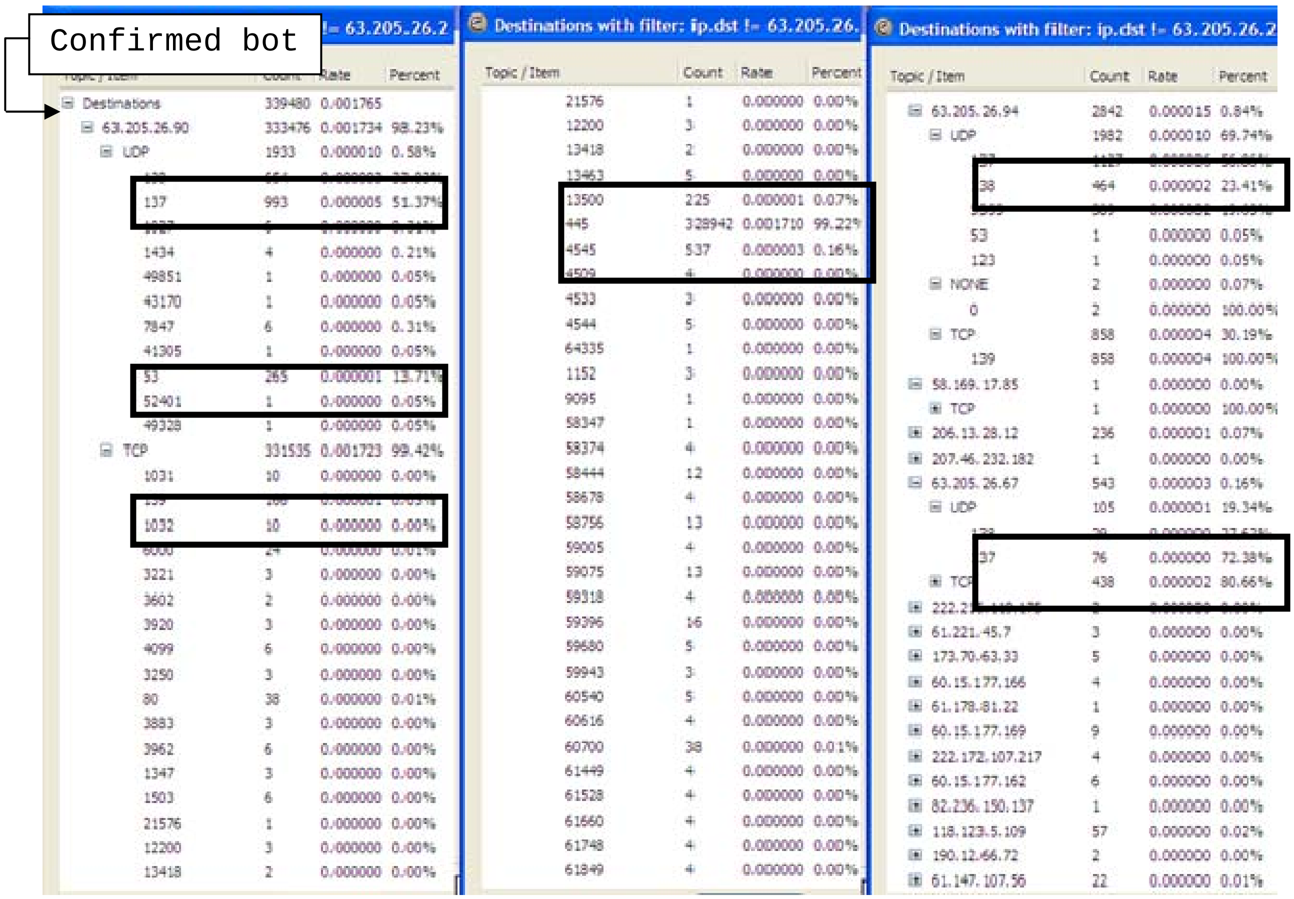

Figure 4. Illustration of Significant Protocols and Ports Used by the Honeypots (IP Addresses 63.205.26.90 and 63.205.26.94) and the 63.205.26.67, Described as an IP Address of Interest

The top center box shows some use of Port 13500/TCP and 4545/TCP packets and almost 329,000 Port 445/TCP packets. This heavy use of Port 445/TCP represents approximately 98 percent of total Sebek excluded traffic. If the Port 445 traffic is eliminated, the other ports of interest are 139/TCP, 137/UDP, 138/UDP and Port 135/TCP. This is consistent with the findings described in [1], although findings from the test bed show a more heavy weighting of Port 445/TCP traffic. The box at top of the right column shows traffic from 63.205.26.94, with interest in UDP Ports 137, 138, and 5355. Although 63.205.26.94 does not trigger a bot alert, the UDP/5355 traffic does suggest possible infection but is inconclusive.

This section has introduced some network traffic statistics and briefly looked at packets and ports used. The Honeywall results in the next section will address a more detailed examination of packets coming to and from the honeynet.

B. HONEYWALL RESULTS

The Honeywall's position, illustrated in Figure 2, in front of the honeypots and on the hub beside BotHunter allowed it to capture all inbound and outbound traffic, as well as the traffic between honeypots. The exception would be any case where a UDP packet collided with another packet. In such an instance, the UDP packet would be lost due to the unreliable nature of UDP. The results presented here show the initial infection or attack, the egg download and other malicious activity.

The Bot phases, as they were detected and alerted on by the Honeywall and BotHunter, are presented in Table 1 for reference in this and following sections. The table differentiates Honeywall results as alerts generated or discovery by forensic analysis. Time is given in UTC and slight variances, less than two minutes, between Honeywall and BotHunter times should be seen as insignificant.

1. Initial Infection

While the Honeywall recorded the initial infection packets, it did not signal an alert. The absence of an alert is a function of the Honeywall's purpose. It is intended to allow attacks in, while recording their actions, and to alert on and slow the outbound propagation of infections.

Figure 5 is a Wireshark screen capture marked to highlight the first instance of a discrepancy in physical address for the WinXP honeypot IP address for the duration of that instance.

Closer inspection of the packets captured shows immediately after an Address Resolution Protocol (ARP) request to resolve the MAC address for IP address 63.205.26.90, the Win2K honeypot, a TCP request is sent from 58.169.17.85 to the WinXP honeypot, IP address 63.205.26.94, with a MAC address of 00:21:9b:79:1d:c1 (pictured in Figure 5 as Dell_79:1d:c1). This MAC address does not belong to the WinXP (63.205.26.94) honeypot and is not used by any other machine in the Honeynet or Bothunter.

		Honeywall		BotHunter
Time (UTC)	**Phases**	**Alert Sent**	**Forensic Analysis**	
	Initial infection			
19:08	Inbound Scan		58.169.17.85 ARP poisoning use 63.205.26.94 as proxy	
19:09			attempted 63.205.26.90 connect to 63.205.26.67	
23:54	Exploit		63.205.26.90 buffer overflow from 63.218.98.110	
	Secondary infection			
23:54	Egg Download	Sebek captures egg download command	63.205.26.90 HTTP .exe download from 212.95.32.104	63.205.26.90 HTTP .exe download from 212.95.32.104
	Malicious activities			
23:54	Outbound Scan	Honeywall msg outbound connection limit reached	63.205.26.90 start 63.205/16 scan	63.205.26.90 outbound scan detected /24
	Maintenance			
23:54	C2 Traffic	Sebek captures connect to ninjawarlord.com	63.205.26.90 bot TCP connect with ninjawarlord.com	
	Peer Coordination			
	Attack Preparation			
23:55:46	Declare Bot	Honeywall msg outbound connection limit reached		63.205.26.90 determined to be bot
23:57:41	Generated Report			report generated, 63.205.26.90 bot

Table 1. Phases of Bot Infection of Win2K (63.205.26.90) Honeypot as Identified From Honeywall and BotHunter.

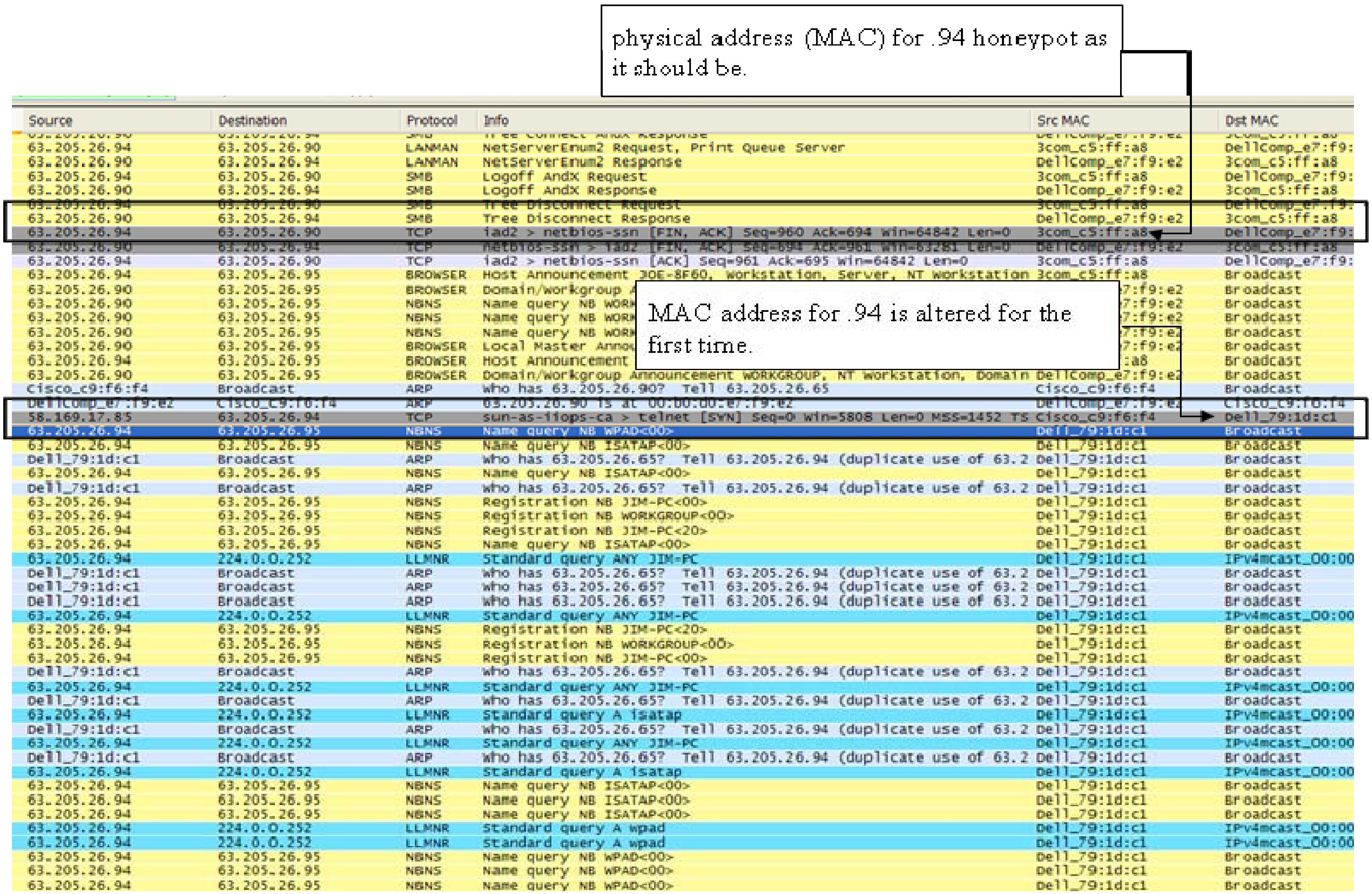

Figure 5. First Sign of ARP Cache Poisoning or MAC Spoofing Involving Honeypot 63.205.26.94 and Collected by the Honeywall

However, the MAC address used for the WinXP honeypot is different from the actual MAC address of 00:60:08:c5:ff:a8. This appears to be a MAC spoofing or ARP cache poisoning.

A description of ARP cache poisoning and MAC spoofing is briefly described below. In the interest of network speed and reduced congestion, the majority of network devices maintain a cache of ARP results identifying the assignment of a MAC address to a specific IP address. Regardless of the IP address assigned to the packets, the MAC address is the next physical destination of the packets in its route. ARP cache poisoning is the process of sending false information in order to replace or submit false MAC addresses for an IP address [24]. MAC spoofing is sending packets with a false or created MAC address that is different from the actual MAC address of the sending machine or receiving machine, thus allowing the machine to impersonate another machine [24].

Different from Figure 5, Figure 6 shows the ARP poisoning occurred on several occasions throughout the experiment. The first column shows the packet numbers, and the second column gives the time of the occurrence of ARP poisoning.

Figure 6 shows the suspicious flow starting at packet number 197. A device masquerading as the WinXP machine with IP address 63.205.26.94, normally MAC address 3com_c5:ff:a8, is sending broadcast messages from MAC address Dell_79:1d:c1. This use of the MAC address for the WinXP honeypot is repeated several times as shown in Figure 6. It should be noted the Honeywall is preventing most of these packets from getting out. Therefore, there will be a large volume of connection attempts that go without response.

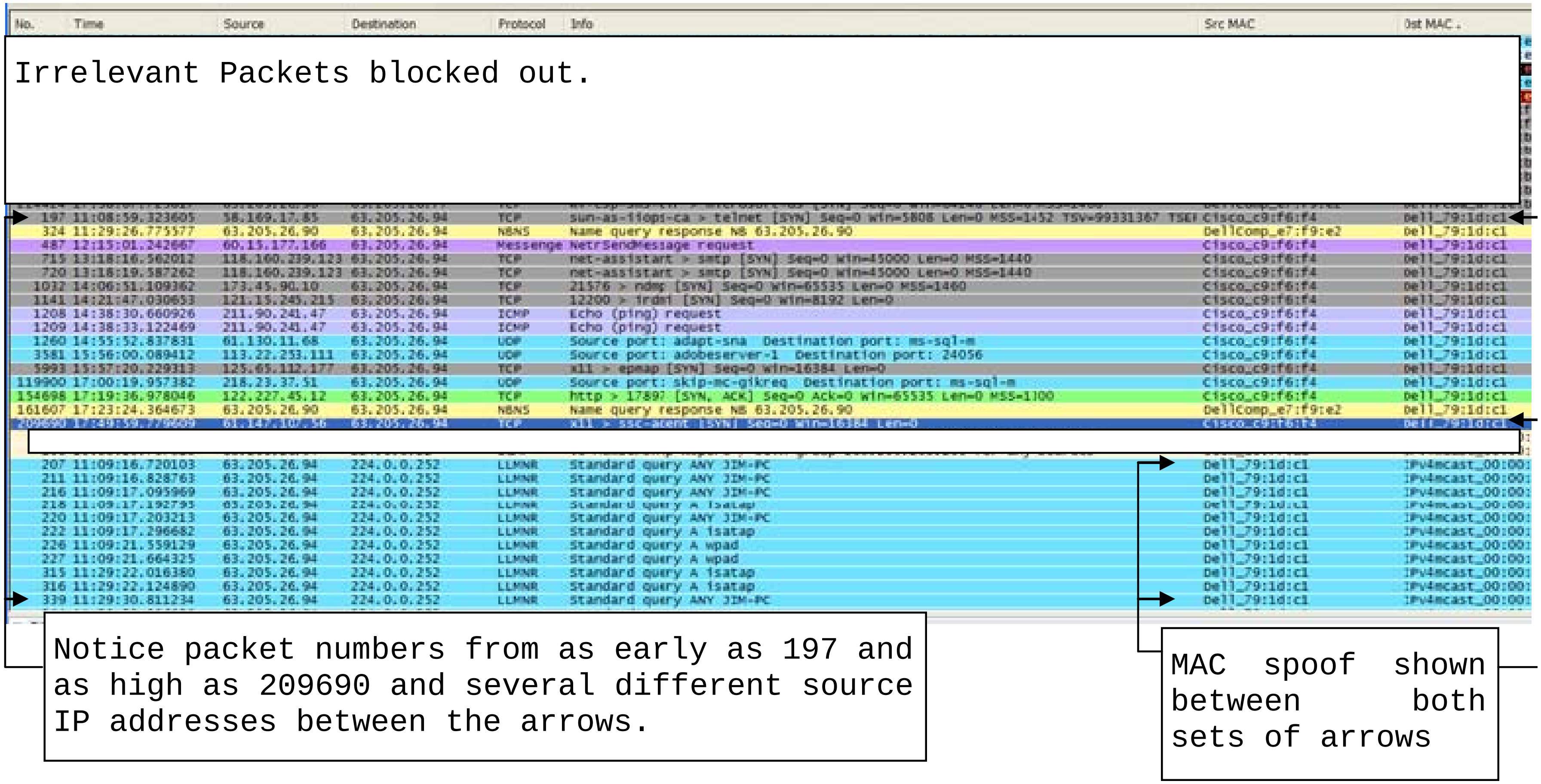

Figure 6. Mac Spoof with packet numbers in first column to illustrate multiple occasions spread across the collection period and could be indicative of a communication Channel or an attack

One possible reason for this is that the machine sending is not 63.205.26.94 but instead another machine on the LAN subnet is MAC spoofing as 63.205.26.94 in an attempt to get past the firewall or to use the MAC address as a communications channel.

Another unusual behavior not shown in Figure 5 or 6 was recorded in packet 249 in which 63.205.26.90 initiates a connection with 63.205.26.67. Prior to this line, there is no known communication between 63.205.26.67 and the honeypots (63.205.26.90 or 63.205.26.94). There is no reason for the honeypots to have knowledge of the existence of 63.205.26.67. They have both sent multiple broadcasts but do not appear to have received any responses from the production side of the honeywall (to include 63.205.26.67).

Figure 7 shows the first clearly observed phase, the initial infection. The initial infection is achieved through an exploit of a vulnerability, which forced a buffer overflow. A buffer overflow and initiation of a subsequent egg download are captured and shown in Figure 7. The buffer overflow exploits a software vulnerability by inputting more data than intended to be received and causing the excess data to be placed into another buffer. This can lead to an attacker gaining access to what would otherwise be restricted code or processes on the computer [24].

Infection of the Win2K honeypot occurs at approximately 2357 UTC on May 27, 2009 (see Figure 7). The attack originates from IP address 63.218.98.110 and is attempted a few times before succeeding.

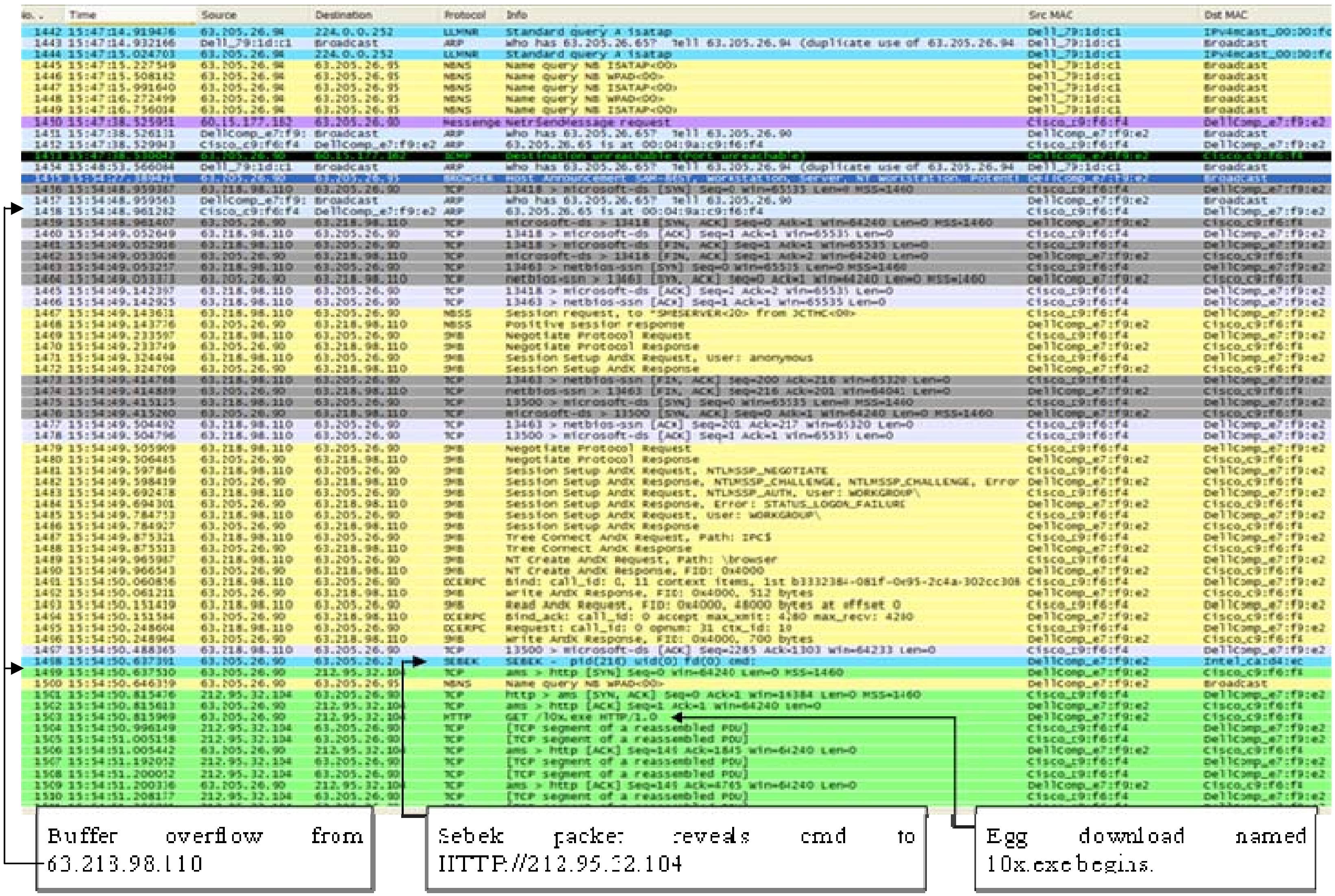

Figure 7. Honeywall Packet Captures Showing Initial Attack via Buffer Overflow, Sebek captures on honeypot 63.205.26.90 and Initiation of Egg Download

2. Secondary Infection

Upon success of the buffer overflow, Sebek captures and reports a command for the honeypot to establish an http (Port 80) connection with IP address 212.95.32.104. This download is an executable named 10x.exe and is shown in Figure 7.

Figure 8 shows the completion of the egg download followed by the beginning of the new bot’s malicious activity. Approximately 6–7 seconds after the download is complete, the honeypot begins what will be a complete Class B (63.205/16) scan.

3. Malicious Activities

Alerts messages, not shown in Figure 7, sent by the Honeywall indicate malicious activity in the form of multiple outbound connection attempts as early as 2354 UTC on May 27, 2009. The Honeywall also sent an alert indicating the maximum number of connection attempts had been exceeded. Forensic analysis shows the Win2K honeypot at 63.205.26.90 begins a full Class B (63.205/16) network enumeration (vulnerability scanning) to include IP addresses internal and external to the 63.205.26.65/27 network containing the honeynet. Sebek packets, shown in Figure 8, also indicated a malicious program on the Win2K honeypot issuing commands. This confirms the Win2K honeypot is infected.

4. Maintenance

Shown in Figure 8, the new bot performs a DNS query to resolve an IP address for ninjawarlord.com and attempts a connection with a response from ninjawarlord.com, a command and control channel for the botmaster.

Not shown in Figure 8, the bot repeats the DNS query every few minutes. In addition, the bot performs a keep alive messages in order to keep a communication channel open to IP address 63.218.98.110. After completing the 63.205/16 network scan, the bot continues to maintain the keep alive messages and to perform the DNS query for ninjawarlord.com. The bot does not appear to meet with any success in propagating; although, this could be heavily influenced by the honeywall’s connection limiting function.

The packets captured by the honeywall, covering May 27 to May 29, 2009, give no evidence of further infections. However, there are numerous additional attempts. The bot’s assignment may be to perform scanning and report to the server.

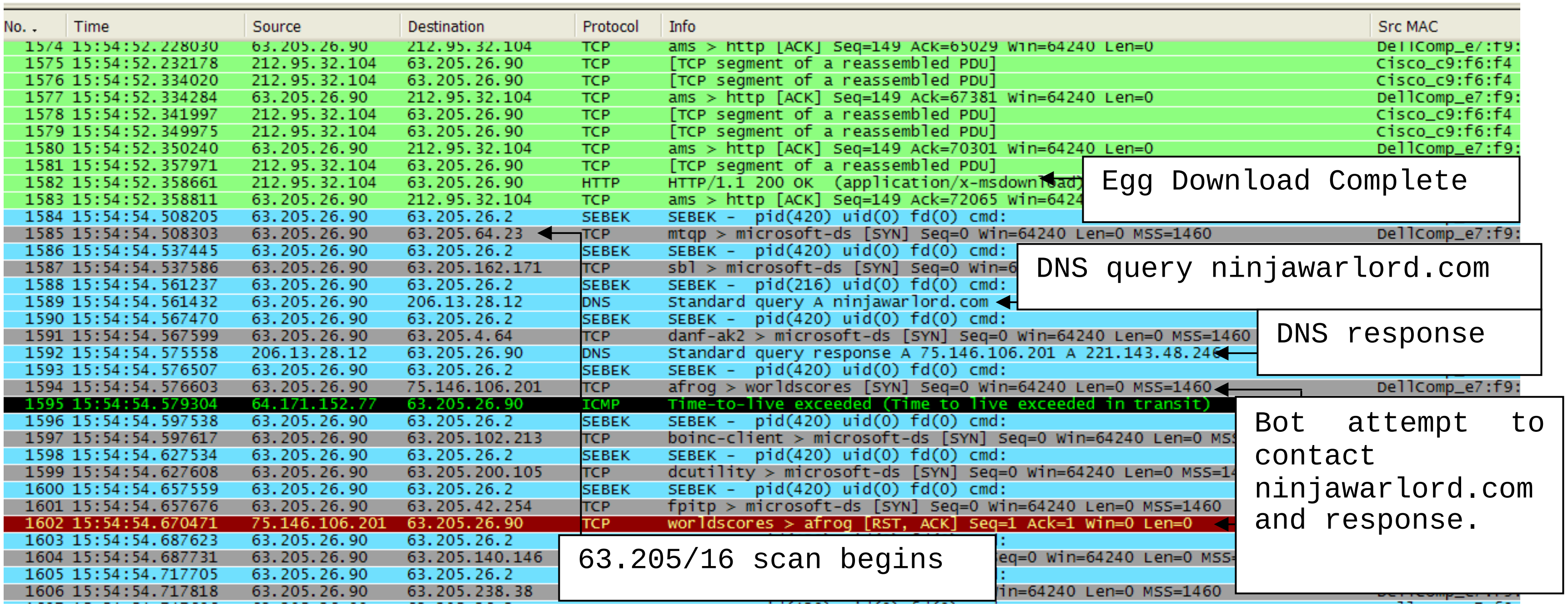

Figure 8. Honeywall Captured Packets Show Egg Download Completion, DNS Query and Response to Resolve an IP Address for ninjawarlord.com in Order to Establish Command and Control, and Initiation of 63.205/16 Network Scanning

The results of performing Wireshark's IPv4 Conversations function, shown in Figure 9, on the captured packets from May 27 to May 29, 2009 yields some information of interest. The goal was to find or confirm infection downloads or C2 channels. Figure 9 shows only the two-way conversations, with the exception of Sebek and broadcast packets that were left in to give some idea of what would be expected in the way of return traffic if the honeywall did not limit the rate of outbound packets. The majority of packets from the 63.205/16 network scan are one-way and, therefore, eliminated from the figure. In some cases, a download or conversation could be one-way and would be overlooked. Conversations are loosely defined as a packet sent to a destination and a packet received from that destination. Some conversations are failed attempts of establishing a connection, whether it be for an exploit, egg download or C2 channel.

5. Honeywall Analysis

Due to possibly encrypted channels, the author draws the conclusion that conversations between the confirmed bot (63.205.26.90) and IP addresses outside of the honeynet subnet are possibly efforts by the bot to check into C2 channels. The expectation for a C2 channel is a relatively low number of outbound (bot initiated) keep alive messages or some other packets sent periodically, spanning a large period of time. Due to the outbound connection rate limitation of the honeywall, a smaller number of responses would be expected than the number of outbound attempts. Shown in Figure 9, conversations that appear to meet this C2 channel description are between: a) 63.205.26.90 and

63.218.98.110, with 465 packets exchanged in 7 hours (66.4 packets per hour (pph)), b) 63.205.26.90 and 118.123.5.109 with 87 packets across a 19.4 hour period (4.5 pph), and c) 63.205.26.90 attempted connections to ninjawarlord.com which DNS query resolves to IP addresses 221.143.48.246 and 75.146.106.201 as previously shown in Figure 8.

Vulnerability scanning or attacks could be characterized by either a barrage of different inbound connections in search of the correct input to trigger a desired response such as a buffer overflow, or periodic inbound unsolicited packets in an attempt to stay below any detection thresholds. This appears to be the case with inbound connection attempts between 121.15.245.215 and 63.205.26.90, with 6 packets exchanged during 18.7 hours (0.32 pph).

An egg download would be characterized by a large amount of data transferred in a short period of time and possibly with large packet sizes. An attempt at a buffer overflow could also be characterized by large packets being sent. If there are multiple attempts at the buffer overflow over a long period of time, it could appear similar to an egg download. An example is shown in Figure 9 by the 63.205.26.90 to 212.95.32.104 conversation, which has already been identified as an egg download. In this instance, a download of approximately 6390 bytes is carried out in a conversation that lasts less than 12 seconds.

Address A	Address B	Packets	Bytes	Packets A->B	Bytes A->B	Packets A<-B	Bytes A<-B	Rel Start	Duration	bps A->B	bps A<-B
63.205.26.90	80.2.69.176	1966	244091	800	81560	1166	162531	36310.712753000	206.3785	3161.57	6300.31
63.205.26.90	63.205.26.94	1034	137809	518	63267	516	74542	1433.103838000	190571.5120	2.66	3.13
63.205.26.90	88.210.85.234	829	63612	285	20052	544	43560	39446.999735000	127.5498	1257.67	2732.11
63.205.26.67	63.205.26.94	793	107782	400	50828	393	56954	3368.013959000	92539.1340	4.39	4.92
63.205.26.90	63.218.98.110	465	30462	232	15373	233	15089	19290.195866000	25349.2985	4.85	4.76
63.205.26.67	63.205.26.90	320	39170	143	19138	177	20032	2893.239255000	34798.6084	4.40	4.61
63.205.26.90	221.143.48.246	286	17724	285	17670	1	54	19378.869633000	16110.0656	8.77	N/A
63.205.26.90	75.146.106.201	246	15228	243	15066	3	162	19295.813082000	16011.0704	7.53	0.08
63.205.26.90	118.123.5.109	87	10427	33	3326	54	7101	8591.459112000	69688.3941	0.38	0.82
63.205.26.90	212.95.32.104	85	76992	31	2004	54	74988	19291.873989000	11.7301	1366.74	51142.21
63.205.26.90	125.65.112.177	33	3368	13	1212	20	2156	19441.468658000	25.4947	380.31	676.53
63.205.26.90	89.149.236.107	6	366	3	180	3	186	12009.331789000	8.9978	160.04	165.37
63.205.26.90	93.43.146.218	6	378	3	180	3	198	44316.029103000	1.7494	823.16	905.47
63.205.26.90	118.160.239.123	6	366	3	180	3	186	9897.763350000	1.3961	1031.45	1065.83
63.205.26.90	121.15.245.215	6	336	2	120	4	216	13675.982692000	67166.4203	0.01	0.03
63.205.26.90	173.70.63.33	6	346	2	122	4	224	5411.859006000	0.9314	1047.92	1924.05
63.205.26.90	212.62.102.221	6	366	3	180	3	186	25778.260548000	9.0692	158.78	164.07
60.15.177.165	63.205.26.90	5	1706	3	1426	2	280	36345.177696000	19419.3322	0.59	0.12
63.205.26.90	63.205.53.55	5	302	4	248	1	54	28589.831619000	3832.7619	0.52	N/A
63.205.26.90	63.205.170.86	5	302	4	248	1	54	25170.070663000	9269.6350	0.21	N/A
63.205.26.90	63.205.153.120	13	798	12	744	1	54	20257.734680000	11848.9216	0.50	N/A
60.15.177.169	63.205.26.90	11	2916	6	2286	5	630	7413.424894000	2570.2888	7.12	1.96
63.205.26.90	63.205.53.83	9	550	8	496	1	54	22615.817014000	9723.6614	0.41	N/A
63.205.26.90	63.205.76.153	9	550	8	496	1	54	27136.801530000	6491.7601	0.61	N/A
60.15.177.162	63.205.26.90	7	1846	4	1496	3	350	18859.762430000	17999.8901	0.66	0.16
63.205.26.90	221.195.73.68	7	384	1	60	6	324	11402.107951000	68883.0079	N/A	0.04
61.221.45.7	63.205.26.90	6	366	3	186	3	180	4722.454665000	2.2285	667.70	646.16
60.194.196.115	63.205.26.90	4	300	2	150	2	150	25357.919661000	0.2750	4363.38	4363.38
60.15.177.166	63.205.26.90	4	1412	3	1342	1	70	6102.502972000	66802.0089	0.16	N/A

Figure 9. Results of Wireshark’s Conversations Function Performed on All Packets Captured Limited to a Minimum of 4 Packets Per Conversation for Inclusion with Sebek Packets and Standard DNS Queries Blocked Out

The downloaded packets were reassembled using Wireshark and submitted to Symantec Corporation's online malware evaluation(https://submit.symantec.com/websubmit/retail.cgi) system to determine if it is a previously known threat. When scanned by Symantec Anti-Virus software, the file is determined to be the W32.Randex worm.

This section of the results looked at packets captured by the honeywall primarily down to the traffic level. It showed the attack, egg download and the new bot's network scanning. The next section will look at BotHunter's results.

C. BOTHUNTER RESULTS

This section presents the results of BotHunter and discusses the bot findings. The reader should recall that BotHunter is run in front of the Honeywall and not behind a firewall. In addition, a correlation score based upon the detection sequencing together of bot infection behavior is generated in order to render a relative confidence level. A score between 0.8 and 3.8 is required to trigger a bot declaration, with a higher correlation score indicating greater confidence [22].

Figure 10 presents a screen shot of the BotHunter GUI, with multiple declared bots listed under "Victim IP." The bots are sorted in order of correlation scores and not with respect to when they were declared, labeled as "Gen time."

The Bothunter indicated its first bot detection at approximately 2357 UTC on May 27, 2009 on the Win2K (IP 63.205.26.90) honeypot (see Figure 2). The Bot declaration was based upon detection of a HTTP based executable download (egg download) and subsequent outbound scanning of first 10 and then 21 IP addresses in a /24 network within 5-7 seconds of the egg dowload. This does not contradict the Honeywall results above, rather it shows BotHunter’s sensitivity is high enough that it triggered a bot detection before the scanning went beyond the first /24 subnet of the larger 63.205/16 network. The egg download was detected coming from IP address 212.95.32.104 with a correlation score of 1.8. Information about the inbound network scan or type of exploit used to gain access to the 63.205.26.90 honeypot is not shown by BotHunter. This is likely due to BotHunter not being located behind and firewall. During setup, BotHunter requires a setting to indicate whether or not it is behind a firewall. When it is not located behind a firewall, its sensitivity to inbound attacks is decreased.

The second bot declaration is shown with a correlation score of 1.3 but with less information. Not shown in Figure 10, this declaration is based exclusively on the detection of intense network IP address and port scanning originating from the 63.205.26.90 honeypot.

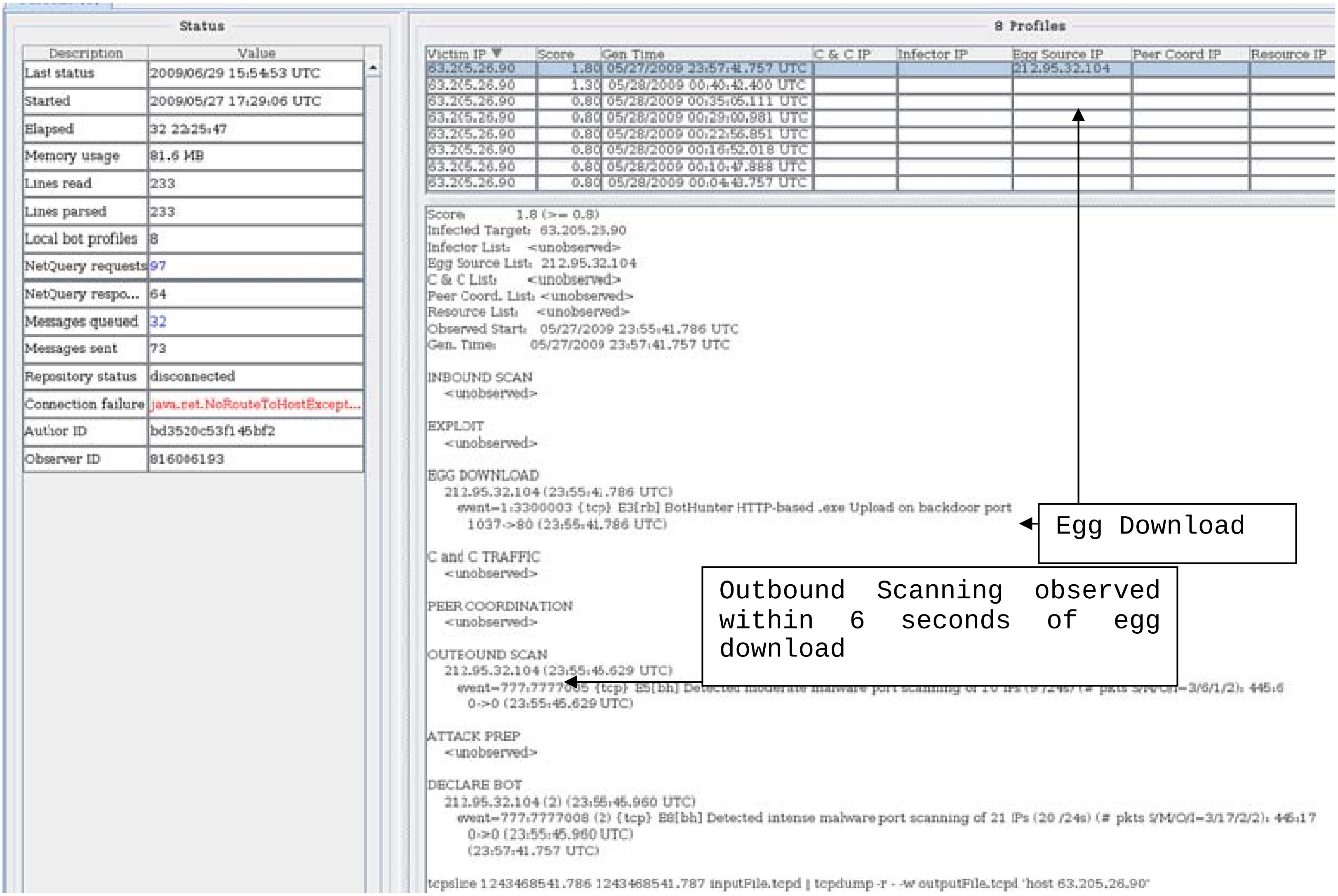

Figure 10. BotHunter Screen Shot Illustrating Multiple Detections of a Bot on 63.205.26.90

Additional bot detections are declared by Bothunter at 00:04 UTC on May 28, 2009 and at approximately 6 minute intervals thereafter until 00:40 UTC May 28, 2009. Of note, almost all of the declarations barely meet the 0.80 correlation score. These additional bot detections do not include the egg download or any other information other than intense outbound IP address and port scanning. The additional bot detections are likely repeat declarations of the same bot infection, given that the subnets (not shown) scanned are all within the larger 63.205/16 network. The absence of peer coordination or C2 information could be a result of channel encryption.

Albeit brief, this section covered the BotHunter's bot detection. The results clearly indicate a single bot infection and suggest that the additional declarations are due to additional scanning of subnets within the 63.205/16 network. The next section will combine the results from the honeynet and BotHunter.

D. HONEYNET AND BOTHUNTER RESULTS COMBINED

This section will focus on the synthesis of an overall result, using a fusion of BotHunter and honeynet results from this low data rate network.

The review of traffic and Sebek packets captured by the honeywall did verify the first bot declaration by BotHunter. The Sebek packets recovered from the Sebek server on the honeywall also identified the egg download command and verified the egg source IP, as reported by BotHunter, immediately following the buffer overflow that was not detected by BotHunter.

A closer look at the packets captured with the honeywall showed a full 63.205/16 network scan with what could be multiple attempts at command and control connections imbedded within the scanning. Although none were detected by BotHunter, two honeywall results suggested the existence of C2 channels. The first suggestion of C2 channels was shown in the keep alive messages to IP address 63.218.98.110, the same IP address that originated the successful buffer overflow. As seen in Figure 9, this connection was maintained over a 7 hour period. The second suggestion of C2 channels is seen in several other conversations, shown in Figure 9, to last over a period of several hours. Yet another way these C2 channels maybe seen is in the 63.205.26.90 honeypot outbound connection attempts to IP addresses outside of the 63.205/16 network scan.

In summary, this chapter gives honeynet results, followed by BotHunter results and then combines the two for a more complete picture. The honeynet traffic shows a cost of approximately 100 bytes per second for the bot infection. A closer look at the packets captured by the honeynet's honeywall shows the attack and subsequent egg download, confirming the egg download and source detected by BotHunter. It does not provide any clarity for why Bothunter did not detect the attack. The honeynet traffic analysis supports the idea that the additional BotHunter declarations with low correlation scores are repeated detections of the same infection. The need for a honeynet within higher bandwidth areas of the tactical network is supported by the fact that the honeynet collects all packets and enables better analysis than BotHunter.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS

This thesis developed a test bed for the detection of botnet infections at the low data rate end of tactical networks. The test bed was developed with the use of BotHunter and a honeynet. BotHunter is a tool that employs a correlation algorithm, intrusion detection system definitions and characteristics of basic botnet behavior. The honeynet is included in order to emulate results that would be seen in parts of a tactical network that operate at higher data rates. The results of the test bed validated the effectiveness of BotHunter for botnet detection in low bandwidth areas of tactical networks and the usefulness of honeynet employment within tactical networks where connections with higher data rates exist.

A. SIGNIFICANT RESULTS

The most significant result of the test bed is the successful validation of the test bed architecture as a means of detecting botnet infections in low data rate networks such as those found in tactical environments. BotHunter continued to detect the bot infection after the periodic loss of connections caused by the honeynet’s connection rate limits. In addition, BotHunter detected a type of bot infection that could be particularly hazardous to tactical networks. The origination of the bot secondary infection and its malicious action of performing a Class B network scan were detected within a matter of minutes.

Traffic analysis of all packets captured by the honeywall allowed determination of the network cost in terms of malicious traffic caused to the test bed by the single bot infection. The traffic cost for the particular bot infection captured in this thesis was measured to be 112 Bytes per second.

The requirement for positioning of a honeynet or honeynets within the test bed network architecture is validated by BotHunter's failure to capture all of the bot behavior, such as the attacking IP address. The honeynet captured all the bot infection phases, including those not seen by BotHunter, for a previously known bot infection. Use of BotHunter does come with some risk of failure to detect a previously unseen bot attack technique. Employment of a honeynet in the higher data rate areas of a network mitigate the risk of a missed bot detection by providing depth and greater information. As explained by Spitzner in [16], activity on a honeypot is by definition suspicious and likely to be malicious.

With a relatively low sensitivity setting, BotHunter successfully detected and reported a bot infection within six minutes of the initial infection on a data rate limited connection of 180 kbps. BotHunter further provided the ability to detect a bot infection without an additional traffic cost as seen by use of the rootkit, Sebek, with the Honeynet.

B. FUTURE WORK

Previous research has looked at bot/botnet detection within well established high data rate networks; this thesis differs from previous research by focusing on characteristics of tactical networks. Tactical networks are characterized by low data rate connections and periodic losses of connectivity. A progression for future work would be to add complexity in a manner that more closely resembles a tactical network.

1. Employ a Honeynet Consisting of a Homogenous Network of Honeypots

The test bed was designed with a non-homogeneous honeynet consisting of two honeypots with one of each operating system, Win2K and WinXP. The method of attack in the initial infection by a bot is generally based upon a specific operating system or other software vulnerability that is likely to change between version releases. A tactical network would typically have a high degree of homogeneity, with the majority of computers having the same operating systems with similar level of updates and antiviral signatures. The test bed should be modified to include multiple instances of any operating system (and other software) versions of interest in order to observe propagation of bot infections and more closely resemble a tactical network.

2. Position BotHunter Between Subnets

The BotHunter was positioned outside of the honeynet and was not behind a firewall. Successful propagation of the bot is not seen by BotHunter. The new location for BotHunter should be behind a firewall and between trusted production subnets (with no firewall between them) on a tactical network or between separate honeynets.

This can be done by establishing two honeynets on separate subnets under a common /16 network, both with separate firewalled access to the Internet, with a non-firewalled link between honeynets. Such a network would allow for better simulation of a tactical network and to further test whether BotHunter could be used as an early warning tool at the low data rate end of the network. In addition, multiple bot infections could be deliberately introduced behind the honeynets in order to observe bot behavior.

3. Addition of a Malware Collection Tool

The Honeywall’s use of Sebek for collection of malware is unreliable because Sebek uses UDP. Without the reliability, collisions, dropped or missed packets for any number of reasons can result in a loss of malware binary. In addition, BotHunter does not provide the capability to collect the actual malware code/files. To fix this, the test bed could be modified to include Nepenthes. The reader is reminded Nepenthes is a malware collection tool that can be setup to collect the malware and pass the malware to a central collection point for analysis [9].

APPENDIX. EQUIPMENT AND SOFTWARE SETTINGS

A. HONEYWALL

1. Honeywall CDROM Root Install

root password: !#79RuuB4me

roo password: Victory1/5! / !#79RUUBme5

Note: Port and IP address numbers are separated by spaces. Do not include colons, semicolons or commas.

TCP allowed out (port numbers): 22 25 43 80 443

UDP allowed out: 53 123

Connection limiting set to: hour

TCP limit: 24   UDP limit: 23

ICMP limit: 57   Other protocols: 14

Honeypot IPs: 63.205.26.90 63.205.26.94

CIDR: 63.205.26.64/27   Broadcast: 63.205.26.95

Management Interface (Walleye) settings

Management Interface IP: 10.9.8.41

Mask: 255.255.255.0   Default Gateway: 10.9.8.1

System host name: localhost   domain name: localdomain

DNS server IPs: 206.13.28.12 206.13.29.12

Configure SSH: yes

Let root login remote: no

Manage intereface allow inbound ports: 443

Allow IP to login to management interface: 10.9.8.40

Web interface for analysis: yes

Restrict firewall outbound comm: yes

SNORT_Inline: yes

Blacklist: (none)

Whitelist: (none)

Black/white list filtering enable: Yes

Disable “strict” capture filtering: no

Fencelist location: /etc/fencelist.txt (IP addresses and CIDR blocks

Enable Fencelist filtering: no

Enable “Roach motel” mode: no

DNS: unlimited

Limit Honeypot unlimited access to DNS: no

Restrict DNS server: no

Email alerts: yes

Email address: insert your email address here

Alert start auto @ : yes

SEBEK

Dest IP Sebek packets: 63.205.26.2

Dest Port: 1101

Sebek Var: Accept and Log

Oink Code is needed for Snort. Go to Snort Web site to login and request an Oink Code. https://www.snort.org/login

B. HONEYPOTS

1. Windows 2000, Service Pack 3

a. Wipe Hard Drive

First run Hard Drive wiping utility, such as Derik's Boot and Nuke, from http://dban.org in order to clean the hard drive and its boot sectors.

b. Insert and Run Install of Win2k SP3

Delete any partitions

Perform long format

Computer name: Sam

Organization: Sam Group

Product Key: Enter your product key here

Computer Name: Sam-86ST

Admin Password: !#79R()CK!

Choose Typical settings

Check Workgroup option

IP: 63.205.26.90   Mask: 255.255.255.224

Gateway: 63.205.26.65

DNS: 206.13.28.12 206.13.29.12

c. Sebek Install

Run sebek from disk, so that it is never copied onto the hard drive.

Driver name: Sebek

Destination MAC: (MAC Address of NIC 1, inward facing NIC of Honeywall) 00:02:B3:CA:D4:EC

IP: 63.205.26.2

Port: 1101

Magic Value: 3289080092

Configuration program name: services25v

Sam’s dog Password >f1D0! F!d0

Mutt

Unregmp2.exe

Admin Password: !#79R()CK!

Guest: p@ssing!

2. Windows XP, Service Pack 2

a. Wipe Hard Drive

Perform Hard drive Wipe as in Win2k paragraph 1.a.

b. Insert and Run Install of WinXP SP2

Delete all partitions

Perform long format

Name: Joe

Organization: Joe Group

Computer Name: JOE-8F60

Admin Password: C0rn4@a11

Select Typical settings

Select Workgroup

Static IP: 63:205.26.94 Mask: 255.255.255.224

Default Gateway: 63.205.26.65

DNS: 206.13.28.12 206.13.29.12

Create users:

Username	user type	password
Bobby	admin	T1ght@ss
Sue	limited	1L1keU2

c. Sebek Install

Run sebek from disk so that it is never copied onto the hard drive.

Driver name: Sebek

Destination MAC: (MAC Address of NIC 1, inward facing NIC of Honeywall) 00:02:B3:CA:D4:EC

IP: 63.205.26.2

Port: 1101

Magic Value: 3289080092

Configuration program name: services25v

C. BOTHUNTER

Bothunter is installed per the instructions in [21] and the graphical user interface (GUI) instructions in [22]. In this configuration, BotHunter is not run behind a firewall, requiring entry into the custom configuration option per [21].

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] The Honeynet Project. Know your enemy: Tracking botnets (accessed January, 2008), http://www.honenet.org/book/export/html/50, 2005.

[2] A. Karasaridis, B. Rexroad, and D. Hoeflin, "Wide-scale botnet detection and characterization." In Proceedings of the 1st Workshop on Hot Topics in Understanding Botnets, April 2007, Cambridge, MA, USA (accessed February 19, 2009), http://www.usenix.org/events/hotbots07/tech/full_papers/karasaridis/karasaridis.pdf.

[3] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "Bothunter: Detecting malware infection through ids-driven dialog correlation." In Proceedings of the 16th USENIX Security Symposium (Security '07), pp 167-182, August 2007, Boston, MA, USA, (accessed February 19, 2009), http://www.usenix.org/events/sec07/tech/full_papers/gu/gu_html/index.html.

[4] Z. Zhu, G. Lu, Y. Chen, Z.J. Fu, P. Roberts, and K.Han, "Botnet Research Survey." In Annual IEEE International Computer Software and Applications Conference, pp 967-972, July 2008, Turku, Finland.

[5] E. Cooke, F. Jahanian, and D. McPherson, "The zombie roundup: Understanding, detecting and disrupting botnets." In 1st Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI), pp 39-44, July 2005, Cambridge, MA, USA (accessed February 19, 2009), http://www.usenix.org/event/sruti05/tech/full_papers/cooke/cooke.pdf.

[6] T. Holz, M. Steiner, F. Dahl, E. W. Biersack, And F. Freiling, "Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm." In LEET'08: First USENIX Workshop on Large-Scale Exploits and Emergent Threats, April 2008, San Francisco, CA, USA (accessed April 2009), http://www.usenix.org/event/leet08/tech/full_papers/holz/holz_html/.

[7] The Honeynet Project. Know your enemy: Fast-flux service networks (accessed July 13, 2009), http://www.honeynet.org/book/export/html/130.

[8] M. Rajab, J. Zarfoss, F. Monroe, and A. Terzis, "A multi-faceted approach to understanding the botnet phenomenon." In Proceedings of ACM SIGCOMM/USENIX Internet Measurement Conference (IMC'06), Brazil, October 2006.

[9] N. Provos and T. Holz, *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*, .Addison-Wesley, Upper Saddle River, New Jersey, 2008.

[10] The Honeynet Project. Know your enemy: Sebek, A kernel based data capture tool (accessed July 2009), http://www.honeynet.org//papers/sebek, November 17, 2003.

[11] The Honeynet Project. Know your enemy: Honeynets (accessed February 2009), http://old.honeynet.org/papers/honeynet/, May 31, 2006.

[12] B. Shirley and C.D. Mano, "A model for covert botnet communication in a private subnet" in NETWORKING 2008, LNCS 4982, pp. 624–632, 2008.

[13] B. Zdrnja, "Conficker's autorun and social engineering," Handler's Diary, SANS Internet Storm Center; Cooperative Network Security Community (accessed January 15, 2009). http://isc.sans.org/diary.html?storyid=5695

[14] X. Chen, "Conficker Worm using Metasploit payload to spread," Computer Security Research–McAfee Avert Labs Blog (accessed January 15, 2009), http://www.avertlabs.com/research/blog/index.php/2009/01/.

[15] "The Downadup Codex: A comprehensive guide to the threat's mechanics" edition 1.0 in Symantec Security Response (accessed March 13, 2009). http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/the_downadup_codex_ed1.pdf.

[16] L. Spitzner, “Honeypots: Definitions and value of honeypots,” http://www.tracking-hackers.com, May 29, 2003 (accessed January 2009), http://www.tracking-hackers.com/papers/honeypots.html.

[17] G. Gu, J. Zhang, W. Lee, ”Botsniffer: Detecting botnet command and control channles in network traffic.” In Proceedings of the 2008 ISOC Network and Distributed System Security Symposium, February, 2008, San Diego, CA, USA (accessed February 19, 2009), http://www.isoc.org/isoc/conferences/ndss/08/papers/17_botsniffer_detecting_botnet.pdf.

[18] G. Gu, R. Perdisci, J. Zhang, and W. Lee, “Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection.” In Proceedings of the USENIX Security Symposium, pp 139-154, August 2008, San Jose, CA, USA (accessed April 2009), http://www.usenix.org/events/sec08/tech/full_papers/gu/gu_html/index.html.

[19] Sourcefire Inc. (accessed March 2009), http://www.snort.org/.

[20] SRI International. BotHunter unix distribution version 1.0.2. (accessed March 2009), http://bothunter.net.

[21] SRI International. BotHunter User Manual, Version 1.0.2, Unix release (accessed March 2009, http://www.bothunter.net/doc/users_guide-UNIX.html.

[22] SRI International. BotHunter Graphical user interface user manual, Version 1.0.2. (accessed March 2009), http://www.bothunter.net/doc/gui.html.

[23] The Honeynet Project. Honeywall CDROM Roo 1.2 User’s Manual (accessed March 2009), http://yum.honeynet.org/roo/manual/1-overview.html.

[24] Jon Erickson. *Hacking: The Art of Exploitation*. No Starch Press, 2nd ed., 2008.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. John G. Kato
   Naval Information Operations Command Suitland
   Suitland, Maryland

4. John T. Scott
   Naval Information Operations Command Suitland
   Suitland, Maryland

5. Marine Corps Representative
   Naval Postgraduate School
   Monterey, California

6. Director, Training and Education
   MCCDC, Code C46
   Quantico, Virginia

7. Director, Marine Corps Research Center
   MCCDC, Code C40RC
   Quantico, Virginia

8. Marine Corps Tactical Systems Support Activity
   (Attn: Operations Officer)
   Camp Pendleton, California

9. Professor John McEachen
   Naval Postgraduate School
   Monterey, California

10. Professor Murali Tummala
    Naval Postgraduate School
    Monterey, California