Taxonomies of Distributed Denial of Service Networks, Attacks, Tools, and Countermeasures

Stephen Specht
sspecht@princeton.edu

Ruby Lee
rblee@princeton.edu

Department of Electrical Engineering
Princeton Architecture Laboratory for Multimedia and Security

Technical Report CE-L2003-03

May 16, 2003

Abstract

Distributed Denial of Service (DDoS) attacks have become a large problem for users of computer systems connected to the Internet. DDoS attackers hijack secondary victim systems using them to wage a coordinated large-scale attack against victim systems. As new countermeasures are developed to prevent DDoS attacks and help systems that are victims of such an attack, attackers are constantly developing new software and adapting older DDoS attack tools to circumvent these new countermeasures.

In this paper we describe the taxonomies of DDoS attacks, the software tools used to wage a DDoS attack and the countermeasures available. These different taxonomies are presented so that similarities and patterns within DDoS attacks and tools can be better understood. This work is intended to assist in developing solutions that will provide a more generalized approach to countering DDoS attacks so that as new derivative attacks are developed a generic countermeasure model can be used to prevent secondary victims and stop DDoS attacks.

1 Introduction

A Denial of Service (DoS) attack can be characterized as an attack with the purpose of preventing legitimate users from using a specified network resource such as a website, web service, or computer system [1]. A Distributed Denial of Service (DDoS) attack is a coordinated attack on the availability of services of a given target system or network launched indirectly through many compromised computing systems. The services under attack are those of the "primary victim", while the compromised systems used to launch the attack are often called the "secondary victims." The use of secondary victims in a DDoS attack provides the attacker with the ability to wage a much larger and more disruptive attack than a DoS attack while remaining anonymous since the secondary victims actually complete the attack making it more difficult for network forensics to track down the original attacker. As defined by the World Wide Web Security FAQ: *A Distributed Denial of Service (DDoS) attack uses many computers to launch a coordinated DoS attack against one or more targets. Using client/server technology, the perpetrator is able to multiply the effectiveness of the Denial of Service significantly by harnessing the resources of multiple unwitting accomplice computers which serve as attack platforms.*[2] According to the

This work was supported in part by the NSF CCR-0208946.

CIAC (Computer Incident Advisory Capability), the first DDoS attacks occurred in the summer of 1999 [3].

In February of 2000, one of the first major DDoS attacks was waged against Yahoo.com and kept Yahoo off of the Internet for about 2 hours. This attack cost Yahoo.com lost advertising revenue [4]. Another recent DDoS attack occurred on October 20, 2002 against the 13 root servers that manage the Internet. These root servers provide the Domain Name System (DNS) to Internet users around the world. They translate logical addresses such as www.princeton.edu into a physical IP address so that computers can connect to the websites. If all 13 servers were to go down, there would be noticeable problems accessing the World Wide Web. Although the attack only lasted for an hour and the effects were hardly noticeable to the average Internet user, it caused 7 of the 13 root servers to shut down, demonstrating the vulnerability of the Internet to DDoS attacks [5].

The contributions of this paper include the first taxonomies proposed for the different DDoS attacks, tools, and countermeasures. DDoS attacks are relatively new and not at all well understood. By classifying the types of DDoS attacks, the characteristics of the DDoS software tools, and the space of possible countermeasures, we hope to aid significantly in understanding the scope of DDoS attacks. This understanding can help to produce comprehensive solutions or countermeasures to cover both known attacks and those that have not yet occurred. Additionally, this paper is the first to characterize the setup and installation techniques of DDoS attack architectures, identifying both active and passive classes.

In Section 2 we describe the classes of DDoS attack architectures. In Section 3 we present our taxonomy for DDoS attacks. In Section 4 we present an overview of the software characteristics for DDoS attack tools with an emphasis of how these tools are setup on secondary victim systems. In Section 5 we present an overview of the commands issued to and by the DDoS attack tools. In Section 6 we present a brief description of some of the more common DDoS attack tools. In Section 7 we present a taxonomy of the different countermeasures that are available to prevent DDoS attacks. In Section 8 we conclude the paper with suggestions for future work on how these taxonomies can be used to develop generic and comprehensive DDoS countermeasures.

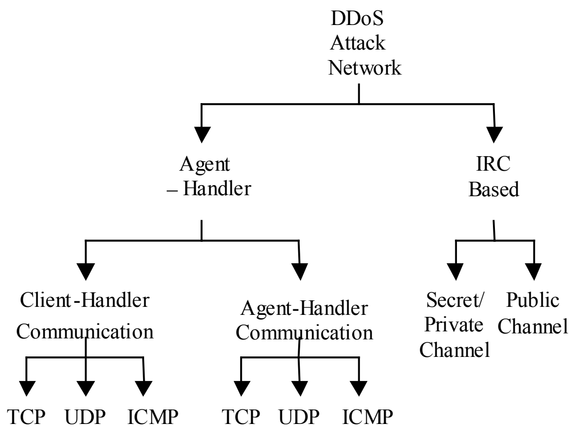
2 DDoS Attack Networks

There are two types of DDoS attack networks: the Agent-Handler model and the Internet Relay Chat (IRC)-Based model.

2.1 Agent-Handler Model

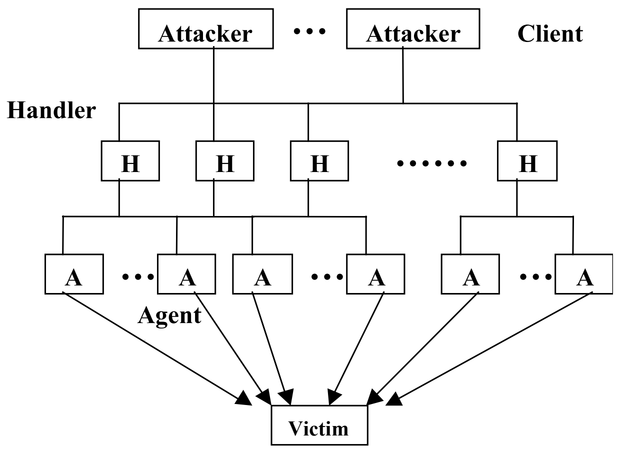
The Agent-Handler model of a DDoS attack consists of clients, handlers, and agents (see Figure 2). The client is where the attacker communicates with the rest of the DDoS attack system. The handlers are software packages located throughout the Internet that the attacker's client uses to communicate with the agents. The agent software exists in compromised systems that will eventually carry out the attack. The attacker communicates with any number of handlers to identify which agents are up and running, when to schedule attacks, or when to upgrade agents. The owners and users of the agent systems typically have no knowledge that their system has been compromised and will be taking part in a DDoS attack. Depending on how the attacker configures the DDoS attack network, agents can be instructed to communicate with a single handler or multiple handlers. Usually, attackers will try

Figure 1: DDoS Attack Network



and place the handler software on a compromised router or network server that handles large volumes of traffic. This makes it harder to identify messages between the client and handler and between the handler and agents. The communication between attacker and handler and between the handler and agents can be via TCP, UDP, or ICMP protocols. In descriptions of DDoS tools, the terms handler and agents are sometimes replaced with master and daemons respectively.

Figure 2: DDoS Agent-Handler Attack Model



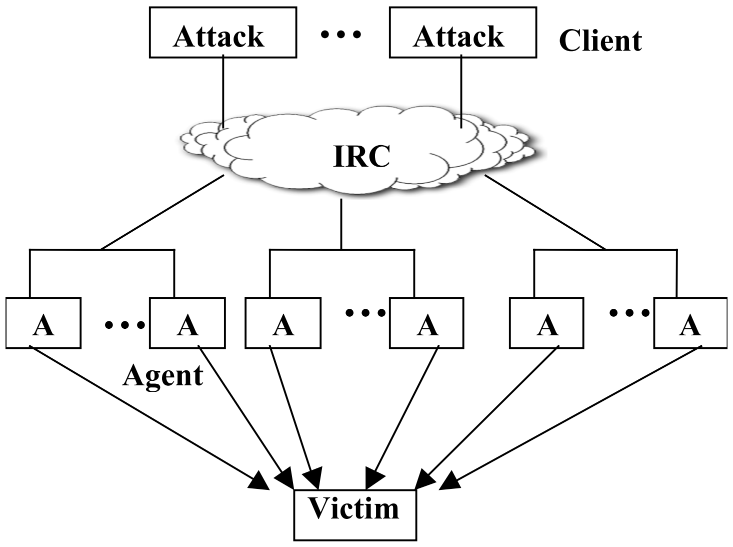
In a DDoS attack network, the systems that have been violated to run the agent software are referred to as the secondary victims. The primary victim is the system that is the target of the DDoS attack. Each agent program uses only some resources (both in memory and bandwidth) when participating in an attack. However, well-designed agent software use up a small proportion of resources so that the secondary-victim users experience minimal change in their system performance.

2.2 IRC-Based DDoS Attack Model

Internet Relay Chat (IRC) is a multi-user, on-line chatting system. It allows computer users to create two-party or multi-party interconnections and type messages in real time to each other [6]. IRC network architectures consist of IRC servers that are located throughout the Internet with channels to communicate with each other across the Internet. IRC chat networks allow their users to create public, secret, and private channels. Public channels are channels where multiple users can chat and share messages and files. Public channels allow users of the channel to see all the IRC names and messages of users in the channel [7]. Private and secret channels are set up by users to communicate with only other designated users. Both private and secret channels protect the names and messages of users that are logged on from users who do not have access to the channel [8]. Although the content of private channels is hidden, certain channel locator commands will allow users not on the channel to identify its existence whereas secret channels are much harder to locate unless the user is a member of the channel.

IRC-Based DDoS attack architecture is similar to the Agent-Handler DDoS attack model except that instead of using a handler program installed on a network server, an IRC communication channel is used to connect the client to the agents. By making use of an IRC channel, attackers using this type of DDoS attack architecture have additional benefits. For example, attackers can use “legitimate” IRC ports for sending commands to the agents [9]. This makes tracking the DDoS command packets much more difficult. Additionally, IRC servers tend to have large volumes of traffic making it easier for the attacker to

Figure 3 : DDoS IRC-Based Attack Model



hide his presence from a network administrator. Another advantage is that the attacker no longer needs to maintain a list of all of the agents, since he can simply log on to the IRC server and see a list of all available agents [9]. The agent software installed in the IRC network usually communicates to the IRC channel and notifies the attacker when the agent is up and running. IRC networks also provide the added benefit of easy file sharing. File sharing is one of the passive methods of agent code distribution that we discuss in Section 4. This makes it easier for attackers to secure secondary victims to participate in their attacks.

In an IRC-based DDoS attack architecture, the agents are often referred to as "Zombie Bots" or "Bots". In both IRC-based and Agent-Handler DDoS attack models, we will refer to the agents as "secondary victims" or "zombies."

3. DDoS Attack Taxonomy

There are a wide variety of DDoS attack techniques. We propose a taxonomy of the main DDoS attack methods in Figure 4. There are two main classes of DDoS attacks: bandwidth depletion and resource depletion attacks. A bandwidth depletion attack is designed to flood the victim network with unwanted traffic that prevents legitimate traffic from reaching the (primary) victim system. A resource depletion attack is an attack that is designed to tie up the resources of a victim system. This type of attack targets a server or process on the victim system making it unable to process legitimate requests for service.

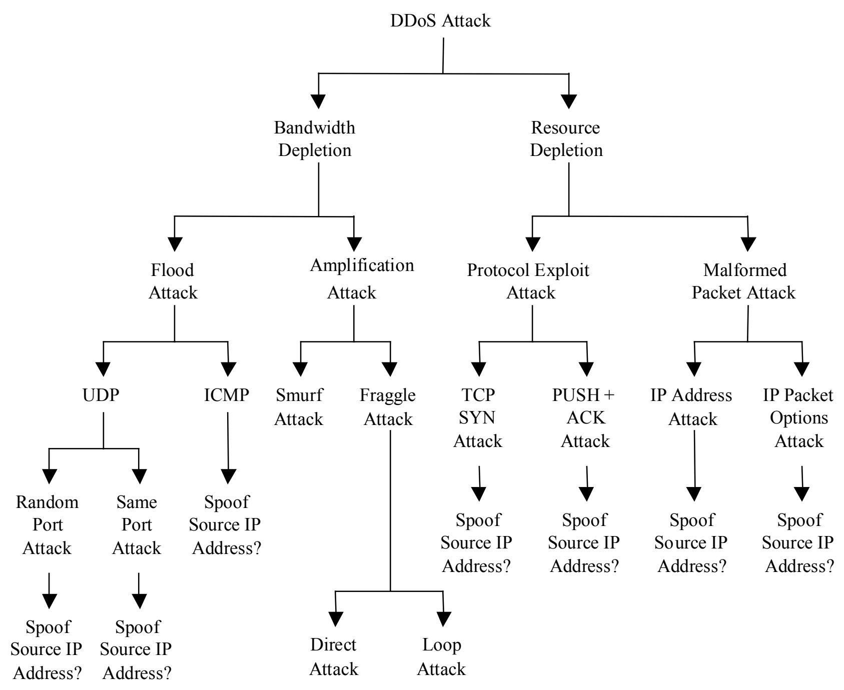
3.1 Bandwidth Depletion Attacks

There are two main classes of DDoS bandwidth depletion attacks. A flood attack involves the zombies sending large volumes of traffic to a victim system, to congest the victim system's bandwidth. An amplification attack involves either the attacker or the zombies sending messages to a broadcast IP address, using this to cause all systems in the subnet reached by the broadcast address to send a reply to the victim system. This method amplifies malicious traffic that reduces the victim system's bandwidth.

3.1.1 Flood Attacks

In a DDoS flood attack the zombies flood the victim system with IP traffic. The large volume of packets sent by the zombies to the victim system slows it down, crashes the system or saturates the network bandwidth. This prevents legitimate users from accessing the victim's resources. Figures 1 and 2 indicate a flood attack for an Agent-Handler attack architecture and an IRC-based attack architecture.

Figure 4: DDoS Attack Taxonomy



UDP Flood Attacks. User Datagram Protocol (UDP) is a connectionless protocol. When data packets are sent via UDP, there is no handshaking required between sender and receiver, and the receiving system will just receive packets it must process. A large number of UDP packets sent to a victim system can saturate the network, depleting the bandwidth available for legitimate service requests to the victim system.

In a DDoS UDP Flood attack, the UDP packets are sent to either random or specified ports on the victim system. Typically, UDP flood attacks are designed to attack random victim ports. This causes the victim system to process the incoming data to try to determine which applications have requested data. If the victim system is not running any applications on the targeted port, then the victim system will send out an ICMP packet to the sending system indicating a "destination port unreachable" message [3].

Often, the attacking DDoS tool will also spoof the source IP address of the attacking packets. This helps hide the identity of the secondary victims and it insures that return packets from the victim system are not sent back to the zombies, but to another computer with the spoofed address.

UDP flood attacks may also fill the bandwidth of connections located around the victim system (depending on the network architecture and line-speed). This can sometimes cause systems connected to a network near a victim system to experience problems with their connectivity.

ICMP Flood Attacks. Internet Control Message Protocol (ICMP) packets are designed for network management features such as locating network equipment and determining the number of hops or round-trip-time to get from the source location to the destination. For instance, ICMP_ECHO_REPLY packets ("ping") allow the user to send a request to a destination system and receive a response with the roundtrip time.

A DDoS ICMP flood attack occurs when the zombies send large volumes of ICMP_ECHO_REPLY packets to the victim system. These packets signal the victim system to reply and the combination of traffic saturates the bandwidth of the victim's network connection [3]. As for the UDP flood attack, the source IP address may be spoofed, so that the return ICMP packets from the victim system are not sent back to the zombies, but to another system.

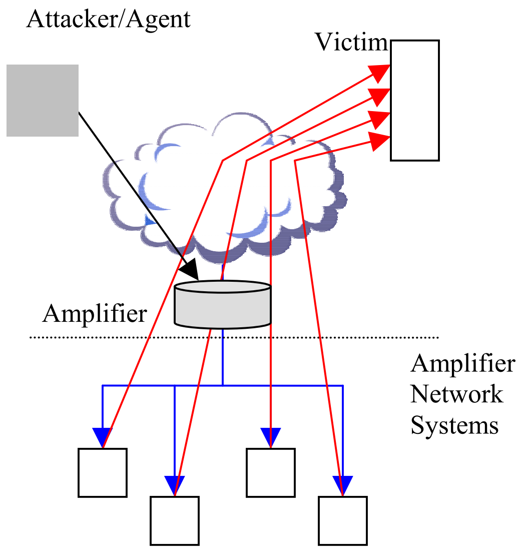
3.1.2 Amplification Attacks

A DDoS amplification attack is aimed at using the broadcast IP address feature found on most routers to amplify and reflect the attack (see figure 5). This feature allows a sending system to specify a broadcast IP address as the destination address rather than a specific address. This instructs the routers servicing the packets within the network to duplicate the packets and send them to all the IP addresses within the broadcast address range.

For this type of DDoS attack, the attacker can send the broadcast message directly, or the attacker can use the agents to send the broadcast message to increase the volume of attacking traffic. If the attacker decides to send the broadcast message directly, this attack provides the attacker with the ability to use the systems within the broadcast network as zombies without needing to infiltrate them or install any agent software. We further distinguish two types of amplification attacks, Smurf and Fraggle attacks.

Smurf Attacks. In a DDoS Smurf attack, the attacker sends packets to a network amplifier (a system supporting broadcast addressing), with the return address spoofed to the victim's IP address. The attacking packets are typically ICMP ECHO REQUESTs, which are packets (similar to a "ping") that request the receiver to generate an ICMP ECHO REPLY packet [10]. The amplifier sends the ICMP ECHO REQUEST packets to all of the systems within the broadcast address range, and each of these systems will return an ICMP ECHO REPLY to the target victim's IP address [11]. This type of attack amplifies the original packet tens or hundreds of times.

Figure 5: Amplification Attack



Fraggle Attacks. A DDoS Fraggle attack is similar to a Smurf attack in that the attacker sends packets to a network amplifier. Fraggle is different from Smurf in that Fraggle uses UDP ECHO packets instead of ICMP ECHO packets [12]. There is a variation of the Fraggle attack where the UDP ECHO packets are sent to the port that supports character generation (chargen, port 19 in Unix systems), with the return address spoofed to the victim's echo service (echo, port 7 in Unix systems) creating an infinite loop [13]. The UDP Fraggle packet will target the character generator in the systems reached by the broadcast address. These systems each generate a character to send to the echo service in the victim system, which will resend an echo packet back to the character generator, and the process repeats. This attack generates even more bad traffic and can create even more damaging effects than just a Smurf attack.

3.2 Resource Depletion Attacks

DDoS resource depletion attacks involve the attacker sending packets that misuse network protocol communications or sending malformed packets that tie up network resources so that none are left for legitimate users.

3.2.1 Protocol Exploit Attacks

TCP SYN Attacks. The Transfer Control Protocol (TCP) includes a full handshake between sender and receiver, before data packets are sent. The initiating system sends a SYN (Synchronize) request (see figure 6a). The receiving system sends an ACK (acknowledgement) with its own SYN request. The sending system then sends back its own ACK and communication can begin between the two systems. If the receiving system is sent a SYN_X packet but doesn't receive an ACK_{Y+1} to the SYN_Y it sends back to the sender, the receiver will resend a new ACK + SYN_Y after some time has passed [14] (see figure 6b). The processor and memory resources at the receiving system are reserved for this TCP SYN request until a timeout occurs.

In a DDoS TCP SYN attack, the attacker instructs the zombies to send such bogus TCP SYN requests to a victim server in order to tie up the server's processor resources, and hence prevent the server from responding to legitimate requests. The TCP SYN attack exploits the three-way handshake between the sending system and the receiving system by sending large volumes of TCP SYN packets to the victim system with spoofed source IP addresses, so the victim system responds to a non-requesting system with the ACK+SYN. When a large volume of SYN requests are being processed by a server and none of the ACK+SYN responses are returned, the server begins to run out of processor and memory resources. Eventually, if the volume of TCP SYN attack requests is large and they continue over time, the victim system will run out of resources and be unable to respond to any legitimate users.

Figure 6a: TCP Synchronization

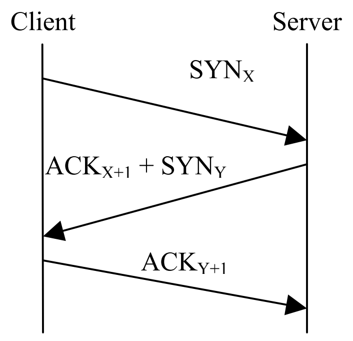
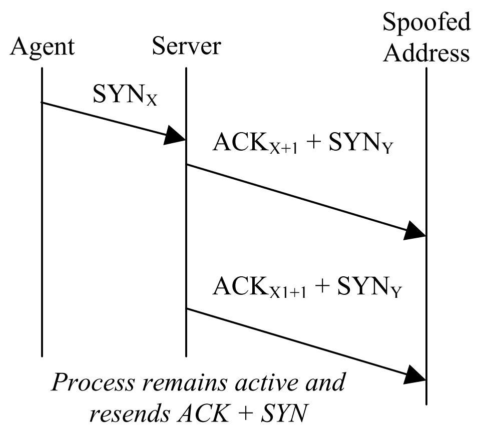


Figure 6b: TCP SYN Attack



PUSH + ACK Attacks. In the TCP protocol, packets that are sent to a destination are buffered within the TCP stack and when the stack is full, the packets get sent on to the receiving system. However, the sender can request the receiving system to unload the contents of the buffer before the buffer becomes full by sending a packet with the PUSH bit set to one. PUSH is a one-bit flag within the TCP header [15]. TCP stores incoming data in large blocks for passage on to the receiving system in order to minimize the processing overhead required by the receiving system each time it must unload a non-empty buffer.

The PUSH + ACK attack is similar to a TCP SYN attack in that its goal is to deplete the resources of the victim system. The attacking agents send TCP packets with the PUSH and ACK bits set to one. These packets instruct the victim system to unload all data in the TCP buffer (regardless of whether or not the buffer is full) and send an acknowledgement when complete. If this process is repeated with multiple agents, the receiving system cannot process the large volume of incoming packets and the victim system will crash.

3.2.2 Malformed Packet Attacks

A malformed packet attack is an attack where the attacker instructs the zombies to send incorrectly formed IP packets to the victim system in order to crash the victim system. There are two types of malformed packet attacks. In an IP address attack, the packet contains the same source and destination IP addresses. This can confuse the operating system of the victim system and cause the victim system to crash. In an IP packet options attack, a malformed packet may randomize the optional fields within an IP packet and set all quality of service bits to one so that the victim system must use additional processing time to analyze the traffic. If this attack is multiplied using enough agents, it can shut down the processing ability of the victim system.

4 Software Characteristics of DDoS Attack Tools

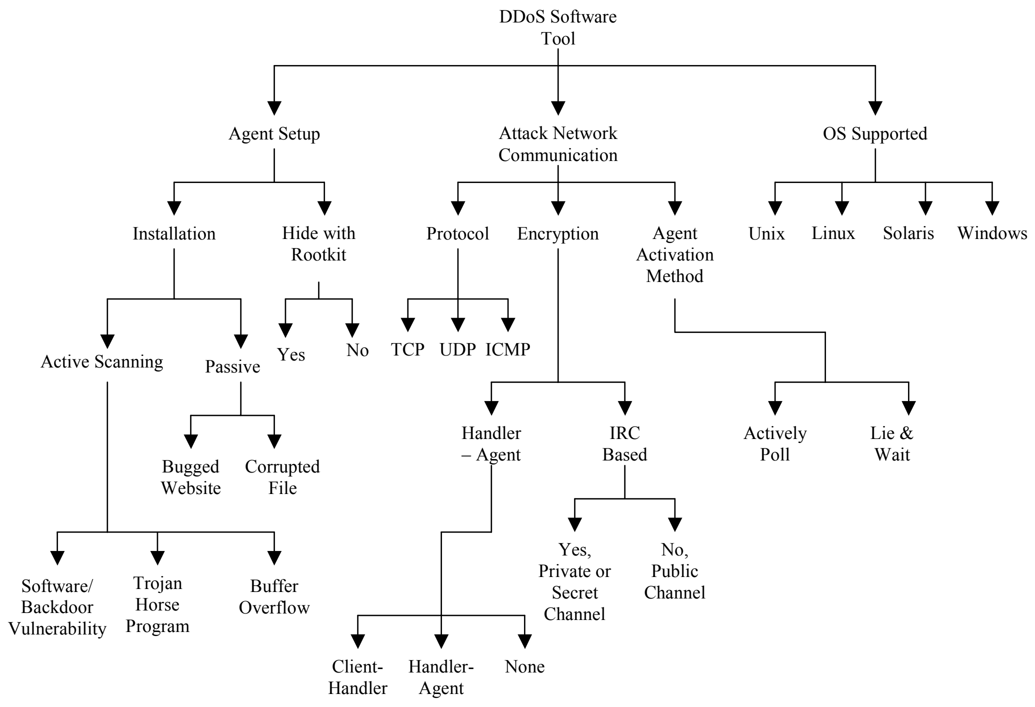
There are a number of software characteristics that are common among DDoS attack tools. These common elements include how agents are setup, agent activation, the operating systems (OS) supported, and whether the communication within the DDoS attack architecture is encrypted. These characteristics can be described in a taxonomy tree as shown in Figure 7.

4.1 DDoS Agent Setup

There are both active and passive methods that attackers use to install malicious code onto a secondary victim system in order to execute a DDoS attack in either the Agent-Handler or the IRC-Based DDoS attack architectures.

Active methods typically involve the attacker scanning the network for systems with known vulnerabilities. Upon identifying such vulnerable systems, the attacker runs scripts to break into the system. Once the attacker has broken into the system, he can stealthily install the DDoS Agent software. Thus the system is compromised as a secondary victim, which can be used as a zombie in a future DDoS attack.

Figure 7: DDoS Software Tools (Characteristics)



Passive methods typically involve the attacker sharing corrupt files or building web sites that take advantage of known vulnerabilities in a secondary victim's web browser. Upon accessing a file or website with an embedded DDoS Agent, the secondary victim system is compromised, and the DDoS agent code may be installed.

4.1.2 Active DDoS Installation

Scanning. Before launching a DDoS attack, attackers must first set up the DDoS attack network. They often run a scanning tool to identify potential secondary victim systems. One common tool attackers use to scan for ports is a software program called Nmap. Attackers can download Nmap from various locations on the web (for example www.insecure.org/nmap/). This tool allows attackers to select ranges of IP addresses to scan. The tool will then proceed to search the Internet for each of these IP addresses. Nmap returns the information that each IP address is broadcasting such as TCP and UDP ports that are open, and the specific OS of the scanned system [16]. An attacker can then examine this list for potential secondary victim systems.

Another tool for scanning the network finds random IP addresses with a known vulnerability. This provides the attacker with a list of victim systems that all share the same common vulnerability. One example of this type of vulnerability scan tool is called Nessus [17]. We describe three types of active DDoS agent setup techniques, as examples.

Software/Backdoor Vulnerability. Once the attacker has scanned for a list of vulnerable systems, he will need to exploit the vulnerability to gain access to the secondary victim system and install the DDoS agent code. There are many sources on the Internet, such as the Common Vulnerabilities and Exposures (CVE) organization, which publicly list all of the known vulnerabilities of different systems. CVE has currently categorized over 2,000 different types of vulnerabilities and they have over 2,000 more waiting for consideration [18]. This research information is available so network administrators can make their systems more secure; however, it also provides attackers with data about which vulnerabilities exist.

One such vulnerability was first reported in November of 2001 by CERT. The vulnerability reported was that the Kaiten IRC-Based DDoS agent software was being installed on Microsoft SQL Servers by making use of a known default password. Attackers could scan for hosts connected with TCP port 1433 (the MS SQL Server port), and find systems where they can then log on to MS SQL Servers using a default administrator password. From this administrator account, the attacker can utilize the xp_cmdshell procedure from the MS SQL Server to initiate an FTP session to download the agent software on the MS SQL Server [19]. Any MS SQL Server that has not had the administrator's password changed or been upgraded with software patches to prevent the default administrator password could be victims of this type of attack.

Trojan Horse Program. A Trojan horse is a program that appears to perform a useful function, but in reality contains hidden code that either executes malicious acts or provides a trap door for unauthorized access to some privileged system function [20]. Trojan horse programs are installed on a victim's system by the attacker and allow the attacker to gain control of a user's computer without the user knowing. In the case of a DDoS attack tool setup, Trojan horse programs already installed on a victim system might be used by the attacker to gain access to a secondary victim's system allowing the attacker to install the DDoS agent code. Trojan Horse programs are typically installed on a secondary victim's system by using the passive setup techniques discussed in Section 4.1.2.

Buffer Overflow. Another vulnerability is the buffer overflow problem. A buffer is a continuous block of memory (with a finite size) that serves as a temporary data storage area within a computer. A buffer overflow is an attack against the buffer that sends more data into the buffer than the size of the buffer. This causes the extra data to overwrite other information adjacent to the buffer [21] in the memory storage stack, such as a procedure return address. This can cause the computer to return from a procedure call to malicious code included in the data that overwrites the buffer. This malicious code can be used to start a program of the attacker's choosing (such as a DDoS Agent) or provide access to the victim's computer so that the attacker can install the DDoS Agent code.

4.1.1 Passive DDoS Installation

Bugged Web Site. One method attackers can use to passively infiltrate a secondary victim computer system is to take advantage of a vulnerability found on web browsers. This method allows the attacker to create websites with code or commands to trap a victim. When the victim's web browser views the web page or tries to access content, the web page indirectly downloads or installs malicious code (e.g., a DDoS Agent.) One example of this type of attack exploits a bug in Microsoft's Internet Explorer (IE) versions 5.5 and 6.0. These versions of IE contain ActiveX, a technology developed by Microsoft to enable control within IE for viewing specific plug-in applications embedded within website code. ActiveX controls can be embedded within a website and allow the IE web browser to automatically download client binary code specified by the website being viewed [22].

An attacker can build malicious code into a web page that can take advantage of ActiveX. Instead of downloading client software for viewing the web page, the attacker can set up ActiveX to download a DDoS agent. The attacker will typically post a website with the malicious ActiveX code somewhere on the Internet to attract victim systems. The malicious html code is used to reference an ActiveX installation package that an IE browser will think is legitimate but actually contains code, such as DDoS agent software or code that allows the attacker to infiltrate the system. The malicious html code could include the DDoS agent, and takes the form:

```
<object classid="clsid:XXXXXXXX" codebase= http://www.webpage.com/myactivex.cab> </object>
```

This command instructs IE to use an ActiveX control with a GUID (GUI Class Identifier) "XXXXXXXX" and if that control has not been downloaded, it provides the address where the control can be downloaded [23]. The download address actually contains the malicious code the attacker wants to install on the victim machine. If the software patch for IE to prevent this problem is not installed, IE will inadvertently download the malicious code in place of the legitimate code. The attacker has now installed the malicious code on the victim's computer. If this code is a DDoS agent, the attacker has now created a secondary victim system that can be invoked for a future DDoS attack.

Corrupted File. Another method of a passive attack that is commonly used is to alter files and include malicious code embedded within them. When the victim system tries to view or execute these files, they will become infected with the malicious code.

There are many tricks to creating infected files. Most attackers are skilled enough to embed a DDoS attack agent or other virus software within a legitimate file. The attackers redesign the Desktop icons for such files, choosing long files names with legitimate extensions interwoven within the filename so that if only part of the file name is displayed, it will appear like a legitimate filename. For instance, one popular technique is for attackers to generate a text file with the binary executable code for a DDoS agent embedded within it. They rename the text file with a very long name and the .txt extension within the name when the real extension is .exe. For instance, the file might be *newfile.txt_this_file_is_really_a_ddos_agent.exe*. If only the first few characters of the file are displayed to the user, it will appear as if this file is really a text file, not an executable file. In this example, the *newfile* name would need to be around 150 chars long so most windows systems would not show the full file name [24]. As soon as a user launches the file, they will become infected with the DDoS agent software. Some attackers are skilled enough to include a legitimate text box to open, so the victim will think the file was legitimate and will not realize that the file contained the DDoS agent.

Corrupt files can be exchanged in a variety of manners. Currently, IRC file sharing and Gnutella networks are two popular file-sharing methods that make it easy for a corrupt file to circulate to many users. An attacker can also send e-mail with corrupt files to victims, hoping the victims will open the files and infect themselves with the DDoS agent code.

4.1.2 Root kits

Root kits are programs that are used by the attacker after installation of handler and/or agent software to remove log files and any other records that might indicate that the attacker was using the system [25]. Attackers may additionally use the root kit tools to create "back-doors" so that they will be able to access the victim's system in the future [26]. Root kit tools are typically used when handler software is installed since one handler can be critical for the DDoS network to work and since handler programs are usually installed within ISP or corporate networks where the possibility of detection may be higher. In comparison, the effort to use a root kit on all of the agents may be time prohibitive and less important since secondary victims are less likely to be aware of the agent software and if some of the agents are discovered, their loss does not impact the DDoS network significantly.

4.2 Attack Network Communication

4.2.1 Protocols Used

The DDoS agents and handlers can communicate to each other via TCP, UDP, and/or ICMP. DDoS handlers and clients can also communicate with each other using the same protocol options.

4.2.2 Encrypted Communication

Some DDoS attack tools have also been developed with support for encrypted communication within the DDoS attack network. Agent-handler DDoS attacks might use an encrypted channel either between the client and the handlers, or between the handlers and the agents. The method of encryption for agent-handler DDoS attacks will be dependent on the communication protocol used by the DDoS tool. IRC-based DDoS attacks may use either a public, private, or secret channel to communicate between the agents and the handlers. Both private and secret IRC channels provide encryption, however private channels (not the data or users) appear in the IRC server's channel list and secret channels do not appear in the IRC server's channel list.

4.2.3 Agent Activation

There are two key methods for the DDoS agents to become active. In some DDoS tools, the agents actively poll the handlers or IRC channel for instructions, whereas in other DDoS tools, the agents will lie and wait for communication from either the handler or the IRC channel.

4.3 OS Supported

DDoS attack tools are typically designed to be compatible with different operating systems (OS). Any OS system (such as Unix, Linux, Solaris, or Windows) may have DDoS agent or handler code designed to work on it. Typically, the handler code is designed to support an OS that would be located on a server or workstation at either a corporate or ISP site. This usually leads to the choice of Unix, Linux, or Solaris. For the agent code, it is also common for it to be compatible with Linux or Solaris with the addition of Windows. Many attackers target residential Internet users with DSL and cable modems (for higher attacking bandwidth) and these users typically use Windows. Agent and handler software can be written for different operating systems.

5 DDoS Attack Software Commands

Each DDoS attack tool has a number of commands. These commands are designed for both the handler and agent software packages. For most DDoS agents, specific commands are entered via a command line interface. Although the specific commands differ for each DDoS attack tool, we generalize these commands as listed in Tables 1 and 2.

Table 1: Handler Commands

Command	Description
Turn On	Instructs the handler to turn on and wait for other commands.
Turn Off	Instructs the handler to shut down. If the handler was actively polling the network looking for agents this command stops the handler from continuing this action.
Initiate Attack	Instructs the handler to contact all agents and have them launch an attack against a specified target.
List Agents	Instructs the handler to poll the network looking for all active agents.
Download Upgrades	Instructs the handler to download an upgrade package, usually an executable file that can be uploaded from a web location.
Set IP Address Spoofing	Instructs the handler to turn IP Spoofing on in the agents. This allows the attacker to set the spoofed IP address.
Set Attack Time	Instructs the handler to set a time when it should communicate to the agents to begin the attack.
Set Attack Duration	Instructs the handler to set a time when it should communicate to the agents to end the attack.
BufferSize	Set the buffer size for packets sent during a flood attack.
Info	Receive information (some type of help/command listing)

Table 2: Agent Commands

Command	Description
Turn On	Instructs the agent to turn on and wait for other commands.
Turn Off	Instructs the agent to shut down. If the agent was in the process of attacking or actively polling the network looking for handler/IRC channels this command stops the agent from continuing this action.
Initiate Attack	This command instructs the agent to launch an attack against a specified target.
Download Upgrades	This command instructs the agent to download an upgrade package, usually an executable file that can be uploaded from a web location.
Set IP Address Spoofing	This allows the attacker to set the IP address to be used for spoofing.
Set Attack Time	Instructs Agent with time to begin attack.
Set Attack Duration	Instructs Agent with time to end attack.
Info	Receive information (some type of help/command listing)

6 Examples of DDoS Attack Tools

There are a wide variety of DDoS attack tools available today. Examples of some common DDoS attack tools are presented and categorized below. These tools are often taken by attackers and modified slightly so there are many derivative DDoS attacks that are created from earlier tools and the core functionality.

6.1 Agent-Handler Attack Tools

Trin00 [3, 25] is credited with being the first DDoS attack tool to be widely distributed and used. Trin00 uses the agent-handler attack architecture, and is a bandwidth depletion attack tool using UDP flood attacks. Early versions of trin00 appear to not support IP source address spoofing. Typically, the trin00 agent gets installed on a system that suffers from remote buffer overrun exploitation [25]. This "bug" in the software allows an attacker to remotely compile and run the agent installation within the secondary victim's system buffer. Early versions of trin00 were found on the Solaris 2.5.1 and Red Hat Linux 6.0 operating systems [25]. The trin00 software is typically set up to communicate with TCP between the attacker's client and the handler systems, and UDP between the handler and agent systems [25]. The trin00 software uses symmetric-key encrypted communication between client and handler and handler and agents.

Tribe Flood Network is a DDoS attack tool that provides the attacker with the ability to wage both bandwidth depletion and resource depletion attacks. The TFN tool provides for UDP and ICMP flooding, as well as TCP SYN, and Smurf attacks [27]. TFN setup has been witnessed by buffer overflow [28]. In the TFN attack tool, the agents and handlers communicate with ICMP_ECHO_REPLY packets. These packets are harder to detect than UDP traffic and have the added ability of being able to pass through firewalls [27]. The TFN attack tool has been discovered on both Red Hat Linux 6.0 and Solaris 2.x operating systems. The TFN tool offers no encryption between agents and handlers or between handlers and clients [28].

Tribe Flood Network 2000 (TFN2K) is a DDoS attack tool based on the TFN architecture. The TFN2K attack tool adds encrypted messaging between all of the attack components [27]. In addition, it provides the handlers and agents with the ability to communicate with ICMP, UDP, or TCP [27]. There is also the option for random communication protocol selection.

Stacheldraht, German for "barbed wire", is a DDoS attack tool based on earlier versions of TFN. Like TFN, it includes ICMP flood, UDP flood, and TCP SYN attack options [29]. It also has the ability to perform updates on the agents automatically [29]. This means that the attacker can provide the installation file on an anonymous server and when each agent system turns on (or logs on to the Internet), the agent will automatically look for updates and install them. Stacheldraht also provides a secure telnet connection via symmetric key encryption between the attacker and the handler systems [29]. This prevents system administrators from intercepting this traffic and identifying it.

Shaft is a derivative of the trin00 tool. It uses UDP communication between handlers and agents. The attacker communicates with the handlers via a telnet connection. Shaft provides UDP, ICMP, and TCP flooding attack options. The attacks can be run individually, or they can be combined to form one attack with UDP/TCP/ICMP flooding. Shaft provides statistics on the flood attack. These statistics are useful to the attacker to know when the victim system is completely down and allows the attacker to know when to stop adding agent machines to the DDoS attack [30].

6.2 IRC-based DDoS Attack Tools

IRC-based DDoS attack tools were developed after the agent-handler attack tools. Hence, many IRC-based DDoS attack tools are more sophisticated in that they include a wide variety of attack characteristics found in many of the agent-handler attack tools.

Trinity is one such IRC-Based DDoS attack tool. Trinity has a wide variety of attack options including UDP, TCP SYN, TCP ACK, and TCP NUL packet floods as well as TCP fragment floods, TCP RST packet floods, TCP random flag packet floods, and TCP established floods. It has the ability to randomize all 32 bits of the source IP address [31]. Trinity also supports generating TCP flood packets with random control flags set. This provides Trinity with a wider set of TCP based attacks.

Knight is an IRC-based DDoS attack tool that was first reported in July 2001 [31]. The Knight DDoS attack tool provides SYN attacks, UDP Flood attacks, and an urgent pointer flooder [32]. The Knight tool is typically installed by using a Trojan horse program called Back Orifice [31]. Knight is designed to run on Windows operating systems.

Kaiten is another IRC-based DDoS attack tool. It is based on Knight, and was first reported in August of 2001 [19]. Kaiten supports a variety of attacking features. It includes code for UDP and TCP flooding attacks, for SYN attacks, and a PUSH + ACK attack [33]. Kaiten also randomizes the 32 bits of its source address.

7 Taxonomy of DDoS Countermeasures

There are currently a number of proposals and partial solutions available today for mitigating the effects of a DDoS attack. Many of these solutions and ideas assist in preventing certain aspects of a DDoS attack. However, there is no comprehensive method to protect against all known forms of DDoS attacks. Also, many derivative DDoS attacks are continually being developed by attackers to bypass each new countermeasure employed. This is the area where future research is needed, and the purpose for this paper's attempt to shed understanding on the nature and scope of DDoS networks, attacks, and tools. We propose a preliminary taxonomy of DDoS Countermeasures in Figure 8.

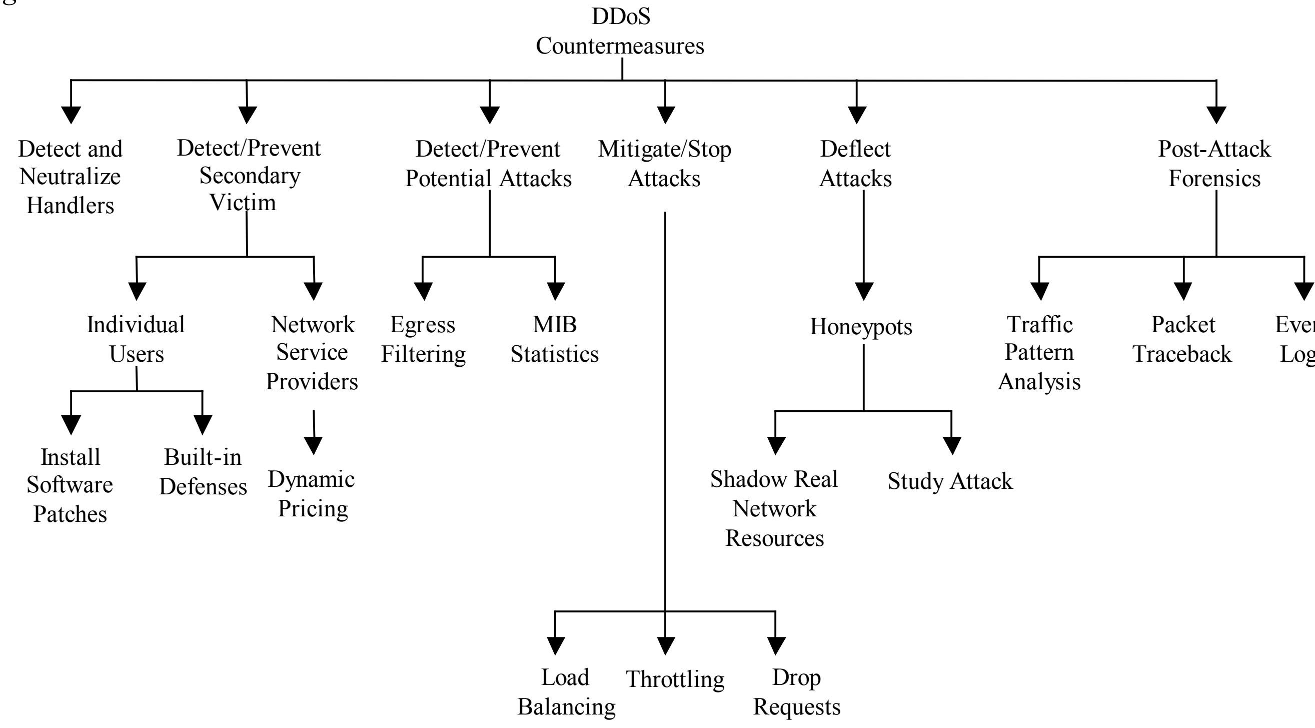
There are three essential components to DDoS countermeasures. There is the component for preventing the DDoS attack which includes preventing secondary victims and detecting and neutralizing handlers. There is the component for dealing with a DDoS attack while it is in progress including detecting/preventing the attack, mitigating/stopping the attack, and deflecting the attack. Lastly, there is the post-attack component which involves network forensics.

7.1 Prevent Secondary Victims

Individual Users. One of the best methods to prevent DDoS attacks is for the secondary victim systems to prevent themselves from participating in the attack. This requires a heightened awareness of security issues and prevention techniques from all Internet users. If attackers are unable to break into and make use of secondary victim systems, then the attackers will have no "DDoS attack network" from which to launch their DDoS attacks.

In order for secondary victims to not become infected with the DDoS agent software, users of these systems must continually monitor their own security. They must check to make sure that no agent programs have been installed on their systems and make sure they are not indirectly sending agent traffic into the network. The task of having all possible secondary victims implement measures to prevent themselves from becoming part of a DDoS attack is quite daunting. The Internet is so de-centralized, and since there are so many different hardware and

Figure 8: DDoS Countermeasures



software platforms, it is quite difficult for users to implement the right protective measures. Typically this would include installing anti-virus and anti-Trojan software and keeping these up to date. In order to successfully do this, the end users must have the resources to afford protective measures and the knowledge to choose the right security protections. Additionally, potential secondary victims must try to detect if they are participating in a DDoS attack, and if so, they need to know how to stop it. These tasks can be viewed as daunting for the average "web-surfer".

Recent work has proposed built-in mechanisms in the core hardware and software of computing systems that can provide defenses against malicious code insertion through buffer overflow violations [34]. This can significantly reduce the probability of a system being compromised as a secondary victim for a DDoS attack network.

Network Service Providers. One strategy currently being discussed is for providers and network administrators to add dynamic pricing to their network usage, to encourage secondary victims to become more active in preventing themselves from becoming part of a DDoS attack. If providers chose to charge differently for the use of different resources, they could charge for access to certain services within their networks. This would allow the providers to only allow legitimate customers on to their networks. This system would make it easier to prevent attackers from entering the network [35]. By altering the pricing of services, secondary victims who would be charged for accessing the Internet may become more conscious of the traffic they send into the network and hence may do a better job of policing themselves to verify that they are not participating in a DDoS attack.

7.2 Detect and Neutralize Handlers

One important method for stopping DDoS attacks is to detect and neutralize handlers. Since the agent-handler DDoS attack tools require the handler as an intermediary for the attacker to initiate attacks, finding and stopping the handlers is a quick method to ending the attack. This can possibly be done by studying the communication protocols and traffic patterns between handlers and clients or handlers and agents in order to identify network nodes that might be infected with a handler. Also, there are usually far fewer DDoS handlers deployed than there are agents, so shutting down a few handlers can possibly render multiple agents useless. Since agents form the core of the attackers ability to wage an attack, neutralizing the handlers to prevent the attacker from using them is an effective strategy to thwarting DDoS attacks.

7.3 Detect Potential Attacks

Egress Filtering. One method for detecting potential attacks is to use egress filtering. Egress filtering refers to the practice of scanning the packet headers of IP packets leaving a network (egress packets) and checking to see if they meet certain criteria. If the packets pass the criteria, they are routed outside of the sub-network from which they originated. If the filter criteria are not met, the packets will not be sent to the intended target. Since one of the features of DDoS attacks is spoofed IP addresses, there is a good probability that the spoofed source addresses of DDoS attack packets will not represent the source addresses of a valid user on a specific sub-network. If the network administrator places a firewall or packet sniffer in the sub-network that filters out any traffic without an originating IP address from this subnet, many DDoS packets with spoofed IP source addresses will be discarded.

MIB Statistics. Another method currently being looked at to identify when a DDoS attack is occurring uses the MIB (Management Information Base) data from routers. The MIB data from a router includes parameters that indicate different packet and routing statistics. Current research has focused on identifying statistical patterns in different parameters during a DDoS attack [36]. It looks promising for possibly mapping ICMP, UDP, and TCP packet statistical abnormalities to specific DDoS attacks.

Accurate statistical models based on the MIB parameters from routers are still being studied to understand how accurately they can monitor DDoS attack traffic and predict when a DDoS attack is happening. Work in this area could provide important information and methods for identifying when a DDoS attack is happening and how to filter or adjust the network to compensate for the attacking traffic.

7.3 Mitigating the Effects of DDoS Attacks

Load Balancing. For network providers, there are a number of techniques used to mitigate the effects of a DDoS attack. Providers can increase bandwidth on critical connections to prevent them from going down in the event of an attack. Replicating servers can help provide additional failsafe protection in the event some go down during a DDoS attack. Balancing the load to each server in a multiple-server architecture can improve both normal performance as well as mitigate the effect of a DDoS attack.

Throttling. One proposed method to prevent servers from going down is to use Max-min Fair server-centric router throttles [37]. This method sets up routers that access a server with logic to adjust (throttle) incoming traffic to levels that will be safe for the server to process. This will prevent flood damage to servers. Additionally, this method can be extended to throttle DDoS attacking traffic versus legitimate user traffic for better results. This method is still in the experimental stage, however similar techniques to throttling are being implemented by network operators. The difficulty with implementing throttling is that it is still hard to decipher legitimate traffic from malicious traffic. In the process of throttling, legitimate traffic may sometimes be dropped or delayed and malicious traffic may be allowed to pass to the servers.

7.4 Deflect Attacks

Honeypots. Another area being researched is Honeypots. Honeypots are systems that are set up with limited security to be an enticement for an attacker so that the attacker will attack the Honeypot and not the actual system. Honeypots typically have value not only in deflecting attacks from hitting the systems they are protecting, but also in serving as a means for gaining information about attackers by storing a record of their activity and learning what types of attacks and software tools the attacker is using. Current research discusses the use of honeypots that mimic all aspects of a legitimate network (such as web servers, mail servers, clients, etc.) in order to attract potential DDoS attackers [38]. The goal of this type of honeypot is to attract a DDoS attacker and get him to install either handler or agent code within the honeypot. This prevents some legitimate systems from getting compromised and allows the honeypot owner to track the handler or agent behavior and better understand how to defend against future DDoS installation attacks.

7.5 Post-Attack Forensics

Packet Traceback. If traffic pattern data is stored during a DDoS attack, this data can be analyzed post-attack to look for specific characteristics within the attacking traffic. This characteristic data can be used for updating load balancing and throttling countermeasures to increase their efficiency and protection ability. Additionally, DDoS attack traffic patterns can help network administrators develop new filtering techniques for preventing DDoS attack traffic from entering or leaving their networks.

Tracing the Attacker. Another set of theories to deal with preventing DDoS attacks and for assisting in identifying the attackers deals with using traces [39]. The concept of tracing is that Internet traffic could be traced back to the true source (rather than that of a potentially spoofed source IP address). This allows back tracing the attacker's traffic and possibly identifying the attacker. Additionally, when the attacker sends vastly different types of attacking traffic, this method assists in providing the victim system with information that might help develop filters to block the attack.

A model for developing a Network Traffic Tracking System that would identify and track user traffic throughout a network has been proposed [40]. This type of system has been identified as being very successful within a closed network environment, such as a corporate network where internal client systems can be fully managed by a central network administrator who can track individual end-user actions. This method begins to break down as the network becomes widely distributed [40]. Traffic tracking on the Internet or large extranets would be difficult to implement. The cost of setting up such as system would be daunting, and since different network administrators control different sections of the Internet, it would be difficult to determine who would be responsible for monitoring the traffic. Also, the loss of privacy on the Internet would meet with an unfavorable response from most network users.

Event Logs. Network administrators can keep logs of the DDoS attack information in order to do a forensic analysis and in order to assist law enforcement in the event the attacker does severe financial damage. Using both Honeypots as well as other network equipment such as firewalls, packet sniffers, and server logs, providers can store all the events that occurred during the setup and execution of the attack. This will allow the network administrators to discover what type of DDoS attack (or combination of attacks) was used.

8 Summary and Conclusions

A number of conclusions can be drawn from understanding DDoS attacks and from looking at some of the current defensive measures that are currently being researched.

DDoS attacks are quite advanced methods of attacking a network system to make it unusable to legitimate network users. These attacks are an annoyance at a minimum, and if they are against a critical system, they can be quite damaging. Loss of network resources costs money, delays work, and cuts-off communication between network users. The negative effects of a DDoS attack make it important that solutions and security measures be developed to prevent these types of attacks.

Detecting, preventing, and mitigating DDoS attacks is important for national security. The Internet is an important component of the United States communication infrastructure. In the same manner that the Internet has become more user friendly over the last 10 years, and more individuals, businesses, and government agencies make use of it, so has hacking and disrupting network traffic. In a recent report to the US Senate (March 2000), the FBI highlighted that "While remote cracking once required a fair amount of skill or computer knowledge, hackers can now download attack scripts and protocols from the World Wide Web and launch them against victim sites. Thus while attack tools have become more sophisticated, they have also become easier to use." [41]. DDoS attacks are easy for attackers and script kiddies to obtain and the potential for other attacks like the October 20th attack against the 13 root servers is quite high. Finding methods for preventing and stopping DDoS attacks will be important for national security. Understanding DDoS attacks is a first step towards this, and the main contribution of this paper.

There are some new legal issues that are being debated due to Internet attacks such as DDoS. Since victims of the attacks usually cannot trace back to the attacker, there is a question of who is liable for an attack, in terms of contributory negligence. Since some DDoS attacks can be traced back to the secondary victims, can the owners or corporations responsible for secondary victims be held liable for participating in an attack? Are software vendors liable for vulnerabilities in their code? Are hardware vendors responsible for providing defenses against malicious intrusion and use of the machines they sell by remote parties other than the owners? Do network providers have an obligation to prevent their networks from allowing secondary victims to send DDoS packet traffic into the network? Should owners or laws require "best effort" defensive practices from software, hardware, and network suppliers and operators?

One of the most important issues that will impact how defenses against DDoS attacks are deployed will be the cost of solutions and preventive measures. If DDoS prevention strategies cost companies and individuals huge sums of money than these systems will not see quick wide scale deployment. It will take time before industry and government agencies buy new products. Additionally, as each new product is developed, the attackers build methods to counter the security measures. This leads to a cyclical pattern of new security systems being deployed that are capable of defending against all currently identified attacks. As new security is developed, the attackers develop new attacks, which in turn lead to new security fixes. This pattern can get incredibly costly for companies and institutions that are trying to keep up with security measures.

In summary, there are many DDoS attack tools that are currently available to the "hacking community". These tools are easy to implement and can have disastrous effects against networks. There are many new methods of preventing these attacks from succeeding, however many of these new methods are still being developed and evaluated. It will be essential that as the Internet and Internet usage expand, that solutions and countermeasures to DDoS attacks be verified and implemented. For future work, we plan to write a simulator that allows accurate and parameterized simulation of DDoS attacks. This can then be used to investigate new countermeasures and comprehensible solutions.

10 References

[1] David Karig and Ruby Lee, "Remote Denial of Service Attacks and Countermeasures," *Princeton University Department of Electrical Engineering Technical Report CE-L2001-002*, October 2001.

[2] Lincoln Stein and John N. Stuart. "The World Wide Web Security FAQ", Version 3.1.2, February 4, 2002. http://www.w3.org/security/faq/ (8 April 2003).

[3] Paul J. Criscuolo. "Distributed Denial of Service Trin00, Tribe Flood Network, Tribe Flood Network 2000, And Stacheldraht CIAC-2319". Department of Energy Computer Incident Advisory Capability (CIAC), UCRL-ID-136939, Rev. 1., Lawrence Livermore National Laboratory, February 14, 2000.

[4] "Yahoo on Trail of Site Hackers", *Wired.com, February 8, 2000.* http://www.wired.com/news/business/0,1367,34221,00.html (15 May 2003).

[5] "Powerful Attack Cripples Internet". *Associated Press for Fox News* 23 October 2002. http://www.foxnews.com/story/0,2933,66438,00.html. (9 April 2003).

[6] Joseph Lo and Others. "An IRC Tutorial", *irchelp.com*. 1997. http://www.irchelp.org/irchelp/irctutorial.html#part1. (8 April 2003).

[7] Nicolas Pioch. "A Short IRC Primer". Edition 1.2, January 1997. http://www.irchelp.org/irchelp/ircprimer.html#DDC. (21 April 2003).

[8] Kleinpaste, Karl, Mauri Haikola, and Carlo Kid. "The Original IRC Manual". March 18, 1997. http://www.user-com.undernet.org/documents/irc-manual.html#seen (21 April 2003).

[9] Kevin J. Houle. "Trends in Denial of Service Attack Technology". *CERT Coordination Center, Carnegie Mellon Software Engineering Institute.* October 2001. www.nanog.org/mtg-0110/ppt/houle.ppt. (14 March 2003).

[10] TFreak. "smurf.c", *www.phreak.org*. October 1997. http://www.phreak.org/archives/exploits/denial/smurf.c (6 May 2003).

[11] Federal Computer Incident Response Center (FedCIRC), "Defense Tactics for Distributed Denial of Service Attacks". *Federal Computer Incident Response Center.* Washington, DC, 2000.
[12] TFreak. "fraggle.c", www.phreak.org. http://www.phreak.org/archives/exploits/denial/fraggle.c (6 May 2003).
[13] Martin, Michael J., "Router Expert: Smurf/Fraggle Attack Defense Using SACLS", *Networking Tips and Newsletters, www.searchnetwork.techtarget.com.* October 2002. http://searchnetworking.techtarget.com/tip/1,289483,sid7_gci856112,00.html (6 May 2003).
[14] Chen, Y. W. "Study on the Prevention of SYN Flooding by Using Traffic Policing", *Network Operations and Management Symposium, 2000. NOMS 2000. 2000 IEEE/IFIP, pp. 593-604.* 2000.
[15] RFC 793, "Transmission Control Protocol DARPA Internet Program Protocol Specification". Arlington, Virginia. September 1981.
[16] "Nmap Stealth Port Scanner Introduction", *Insecure.org*. August 2002. http://www.insecure.org/nmap/. (8 April 2003).
[17] "Nessus Documentation", *Nessus*. 2002. http://www.nessus.org/. (8 April 2003).
[18] "CVE (version 20020625)", *Common Vulnerabilities and Exposures*. March 27, 2002. http://cve.mitre.org/cve/. (9 April 2003).
[19] "CERT® Incident Note IN-2001-13". *CERT Coordination Center, Carnegie Mellon Software Engineering Institute.* November 27, 2001. http://www.cert.org/advisories/CA-2001-20.html. (14 March 2003).
[20] Colon E. Pelaez and John Bowles, "Computer Viruses", *System Theory, 1991, Twenty-Third Southeastern Symposium, pp. 513-517,* Mar 1999.
[21] Crispin Cowan, Perry Wagle, Calton Pu, Steve Beattie, and Jonathan Walpole, "Buffer Overflows: Attacks and Defenses for the Vulnerability of the Decade", *DARPA Information Survivability Conference and Exposition, 2000. Vol. 2, pp. 119-129,* 2000.
[22] Microsoft. "How to Write Active X Controls for Microsoft Windows CE2.1", *Microsoft Corporation*. June 1999. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnce21/html/activexce.asp. (5 April 2003).
[23] "Executing arbitrary commands using ActiveX "codebase=" parameter", *EdenSoft™,* 2 April 2002. http://www.edensoft.com/exploit.html. (9 April 2003).
[24] Dancho Danchev. "The Complete Windows Trojans Paper", *BCVG Network Security*. October 22, 2002. http://www.ebcvg.com/articles.php?id=91. (9 April 2003).
[25] David Dittrich. "The DoS Project's "trinoo" Distributed Denial of Service Attack Tool". University of Washington, October 21, 1999. http://staff.washington.edu/dittrich/misc/trinoo.analysis.txt (8 April 2003).
[26] Alex Noordergraff. "How Hackers Do It: Tricks, Tools, and Techniques", *Sun BluePrints™ OnLine. Part No.: 816-4816-10, Revision 1.0.* May 2002. http://www.sun.com/solutions/blueprints/0502/816-4816-10.pdf. (8 April 2003).
[27] Frank Kargl, Joern Maier, and Michael Weber, "Protecting Web Servers from Distributed Denial of Service Attacks", *Proceedings of the Tenth International Conference on World Wide Web*, April 2001.
[28] David Dittrich. "The "Tribe Flood Network" Distributed Denial of Service Attack Tool". University of Washington, October 21, 1999. http://staff.washington.edu/dittrich/misc/trinoo.analysis.txt (8 April 2003).
[29] David Dittrich. "The "stacheldraht" Distributed Denial of Service Attack Tool". University of Washington, December 31, 1999. http://staff.washington.edu/dittrich/misc/stacheldraht.analysis.txt (8 April 2003).
[30] Sven Dietrich, Neil Long, and David Dittrich, "Analyzing Distributed Denial of Service Tools: The Shaft Case", *USENIX Association, Proceedings of the 14th Systems Administration Conference (LISA 2000), New Orleans, Louisiana*, December 2000.

[31] "CERT® Advisory CA-2001-20 Continuing Threats to Home Users", *CERT Coordination Center, Carnegie Mellon Software Engineering Institute.* July 23, 2001. http://www.cert.org/advisories/CA-2001-20.html. (14 March 2003).

[32] Bysin. "Knight.c Sourcecode", *PacketStormSecurity.nl.* July 11, 2001. http://packetstormsecurity.nl/distributed/knight.c. (18 March 2003).

[33] Contem. "kaiten.c Sourcecode", *PacketStormSecurity.nl.* December 2001. http://packetstormsecurity.nl/irc/indexsize.shtml. (8 April 2003).

[34] Ruby Lee, David Karig, Patrick McGregor and Zhijie Shi, "Enlisting Hardware Architecture to Thwart Malicious Code Injection", *Proceedings of the International Conference on Security in Pervasive Computing (SPC-2003), pp. N/A,* March 2003.

[35] David Mankins, Rajesh Krishnan, Ceilyn Boyd, John Zao, and Michael Frentz, "Mitigating Distributed Denial of Service Attacks with Dynamic Resource Pricing", *Computer Security Applications Conference, 2001. ACSAC 2001. Proceedings 17th Annual, pp. 411-421,* 2001.

[36] Joao B. D. Cabrera, Lundy Lewis, Xinzhou Qin, Wenke Lee, Ravi K. Prasanth, B. Ravichandran, and Ramon K. Mehra, "Proactive Detection of Distributed Denial of Service Attacks Using MIB Traffic Variables – A Feasibility Study", *Integrated Network Management Proceedings, pp. 609-622*, 2001.

[37] David K. Yau, John C. S. Lui, and Feng Liang, "Defending Against Distributed Denial of Service Attacks with Max-min Fair Server-centric Router Throttles", *Quality of Service, 2002 Tenth IEEE International Workshop, pp. 35-44,* 2002.

[38] Nathalie Weiler. "Honeypots for Distributed Denial of Service", *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2002. WET ICE 2002. Proceedings. Eleventh IEEE International Workshops, 2002. pp. 109-114.* 2002.

[39] Vern Paxon, "An Analysis of Using Reflectors for Distributed Denial of Service Attacks", *ACM SIGCOMM Computer Communication Review, Vol. 31, Iss. 3*, July 2001.

[40] Thomas E. Daniels and Eugene H. Spafford, "Network Traffic Tracking Systems: Folly in the Large?", *Proceedings of the 2000 Workshop on New Security Paradigms,* February 2001.

[41] Freeh, Louis J. "Statement for the Record of Louis J. Freeh, Director Federal Bureau of Investigation on Cybercrime Before the Senate Committee on Judiciary Subcommittee for the Technology, Terrorism, and Government Information." Washington, D.C., March 28, 2000.