

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

Approved for public release; distribution is unlimited

	CUMENTATION PAGE	_		
			orm Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leav	ve blank) 2. REPORT DATE September 200		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 6. AUTHOR(S) Russell W. Becker 			. FUNDING NUMBERS	
<pre>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000</pre>			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		10	10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
			is are those of the author and artment of Defense or the U.S.	
12a. DISTRIBUTION / AVA	ILABILITY STATEMENT	12	2b. DISTRIBUTION CODE	
Approved for public rele 13. ABSTRACT (maximum 20	ease; distribution is unli	mited		
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.				
significant threat t presents a network and of malicious activit limited environment significant burden to A test bed is of driven correlation connections, rather to in a high data rate architecture to ident	rchitecture with the ob- ty and its propagation with periodic losses the network. constructed that makes u tool, BotHunter, fo than solely on inbound area of a tactical net tify malicious activitie	tactical jective of within of co se of an cused of connecti work. T es is val	of near real-time detection a data rate (bandwidth) nnectivity without adding intrusion detection system on outbound and inbound ons and a honeynet located he ability of the proposed lidated when both BotHunter	
significant threat t presents a network an of malicious activit limited environment significant burden to A test bed is of driven correlation connections, rather to in a high data rate architecture to ident and the Honeynet succ 14. SUBJECT TERMS	rchitecture with the ob- ty and its propagation with periodic losses the network. constructed that makes u tool, BotHunter, fo than solely on inbound area of a tactical net tify malicious activitie	tactica jective of within of co se of an cused of connecti work. T es is val nfection	of near real-time detection a data rate (bandwidth) nnectivity without adding intrusion detection system on outbound and inbound ons and a honeynet located the ability of the proposed lidated when both BotHunter 15. NUMBER OF PAGES	
significant threat t presents a network and of malicious activity limited environment significant burden to A test bed is of driven correlation connections, rather to in a high data rate architecture to ident and the Honeynet succe 14. SUBJECT TERMS	rchitecture with the obj ty and its propagation with periodic losses the network. constructed that makes u tool, BotHunter, fo than solely on inbound area of a tactical network tify malicious activities cessfully detect a bot i	tactica jective of within of co se of an cused of connecti work. T es is val nfection	of near real-time detection a data rate (bandwidth) nnectivity without adding intrusion detection system on outbound and inbound ons and a honeynet located the ability of the proposed lidated when both BotHunter 15. NUMBER OF PAGES	
significant threat t presents a network and of malicious activity limited environment significant burden to A test bed is of driven correlation connections, rather to in a high data rate architecture to ident and the Honeynet succe 14. SUBJECT TERMS Botnet, Tactical Network	rchitecture with the obj ty and its propagation with periodic losses the network. constructed that makes u tool, BotHunter, fo than solely on inbound area of a tactical network tify malicious activities cessfully detect a bot i	tactical jective of n within of consection se of an cused of connection work. These is values infection neypot, Long 19. SEC	of near real-time detection a data rate (bandwidth) nnectivity without adding intrusion detection system on outbound and inbound ons and a honeynet located he ability of the proposed lidated when both BotHunter 15. NUMBER OF PAGES 79 16. PRICE CODE URITY ICATION OF ABSTRACT	

Prescribed by ANSI Std. 239-18

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

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.

v

TABLE OF CONTENTS

I.	INTRO	DDUCTION1
	Α.	MOTIVATION2
	в.	OBJECTIVES
	c.	RELATED WORK
	D.	ORGANIZATION6
II.	OVER	/IEW OF BOTNETS7
	Α.	CREATING AND MAINTAINING A BOTNET7
	в.	METHODS OF DETECTING, CAPTURING MALICIOUS CODE AND
		COMBATING BOTNETS9
		1. Use of Antivirus and Firewalls10
		2. Intrusion Detection Systems10
		3. Forensic Analysis of Network Traffic11
		4. Honeypots and Honeynets11
		5. Correlation tools12
III.	TEST	BED
	А.	RELATIONSHIP OF TEST BED TO TACTICAL NETWORK
		ARCHITECTURE
	в.	TEST BED IMPLEMENTATION19
IV.	тъст	BED IMPLEMENTAION RESULTS
T A A	A.	TRAFFIC STATISTICS
	в.	HONEYWALL RESULTS
		1. Initial Infection
		 Secondary Infection
		3. Malicious Activities
		4. Maintenance
		5. Honeywall Analysis
	c.	BOTHUNTER RESULTS
	D.	HONEYNET AND BOTHUNTER RESULTS COMBINED44
v.	CONCI	LUSIONS
••	A.	SIGNIFICANT RESULTS
	в.	FUTURE WORK
		1. Employ a Honeynet Consisting of a Homogenous
		Network of Honeypots
		2. Position BotHunter Between Subnets
		3. Addition of a Malware Collection Tool50
אישרות ג	ттv	EQUIPMENT AND SOFTWARE SETTINGS
APPEr	A.	HONEYWALL
	п.	1. Honeywall CDROM Root Install
	в.	HONEYPOTS
	5.	1. Windows 2000, Service Pack 3
		T. MINGOWS 2000, BELVICE FACK 3

	a.	Wipe Hard Drive53
	b.	Insert and Run Install of Win2k SP353
	c.	Sebek Install
	2. Wind	ows XP, Service Pack 254
	a.	Wipe Hard Drive54
	b.	Insert and Run Install of WinXP SP254
	C.	Sebek Install
C.	BOTHUNTER	
LIST OF	REFERENCES .	
INITIAL	DISTRIBUTIO	N LIST61

LIST OF FIGURES

Figure	1.	Typical Tactical Network Architecture With Addition of a Honeynet in the Dashed Box16
Figure	2.	Test Bed Developed to Emulate the Low Data Rate Side of a Tactical Network and Detect bot infections (After [11])
Figure	3.	Gross Traffic Statistics With Sebek Packets Included on the Left Side and Filtered Out on the Right
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
Figure	5.	First Sign of ARP Cache Poisoning or MAC Spoofing Involving Honeypot 63.205.26.94 and Collected by the Honeywall
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 attack32
Figure	7.	Honeywall Packet Captures Showing Initial Attack via Buffer Overflow, Sebek captures on honeypot 63.205.26.90 and Initiation of Egg Download
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 Scanning37
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 Out40
Figure	10.	BotHunter Screen Shot Illustrating Multiple Detections of a Bot on 63.205.26.9043

LIST OF TABLES

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 xiii 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. honeynet Implementation of a 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.

xiv

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.

xv

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 In recent research, Holz et al. met with some success [5]. in disrupting the communications of a P2P botnet [6]. Fastflux 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 of old collecting unknown exploits 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. Manv computers are still left exposed to the Internet without a firewall, giving attackers easy access. Even with а firewall, there are other methods by which the attacker can qain 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 eqq 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 eqq 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, IΡ previously known addresses, and 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 siqnal numerous attacks, but do not do well at successful discriminating between 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 The bridging function decreases the likelihood connections. 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]. Τf 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 The outbound connection limiting is established criteria. performed by simply having the honeywall drop the packets after the recording step. А 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 for exploit, outbound connections 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 because of the tunneling architecture could two 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 of use 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. A11 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 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.

File Name: Length: Format: Packet size limit:	48490511 libpcap (to	Lbytes cpdump, Ethereal,		o\1249404376 complete data set	: May 27 to May 29.pcap
Time First packet: Last packet: Elapsed:	2009-05-	29 16:59:24			
Capture Interface: Dropped packets: Capture filter:	unknown				_
Display Display filter: Marked packets:	-	63.205.26.2	•	Sebek packets filtered out.	
Traffic		Captured	Displayed		
Between first and I Packets Avg. packets/sec Avg. packet size Bytes Avg. bytes/sec Avg. MBit/sec	ast packet	505282	192365.789 sec 339459 1.765 63.941 bytes 21705369 112.834 0.001		

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.

Confirmed	200	Rate	Percent	Topic / Item	Count	Rate	Percent	Topic / Item	Count	Rate	Deuters
Destinations	339480	0.001765		21576	1	0.000000			2120325		Percen
■ 63.205.26.90		0.001734	Sector and	12200	3	0.000000	0.05.00000	63.205.26.94	2842	0.000015	
E UDP	1933	0.000010		13418	2	0.000000	0.00%	E UDP	1982	0.000010	69.74
	100	0.000000		13463	5	0.000000	0.00%				
137	993	0.000005	51.37%	13500	225	0.000001	0.07%	38	464	0.000002	23.4
1007			0.717	445	328942	0.001710	99.229			0.0000000	0.00
1434	4	0.000000	0.21%	4545	\$37	0.000003	0.16%	53 123	1	0.0000000	
49851	1	0.000000	0.05%	4509	4	0.000000	0.00%	E NONE	2	0.0000000	
43170	1	0.000000	0.05%	4533	3	0.000000	0.00%	0	-	0.0000000	
7847	6	0.000000	0.31%	4544	5	0.000000	0.00%	E TCP	858	0.0000004	
41305	1	0.000000	0.05%	64335	1	0.000000	0.00%	139	858	0.0000004	
53	265	0.000001	13.71%	1152	3	0.000000	0.00%	E 58,169, 17.85	1	0.0000000	
52401	1	0.000000	0.05%	9095	1	0.000000	0.00%	TCP	1	0.0000000	
49328	1	0.000000	0.05%	58347	1	0.000000	0.00%	E 206.13.28.12	236	0.000001	
I TCP	331535	0.001723	99.42%	58374	4	0.000000	0.00%	207.46.232.182	1	0.0000000	
1031	10	0.000000	0.00%	58444	12	0.000000	0.00%	E 63.205.26.67	543	0.000003	
+33	100	0.000001	0.0318	58678	4	0.000000	0.00%	E LOP	105	0.000001	
1032	10	0.000000	0.00%	58756	13	0.000000	0.00%	172	20	0.000000	
6000	24	0.000000	0.01%	59005	4	0.000000		37	76	0.000000	72.3
3221	3	0.000000	0.00%	59075	13	0.000000	0.00%	TCP	438	0.000002	
3602	2	0.000000	0.00%	59318	4	0.000000	0.00%	E 222.21 110 175			
3920	3	0.000000	0,00%	59396	16	0.000000	0.00%	61.221.45.7	3	0.000000	0.00
4099	6	0.000000	0.00%	59680	5	0.000000	0.00%	173.70.63.33	5	0.000000	0.00
3250	3	0.000000	0.00%	59943	3	0.000000	0.00%	■ 60, 15, 177, 166	4	0.000000	0.00
80	38	0.000000	0.01%	60540	5	0.000000	0.00%	Ⅲ 61.178.81.22	1	0.000000	
3883	3	0.000000	0.00%	60616	4	0.000000	0.00%	E 60.15.177.169	9	0.000000	0.00
3962	6	0.000000	0.00%	60700	38	0.000000		■ 222.172.107.217	4	0.000000	0.00
1347	3	0.000000	0.00%	61449	4	0.000000	1000	B 60.15.177.162	6	0.0000000	
1503	6	0.000000	0.00%	61528	4	0.000000		82,236, 150, 137	1	0.000000	0.00
21576	1	0.000000	0.00%	61660	4	0.000000	1.1.1.1.1.1.1.1	H 118.123.5.109	57	0.000000	
12200	3	0.000000	0.00%	61748	4	0.000000		III 190.12.66.72	2	0.000000	
13418	2	0.000000	0.00%	61849	4	0.000000	0.00%	IB 61.147.107.56	22	0.0000000	0.01

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) for the MAC address IΡ address request to resolve 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.

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

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

physical address (MAC) for .94 honeypot as it should be.

,					
Source	Destination	Protocol		Src MAC	Dst MAC
63.205.26.94	63.205.26.90	LANMAN	IT EE CONTECC ANUA NESPONSE	3com_c5:ff:a8	DellComp_e7:f9:
					3com_c5:ff:a8
63.205.26.90	63.205.26.94	LANMAN		DellComp_e7:f9:e2 3com_c5:ff:a8	
63.205.26.94	63.205.26.90	SMB SMB			DellComp_e7:f9 3com_c5:ff:a8
63.205.26.90	63.205.26.94	5416		DellComp_e7:f9:e2	Delicomp_e7:f9
63.205.26.90	63.205.26.94	SMB		DellComp_e7:f9:e2	3com_c5:ff:a8
63.205.26.94	63.205.26.90	TCP		3com_c5:ff:a8	DellComp_e7:f9:
63.205.28.90	63.205.26.94	TCP		Delicomp_e/:f9:e2	SCON_CS:TT:a8
63.205.26.94	63.205.26.90	TCP		3com_c5:ff:a8	DellComp_e7:f9
63.205.26.94	63.205.26.95	BROWSER	Host Announcement JOE-8F60, Workstation, Server, NT Workstation		Broadcast
63.205.26.90	63,205,26,95	BROWSER	Domain/workgroup A	7:f9:e2	Broadcast
63.205.26.90	63.205.26.95	NBNS	NAME AND ADDRESS ADDRESS	7.50	Broadcast
63.205.26.90	63.205.26.95	NBNS	Name query NB WORM MAC address for .94 is altered for	the 7:f9:e2	Broadcast
63.205.26.90	63.205.26.95	NENS	Name query NB WORK	7:19:e2	Broadcast
63.205.26.90	63.205.26.95	BROWSER	Local Master Annos first time.	7:f9:e2	Broadcast
63.205.26.94	63.205.26.95	BROWSER	Host Announcement	188	Broadcast
63.205.26.90	63.205.26.95	BROWSER	Domain/Workgroup Announcement WORKGROUP, NT Workstation, Domain (Broadcast
Cisco_c9:f6:f4	Broadcast	ARP		cisco_c9:f6:f4	Broadcast
beilicomp_e/:r9:e2	C15C0_C9:10:14	AKP		berncomp_e/:19:ez	C15C0_C3:10:14
58.169.17.85	63.205.26.94	TCP	sun-as-fiops-ca > telnet [SYN] Seq=0 win=5808 Len=0 MSS=1452 TS (Dell_79:1d:c1
63.205.26.94	63.205_26.95	NEINS		Dell_79:1d:c1	Broadcast
63.205.26.94	63.205.26.95	NEINS		Dell_79:1d:c1	Broadcast
Dell_79:1d:c1	Broadcast	ARP	who has 63.205.26.65? Tell 63.205.26.94 (duplicate use of 63.2)		Broadcast
63.205.26.94	63.205.26.95	NBINS		Dell_79:1d:c1	Broadcast
Dell_79:1d:<1	Broadcast	ARP	who has 63.205.26.65? Tell 63.205.26.94 (duplicate use of 63.21		Broadcast
63.205.26.94	63.205.26.95	NBINS		Dell_79:1d:c1	Broadcast
63.205.26.94	63.205.26.95	NBNS	Registration NB WORKGROUP<00>	Dell_79:1d:c1	Broadcast
63.205.26.94	63.205.26.95	NEINS		Dell_79:1d:c1	Broadcast
63.205.26.94	63.205.26.95	NBNS		Dell_79:1d:c1	Broadcast
63.205.26.94	224.0.0.252	LLMNR	Standard query ANY JIM-PC	Dell_79:1d:c1	IPv4mcast_00:0
Dell_79:1d:cl	Broadcast	ARP	who has 63.205.26.65? Tell 63.205.26.94 (duplicate use of 63.21		Broadcast
Dell_79:1d:c1	Broadcast	ARP	who has 63.205.26.65? Tell 63.205.26.94 (duplicate use of 63.21	Dell_79:1d:c1	Broadcast
Dell_79:1d:c1	Broadcast	ARP	who has 63.205.26.65? Tell 63.205.26.94 (duplicate use of 63.21	Dell_79:1d:c1	Broadcast
63.205.26.94	224.0.0.252	LLMNR		De11_79:1d:c1	IP-v4mcast_00:0
63.205.26.94	63.205.26.95	NBNS	Registration NB JIM-PC<20>	Dell_79:1d:c1	Broadcast
63, 205, 26, 94	63, 205, 26, 95	NBNS		Dell_79:1d:c1	Broadcast
63.205.26.94	63.205.26.95	NBNS		Dell_79:1d:c1	Broadcast
Dell_79:1d:c1	Broadcast	ARP	who has 63.205.26.65? Tell 63.205.26.94 (duplicate use of 63.21	Dell 79:1d:c1	Broadcast
63, 205, 26, 94	224.0.0.252	LLMNR		pell 79:1d:c1	IPv4mcast 00:0
Dell_79:1d:c1	Broadcast	ARP	who has 63.205.26.65? Tell 63.205.26.94 (duplicate use of 63.21		Broadcast
63.205.26.94	224.0.0.252	LLMNR		Dell_79:1d:c1	IPv4mcast_00:0
Dell_79:1d:c1	Broadcast	ARP	who has 63.205.26.65? Tell 63.205.26.94 (duplicate use of 63.21		Broadcast
63.205.26.94	224.0.0.252	LLMNR		Dell_79:1d:c1	IPv4mcast_00:0
Dell_79:1d:c1	Broadcast	ARP	who has 63.205.26.65? Tell 63.205.26.94 (duplicate use of 63.21		Broadcast
63.205.26.94	224.0.0.252	LLMNR		Dell_79:1d:c1	IPv4mcast_00:0
63.205.26.94	63.205.26.95	NBNS	Name query NB ISATAP<00>	Dell_79:1d:c1	Broadcast
63.205.26.94	63, 205, 26, 95	NBNS		Del1_79:1d:c1	Broadcast
63.205.26.94	63.205.26.95	NBNS		Dell_79:1d:cl	Broadcast
		LLMNR			IPv4mcast_00:0
63.205.26.94	224.0.0.252			Dell_79:1d:c1	
63.205.26.94	224.0.0.252	LLMNR		Del1_79:1d:c1	IPv4mcast_00:0
63.205.26.94	63.205.26.95	NBNS		Dell_79:1d:c1	Broadcast
63.205.26.94	63.205.26.95	NBNS		Del1_79:1d:c1	Broadcast
63.205.26.94	63.205.26.95	NENS	Name query NB wPAD<00>	Dell_79:1d:c1	Broadcast

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 machine, thus allowing the or receiving 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.

		Protocol	b/o	Src MAC	Ost MAC +
D 1	1.7.1				
Packets	B DIOCKE	ea ou	τ.		
38.169.17.85	63.205.26.94	TCP	sun-as-110p1-ca > telnet [SYN] Seq=0 win=5808 Len=0 MSS=1452 TSV=9	9331367 TSEF C15C0_C9:16:14	pel1_79:1d:
63.205.26.90	63,205,26,94	NBNS	Name query response N8 63.205.26.90	DellComp_e7:f9:e2	Del1_79:1d:
					pell_79:1d: pell_79:1d:
		TCP	net-assistant > smtp [SYN] Seg=0 win=45000 Len=0 MSS=1440	C15C0_C9:f6:f4	0e11_79:1d:
173.45.90.10	63.205.26.94	TCP	21576 > ndmp (SYN) Seq=0 win=65535 Len=0 M55=1460	Cisco_c9:f6:f4	0e11_79:1d:
					Del1_79:1d:
			Echo (ping) request		Del1_79:1d: Del1_79:1d:
61, 130, 11, 68	63,205,26,94	UDP	Source port: adapt-sna Destination port: ms-sol-m		pel1_79:1dt
113.22.253.111		UDP	Source port: adobeserver-1 Destination port: 24056	Cisco_c9:f6:f4	pell_79:1d:
	63,205,26,94	TCP	x11 > epmap [SYN] Seq=0 win=16384 Len=0	Cisco_c9:f6:f4	Dell_7911dt
					pel1_79:1d:
					Dell_79:1d: Dell_79:1d:
		TCP	x11 > ssc-acent (Syn) Seg=0 win=16384 Len=0	C1sco_c9:t6:t4	0e11_79:10:
63.205.26.94	224.0.0.252	LEMNR	Standard query ANY JIM-PC	Del1_79:1d:cl	IPv4mcast_0
					IPv4scast_0
					IPv4mcast_0 IPv4mcast_0
63.205.26.94	224.0.0.252	LLMNR	Standard guery ANY JIM-PC	Del1_79:1d:cl	IPv4mcast_0
63.205.26.94	224.0.0.252	LLMNR	Standard query A 1satap	Del1_79:1d:cl	IPv4mcast_0
					1Pv4scast_0
63.205.26.94 63.205.26.94	224.0.0.252 224.0.0.252	LLMNR	Standard query A wpad Standard query A isatap	Dell_79:1d:cl Dell_79:1d:cl	IPv4mcast_0 IPv4mcast_0
	224.0.0.252	LLMNR	Standard query A Isatap		
63.205.26.94	224.0.0.252	LLMNR		Del1_79:1d:c1	IPv4ecast_0
	63.205,26.90 60.15.177.166 60.15.177.166 118.160.239.123 118.160.239.123 118.160.239.123 118.160.241.47 61.130.11.68 113.22.233.111 125.65.112.177 216.23.25.12 63.205.26.90 63.205.26.94 63.205.26.94 63.205.26.94 63.205.26.94 63.205.26.94 63.205.26.94 63.205.26.94 63.205.26.94 63.205.26.94 63.205.26.94 63.205.26.94 63.205.26.94	63, 205, 26, 90 63, 205, 26, 94 118, 140, 239, 123 63, 205, 26, 94 118, 140, 239, 123 63, 205, 26, 94 118, 140, 239, 123 63, 205, 26, 94 121, 15, 245, 215 63, 205, 26, 94 121, 90, 241, 47 63, 205, 26, 94 121, 90, 241, 47 63, 205, 26, 94 125, 253, 111 63, 205, 26, 94 125, 253, 111 63, 205, 26, 94 125, 26, 94 124, 00, 252 63, 205, 26, 94 124, 00, 252 137, 254, 94 124, 00, 252 137, 255, 26, 94 137, 224, 00, 252 137, 255, 26, 94 137, 224, 00, 252 137, 255, 26, 94 147, 224, 00, 252 137, 255, 26, 94 147, 224, 00, 252 147, 257, 257, 257, 257, 257, 257, 257, 25	63.205.26.90 63.205.26.94 N845 60.15.17.7166 63.205.26.94 Messenge 118.160.239.123 63.205.26.94 TCP 118.140.239.123 63.205.26.94 TCP 118.140.239.123 63.205.26.94 TCP 121.15.245.215 63.205.26.94 TCP 121.90.241.47 63.205.26.94 TCP 211.90.241.47 63.205.26.94 UCP 113.22.253.111 63.205.26.94 UCP 123.25.253.111 63.205.26.94 UCP 123.52.253.111 63.205.26.94 UCP 123.52.253.111 63.205.26.94 UCP 123.52.253.111 63.205.26.94 UCP 125.52.37.51 63.205.26.94 UCP 125.25.26.94 63.205.26.94 UCP 125.25.26.90 63.205.26.94 UCP 125.25.26.94 224.00.252 LLMMR 63.205.26.94 224.00.252 LLMMR 63.205.26.94 224.00.252 LLMMR 63.205.26.94 224.00.252 LLMMR 63.205.26.94 224.00.252 LLMMR 63.205.26.94 224.00.252 LLMMR 63.205.26.94 224.00.252 LLMMR	63.205,26.90 63.205,26.94 NBAS Name query response NB 63.205,26.90 60.15.177,166 63.205,26.94 NESSenge NetrSendMestage request 118.160,239,123 63.205,26.94 TCP net-assistart > Smtp [SYN] Seq=0 Win=45000 Len=0 MSS=1440 118.160,239,123 63.205,26.94 TCP net-assistart > Smtp [SYN] Seq=0 Win=45000 Len=0 MSS=1440 113.45.90.10 63.205,26.94 TCP 11276 > ndmg [SYN] Seq=0 Win=8192 Len=0 121.15,245,215 63.205,26.94 TCP 12200 > indmi [SYN] Seq=0 Win=8192 Len=0 121.90.241.47 63.205,26.94 ICP Echo (ping) request 01.130.11.68 63.205,26.94 UCP Echo (ping) request 01.130.11.68 63.205,26.94 UCP Source port: adopt=sna Destination port: ms=sql=m 113.222.53.111 63.205,26.94 UCP Source port: adopt=sna Destination port: 24056 125.65.112.177 63.205,26.94 UCP Source port: sksp=mc-gikreq Destination port: ms=sql=m 122.222.45.12 63.205,26.94 UCP Source port: sksp=mc-gikreq Destination port: ms=sql=m 122.222.65.90 63.205,26.94 NBAS Name query response NS 63.205,26.90 000 UP#1040050 13000500000000000000000000000000000	63.205.26.90 63.205.26.94 NBMS Name query response NS 63.205.26.90 DellComp.e7:19:e2 60.15.177.166 63.205.26.94 Nessenge NetTSendMessage request C1sco.29:16:14 118.160.239.123 63.205.26.94 TCP net-assistart > smtp [SYN] Seq=0 win=45000 Len=0 MSS=1440 C1sco.29:16:14 118.160.239.123 63.205.26.94 TCP net-assistart > smtp [SYN] Seq=0 win=45000 Len=0 MSS=1440 C1sco.29:16:14 121.45.25.215 63.205.26.94 TCP 112:00 > indm [SYN] Seq=0 win=45000 Len=0 MSS=1440 C1sco.29:16:14 121.90.241.47 63.205.26.94 TCP 12:00 > indm [SYN] Seq=0 win=45000 Len=0 MSS=1460 C1sco.29:16:14 211.90.241.47 63.205.26.94 TCP 12:00 > indm [SYN] Seq=0 win=8192 Len=0 C1sco.29:16:14 211.90.241.47 63.205.26.94 UCP Echo (ping) request C1sco.29:16:14 211.90.241.47 63.205.26.94 UCP Source port: adobeserver-1 bestination port: ms-sql=m C1sco.29:16:14 123.22.23.111 63.205.26.94 UCP Source port: skip=mc-gikreq Destination port: ms-sql=m C1sco.29:16:14 123.22.27.35 63.205.26.94 TCP kit > epaap [SN] Seq=0 win=6384 Len=0 C1sco.29:16:14

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 63.205.26.67 known communication between and no 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.

io	Time	Source	Destination	Protocol	2rfo	Sirc MAC	Dist MAC
	15:47:14.919476	63.205.26.94	224.0.0.252	LUMINE	Standard guery A isatap	Del1_79:1d:c1	IPv4ecast_00:00:f
1443	15:47:14.932166	Dell_79:1d:c1	Broadcast	ARP	who has 63.205.26.657 Tell 63.205.26.94 (duplicate use of 63.205.26.94	Del1_79:1d:c1	Broadcast
1444	15:47:15.024703 15:47:15.227549	63.205.26.94	224.0.0.252 63.205.26.95	ELMNR.	standard query a isatap	Dell_79:1d:cl Dell_79:1d:cl	1Pv4ecast_00:00:f
1445	15:47:15.508182	63.205.26.94 63.205.26.94	63.205.26.95	NENS NENS	Name query NS ISATAP(00)	Dell_79:1d:cl	Broadcast Broadcast
	15:47:15.991640	63,205,26,94	63,205,26,95	MINE	Name query NB WPAD<00> Name query NB ISATAP<00>	Del1_79:1d:c1	Broadcast
	15:47:16.272499	63.205.26.94	63.205.26.95	NENS	Name query NE WPAD <do></do>	Dell_79:1d:cl	Broadcast
1449	15:47:16.756014	63.205.26.94	63.205.26.95	NINS	Name query NE ISATAP-00>	Del1_79:1d:cl	Broadcast
	15:47:38.525011	60.15.177.162	63.205.26.90	Pessenge	e NetrSendlessage request	C1sco_19:f6:f4	DellComp_e7:f9:e2
1411	15:47:38.526131	Dellconp_e7:f9:	Broadcast	AD.P	who has 63.205.26.657 1e11 63.205.26.90	Dellcorp_e7:f9:e2	Broadcast
1412	15:47:38.529943	C1sto_c9:f6:f4	DellComp_e7:f9:e2	2 (ARP)	63.205.26.65 is at 00:04:9a:c9:16:14	cisco_t9:f6:f4	DellComp_e7:f9:e2
1.11.1	15:48:53.566084	Gell_79:1d:t1	Broadcast	42.9	who has 63.205.26.657 Tell 63.205.26.94 (duplicate use of 63.205.26.94	Dellicomp e71191e2	Broadcast
12317	ISSISTER OF BUILDEDING	61.705.20.90	63.205.26.95	COLUMN TWO IS NOT	Host Arrows emert Sam Afst, workstation, Server, bt workstation, Potenti	Del Condite Al Lite	Eroadcast
1436	15:54:48,939367	63,218,98,110	63.205.26.90	TCP	Host Announcement Sam Ador, Norestation, Server, NT workstation, Potenti 13418 > microsoft-ds [SYN] Seq=0 win+65333 Len+0 MSS=1440	C15c0_191f6:f4	DellComp_e7:f9:e2
	15:54:48.959563	Dellcomp_e7:f9:	Broadcast	ARP	who has 63.205.26.657 Tell 63.205.26.90	DellComp_e7:f9:e2	Broadcast
1418	15:54:48.961282		Dellcomp_elif9:el		63.205.26.65 is at 00:04:9a:c9:16:f4	cisco_c9:f6:f4	DellComp_e7:f9:e2
	15154148.961407	63.205.26.90	63,218,98,110	XP	microsoft-ds > 11418 [SYN, ACK] Seq+0 Ark=1 Win+64240 Ltn+0 MSS+1460	DellConp_e71f91e2	Cisco_c9:f6:f4
1400	15:54:49.052649	63,218,98,110	63.205.26.90	TCP TCP	13418 > microsoft-ds (ACK) Seq=1 ACK=1 vin=65533 Len=0 13418 > microsoft-ds (FIN, ACK) Seq=1 ACK=1 win=65533 Len=0	C1sco_r9:f6:f4 C1sco_r9:f6:f4	DellComp_e7:f9:e2 DellComp_e7:f9:e2
1442	15:54:49.053006	63.205.26.90	63.218.98.110	TCP.	microsoft-ds > 13418 [FIN, ACK] Seg-1 Ark+2 Win-64240 Lin+0	DellComp_e7:f9:e2	Cisco,c9:f6:fi
	15:54:49.053217	63.218.95.110	63.205.26.90	TCP .	13463 > netbios-ssn [svs] Seg=0 win=65535 Len=0 #55=1460	Cisco_r9:f6:f4	pellcomp_e7:f9:e2
	15:54:49.033373	63.205.26.90	03.218.95.110	TC#	netbios-ssn > 13463 [SYN, ACK] seg=0 Ack=1 win=64240 Len=0 #55=1460	DellComp_e7:f9:e2	CISCO_C9:f6:f8
1465	15:54:49,142397	63.218.98.110	63,205,26,90	TCP	13418 > microsoft-ds [ACK] Seg=2 Ack=2 vin=65533 Len=0	C1sco_t9:f6:f4	DellComp_e7:f9:e2
1466	15:54:49.142925	63,218,98,110	63.205.26.90	TCP	13463 > netb1os-ssn [AC#] Seg=1 Ack=1 win=65535 Len=0	C1sco_c9:f6:f4	DellComp_e7:f9:e2
	15:54:49.143631	63.218.98.110	63.205.26.90	NBSS	Session request, to "SMESERVER<20> from SCTHC<00>	C1sc0_191f61f4	DellComp_e7:f9:e2
	15:54:49.143776	63.205.26.90	63.218.98.110	NESS	Positive session response	Dellcomp_e7:f9:e2 Clsco_c9:f6:f4	C11CO_C91F6:F1
1469	15:54:49.233597	63.218.98.110	63.205.26.90	948	Negotiate Protocol Request	C1sco_c9:f6:f4	DellComp_e7:f9:e2 Clsco_c9:f6:f8
1470	15:54:49.233749	63.205.26.90	63.218.98.110	248	Negotiate Protocol Response	DellComp_e7:f9:e2	CISCO_C9:F6:FE
	15:54:49.324494 15:54:49.324709	63,218,98,110 63,205,26,90	63,205,26,90 63,218,98,110	9/8	Session Setup Andx Request, User: anonymous	Cisco_t9:f6:f4 DellCorp_e7:f9:e2	DellComp_e7:f9:e2 Cisco_c9:f6:f1
	15154149.414768	63.218.95.110	63.205.26.90	TOP .	Session Setup Andx Response 13463 > netbios-ssn (FIN, ACK) seg-200 Ack-216 xin+65320 Len=0	Cisco_(9)f6)f4	DellComp_e7:f9:e2
	15:54:49,414889	63,205,26,90	63.218.98,110	TCP	netbios-ssn > 13463 (FIN, ACK) seq=216 Ack=201 win=64041 Len=0	DellComp_e7:f9:e2	C1sco_c9if6if4
	15:54:49.415125	63.218.98.110	63,205,26,90	1CP	13500 > m1crosoft-ds [SW] Seg=0 win=63535 Len=0 M55=1460	C1sco_19:f6:f4	DellComp_e7:f9:e2
1470	15:54:49.415200	63.205.20.92	63.218.98.110	TCP.	microsoft-ds > 11500 [57N, ACK] Seg=0 Ack=1 win=64240 Len=0 MSS=1460	DellComp_e7:f9:e2	C1sco_c9:f6:f1
	15:54:49.504492	63,218,98,110	63.205.26.90	7C.P	13463 > netbios-ssn [Acx] seq=201 Ack=217 win=65320 Len=0	C1sco_c9:f6:f4	DellComp_e7:f9:e2
	15:54:49.504796	63.218.98.110	63.205.26.90	TCP	13500 > microsoft-ds [ACK] Seq=1 Ack=1 vin+65535 Len=0	Cisco_19:f6:f4	DellComp_e7:f9:e2
	15:54:49.505909	63,218,98,110	63.205.26.90	948	Negotiate Protocol Request	C1sco_19:f6:f4	Dellcomp_e7:f9:e2
	15:54:49.506485	63.205.26.90	63.218.98.110	948	Negotiate Protocol Response	DellCorp_e7:f9:e2	Cisco_C9:f6:f4
1111	15:54:49.597846 15:54:49.598419	63.218.98.110 63.205.26.90	63.205.26.90 63.218.98.110	918	Session Setup Andx Request, NTLNSSP_NEGOTIATE	Cisco_t9:f6:f4	DellComp_e7:f9:e2 Cisco_c9:f6:f4
	15:54:49.692478	63.218.98.110	63.205.26.90	918	Session Setup Andx Response, NTLMSSP_CHALLENGE, NTLMSSP_CHALLENGE, Error Session Setup Andx Request, NTLMSSP_AUTH, User: WORKGROUP\	Cisco ratfaif4	DellComp e7:f8:e2
1484	15:54:49.694301	63.205.26.90	63.218.95.110	246	Session Setup Andx Response, Error: STATUS_LOGOK_FAILURE	Dellcomp_e7:f9:e2 Cisco_t9:f6:f4	DellComp_e7:f9:e2 Cisco_c9:f6:f4
1445	15:54:49.784753	63.218.98.110	63.205.26.90	948	Session Setup Andx Request, User: WORKGROUP\	C15co_19:f6:f4	DellComp_e7:f9:e2
	15:54:49.784927	63.205.26.90	63.218.98.110	348	Session Setup Andk Response	DellCorp_e7:f9:e2 C1sco_t9:f6:f4	Clsco_c9:f6:fi
	15:54:49.875321	63, 218, 98, 110	63.205.26.90	248	Tree Comect Andx Request, Path: IPCS	C15C0_£9:f6:f4	DellComp_e7:f9:e2
	15:54:49.875513	63.205.26.90	63.218.98.110	918	Tree Cornect Andk Response	DellComp_e7:f9:e2	cisco_c9:f6:f4
1489	15:54:49.965987 15:54:49.966543	63.218.98.110 63.205.26.90	63.205.26.90 63.218.98.110	218 216	NT Create AndX Request, Path: \browser NT Create AndX Response, FID: 0x4000	Cisco_t9:f6:f4 DellCorp_e7:f9:e2	DellComp_e7:f9:e2 Cisco_c9:f6:f4
	15:54:50.060836	63.218.98.110	63.205.26.90	CERPC	Bind: call_id: 0, 11 cortext items, 1st b3332384-081f-0e95-2c4a-302cc308	cisco rotfatfa	DellComp_e7:f9:e2
	15:54:50.061211	63,105,26,90	63.218.98.110	245	write Andx Response, FID: 0x4000, 512 bytes	Dellcomp_e7:f9:e2	Cisco_C9:f6:f4
	15:54:50.151419	61,718,98,110	63.205.26.90	948	Read Andx Request, FID: 0x4000, 48000 bytes at offset 0	Cisco_19:f6:f4	DellComp_e7:f9:e2
	15:54:50.151584	63.205.26.90	63.218.98.110	DEERPE	sind_ack: call_id: 0 accept max_xmit: 4260 max_recv: 4260	DellCorp_e7:f9:e2	Cisco_c9:f6:fi
1495	15:54:50.248604	63,218,98,110	63.205.26.90	DCERPC	Request: call_fd: 0 opnum: 31 ctx_fd: 10	C1sco_r9:f6:f4	pellcomp_e7:f9:e2
1496	15:54:50.248964	63.205.26.90	63,218,98,110	948	write AndX Response, FID: 0x4000, 700 bytes	DellComp_e7:f9te2	Cisco_c9:f6:f4
1497	15:54:50.488365	63.218.98.110	63.205.26.90	TCP	13500 > microsoft-ds (ACK) Seq=2285 Ack-1303 win=64233 Lem=0 SEBEK - p1d(21d) wid(0) fd(0) emd:	C1sco_19:f6:f4	DellComp_e7:f9:e2 Intel_caid4:ec
1418	15:54:50.637391	63.205.26.90	63.205.26.2	SEREK	SLELK - p10(210) w10(0) f0(0) (ed)	Dellcorp_e7:f9:e2 Dellcorp_e7:f9:e2	Intel_cald4lec
	15:54:50.637510 15:54:50.646319	63.205.26.90	212.95.32.10	NENS 1	ans > http [SYN] Seq=0 win+64240 Len=0 MSS=1460 Name query NS WPAD<00>	Dellcorp_e7:f9:e2	Clsco_c9:f6:f8 Broadcast
	15:54:50.815476	212.95.32.104	63.205.26.90	TCP.	http > ans [SYN, ACK] Seg=0 Ack=1 win=16384 Len=0 MSS=1460	cisco_t9:f6:f4	DellComp_e7:f9:e2
1562	15:54:50.815613	63,205,26,90	212.95.32.10	TEP	ans > http [ack] tep=1 ack=1 win=64240 ten=0	DellComp_e7:f9:e2	Cisco_c9:f6:fi
1503	15:54:50.815949	63.205.26.90	212.95.32.104	NTTP	GET /10x exe HTTP/1.0	DellComp_e7:f9:e2	C15C0_C9:F6:F8
1504	15:54:50.996149	212.95.32.104	63.205.26.90	TCP	[TCP segment of a reassenbled PDU]	C1sco_19:f6:f4	DellComp_e7:f9:e2
	15:54:51.005118	212.95.32.104	63.205.26.90	TCP	[TCP segment of a reassembled POU]	C1sco_191f61f4	DellComp_e7:f9:e2
	15:54:51.005442	63.205.26.90	212.95.32.10	TCP .	ans > http [ACK] Seq=149 Ack=1845 win=6i240 Len=0	Dellconp_e7:f9:e2	Cisco_c9:f6:f4
	15:54:51.192052	212.95.32.154	63.205.26.90	TCP	[TCP segment of a reassembled POU]	c1sco_191f61f4	DellComp_e7:f9:e2
	15:54:51.200012	212.95.32.104	63.205.26.90	X.P	[TCP segment of a reassembled POU]	Cisco_191f6:f4	DellComp_e7:f9:e2
1 570	15:54:51.200336 15:54:51.208137	63.205.26.90 212.95.32.104	212.95.32.10	XCP XCP	ans > http (ACK) Seq-149 Ack=4765 win=6/240 Len=0 (TCP segment of a reassembled PDU)	DellComp_e7:f9:e2 Cisco_19:f6:f4	cisco_c9:f6:fi bellcomp_e7:f9:e2
1000	13134134.2V0217						Contraction of the second
- I B	uffer ov	erflow	from	Sebel	t packet reveals cmd to TEg	z download	named
						-	nonico
<u> </u>	3.213.98.11	0		TLAL	P.//212.95.02.104 10z	s.exebegins.	I
10-	0.210.20.11	0		11111		A. CAC UCXIIIA.	I

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 Forensic analysis shows the Win2K honeypot been exceeded. 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.

. Time	Source	Destination	Protocol	Info	Src MAC
15/4 15:54:52.228030	63.205.26.90	212,95,32,104	TCP	ams > http [ACK] Seq=149 Ack=65029 Win=64240 Len=0	DellComp_e/:t9:
1575 15:54:52.232178	212.95.32.104	63.205.26.90	TCP	[TCP segment of a reassembled PDU]	cisco_c9:f6:f4
1576 15:54:52.334020	212.95.32.104	63.205.26.90	TCP	[TCP segment of a reassembled PDU]	cisco_c9:f6:f4
1577 15:54:52.334284	63.205.26.90	212.95.32.104	TCP	ams > http [ACK] Seq=149 Ack=67381 Win=64240 Len=0	<pre>DellComp_e7:f9:</pre>
1578 15:54:52.341997	212.95.32.104	63.205.26.90	TCP	[TCP segment of a reassembled PDU]	Cisco_c9:f6:f4
1579 15:54:52.349975	212.95.32.104	63.205.26.90	TCP	[TCP segment of a reassembled PDU]	Cisco_c9:f6:f4
1580 15:54:52.350240	63.205.26.90	212.95.32.104	TCP	ams > http [ACK] seq=149 Ack=70301 Win=6424 <u>0 Len=0</u>	DellComp_e7:f9:
1581 15:54:52.357971	212.95.32.104	63.205.26.90	TCP	[TCP segment of a reassembled PDU]	
1582 15:54:52.358661	212.95.32.104	63.205.26.90	HTTP	нттр/1.1 200 ок (application/x-msdown tad) Egg Download Co	mplete
1583 15:54:52.358811	63.205.26.90	212.95.32.104	TCP	ams > http [ACK] Seq=149 Ack=72065 Win=6424	-
1584 15:54:54.508205	63.205.26.90	63.205.26.2	SEBEK	SEBEK - pid(420) uid(0) fd(0) cmd:	
1585 15:54:54.508303	63.205.26.90	63.205.64.23 🗲		mtqp > microsoft-ds [SYN] Seq=0 Win=64240 Len=0 MSS=1460	DellComp_e7:f9:
1586 15:54:54.537445	63.205.26.90	63.205.26.2	SEBEK	SEBEK - pid(420) uid(0) fd(0) cmd:	_
1587 15:54:54.537586	63.205.26.90	63.205.162.171	TCP	sbl > microsoft-ds [SYN] seq=0 win=6 DNS query ninjawar.	Lord.com
1588 15:54:54.561237	63.205.26.90	63.205.26.2	SEBEK	SEBEK - p1d(216) u1d(0) fd(0) cmd:	
1589 15:54:54.561432	63.205.26.90	206.13.28.12	DNS	Standard query A ninjawarlord.com 🗲	
1590 15:54:54.567470	63.205.26.90	63.205.26.2	SEBEK	SEBEK - pid(420) uid(0) fd(0) cmd:	
1591 15:54:54.567599	63.205.26.90	63.205.4.64	TCP	danf-ak2 > microsoft-ds [SYN] Seq=0 win=64240 Len=0 MSS=1460 DNS 1	response
1592 15:54:54.575558	206.13.28.12	63.205.26.90	DNS	Standard query response A 75.146.106.201 A 221.143.48.24	-
1593 15:54:54.576507	63.205.26.90	63.205.26.2	SEBEK	SEBEK - pid(420) uid(0)_fd(0) cmd:	
1594 15:54:54.576603	63.205.26.90	75.146.106.201	TCP	afrog > worldscores [SYN] Seq=0 win=64240 Len=0 MSS=1460	DellComp_e7:f9:
1595 15:54:54.579304	64.171.152.77	63.205.26.90	ICMP	Time-to-live exceeded (Time to live exceeded in transit)	
1596 15:54:54.597538	63.205.26.90	63.205.26.2	SEBEK	SEBEK - pid(420) uid(0) fd(0) cmd: Bot a	attempt to
1597 15:54:54.597617	63.205.26.90	63.205.102.213	TCP	boinc-client > microsoft-ds [SYN] Seq=0 Win=64240 Len=0 MS	-
1598 15:54:54.627534	63.205.26.90	63.205.26.2	SEBEK	SEBEK - pid(420) uid(0) fd(0) cmd: contac	t
1599 15:54:54.627608	63.205.26.90	63.205.200.105	TCP	dcutility > microsoft-ds [SYN] Seq=0 Win=64240 Len=0 MSS=14	
1600 15:54:54.657559	63.205.26.90	63.205.26.2	SEBEK	SEBEK - pid(420) uid(0) fd(0) cmd: ninjaw	arlord.com
1601 15:54:54.657676	63.205.26.90	63.205.42.254	TCP	fpitp > microsoft-ds [SYN] Seq=0 Win=64240 Len=0 MSS=1460	
1602 15:54:54.670471	75.146.106.201	63.205.26.90	тср	worldscores > afrog [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 🔶 and re	sponse.
1603 15:54:54.687623	63.205.26.90	63.205.26.2			
1604 15:54:54.687731	63.205.26.90	63.205.140.146	63.20	05/16 scan begins eq=0 win=64240 Len=0 MSS	
1605 15:54:54.717705	63.205.26.90	63.205.26.2			
1606 15:54:54.717818	63.205.26.90	63.205.238.38		(in=64240 Len=0 MSS=1460	

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

Wireshark's The results of performing 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 Conversations are loosely defined as a packet overlooked. sent to a destination and a packet received from that Some conversations are failed attempts of destination. 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 attacks could or 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<-8	Bytes A<-B	Rel Start	Duration	bps A->8	bps A<-B
53.205.26.90	80.2.69.176	1966	244091	800	81560	1166	162531	36310.712753000	206.3785	3161.57	6300.31
53.205.26.90	63.205.26.94	1034	137809	518	63267	516	74542	1433.103838000	190571.5120	2.66	3.13
53.205.26.90	88.210.85.234	829	63612	285	20052	544	43560	39446.999735000	127.5498	1257.67	2732.11
53.205.26.67	63.205.26.94	793	107782	400	50828	393	56954	3368.013959000	92539.1340	4.39	4.92
53.205.26.90	63.218.98.110	465	30462	232	15373	233	15099	19290.195866000	25349.2985	4.85	4.76
53.205.26.67	63.205.26.90	320	39170	143	19138	177	20032	2893.239255000	34798.6084	4.40	4.61
53.205.26.90	221.143.48.246	286	17724	285	17670	1	54	19378.869633000	16110.0556	8.77	N/A
53.205.26.90	75.146.106.201	246	15228	243	15066	3	162	19295.813082000	16011.0704	7.53	0.08
53.205.26.90	118.123.5.109	87	10427	33	3326	54	7101	8591.459112000	69688.3941	0.38	0.82
53.205.26.90	212.95.32.104	85	76992	31	2004	54	74988	19291.873989000	11.7301	1366.74	51142.21
53.205.26.90	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	-	366	3	180	3	186	12009.331789000		160.04	165.37
63.205.26.90	93.43.146.218	6	378	3	180	3	198	44316.029103000		823.16	905.47
63.205.26.90	118.160.239.123	-	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		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		158.78	164.07
60.15.177.165	63.205.26.90	5	1706	3	1426	2	280	36345.177696000		0.59	0.12
63.205.26.90	63.205.53.55	5	302	4	248	1	54	28589.831619000		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		4	300	2	150	2	150	25357.919661000		4363.38	4363.38
	63.205.26.90	4	1412	4 3	1342	1	20	6102.502972000		9303.30	4303.30 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(<u>https://submit.symantec.com/websubmit/retail.cgi</u>) 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.

Status		8 Profiles								
Description	Value	Victim IP V Score Gen Time C & C IP Infector IP Egg Source IP Peer Coord IP Resource								
Last status	2009/06/29 15:54:53 UTC	63.2(5.26.90 1.80 05/27/2009 23:57:4:757 UTC 212.95.32.104 63.2(5.26.90 1.30 05/28/2009 00:40:42.400 UTC								
started	2009/05/27 17:29:06 UTC	63.2(5.26.90 0.80 05/28/2009 00:35:05.111 UTC								
Elapsed	32 22 25:47	63.2(5.26.90 0.80 05/28/2009 00:29:00.981 UTC 63.2(5.26.90 0.80 05/28/2009 00:22:56.851 UTC								
temory usage	81.6 MB	63.2(5.26.90 0.80 05/28/2009 00:16:52.018 UTC								
ines read	233	63.2(5.26.90 0.80 05/28/2009 00:10:47.888 UTC 63.2(5.26.90 0.80 05/28/2009 00:04:43.757 UTC 63.2(5.26.90 0.80 05/28/2009 00:05.2(5.26.90 0.80 05/28/2009 00:05.2(5.26.90 0.80 05.2(5.26.90 0.80 05.2(5.26.90 0.80 05.2(5.26.9000005.2(5.26.9000000005.2(5								
ines parsed	233	Score 1.8 (>= 0.8) Infected Target: 63.205.25.90								
ocal bot profiles	8	Infector List. <unobserved></unobserved>								
etQuery request	97	Egg Source List: 212.95.32.104 C & C List: <unobserved></unobserved>								
etQuery respo	64	Peer Coord. List: <unobserved></unobserved>								
lessages queued	32	Resource List 								
lessages sent	73	Gen. Time: 05/27/2009 23:57:41.757 UTC								
epository status	disconnected	INBOUND SCAN								
	java.ret.NoRouteToHostExcept	<unblacktored></unblacktored>								
utior ID	bd3520c53f145bf2	EXPLOIT								
bserver ID	816006193	<unobserved></unobserved>								
		212.95.32.104 (23:55:41.786 UTC) event=1:3300003 (tcp) E3(rb) BotHunter HTTP-based .exe Upload on backdoor port 1037->80 (23:55:41.786 UTC) C and C TRAFFIC								



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 attempts could be multiple 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 is suggestion of C2 channels 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.

this chapter gives In summary, 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, the download and source confirming egg 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 intrusion correlation algorithm, detection а 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 employment within tactical networks honeynet 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. Α 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 nonfirewalled 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

52

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 <u>http://dban.org</u> 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@all

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:
<u>Username user type password</u>

Bobby	admin	Tlght@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), <u>http://www.usenix.org/events/hotbots07/tech/full_papers</u>/karasaridis/karasaridis.pdf.
- [3] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "Bothunter: Detecting malware infection through idsdriven 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/co oke/cooke.pdf.
- [6] T. Holz, M. Steiner, F. Dahl, E. W. Biersack, And F. Freiling, "Measurements and mitigation of peer-to-peerbased 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/hol z/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), <u>http://www.honeynet.org//papers/sebek</u>, 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/2009/0 1/.
- [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.pd f.

- [16] L. Spitzner, "Honeypots: Definitions and value of honeypots," <u>http://www.tracking-hackers.com</u>, May 29, 2003 (accessed January 2009), <u>http://www.trackinghackers.com/papers/honeypots.html</u>.
- [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), <u>http://www.isoc.org/isoc/conferences/ndss/08/papers/17_</u> botsniffer_detecting_botnet.pdf.
- [18] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "Botminer: Clustering analysis of network traffic for protocoland 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

- Defense Technical Information Center Ft. Belvoir, Virginia
- Dudley Knox Library Naval Postgraduate School Monterey, California
- John G. Kato Naval Information Operations Command Suitland Suitland, Maryland
- John T. Scott Naval Information Operations Command Suitland Suitland, Maryland
- 5. Marine Corps Representative Naval Postgraduate School Monterey, California
- Director, Training and Education MCCDC, Code C46 Quantico, Virginia
- Director, Marine Corps Research Center MCCDC, Code C40RC Quantico, Virginia
- Marine Corps Tactical Systems Support Activity (Attn: Operations Officer) Camp Pendleton, California
- 9. Professor John McEachen Naval Postgraduate School Monterey, California
- Professor Murali Tummala Naval Postgraduate School Monterey, California