2008

# Botnets, a Threat to National Security: An Examination of Bots, Botnets and the Threats They Pose to National Security

Louis F. Brooks

FLORIDA STATE UNIVERSITY

COLLEGE OF ARTS & SCIENCES

# BOTNETS, A THREAT TO NATIONAL SECURITY. (AN EXAMINATION OF BOTS, BOTNETS AND THE THREATS THEY POSE TO NATIONAL SECURITY)

By

LOUIS F. BROOKS

A Thesis submitted to the
Department of Computer Science
in partial fulfillment of the
requirements for the degree of
Masters of Science

Degree Awarded:
Spring Semester, 2008

The members of the Committee approve the Thesis of Louis Brooks defended on April 11, 2007.

_____
Alec Yasinsac
Professor Directing Thesis

_____
Sudhir Aggarwal
Committee Member

_____
Zhenghao Zhang
Committee Member

_____
David Whalley
Chair, Department of Computer Science

_____
Joseph Travis
Dean, College of Arts & Sciences

The Office of Graduate Studies has verified and approved the above named committee members.

## ACKNOWLEDGEMENTS

I would like to acknowledge my wife, Mafe Brooks, for supporting me while I returned to graduate school to obtain this degree.

## TABLE OF CONTENTS

## LIST OF FIGURES

## ABSTRACT

In this paper we show that botnets and specifically peer-to-peer botnets, pose a considerable threat to our national security. We examine the rise of the botnet threat and the technology that makes this threat possible. We also examine the tools available to the criminals that control botnets, botnet offensive and defensive capabilities and the latest trends in botnet development. Then we show how this technology can be used as an offensive weapon against the United States and our allies.

# 1. INTRODUCTION

Stormworm, which first appeared early in 2007, is the first widely spread peer-to-peer style botnet. My reasons for researching this specific botnet are due to the novel architecture that it uses for command and control. While there have been other P2P botnets, none have had as wide spread distribution or impact as Stormworm. At the time that I began my research there was little known about the functionality or capabilities of this new bread of botnet. I will show my research methodology into the peer-to-peer functionality of Stormworm including the discovery of a peers blacklist which is as far as we know is a new discovery. I will show how a botnet modeled after this architecture could be used as an offensive weapon; why this type of weapon would be attractive to use due to its scalability, resilience to attack, low cost to setup and operate and the anonymity it provides to its owner; and compare the capabilities of a Stormworm style botnet to the botnet attack against Estonia that occurred in 2007.

Over the past few years the focus of malware developers and hackers has changed from mischief and causing general mayhem to developing large groupings of compromised hosts, called bots, drones or zombies into what are known as botnets. These large networks of compromised computers, often numbering in the hundreds of thousands of hosts are increasingly being used for criminal activity. [BY06] Dr. Merrick Furst has stated that the emerging threat from botnets is the number one danger to the Internet. [M06] Researches estimate that around 30,000 bots, which is equivalent to the computational resources of a major U.S. university, are added to botnets world wide each day. [CJM2005]

In the winter of 2005 the Dutch National Criminal Investigation unit discovered a botnet made up of over 1.5 million hosts after it was used in a distributed denial of service attack against a U.S. company. [SANS05] Earlier in 2005 the Washington Post reported that a major online advertising agency was suing seven people for illegally installing its adware on victim's computers without their knowledge through the use of botnets. [WP05]

With cyber-armies numbering in the millions, botnets pose a significant threat to the security of our national information technology infrastructure. Foreign powers have already

shown a willingness to use botnet cyber warfare against other nations. However the development and control of botnets able to attack the United States is not limited to large nation states and is well within reach of a terrorist organization such as Al-Qaeda. With our communications and financial infrastructure becoming reliant upon network technology this threat continues to grow.

## 2. BOTS & BOTNETS, THE TECHNOLOGY BEHIND THE THREAT

Zou and Cunningham describe botnets as a collection of compromised computers that are controlled by a single entity. The individual bots are software programs that run on a host computer allowing the bot herder to control the actions of the host. A host itself may be infected by several bots and be a member of several botnets. Controllers can use these bots to perform several types of attacks including DDOS, email spamming, key logging, abusing online advertisements, spreading new malware, and identity theft. [ZC06]

### 2.1 Bot Propagation

Botnets blur the distinction between different forms or malware such as worms, trojans and viruses. [DZL2006] Vertical and horizontal scanning of an IP space by infected hosts looking for other vulnerable hosts to compromise is still the primary means of propagation. It is estimated that from 27% to as high as 75% of all scanning on the internet is conducted by botnets. [RZMT06] The information gleaned from these scans can be used to identify potential victims and then compromise these hosts on an individual basis. Another technique, which may have been used by the authors of the Witty Worm, is to compose a list of potential victims and then mount a distributed assault on these targets from multiple sources. It is believed that the witty worm, which is one of the fastest spreading worms in internet history, may have used this technique since none of the originating hosts were vulnerable to this exploit. [SPW02]

Trojans, either through email, instant messaging, or malicious web pages have long been a favorite delivery vector for malware authors. A recent example of this is the postcard.exe trojan that masqueraded as an online postcard but actually installed a root kit and peer-to-peer based bot on the computer.

Once a host is compromised, the bot application is downloaded to the computer's hard drive and then executed. This software contains the network client and information needed to

connect to the botnet, as well as offensive or defensive weapons that the bot may need to carry out its owner's directives. The bot software may also disable any virus protection, firewalls or other safety precautions the host may have installed. It is often the behavior of the bot herder to then harden the host so that it can not be compromised by a competitor through the same vulnerability. [P03].

### 2.2 Command & Control Architecture

The value of a botnet can be seen both in its ability to use distributed computer power for its owners gain and the ability to retain some form of anonymity through the use of a multi-tier command and control structure.

Distributed computing allows a user to divide the load for accomplishing a task across the resources of many hosts. A DDOS performed by a botnet comprised of several hundred thousand bots could cripple any defense to defeat such an attack currently in use. A botnet composed of a million or more hosts could be used as an attack by terrorists or as an act of war against a nation state to cripple its network infrastructure. [SPW02]

By using a multi-tiered command and control (C&C) structure the bot herder is able to hide her identity from authorities. [S06] The bots themselves are not physically owned by the controller and may be located in private residences, corporations and government agencies spanning the globe. Differences in language, customs, time zones and laws make it difficult to track the attackers across international boundaries. Should the C&C Server be taken down the bot herder can set up another C&C, sometimes within minutes, and regain control of the botnet.

Currently two primary means of C&C are in use by bot herders. The most common is an IRC based hierarchical model. A newer model that has only been in circulation since 2006 is a peer-to-peer based distributed C&C model.

## 2.3 IRC Based Hierarchical Model

Developed by Jarkko Oikarinen as a method for text based conferencing over a network the Internet Relay Chat (IRC) protocol has become the standard C&C mechanism for botnets. IRC is designed in the classic server client architecture where users log into an IRC server through the use of IRC clients. (See Figure 2.3.1) Each IRC server hosts various channels which are venues for conversation by the server's users. IRC is a flexible, extensible and easy to administer communication system that allows both one-to-one and one-to-many communication. [M06] Several open source versions exist so cost of implementation is minimal.

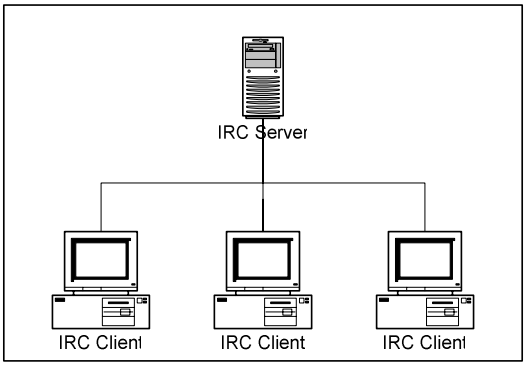


**Figure 2.3.1 – IRC Network**

Robots, or bots, were originally developed for use in IRC [OR93] to automate common tasks such as managing access lists, moving files, sharing user and channel information, etc. [P03] Later more malicious uses for bots were developed in efforts to knock other IRC users out of a channel or to deny access to an IRC channel by its users. These early denials of service attacks (DOS) were primarily done as a form of mischief or in retribution for some action by the victim. Larger victims required greater resources and the Distributed Denial of Service (DDOS) attack was invented. As time went by the controllers of these networks discovered the potential

financial gains that could be made through extortion, identity theft, spam and other forms of malicious behaviors. This has spurred a more aggressive growth in the number of bots. [CJM05]

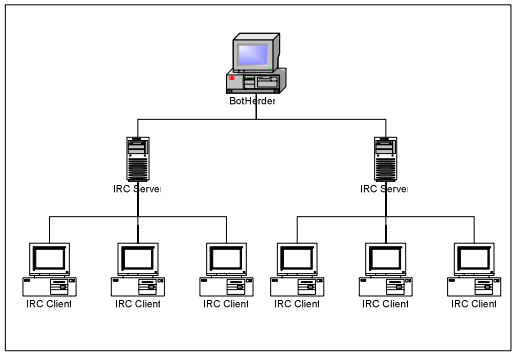


**Figure 2.3.2 – IRC Based Botnet**

Through the use of an IRC client (See Figure 2.3.2) the bots join an IRC Server controlled by the bot herder. Once connected the drones can receive commands in the chat channel or through a private message. IRC allows the first person to enter a channel to control or moderate the channel. This makes it important that the bot herder always be the first client to sign into the channel. Should the bot herder be unable to connect to the channel, she could lose control of her botnet to a rival. To combat this, botnets often employ simple password based authentication to prevent someone from entering the channel unauthorized.

The wide range of commands available through IRC allows the bot herder complete control over her bots. These capabilities include communication, reconnaissance, file transfer, attack capabilities and system monitoring. Often the IRC systems used by bot herders do not conform to the IRC protocol but are stripped down versions with several unwanted features removed for convenience. [RZMT06]

There are several thousand bot packages that are available on the internet. I have personally examined over 200 different bot packages which were downloaded from various websites. This was only a small sample of what is available to the public. The majority of these are derivatives of a few common bots. Often all that is needed to set up a bot is to enter the IRC, C&C, DNS IP addresses and server password in a configuration file. Include any special capabilities desired from off the shelf modules. Then compile the bot executable for the desired platform.

One of the primary weaknesses of IRC as a form of C&C is the IRC server which is a single point of failure for the botnet. One method that has been devised by the blackhat community is the use of DNS names instead of static IPs in the configuration of the bot. If an IRC server is disabled she can set up another IRC server, change the DNS name to point to the address of this new server and be back in control of her botnet in a matter of minutes.

### 2.4 Peer-to-peer Based Distributed Model

A more recent development in botnet technology has been the use of a peer-to-peer (P2P) C&C structure. These new forms of botnets feature anonymous, secure and encrypted communication and are resilient to attack. [LEKR05]

### 2.5 Gnutella Based Botnets

Early peer-to-peer botnets were created in a structure modeled on Gnutella a popular file sharing network protocol. (See Figure 2.5.1)

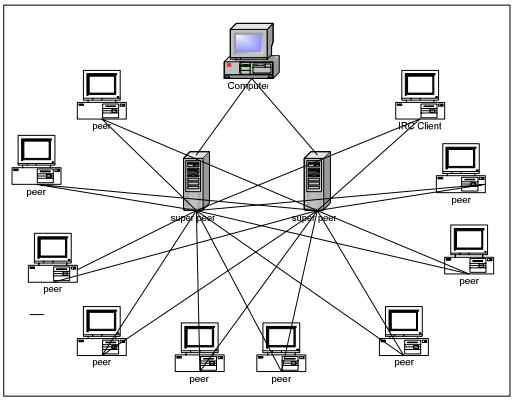


**Figure 2.5.1 Gnutella Style Peer-to-peer Based Botnet**

A Gnutella style botnet could be comprised of approximately $2^{20}$ nodes with a maximum of 6 hops between any node on the network. This architecture uses a peer - ultra peer structure in which some hosts are ultra peers and form the backbone of the P2P network. Each ultra peer can be connected to up to 30 other ultra peers. Leafs in the network are normal peers and are connected to a smaller subset of ultra peers. It is estimated that this type of network could survive the loss of 60% of its ultra peers and almost 90% of all nodes and still remain functional. [LEKR05] With no single point of failure this style of botnet provides high resilience, difficulty in tracking and scalability to the bot herder but at the expense of increased complexity. [ZC06]

## 2.6 Kademlia Based Botnets

The second generation of P2P based botnets is using a true distributed P2P network based on the Kademlia protocol. Developed by Petar Mayounkov and David Mazieres [MM02], Kademlia is an overlay network built on top of the TCP/IP protocol which uses a Distributed Hash Table (DHT) for routing. Each node in the overlay network has a 160 bit quasi-unique node ID that is used for network identification. This allows for $2^{160}$ total nodes per network as opposed to $2^{20}$ nodes for a Gnutella based botnet. Closeness in the Kademlia protocol is defined by

XORing the node ID of any two nodes in the network. (See Figure 2.6.1) Nodes in the same subtree are closer than nodes in two different subtrees.

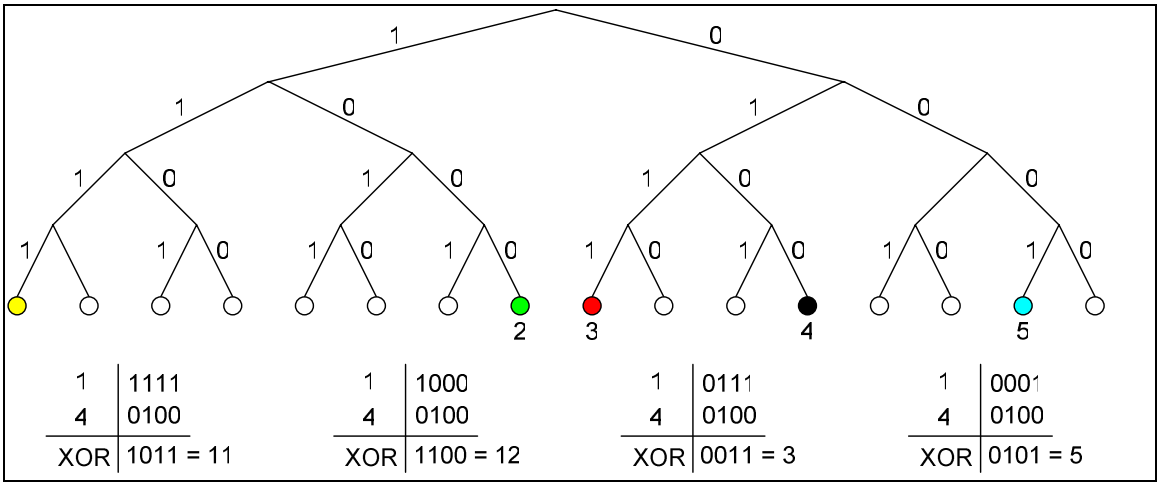


**Figure 2.6.1 Distance between five nodes in a Kademlia network using a 4 bit key space.**

## 2.7 Distributed Denial of Service

A denial of service (DOS) attack attempts to deny access to legitimate resources or services. [G84] The attacks can either focus on the network level or the OS level. [DM03] For this paper we will be dealing strictly with network level attacks. Network based attacks can be divided into two categories, exhaustion of computer resources and exhaustion of bandwidth.

An exhaustion of computer resource attack aims to deplete a target's resources by exploiting a vulnerability in the target system causing it to crash or by exploiting a design flaw tying up the target's resources so they are unavailable for legitimate users. This type of attack can be accomplished by a single node due to the fact that it is abusing a flaw in the software running

on the node. An example of this type of attack is the Ping – Of – Death attack that uses a malformed Internet Control Message Protocol (ICMP) to crash a vulnerable system. [PLR07].

An exhaustion of bandwidth attack uses massive amounts of traffic to inundate a target's available bandwidth so legitimate users are unable to access the server's resources. A distributed denial of service (DDOS) attack uses multiple hosts to provide greater computational resources and bandwidth for use in the attack. [DM03] Most bots are equipped with several forms of flooding or bandwidth attacks as well as packet shaping capabilities. [FHW05]

As opposed to the direct DDOS attack (SeeFigure 2.7.1) a reflection DDOS attack uses abuses the

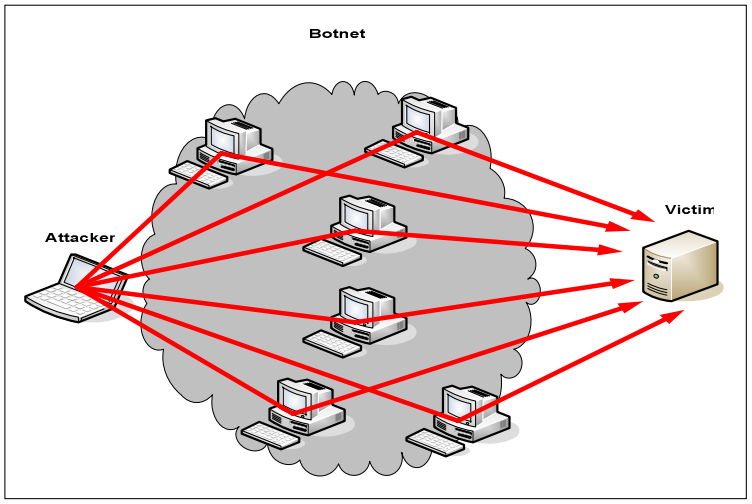


**Fig. 2.7.1 – Direct DDOS bandwidth exhaustion attack.**

TCP/IP protocol to magnify the power of her attack. In this example (See Figure 2.7.2) the members of the botnet transmit traffic to the broadcast domain of several class B networks using

the victim's IP address as the source of the traffic. The nodes in the class B networks will then respond, to the spoofed IP address flooding the victim with return replies. [PLR07]

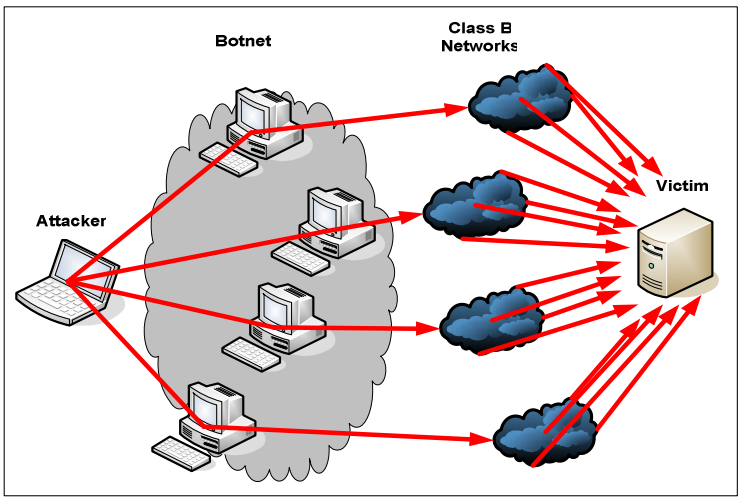


**Figure 2.7.2 – Reflection DDOS bandwidth exhaustion attack.**

Defense against DDOS attacks can prove difficult for several reasons including the volume of traffic, the distributed nature of the attack and the difficulty in correctly identifying malicious versus legitimate traffic. The volume of traffic that can be generated by large botnets can often exceed 10 gb/s. This amount of traffic exceeds the capabilities of most currently available network infrastructure and security equipment. The distributed nature of botnets helps to hide the source of the attack since botnets often span multiple continents and political boundaries. This makes it difficult to point to any one node and identify it as an attacker. The large numbers of hosts used also ensures a higher success rate, even if a few attacking hosts are disabled. [FHW05] Lastly, since the attacks are based on valid protocols or service requests, there is the difficulty in differentiating between legitimate and malicious traffic. [PLR07]

## 2.8 Botnet Defense through Fast Flux Networking Services

A fast flux network is one that abuses the DNS protocol in an effort to obfuscate or protect important network assets. First identified by The Honeynet Project Research Alliance [HPRA07] this type of network uses rapidly changing, short lived DNS records to randomly rotate the IP address mapped to a Fully Qualified Domain Name (FQDN). Fast flux networks also employ network protocol redirects which are transparent in an effort to further obscure valuable network assets. The Honeynet Project has identified two types of fast flux networks, single-flux and double-flux networks.

## 2.9 Single-Flux Networks

In a single-flux network (See Figure 2.9.1) a blackhat will have a service, such as a website hosting illegal pornography, a phishing scheme or malware, hosted on several compromised machines. These zombies will form a pool of available IP addresses to which the FQDN of the website can be constantly cycled through. Each IP may be assigned to the FQDN for only a few minutes. A user accessing this website at different times, even though using the same FQDN, will actually be connecting to different computers under the blackhat's control. An alternative setup is to host the site on one of more central servers and use the pool of compromised computers as transparent redirects. This further obfuscates the backhat's assets and gives them greater control with easier management of the website.

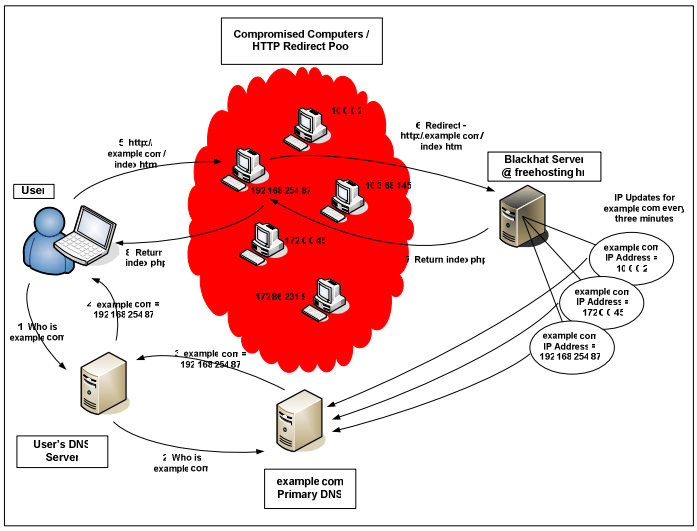


**Figure 2.9.1 Single-Flux Network with HTTP Redirect**

## 2.10 Double-Flux Networks

The double-flux network (See Figure 2.10.1) adds a second layer of obfuscation to the mix. In this scenario, not only is the IP address associated with a service's FQDN in flux, but the IP address of the primary domain name server for that domain is also constantly changing. This is accomplished by having the pool of compromised computers act as DNS redirects to a DNS server under the Blackhat's control. The IP addresses for the compromised computer are rotated on a regular bases as the primary DNS for that domain with the top level domain servers (.edu, .com, .hr, etc).

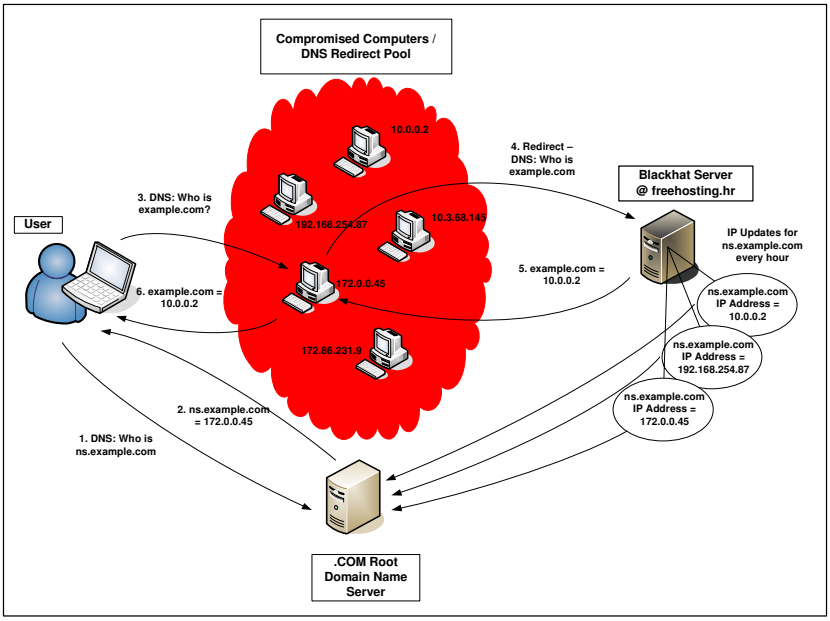


**Figure 2.10.1 Double-Flux Network with DNS Redirect.**

(See Figure 2.9.1 for HTTP half of Double-Flux Network)

## 3. STORMWORM, A KADEMLIA BASED BOTNET

Stormworm is the first widely spread truly pure peer-to-peer botnet. By pure peer-to-peer botnets, Stormworm has no central servers or super peers such as a Gnutella botnet would contain. This new, novel architecture is what drew me to study Stormworm. I wanted to see how this new architecture worked as well as its capabilities. At the time that I started my studies of the Stormworm botnet there was no available information on how it operated in the literature. My research into this new type of command and control structure allowed me to understand the capabilities of this model for a botnet and what the potential of this new style of architecture could provide to its controllers.

As part of my graduate research the Security & Assurance in Information Technology Laboratories (SAIT Labs) at the Florida State University's Department of Computer Science allowed me to use their network and IT resources to set up a honeynet to study hacker methods and tactics. A honeynet is a network set up with no other purpose than to be compromised so that the attacker can be closely monitored to track their activities.

The SAIT Labs honeynet network (See Figure 3.1.) is made up of sixteen publicly accessible IP addresses connected to the WAN through a Cisco router. The honeypots, or machines that are made available for compromise, are 386 based PCs loaded with various operating systems depending on the desired focus of the research. The primary monitoring and control mechanism for the honeynet is the honeywall. This machine acts as a bridge from the WAN router to the honeynet network and all traffic passing into or out of the honeynet must pass through this bridge. To keep the honeywall from being discovered the network interfaces do not have IP addresses but uses passive sniffing and forwarding to pass traffic through the bridge. The honeywall uses tcpdump, snort and iptables to monitor and control this traffic. Due to the fact that the honeynet does not have any legitimate users, all traffic into or out of the network is considered suspect. As opposed to traditional firewalls the iptables implementation on the Honeywall acts to keep the intruder from using the honeypot as a launching point for attacks

against other computers. The honeypots themselves are set up to allow for quick forensic analysis by keeping a database of hashes to identify all legitimate files stored on the computer.

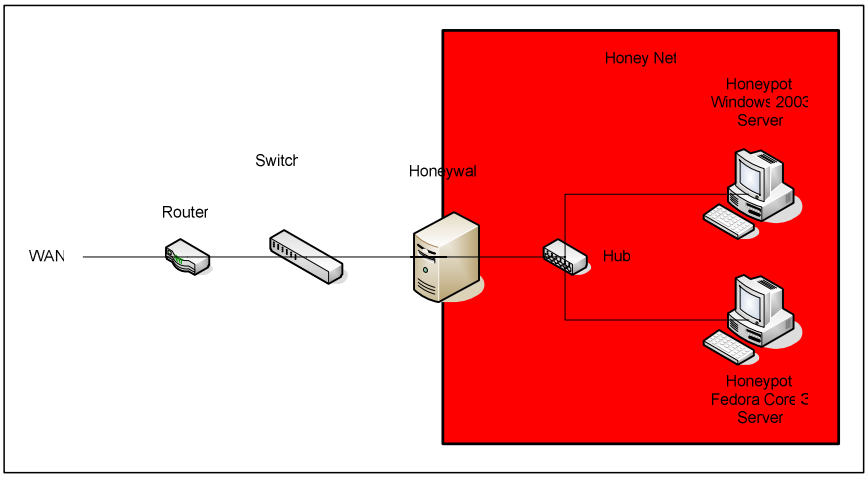


**Figure 3.1 – SAIT Labs Honeynet**

Once it has been determined that a honeypot has been compromised it is pulled from the network for analysis. First a live analysis is conducted while the system is still powered on. Running process are recorded, network connections examined and a preliminary search of the file system is conducted. Following the live analysis, a combination of static analysis using the forensics tool Autopsy and an exact bit for bit duplicate of the hard drive from the compromised system coupled with a live analysis of the system using a forensically sound bootable linux cd-rom is conducted. This allows us to identify files which may not have been detectable during the live analysis using the host's OS. Using file modify-access-create (MAC) timestamps, network logs from the Honeywall and files recovered from the honeypot, the hacker's activities can be reconstructed and studied.

I have been studying botnets for the past five years using the honeynet and for the past two years as a member of Florida State University's IT Security Team. Part of my responsibilities with the IT Security Team is to perform incidence response and forensic analysis on University owned computers that are suspected to have been compromised. It was in this fashion that the Stormworm bot first came to my attention. In early 2007 a computer was brought in for analysis and I discovered that it had been infected with a new bot that I had not seen before. After running the recovered bot executable through Norton Sandbox, an online malware identification tool, I was able to identify it as a new bot titled Stormworm. This new bot was named after the emails first used to propagate the bot through the internet. Further research of online material indicated that this bot used a P2P C&C architecture as opposed to the more traditional IRC C&C structure. I decided that this new bot warranted further investigation to see what its command structure and capabilities were. One new item that came from this initial analysis was that of a peers blacklist which as far as we know is a new discovery. This peers blacklist contained computers that were prohibited from accessing the botnet. Through my position with the FSU IT Security Team I had recently become a member of an organization composed of security professionals from higher education institutions from around the country. I learned from this group that the black list had not been seen in previous versions of the Stormworm bot.

Since this bot was being propagated by a trojan as opposed to a direct compromise, the traditional method of using a honeynet was not going to be effective. Instead I would need to "seed" the honeynet by directly infecting the honeypot and observing the effects. To this end I set up a honeypot with a standard, fully patched, install of Windows XP Professional. A recent copy of the spam email used to spread this botnet was retrieved from my computer science email account and opened in the Internet Explorer browser on the honeypot. The contents of the spam email indicated that I had received an electronic card from a family member and that I should follow the provided link to view my copy of the e-card.

Clicking on the link directed the browser to a webpage which, upon loading, caused the honeypot to crash and require a reboot. Upon reboot the honeypot started sending out UDP packets to multiple addresses indicating that it had been compromised. The computer was taken

offline and an analysis was performed. Live analysis did not show any signs of a compromise with no unusual processes or network activity identifiable with the system's built in tools. Furthermore, no new files, were found on the system. The static analysis however identified the bot executable and the bot configuration file contained in C:\windows\system32\ folder. The inability to detect the bot files with the system resources from the honeypot's OS indicated that a rootkit may have been used. The static analysis confirmed this showing that several system files had been replaced as part of the compromise.

I was also able to recover a copy of the web page that the browser loaded just before the system crash from the Internet Explorer's cache. Examination of the malicious webpage revealed that a JavaScript bug in the browser had been exploited to install the files on the computer. To determine if the exploit was specific to the Microsoft Internet Explorer web browser, I loaded another honeypot with the same parameters as the first except the addition of the Mozzila Firefox web browser. Following the same procedure as before gave the same results showing that it was not a browser specific exploit. It should be noted that later versions of the Stormworm bot did not crash the operating system. This indicates that the compromise procedure was improved to prevent this possibly undesirable side effect.

A third honeypot was loaded with a standard install of Windows XP but this time I disabled JavaScript functionality in the Internet Explorer browser. This was done to see what would occur when the malicious webpage was loaded without the JavaScript attack vector available. This time the webpage loaded with a message stating that due to the fact that JavaScript was disabled the e-card could not be displayed. The webpage indicated that to view the webpage, JavaScript should be re-enabled or that a copy of the ecard could be manually accessed by clicking on a link displayed in the webpage. Following this link downloaded an executable file to the desktop. When launched this executable installed the bot files on the host and caused the operating system to crash. Analysis of this machine showed the same results as the two that had been compromised by the malicious JavaScript exploit except for the executable that remained on the desktop.

The first honeypot that I had intentionally compromised was reactivated and reconnected to a secondary research network (See Figure 3.2) in order to monitor its activity.

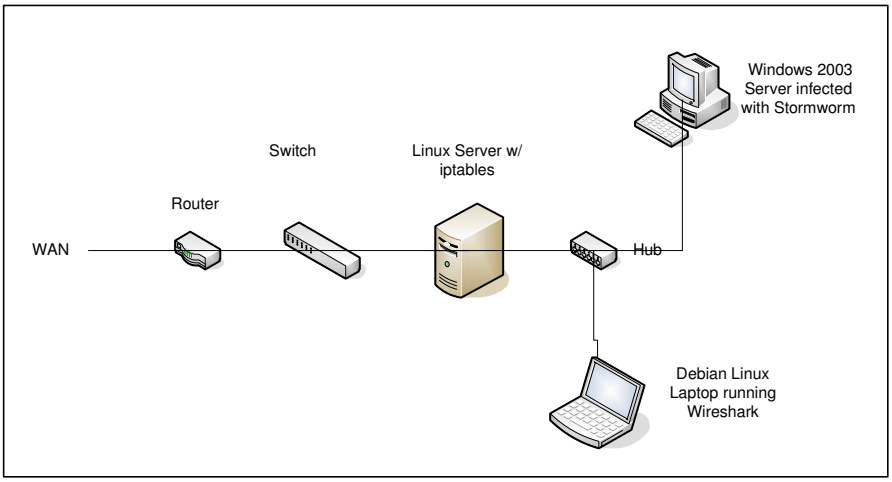


**Figure 3.2 – Secondary Research Network**

The secondary network was used to study the communications activity of the botnet. Through the use of a hub and a Debian Linux laptop running Wireshark, a network packet capture and analyzer tool all network traffic could be monitored and recorded. A Debian Linux server was used as a bridge to block any unwanted traffic from the honeynet. From this traffic I was able to determine that the protocol the bot was using was similar to the one used by the edonkey/emule file sharing network which is based on the Kademlia P2P protocol. I was also able to observe and capture the bot attempting to send large volumes of spam email which was composed of an empty file with a .pdf file attached.

At this time the honeypot was taken down and re-examined to detect if any changes had occurred to the host since the compromise. The only change found was that the peers list, a list of other nodes in the botnet with which the bot could communicate, had grown from thirty-two to

seventy-eight entries. Examining the packets captured by the laptop showed messages from other nodes responding to queries from this bot with lists of other nodes. The names and addresses of the new nodes matched the additions to the peers list. When compared with entrees on the original peers list it also became evident that some nodes had been removed from the list. The packets captured indicated that the bot had tried to contact these nodes but there had been no response. The captured packets also indicated that the bot processed the peers list in order from first to last when trying to make initial contact with other nodes in the botnet.

In an effort to further study the communication of the bots within the botnet I set up a virtual networking environment composed of two virtual machines running a default install of Windows XP connected by a single subnet in VMWare Workstation. (See Figure 3.3)

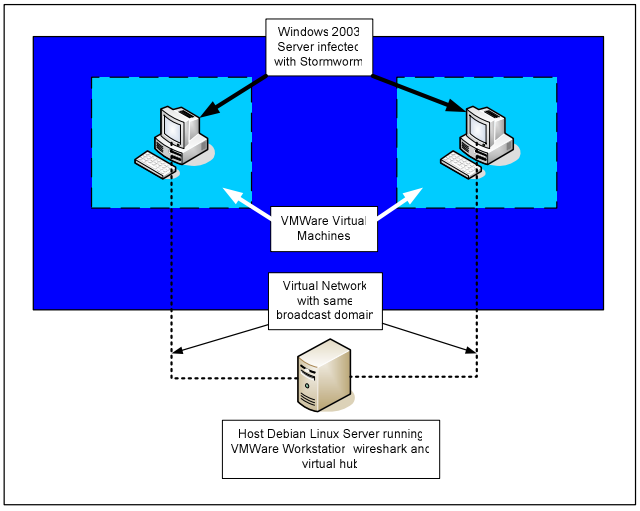


**Figure 3.3 – Single Subnet Virtual Network Architecture**

As part of the installation of Windows XP into the virtual environments I installed a suite of tools provided by VMWare to better integrate the guest operating system with the virtual environment.

In order to be able to monitor the traffic between the two virtual machines I set up the host, a 386 based PC running Debian Linux, to act as the hub connecting the two machines. I was then able to use Wireshark to capture all packets between the two virtual machines. The trojan executable that was manually downloaded to the third honeypot was transferred to the first virtual machine and executed. The trojan began to execute then stopped and deleted itself from the virtual environment. This is a common defense mechanism used by bot developers to prevent analysis of the bot in virtual environments. To combat this I reloaded the two virtual machines but this time I did not install the VMWare tools in the Windows XP operating systems. With this new setup the trojan installed the bot software as previously described. In an effort to facilitate communication between the two nodes I seeded the peers list with the network addresses for the two virtual machines. I had learned how to decipher the contents of the peers list from another researcher belonging to the previously mentioned professional group.

Connecting the two bots to the same broadcast domain did not have the desired results as the two bots refused to communicate. However I did detect the virtual machines sending out a high number arp requests. To further monitor this communication I stopped the virtual environment and replaced one of the nodes with a clean install of Windows XP. Once restarted, the infected node again began to send out gratuitous arp requests to each IP address in the subnet. When the clean node responded the bot attempted to access the root share through a simple brute force of the admin password. Once it had access to the root share it attempted to install the bot software on the new node.

Not obtaining my original goal of getting the two bots to communicate, I redesigned the virtual environment (See Figure 3.4) so that the bots were on separate but connected subnets with the host computer acting as a router between the two networks.

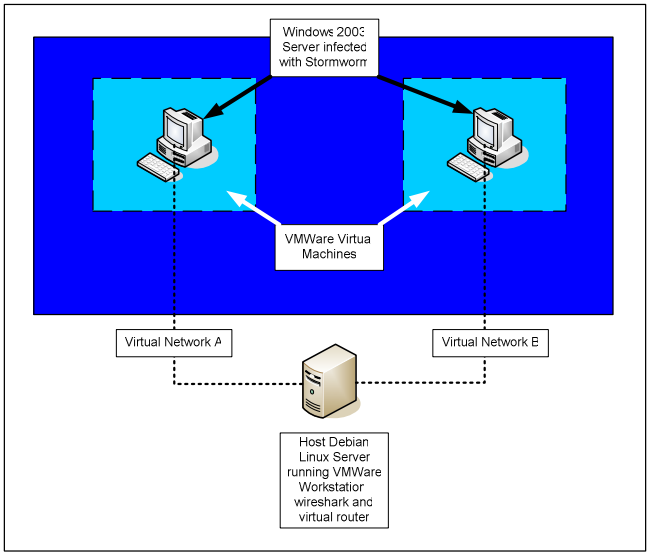


**Figure 3.4 – Dual Subnet Virtual Network Architecture**

With this new network configuration the bots started to communicate immediately. I tried several different configurations of the peers list and found the following about the bot's communication. The identification string used to identify the different nodes do not need to be unique or even related to a specific host. If a node appears using a previously seen ID string with a new IP address, this node is added to the peers list. The bot would also randomly contact nodes on the peers list after initially bootstrapping into the overnet to verify that they were still reachable. Since the protocol used by the Stormworm botnet appeared to be similar to the one used by the edonkey/emule P2P file sharing network I wanted to see if the nat/firewall flags used by Kademlia networks were used by the botnet. In the edonkey/emule network the last two digits in the peers list node address segment identify if the node is publicly available on the network. Resetting these address bits to one would signify that the node is not on a publicly routable network or is behind a firewall. One of the nodes had its peers list modified to change the last digit to a one and the node was restarted. The node attempted to contact the other node on its list

and reported its network status. After this, the second node would not attempt to contact the first node even if it was actually publicly addressable.

I was unable to obtain a copy of the bot used on the HTTP servers for examination. All of the trojans that I was able to obtain copies of installed the standard spam bot. I was also cautioned by other researchers that the HTTP servers would start a DDOS against any machine scanning them. This threat, combined with the use of fast flux redirects made examining the HTTP servers difficult and potentially dangerous so I will leave that to other researchers. Further examination of the botnet was also complicated by the fact that the latest generation of the Stormworm bot uses encryption to hide its communication within the overnet. The rest of this section summarizes my findings from studying the Stormworm botnet.

### 3.1 Propagation

The Stormworm bot has primarily been spread through spam email. These emails will contain a link to a trojan that installs the bot software on the system. The intended victim will receive an email stating that they should follow the enclosed link to download an electronic card. The link takes the user to a webpage, via a fast flux redirect from one of the botnet drones, hosted on a web server controlled by the blackhat. When the webpage begins to load a JavaScript exploit is used to compromise the computer and install the bot software. If the JavaScript exploit fails, a direct link to the trojan, posing as an ecard, is provided so that the user may download it manually. Launching the program manually has the same effect as if the program was compromised by the JavaScript exploit. A second method of propagation has been through shared drives on a local network. Once a machine has been infected it begins to search, via the ARP protocol, for other computers in the same broadcast domain. The bot sends out gratuitous ARPs and waits for a reply. Once a reply is received, the bot tries to compromise the computer via any network drives that are available.

### 3.2 Roles and Behavior

There are two different types of bots that make up the Stormworm botnet, bots used for sending spam email and bots that host trojan software on HTTP servers. The bots used to distribute spam email, which on run on a Windows OS, are the most common bot in existence. These bots then connect to the botnet and wait for orders from their controller. Of interest about the spam email is not the subject but the format of the email. Instead of using plain text, the email is sent as a blank file with a PDF attachment to avoid detection by spam filters. These bots also act as the front end for the fast flux redirect used to protect the HTTP servers. The link to the trojan in the spam email takes the user to a spam bot acting as a proxy for the HTTP servers. The HTTP servers are the actual crown jewel of the botnet and are heavily protected through fast flux relays and the threat of a distributed denial of service.

### 3.3 Communication

Communication between nodes in the botnet is accomplished using the UDP protocol and a configuration file. The configuration file is used to bootstrap the new bot into the Stormworm Overnet. From the configuration file the bot finds its listen port and a peers list of other bots in the network. The peers list (See Figure 3.3.1) contains a node id in the form of a pseudo-randomly generated MD4 hash for each node known to the bot followed by the node's IP address, its listening port and its Firewall/NAT flag. The Firewall/NAT flag indicates if the node is accessible from a publicly routable IP address. If the machine uses a private IP address or is behind a firewall then it sets this value to 01. This indicates to other nodes not to attempt to contact it. The bot also keeps a blacklist or list of nodes that are bared from communicating with the bot.

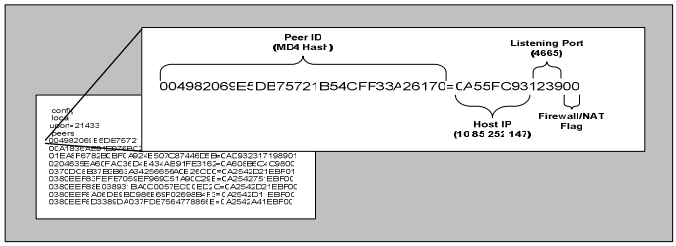


**Figure 3.3.1 – Peers list containg the Peer ID, Host IP, Listening Port & Firewall/NAT flag**

Upon start, the bot sends a PUBLICIZE message to each node listed in the peers list. If the computer is active, it replies with a PUBLICIZE_ACK message. If no response is received the node is removed from the peer list. Once a node has successfully contacted another node it sends a Connect message to the node. The node responds with a CONNECT_ACK message that contains a partial list of the responding node's peers list. The new node then adds this new information to its peers list. Due to the fact that only a subset of the peers list is returned in the CONNECT_ACK, no computer has the complete peers list. Periodically the bot will repeat this process to verify if a node is still active and to refresh its peers list.

In early versions of the Stormworm bot all communication was conducted in clear text. More recent versions have begun to use encryption to hide communication between nodes. Another behavior that may be a defense mechanism is that two bots in the same broadcast domain will not contact each other. This may be done in an effort to avoid detection through network surveillance.

### 3.4 Defense Mechanisms

Besides the encryption mentioned in section 3.3 Stormworm uses several tools to deter detection on the compromised host. First the executable is compressed with a custom packer. This alters the executables signature in an effort to avoid detection by anti-virus solutions. Once a machine has been compromised, the bot installs a root-kit that hides its presence making the bot data files undetectable using system resources. The bot's processing and network activity are also hidden from system resources.

In an effort to defeat live forensic analysis the bot executable can detect if it is running in a virtual computing environment. It accomplishes this by checking the host's registry for signs that it is a virtual machine. If it detects that it is in a virtual machine, the executable will delete its data files and executable file from the hard drive and then exit from running memory.

Just as Stormworm uses a peers list to identify what nodes it may communicate with, it uses a peers blacklist to identify nodes that are off limits for any communication. The bot will not respond to requests from nodes on this list nor will it attempt to communicate with nodes on the list. This list is located in the configuration file that contains the bot's listening port and the regular peers list. Not all versions of the Stormworm bot have used the peers blacklist.

The last form of defense is used strictly by the HTTP servers. If an HTTP server is scanned with a vulnerability assessment tool such as Nessus, the HTTP server will initiate a DDOS attack targeted at the scanning computer.

# 4. BOTNET THREAT TO NATIONAL SECURITY

On April 27, 2007, Estonia removed a war memorial dedicated to Soviet fighters who had died fighting the German invasion during World War II. This event precipitated the first large scale cyber attack against a sovereign state. [S07] For a period of two weeks Estonia withered DDOS attacks against government websites, all major banks, telecoms, media outlets and root DNS servers. Communications for emergency services and first responders failed during the attack, all major news websites were taken down, and ATMs and online banks were knocked off the network. These attacks were also directed at all of Estonia's government ministries including its Ministries of Defense and Foreign affairs. [BBC07] At the height of the attack over 1 million computers slammed Estonia's national internet with over 4 million packets per second, or 200 times the their network's normal load. The botnet that was used for this attack spanned the globe and included a large number of nodes in the United States. This attack crippled Estonia's internet infrastructure and was only abated after Estonia began to block all international traffic from its networks. [D07] Estonia has blamed Russia for the attacks as retaliation for the removal of the war memorial. [BBC07]

Peer-to-peer botnets such as Stormworm drastically increase the threat level as compared to hierarchical botnet models due to their immense scalability and resilience to attack. The botnet that attacked Estonia contained approximately $2^{20}$ nodes.

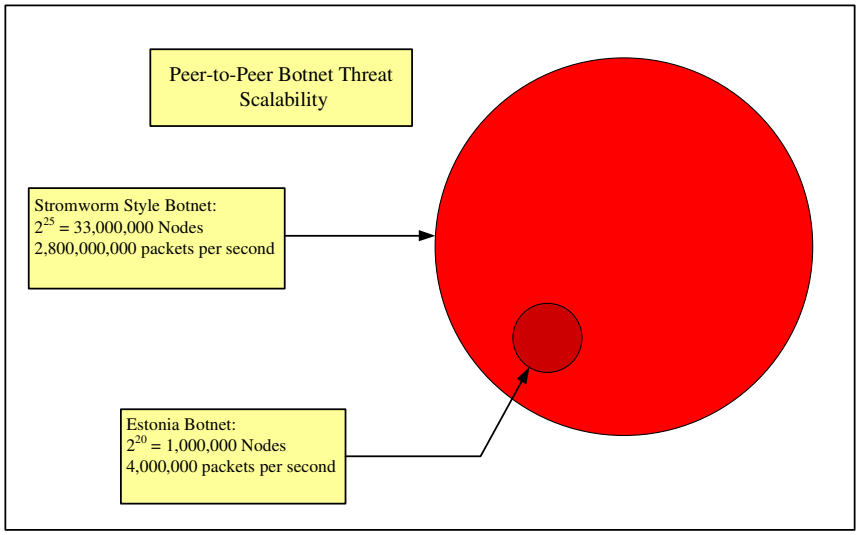


**Figure 4.1: Comparison of Estonia Botnet packets-per-second to a Stormworm Style Botnet with $2^{25}$ nodes.**

It is feasible to see botnets greater than $2^{20}$ nodes in the future as more computers are connected to the internet. If we were to create a botnet modeled after Stormworm we could theoretically have a botnet with $2^{160}$ nodes. However, a botnet of this size is not physically possible due to the limited numbers of computers connected to the internet. It is reasonable however for a Stormworm style botnet containing $2^{25}$ nodes (See Figure 4.1) to be built within the next eight to ten years. If we limit each node to a 100mbit/s network link, our botnet could generate over 2.8 billion packets per second. This is amount of traffic is several magnitudes greater then the estimated 4 million packets per second seen in the Estonia attack.

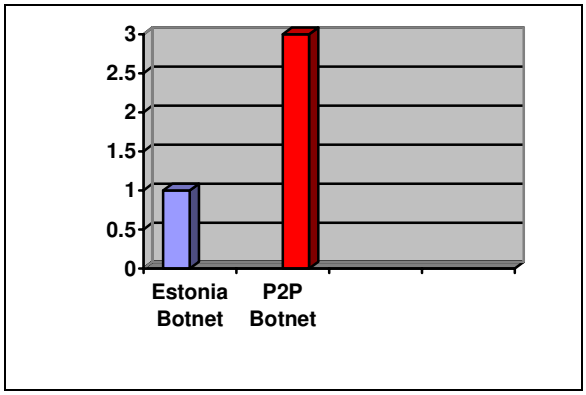


**Figure 4.2: Size in millions of nodes of the Estonia Botnet vs. a Stormworm style botnet after 90% loss**

Even with a loss of 90% of the nodes (See Figure 4.2) our $2^{25}$ node Stormworm style botnet would be larger than the one used to attack Estonia. We would still have three times the number of nodes available to continue the attack giving us a twelve million packets per second attack capability. And since a large number of the bots used to attack Estonia are located in the US, cutting off international internet access such as Estonia did would not be as successful in mitigating our botnet's attack.

The distributed nature of a P2P botnet enhances the anonymity provided to the attacker. The decentralized command and control structure of our botnet means there is no one single point that could indicate a source for the attacks. This combined with the fact that the computers may not even reside in the same hemisphere as the entity that is launching the attack makes it difficult to definitively identify the attacker.

Bot herders use resources that belong to others as their method of attack. Cost of entry is low with freeware versions of botnets readably available. Construction and design of P2P botnet software is well within the capability of most computer science grad students. So the barrier of entry is low and the up keep costs of the botnet are paid by the compromised computer's owners.

This allows those with even modest assets to control a weapon that can shut down a country's network and communications infrastructure.

While most of the damages we have examined have been in the virtual realm it is easy to imagine an attacker using a botnet as part of a larger attack on our country. If the 9/11 attackers had used a botnet in a distributed denial of service attack against government organizations, first responders such as fire and paramedics, media outlets and the military as part of the attack one could speculate that the death toll and destruction could potentially been far greater. Another scenario that is well within the realm of possibility is if China decided to invade Taiwan. A botnet of substantial size could be used to not only attack the Taiwan network infrastructure but also as a means of blocking news of the attack from reaching Taiwan's allies until well after the physical attack had been launched.

## 5. CONCLUSION

In this paper I have shown the original research that I conducted on a new type of botnet, Stormworm. This includes the original discovery of a defense mechanism used by Stormworm, the peers blacklist. I have shown that the newest breed of botnet based on peer-to-peer technology through its scalability and robustness against attack, forms a threat to national security magnitudes greater than botnets used in previous attacks against nation states. Further, we have seen that due to its distributed nature it hides the identity and location of the attacker providing anonymity to its owner. Last, we have seen how the barrier to development for a peer-to-peer based botnet such as Stormworm is low due to the low cost of startup and maintenance and the availability of proven protocols such as Kademlia to base the software upon.

With a large number of infected computers within our own boarder we have to not only prepare for attack from outside but also from within. The United States is one of the most wired countries in the world. However our borders are unprotected and our government agencies, which regularly receive low marks for cyber security, are ill prepared to defeat this threat. With increasing reliance on networks for communication, commerce and government activities this threat grows stronger.

Future research needs to not only focus on how to mitigate the impacts of these types of attacks, but on how to defeat the botnets that launch the attacks. We have shown that botnets can scale their attacks to match any type of defense that involves adding larger resources. This negates the current doctrine of adding more bandwidth, more CPUs and stronger firewalls to cope with large scale DDOS attacks. Instead we should focus on how we can defeat, disrupt and potentially destroy the botnets themselves.

## REFERNECES

[BY06] P. Barford, V. Yegneswaran, "An Inside Look at Botnets" In Series: Advances in Information Security, Springer, 2006, ISBN-10: 0-387-32720-7

[BBC07] BBC News, "Estonia hit by 'Moscow cyber war", BBC News, May 17, 2007

[CJM05] F. Jahanian, D. McPherson, "The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets", Proc. of Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI'05), July 2005

[D07] J. Davis, "Hackers Take Down the Most Wired Country in Europe", Wired Magazine, Issue 15.09, August, 2007.

[DM03] C. Douligeris, A. Mitrokotsa, "DDOS Attacks and defense Mechanisms: A Classification", ISSPIT 2003, December 2003

[DZL06] D. Dagon, C. Zou, W. Lee "Modeling Botnet Propagation Using Time Zones" NDSS 2006, The Internet Society, 2006

[FHW05] F. Freiling, T. Holz, George Wicherski, "Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks" RWTH Aachen, Department of Computer Science, 2005

[G04] V.D. Gligor, "A not on the denial-of-service in operating systems." IEEE Trans Softw. Eng. 10, 3, 320 – 324, 1984
[HPRA07] The Honeynet Project and Research Alliance, "Know You Enemy: Fast-Flux Service Networks", http://www.honeynet.org, July 2007

[LEKR05] J. Li, T. Ehrenkranz, G. Kuenning, P. Reiher, "Simulation and Analysis on the Resiliency and Efficiency of Malnets", Procedings of the Workshop of Principles of Advanced and Distributed Simulation, 2005

[M06] K. McDowell, "Now That We Are All So Well-Educated about Spyware, Can We Put the Bad Guys out of Business?", SIGUCCS'06, November 5-8, 2006, ACM 2006

[MM02] P. Maymounkov and D. Mazieres, "Kademlia: A Peer-to-peer Information System Based on the XOR Metric", IPTPS'02, March 2002.

[OR93] J. Oikarinen and D. Reed, "RFC 1459: Internet Relay Chat Protocol" , 1993

[P03] R. Puri, "Bots & Botnets: An Overview" SANS Institute 2003

[PLR07] T. Peng, C Leckie, K. Ramaohanarao, “Survey of Network-Based Defense Mechanisms Countering the DoS and DDoS Problems.”, ACM Computer Surveys, Vol. 39, No. 1, Article 3, April 2007

[RZMT06] M. Rajab, J. Zarfoss, F. Monrose, A. Terzis, “ A Multifaceted Approach to Understanding the Botnet Phenomenon” IMC’06, October 25 – 27, 2006, Rio de Janeiro, Brazil, ACM 2006

[RF05] A. Ramachandran, N. Feamster, “Understanding the Network-Level Behavior of Spammers”, SIGCOMM’06, September 11-15, 2006, ACM 2006

[SANS05] The SANS Internet Storm Center, http://isc.sans.org/diary.html?storyid=778

[S06] G. Schaffer, “Worms and Viruses and Botnets, Oh My!, Rational Responses to Emerging Internet Threats”, IEEE Security & Privacy. IEEE Computer Society, 2006

[S07] P. Schrank, “Newly Nasty”, The Economist, May 24th, 2007.

[SPW02] S. Staniford, V. Paxson, N. Weaver, “How to 0wn the Internet in Your Spare Time”, Proceedings of the 11th USENIX Security Symposium (Security '02) 2002

[WP05] B. Krebs, “Adware Firm Accuses 7 Distributors of Using ‘Botnets’: Lawsuit Claims Defendants Spread Pop-Ups via Hacked PCs.” washingtonpost.com, 16 Aug. 2005; www.washingtonpost.com/wp-dyn/content/artticle/2005/08/16/AR2005081600727.html

[ZC06] C. Zou, R. Cunningham, “Honeypot-Aware Advanced Botnet Construction and Maintenance.”, Proceedings of the 2006 International Conference on Dependable Systems and Networks, IEEE, 2006

## BIOGRAPHICAL SKETCH

**Louis Brooks**

**Certifications:**
**-** GIAC Certified Incidence Handler

**Employment and Related Experience:**
***Network Security Analyst –*** August 2006 – Present. Florida State University, IT Security Team.

***Systems Administrator*** - August 2003 – August 2006 – Security & Assurance in Information Technology Laboratories, Department of Computer Science, Florida State University.

***Associate MIS Coordinator -*** May 2002 – April 2003, Big Bend Hospice, Tallahassee, Florida

***MIS Assistant*** - April 2001 – May 2002, Big Bend Hospice, Tallahassee, Florida

***Chapter Technical Support / Director of Program Services / Florida Public Affairs Liaison*** - February 1999 – June 1999, Florida Department of Health, Legislative Planning Office, Tallahassee.

**Education:**

**Master's Degree – Florida State University, Department of Computer Science,** Tentative 2008 Computer Security Track.

**Network Management Certificate Program – Tallahassee Community College,** 2002. (3.4 GPA.) Courses include Windows 2000 server, Cisco Networking Academy, Linux, General Networks, Microcomputer Architecture, C++ programming, cabling, troubleshooting and Microsoft Network Infrastructure.

**Master's Degree – Florida State University, School of Social Work,** 1998 (3.5 overall GPA). Concentration in Social Policy and Program Administration. Courses in grant writing, program and policy analysis, statistics, budgeting, and program administration.

**Bachelor's of Science – Florida State University, School of Social Work,** April 1997 (Dean's List)

**AA Degree – Tallahassee, Community College,** April 1994