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

Stephen M. Specht
Electrical Engineering
Princeton University
Princeton, NJ 08544
stephen.specht@us.army.mil

Ruby B. Lee
Electrical Engineering
Princeton University
Princeton, NJ 08544
rblee@princeton.edu

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 primary victim systems. As new countermeasures are developed to prevent or mitigate DDoS attacks, attackers are constantly developing new methods to circumvent these new countermeasures.

In this paper, we describe DDoS attack models and propose taxonomies to characterize the scope of DDoS attacks, the characteristics of the software attack tools used, and the countermeasures available. These taxonomies illustrate similarities and patterns in different DDoS attacks and tools, to assist in the development of more generalized solutions to countering DDoS attacks, including new derivative attacks.

1 INTRODUCTION

A Denial of Service (DoS) attack is 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 that is 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 while remaining anonymous since the secondary victims actually perform the attack making it more difficult for network forensics to track down the real attacker.

In February of 2000, one of the first major DDoS attacks was waged against Yahoo.com, keeping it off the Internet for about 2 hours, costing it lost advertising revenue [2]. Recently, attackers used a series of DDoS attacks against a variety of companies providing anti-spam services [3]. These attacks caused many of them to shut down their services.

DDoS attacks are relatively new and not well understood. This paper proposes taxonomies for understanding different DDoS attacks, tools, and countermeasures. We hope these taxonomies aid in understanding the scope of DDoS attacks, leading to more comprehensive solutions or countermeasures to cover both known attacks and those that have not yet occurred. This paper is also the first to characterize the setup and installation techniques of DDoS attack architectures, identifying both active and passive classes.

In Section 2 we describe classes of DDoS attack architectures. In Section 3 we present our taxonomy for DDoS attacks. In Section 4 we present the software characteristics for DDoS attack tools emphasizing how these tools are setup on secondary victim systems. In Section 5 we present a taxonomy of different DDoS countermeasures. We conclude in Section 6.

2 DDoS ATTACK ARCHITECTURES

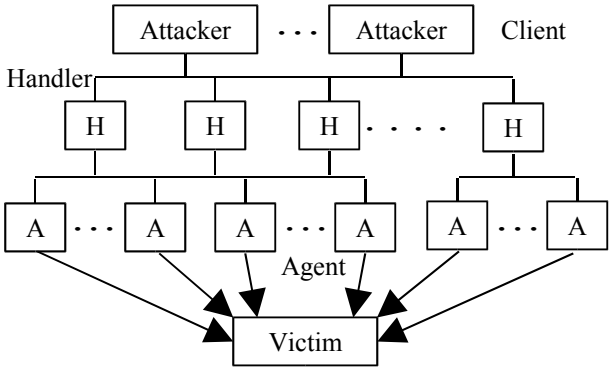
Two types of DDoS attack networks have emerged: the Agent-Handler model and the Internet Relay Chat (IRC)-based model.

The *Agent-Handler model* of a DDoS attack consists of clients, handlers, and agents (see Figure 1). 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

This work was supported in part by NSF CCR-0208946.
Stephen Specht is now a Computer Engineer with the US Army Information Operations Division, Fort Monmouth, NJ.

Stephen M. Specht and Ruby B. Lee, Distributed Denial of Service: Taxonomies of Attacks, Tools, and Countermeasures. *Proceedings of the 17th International Conference on Parallel and Distributed Computing Systems, 2004 International Workshop on Security in Parallel and Distributed Systems*, pp. 543-550, September 2004

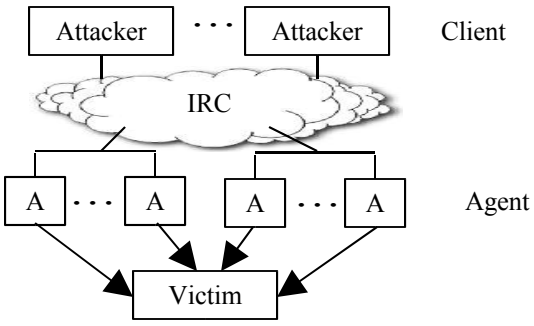
Figure 1: DDOS Agent-Handler Attack Model



instructed to communicate with a single handler or multiple handlers. Usually, attackers will try to 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. In descriptions of DDoS tools, the terms "handler" and "agents" are sometimes replaced with "master" and "daemons", respectively.

The *IRC-based DDoS attack architecture* is similar to the Agent-Handler model except that instead of using a handler program installed on a network server, an IRC (Internet Relay Chat) communication channel is used to connect the client to the agents. An IRC channel provides an attacker with additional benefits such as the use of "legitimate" IRC ports for sending commands to the agents [4]. This makes tracking the DDoS command packets more difficult. Additionally, IRC servers tend to have large volumes of traffic making it easier for the attacker to hide his presence. Another advantage is that the attacker does not need to maintain a list of the agents, since he can log on to the IRC server and see a list of all available agents [4]. 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.

Figure 2: DDoS IRC-Based Attack Model



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 refer to the agents as "secondary victims" or "zombies", and the target of the DDoS attack as the "primary victim". Well-designed agent software uses only a small proportion of resources (memory and bandwidth) so that the users of secondary-victim systems experience minimal performance impact when their system participates in a DDoS attack.

3 DDoS ATTACK TAXONOMY

There are a wide variety of DDoS attacks. We propose a taxonomy of the main DDoS attack methods in Figure 3. 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. A *resource depletion attack* is an attack that is designed to tie up the resources of a victim system making the victim unable to process legitimate requests for service.

3.1 Bandwidth Depletion Attacks

Bandwidth depletion attacks can be characterized as flood attacks and amplification attacks.

Flood Attacks. A *flood attack* involves zombies sending large volumes of traffic to a victim system, to congest the victim system's network bandwidth with IP traffic. The victim system slows down, crashes, or suffers from saturated network bandwidth, preventing access by legitimate users. Flood attacks have been launched using both UDP (User Datagram Protocol) and ICMP (Internet Control Message Protocol) packets.

In a *UDP Flood attack*, a large number of UDP packets are sent to either random or specified ports on the victim system. The victim system tries to process the incoming data to determine which applications have requested data. If the victim system is not running any applications on the targeted port, it will send out an ICMP packet to the sending system indicating a "destination port unreachable" message [5].

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 since return packets from the victim system are not sent back to the zombies, but to the spoofed addresses. UDP flood attacks may also fill the bandwidth of connections located around the victim system. This often impacts systems located near the victim.

An *ICMP flood attack* occurs when the zombies send large volumes of ICMP_ECHO_REPLY packets ("ping") 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 [5]. During this attack, the source IP address of the ICMP packet may also be spoofed.

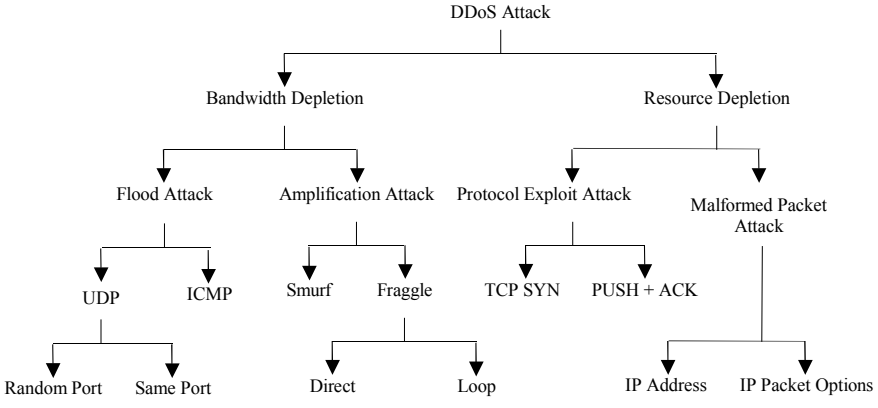
Amplification Attacks. An amplification attack involves the attacker or the zombies sending messages to a broadcast IP address, using this to cause all systems in the subnet reached by the broadcast address to send a reply to the victim system. The broadcast IP address feature is found on most routers; when a sending system specifies a broadcast IP address as the destination address, the routers replicate the packet and send it to all the IP addresses within the broadcast address range. In this attack, the broadcast IP address is used to amplify and reflect the attack traffic, and thus reduce the victim system's bandwidth.

The attacker can send the broadcast message directly, or 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.

A DDoS *Smurf attack* is an example of an amplification attack where 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 [6]. 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 [7]. This type of attack amplifies the original packet tens or hundreds of times.

Another example is the DDoS *Fraggle attack*, where the attacker sends packets to a network amplifier, using UDP ECHO packets [8]. 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 [9]. 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 send an echo packet back to the character generator, and the process repeats. This attack can generate more bad traffic and cause more damage than a Smurf attack.

Figure 3: DDoS Attack Taxonomy



3.2 Resource Depletion Attacks

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

Protocol Exploit Attacks. We give two examples, one misusing the TCP SYN (Transfer Control Protocol Synchronize) protocol, and the other misusing the PUSH+ACK protocol.

In a DDoS *TCP SYN attack*, the attacker instructs the zombies to send 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 eventually runs out of processor and memory resources, and is unable to respond to legitimate users.

In a *PUSH + ACK attack*, the attacking agents send TCP packets with the PUSH and ACK bits set to one. These triggers in the TCP packet header 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.

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 it.

There are at least 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 can 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, it can exhaust the processing ability of the victim system.

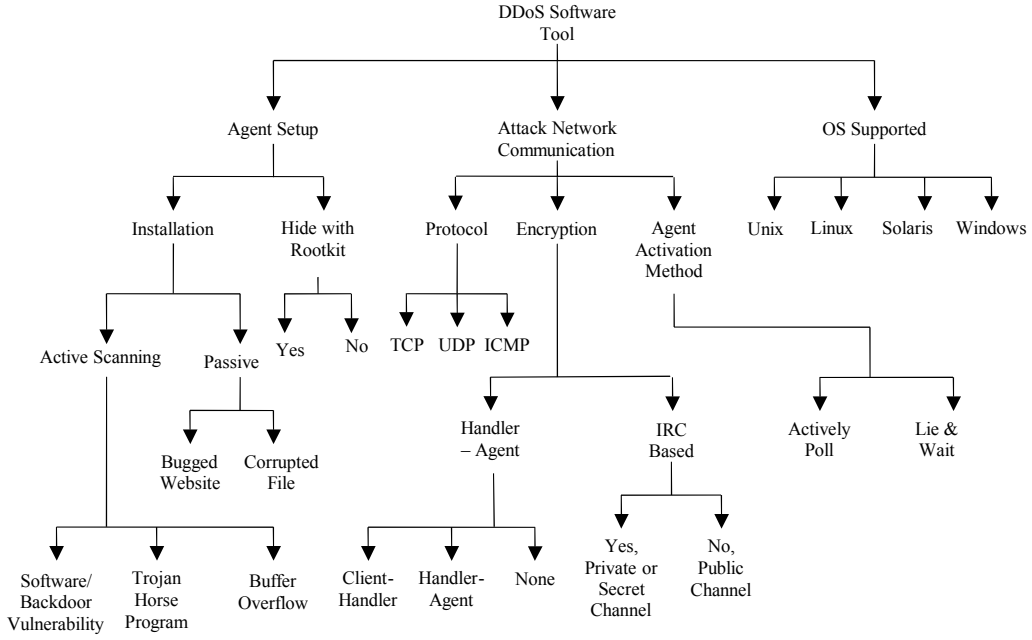
4 DDoS ATTACK TOOLS

DDoS attack tools include a number of common software characteristics. Figure 4 shows some of these common elements: how agents are setup, agent activation, whether the communication is encrypted, and which operating systems (OS) are supported.

4.1 DDoS Agent Setup

We classify the ways that attackers install malicious DDoS agent code onto a secondary victim system as either active or passive. *Active DDoS agent installation* methods typically involve the attacker scanning the network for systems with known vulnerabilities, running scripts to break into the system, and stealthily installing the DDoS agent software. In *passive DDoS installation* methods, the secondary victim unwittingly causes the DDoS agent software to be installed by opening a corrupted file or visiting a corrupted web-site.

Figure 4: DDoS Software Tools (Characteristics)



Active Scanning. Before installing DDoS software, attackers first run scanning tools, such as the port scanner Nmap, to identify potential secondary victim systems. Nmap 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 and return information that each IP address is broadcasting such as open TCP and UDP ports and the specific OS of the scanned system [10]. An attacker selects secondary victim targets from this list, targeting software and backdoor vulnerabilities.

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 lists over 2,000 of the known vulnerabilities of different systems [11]. This information is available so network administrators can make their systems more secure; however, it also provides attackers with data about existing vulnerabilities.

A common software 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 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 in the memory stack, such as a procedure return address [13]. 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.

An example of a backdoor vulnerability is a *Trojan horse program*. This is a program that appears to perform a useful function, but in reality contains hidden code that either executes malicious acts or provides a backdoor for unauthorized access to some privileged system function [12].

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.

Passive DDoS Agent Installation. 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.

A *corrupted file* has malicious code embedded within it. When the victim system tries to view or execute this file, it will become infected with the malicious code. One popular technique is for attackers to generate a text file with the name of the binary executable code for a DDoS agent embedded within it. They rename the text file with a very long name with the .txt extension within the name when the real extension is .exe. For instance, the file might be *newfile.txt_this_is_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 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 [15]. As soon as a user launches the file, his system will become infected with the DDoS agent software. Corrupt files can be exchanged in different ways such as IRC file sharing, Gnutella networks, and by e-mailing corrupt files to victims.

Another passive DDoS agent installation technique uses a *bugged web site*. A vulnerability found on web browsers 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 by allowing the IE web browser to automatically download client binary code specified by the website being viewed [14]. Attackers use malicious code inserted into a web page to take advantage of ActiveX and instead of downloading legitimate client software, the attacker can set up ActiveX to download hostile code, such as a DDoS agent.

Root kits are programs that are used by the attacker after installation of handler or agent software to remove log files and any other records that might indicate that the attacker was using the system [16]. Attackers may additionally use the root kit tools to create backdoors so that they will be able to access the victim's system in the future [17]. 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 is higher.

4.2 Attack Network Communication

The DDoS agent-handler and handler-client communication can be via TCP, UDP, and/or ICMP protocols.

Some DDoS attack tools also make use of *encrypted communication* within the DDoS attack network. Agent-handler DDoS attacks might use encrypted communications either between the client-handlers and/or between the handlers-agents. 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; private channels (not the data or users) appear in the IRC server's channel list but secret channels do not.

There are two key methods for DDoS *agent activation*. In some DDoS tools, the agents actively poll the handlers or IRC channel for instructions, whereas in other DDoS tools, the agents will wait for communication from either the handler or the IRC channel.

4.3 Operating Systems 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 (i.e.Unix, Linux, or Solaris). Agent code is usually designed for a Windows platform since many attackers target residential Internet users with DSL and cable modems (for higher attacking bandwidth) and these users typically use Windows.

5 DDoS COUNTERMEAUSRES

The countermeasures proposed for preventing a DDoS attack are currently partial solutions at best. There is currently 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. More research is needed, and the purpose for this paper is to characterize the nature and scope of DDoS attacks and tools to facilitate such research. We propose a preliminary taxonomy of DDoS Countermeasures in Figure 5.

There are three categories of DDoS countermeasures. First, preventing the setup of the DDoS attack network, including preventing secondary victims and detecting and neutralizing handlers. Second, dealing with a DDoS attack while it is in progress, including detecting or preventing, mitigating or stopping, and deflecting the attack. Third, there is the post-attack category involving network forensics.

5.1 Prevent Secondary Victims

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.

To prevent secondary victims from becoming infected with the DDoS agent software, 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. Because the Internet is so de-centralized, with so many different hardware and software platforms, it is quite difficult for users to implement the right protective measures such as anti-Trojan software. To be successful, end users must have the resources to afford protective measures and the knowledge to choose the right protections. Additionally, secondary victims must identify when they are participating in a DDoS attack, and if so, they need to know how to stop it. These tasks are 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 [18]. This can significantly reduce the probability of a system being compromised as a secondary victim for a DDoS attack.

Another strategy is for network service providers and network administrators to add dynamic pricing to 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 would be better able to identify legitimate users. This system would make it easier to prevent attackers from entering the network [19]. Additionally, secondary victims who might 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.

5.2 Detect and Neutralize Handlers

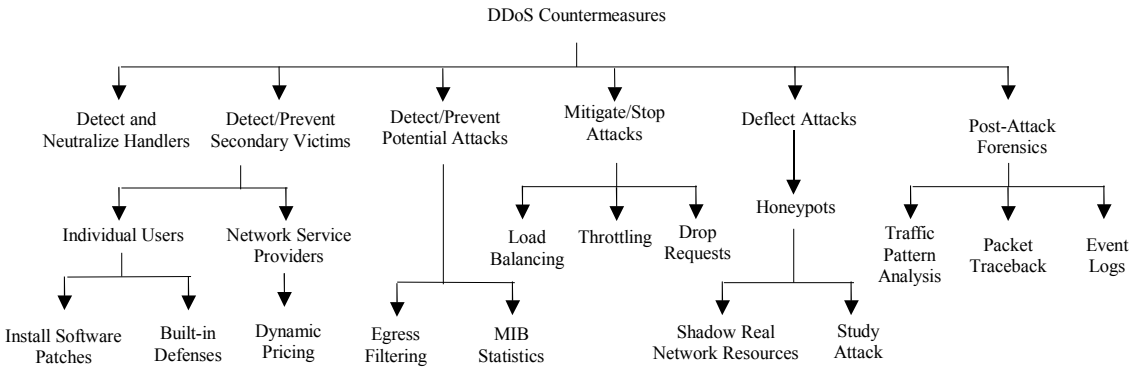
An important method for stopping DDoS attacks is to detect and neutralize handlers. One technique is to study the communication protocols and traffic patterns between handlers and clients or handlers and agents, in order to identify network nodes that might be infected with handler code. Since there are far fewer DDoS handlers deployed than agents, shutting down a few handlers can render multiple agents useless thereby neutralizing a DDoS attack.

5.3 Detect or Prevent Potential Attacks

To detect or prevent a potential DDoS attack that is being launched, egress filtering and MIB (Management Information Base) statistics can be used.

Egress filtering refers to the scanning of IP packet headers leaving a network 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. Otherwise, the packets will not be sent. Since DDoS attacks often use spoofed IP addresses, there is a good probability that the source addresses of DDoS attack packets will not represent the source address of a valid user on a specific sub-network. If the network administrator places a firewall in the sub-network to filter out any traffic without an originating IP address from the subnet, many DDoS packets with spoofed IP addresses will be discarded.

Figure 5: DDoS Countermeasures



Another method being studied to identify when a DDoS attack is occurring uses

the MIB statistics from routers. Router MIB data includes parameters that indicate different packet and routing statistics. Identifying statistical patterns in different parameters during a DDoS attack [20] looks promising for possibly mapping ICMP, UDP, and TCP packet statistical abnormalities to specific DDoS attacks. Work in this area could provide methods for identifying when a DDoS attack is happening and how to adjust network parameters to compensate for the attacking traffic.

5.4 Mitigating the Effects of DDoS Attacks

Load balancing can improve both normal performance as well as mitigate a DDoS attack. Network providers can increase bandwidth on critical connections to prevent them from going down in an attack. Additionally, providers can replicate servers and provide additional failsafe protection if some go down during a DDoS attack.

Throttling is another technique proposed to prevent servers from going down. The Max-min Fair server-centric router throttle method[21] 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 can 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 hard to decipher legitimate traffic from malicious traffic.

5.5 Deflect Attacks

Honeypots are systems intentionally set up with limited security to be an enticement for an attacker's attack. Honeypots serve to deflect attacks from hitting the systems they are protecting as well as 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 [22]. The goal of this type of honeypot is to lure an attacker to install either handler or agent code within the honeypot, thereby allowing the honeypot owner to track the handler or agent behavior and better understand how to defend against future DDoS installation attacks.

5.6 Post-Attack Forensics

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.

To help identify the attackers, *packet traceback* techniques are proposed [23]. The concept involves tracing Internet traffic back to it's source (rather than that of a spoofed source IP address). This technique helps to identify the attacker. Additionally, when the attacker sends 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 [24]. This model can be very successful within a closed network environment 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 over widely distributed networks [24].

Network administrators can also keep *event logs* of the DDoS attack information in order to do a forensic analysis and to assist law enforcement in the event the attacker does severe damage. Using Honeypots and 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 allows network administrators to discover what type of DDoS attack (or combination of attacks) was used.

6 CONCLUSION

DDoS attacks make a networked system or service unavailable to legitimate users. These attacks are an annoyance at a minimum, or can be seriously damaging if a critical system is the primary victim. Loss of network resources causes economic loss, work delays, and loss of communication between network users. Solutions must be developed to prevent these DDoS attacks.

We have proposed taxonomies of DDoS attacks, tools, and countermeasures in this paper to help scope the DDoS problem and to facilitate more comprehensive solutions. More detailed descriptions and DDoS examples are available in [25]. There are many DDoS attack tools available to attackers. These tools are easy to implement and can have disastrous effects. There are methods of preventing these attacks from succeeding, however, many of these are still being developed and evaluated. It is essential, that as the Internet and Internet usage expand, more comprehensive solutions and countermeasures to DDoS attacks be developed, verified, and implemented.

7 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*, Oct 2001.

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

[3] Brunker, Mike. "Spam Block Lists Bombed to Oblivion". *MSNBC.COM* 24 Sept 2003. http://www.msnbc.msn.com/id/3088113/ (Jul 5, 2004).

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

[5] 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, Feb 14, 2000.

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

[7] Federal Computer Incident Response Center (FedCIRC), "Defense Tactics for Distributed Denial of Service Attacks". Washington, DC, 2000.

[8] TFreak. "fraggle.c", *www.phreak.org*. www.phreak.org/archives/exploits/denial/fraggle.c (6 May 2003).

[9] Martin, Michael J., "Router Expert: Smurf/Fraggle Attack Defense Using SACLS", *Networking Tips and Newsletters, www.searchnetwork.techtarget.com*. Oct 2002. http://searchnetworking.techtarget.com/tip/1,289483,sid7_gci856112,00.html (6 May 2003).

[10] "Nmap Stealth Port Scanner Introduction", *Insecure.org*. August 2002. http://www.insecure.org/nmap/. (8 Apr 2003).

[11] "CVE (version 20020625)", *Common Vulnerabilities and Exposures*. Mar 27, 2002. http://cve.mitre.org/cve/. (9 Apr 2003).

[12] Colon E. Pelaez and John Bowles, "Computer Viruses", *System Theory, 1991, Twenty-Third Southeastern Symposium, pp. 513-517,* Mar 1999.

[13] 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.

[14] Microsoft. "How to Write Active X Controls for Microsoft Windows CE2.1", *Microsoft Corporation*. Jun 1999. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnce21/html/activexce.asp. (5 Apr 2003).

[15] Dancho Danchev. "The Complete Windows Trojans Paper", *BCVG Network Security*. Oct 22, 2002. http://www.ebcvg.com/articles.php?id=91. (9 Apr 2003).

[16] David Dittrich. "The DoS Project's "trinoo" Distributed Denial of Service Attack Tool". University of Washington, Oct 21, 1999. http://staff.washington.edu/dittrich/misc/trinoo.analysis.txt (8 Apr 2003).

[17] 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 Apr 2003).

[18] 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), LNCS 2802, pp. 237-252, Springer Verlag,* March 2003.

[19] 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.

[20] 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.

[21] 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.

[22] 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.*

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

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

[25] Stephen Specht and Ruby Lee, "Distributed Denial of Service: Taxonomies of Networks, Attacks, Tools, and Countermeasures," *Princeton University Department of Electrical Engineering Technical Report CE-L2003-03,* May 2003