

Only a matter of time remains for a major site that has yet to suffer a denial-of-service attack. The more well known the site, the higher the chance that it will be hit. When such an attack arrives, a site, no matter the amount of volume it may handle, will be unable to accept new connections and will likely be unable to service the connections it already has. If the site is a portal of commerce, then it is tenable to say that every minute the site cannot provide its service is a dollar-amount lost in sales. What can be done to prevent these denial-of-service attacks and what can be done to protect against them?

Denial-of-service attacks are a very real and very costly threat to the Internet today. To ignore their presence is to tempt fate. To dwell on their presence is to challenge the next attacker; advertising one's impregnability to infiltration and/or imperviousness to denial-of-service is to tempt fate. But is it practical for a business to spend the effort necessary to secure itself and then refrain from advertising it to their clients? Surely, such security would attract customers. The question of how many resources to expend to defend against denial-of-service versus how much to advertise this defense is answerable only by the service provider that chooses to recognize the threat of denial-of-service.

For those that have chosen to defend themselves, the most important topic is *what is denial-of-service?* How is it possible with today's high-speed hardware? How is it possible with today's robust and reliable technologies?

1.0 What does DENIAL-OF-SERVICE mean, exactly?

Think of a server. It provides a World-Wide-Web interface for all the Internet users to the stock market. Users connect and create an account. Every time henceforth that they wish to view stocks or to buy and sell stocks, they log in and move about on the site to do their bidding. Stocks are bought and stocks are sold. Moreover, every transaction that takes place through this site carries a service fee. This is an e-business: the custodians of the site are in the business of providing stock speculators with ability to buy and sell at their inclination.

Then something goes wrong. The traffic to the site increases to an overwhelming volume. The influx is so great that the server can scarcely handle it. Users that have already logged in now must wait a minute (perhaps more) for a page to load each time they navigate through the site. Potential users (i.e., users that have not yet logged in but are attempted to connect to the site in order to do so) must also wait a minute or more to see the main entry portal to the site.

Think that is not so bad? Think that a minute-long wait will not kill anybody, or that commerce will not suffer for it? Maybe sales will not decrease. Maybe every one of the clients, whether logged in or soon-to-be logged in, understands that the Information Superhighway, just like the interstate highway, has rush hours, too, and that the best things come to those who wait.

Then something else goes wrong. The traffic increases. The overwhelming volume of traffic that the server could scarcely handle just doubled. The server comes to a screeching halt. Logged-in users are sitting ducks while they remain on the site. Users trying to log in can wait as long as they want, but they are not going to see the main page. Any potential clients that are attempting to view the site for the first time will now have the first impression that the site is *not* moving at the speed of business, is *not* cutting-edge, and, most importantly, is *not* worth using.

This is denial-of-service. To be more precise, this was a distributed denial-of-service. Not all denial-of-service attacks rely on floods of traffic to cripple the site. There are subtler ways of doing it.

Denial-of-service is an attack that causes a service to deny users its data. The service will not, or cannot, accept requests or process transactions. The affected process has been forced to *deny its service* to its users.

1.1 The Basic Idea

What is crucial to understanding denial-of-service is that legitimate traffic can overwhelm a server – no attack necessary. This means that at any given hour, half of the Internet-capable world may log on. If a site cannot handle a thousand users connecting to it within the span of a few minutes, it will crash. This is a reality of e-business. To counter such a reality, sites often maintain mirror servers. The mirror-oriented structure can alleviate the pressures of traffic floods by dispersing the users among its multiple servers. Obviously, the more servers a site has, the more traffic it can handle. Now a site is robust against a natural flood. It is *robust*, not invulnerable.

If legitimate traffic can cause a server to crash, then perhaps someone can cause a flood of illegitimate traffic that looks no different than a flood of legitimate traffic. If someone does this, then a service provider may lose customers to denial-of-service without ever knowing it. Perhaps this has frustrated victims more than anything: the impotence to deliver justice to those who would inflict harm.

Denial-of-service comes in a variety of flavors. An e-business may protect against one attack but leave itself wide open to another. An e-business may try to protect itself against all of the various attacks but find itself unwilling to do so: protection against certain types of attacks comes only at the cost of functionality.

Some attacks are easy to detect. Some are also easy to trace. The problem is that the source of the attack will not reveal the mind behind the attack. Some distributed denial-of-service attacks take this form. An administrator of a network (on which the servers lie) may trace the attack back to each of the three hundred machines that launched it, but may still have no idea who signaled the launch. This is what is meant by a distributed attack: an attacker controls hundreds to thousands of machines from which the payload will be launched. No single machine was responsible for all of the illegitimate traffic. The responsibility is distributed among all of the compromised machines.

1.2 Understanding the Threat

Before a business can protect itself from the sometimes-arbitrary assault of a denial-of-service attack, it must know the dynamics of such an assault. Protecting itself against the unknown can be foolish, especially when what it defends itself against does not have to be unknown.

The main focus will be distributed denial-of-service attacks, or those attacks that involve hundreds and even thousands of compromised machines. These attacks have the most potential in terms of damage due to the sheer volume of their payloads. However, other types of denial-of-service attacks will not go unmentioned. It is critical to understanding distributed attacks that one has a solid base of knowledge concerning denial-of-service in general. Henceforth denial-of-service will be referred to as DoS, and distributed denial-of-service will be referred to as DDoS.

DDoS involves several stages. In order to conduct an attack from hundreds of computers, an attacker must break into the hundreds of computers. This is the infiltration. Once the attacker has compromised the protection on a machine, the attacker must get the DDoS tools onto that machine. This is the propagation. Once all of the tools are in place, the system needs a method by which it may maintain and conduct itself. With a discussion of the architecture of the DDoS framework comes a summary of the various means of communication used in such architectures. Next comes the payload.

Once the various munitions used in an attack are understood, only then can the topic of defense be discussed. To think of defending against such a menace without having the slightest notion of the nature of the menace is laughable at best. Many defenses exist only at the cooperation of networks across the Internet. These defenses are preventative measures, and few measures exist that effectively provide direct protection against DDoS.

2.0 DDoS: Distributed Denial-of-Service

If DDoS is distributed, then what types of attacks are not distributed? Since many of the non-distributed attacks provide the basis for the distributed attacks, little attention will be paid to the non-distributed ones, except to elucidate between repressive and oppressive methods of causing a denial of service. This notion of oppressive and repressive goes hand-in-hand with the ideas of triviality and non-triviality.

The survey of DDoS attack technology is split into several stages. It is easier to understand the beast if one knows its roots and its childhood. DDoS is not possible without a multitude of compromised machines, and so the first question is: how does an attacker ensnare a multitude of machines under its thumb? The art of compromising the machines has been separated into the aspects of infiltration and propagation so that due and sufficient elaboration occurs for each.

DDoS frameworks, once constructed, may implement attacks that use two techniques: reflection and amplification. Amplification allows the weakest of machines to crash even a state-of-the-art server, given an enough time. Reflection hinders investigation into the identity and location of the attacker. With the two techniques combined, an attacker may use a DDoS assault that is both untraceable and unstoppable.

The payloads themselves vary widely. There are attacks that use only IP. There is TCP/IP attacks, UDP/IP attacks, and ICMP attacks. For the benefit of the reader, more than just DDoS-specific payloads will be discussed. It is helpful to know of other ways a server can be crashed.

What can be done about DDoS? There is filtering techniques, packet traces, and encryption, for starters. Most defenses will not protect a machine from an attack, but will help prevent attacks on other machines. If networks across the Internet took such preventative measures, then DDoS might fade away in obsolescence. In the meantime, a business will need more immediate protection. Some forms of protection are not very helpful, and others are, although they require shutting down useful services.

2.1 Infiltration

The first thing an attacker must do is find out which servers are candidates for membership in the DDoS network. What characteristics would make a machine an ideal candidate?

First, the attacker wants a machine with a decent bandwidth. The higher the bandwidth, the better. This immediately places most servers and any router on the list of candidates. That is a huge list, and, as of yet, such a list is useless.

Second, the attacker needs to know which of the chosen machines are feasible to use. The attacker cannot simply log on to a machine and download the DDoS tool. Rather, in most cases the server may not be accessible to the general public. This means that such a server might require authentication for any type of access. The server might exist to facilitate business-to-business transactions. What does this mean for the attacker? If most machines on the Internet are hard against strange users anonymously downloading software onto them, then where does that leave the attacker? The attacker does not need explicit access to a machine. Most software has some sort of glitch in it that can be exploited. If the software is a server program, and the bug is a buffer overflow, for example, then the attacker need only know that the machine was running that particular server program. Many such bugs exist. Now the attacker recompiles the list of candidates to include only the machines running vulnerable software.

It is important to keep software up to date. When a company announces a new patch, download it and install it. When a newer version of the software hits the scene, try to buy it. Often, newer versions do not contain the well-known bugs of the earlier versions. Patches exist to correct problems, and hopefully the problem corrected is a bug that allowed a program to be compromised.

The attacker now has a list of machines ready for infiltration. What next? If most DDoS attacks involve hundreds of machines, then how long would it take for an attacker to infiltrate each machine, one by one? It would take too long; attackers will not perform the infiltration of each machine individually. The attacker will not because he or she can write a script or a program to automate the task. The automated procedure goes down the list, exploiting the bugs on each machine, and then moving on to the next bug. When a bug is exploited, the next step is to download the tools. This is called propagation.

2.2 Propagation

There are two basic ways to move the DDoS software to the compromised machines. These two methods apply to areas outside of denial-of-service attacks also: worms may use them.

The first method is central source propagation[9]. Once the attacker has downloaded a tool to the compromised machine, the tool initiates a transfer of the DDoS code from a central source. Scripts automate this installation process[9]. Because all infected machines must download from the same source, this method of propagation is not reliable. If the source should vanish, then no further downloads may occur.

The second method is back-chaining propagation[9]. The attacking machine sends the tool to the compromised machine. The tool then initiates a download from the attack machine to download the DDoS code. This method has more reliability[9]. The source of the code is the attacking machine, and not a central point. Even if one arm of the DDoS web is broken, other arms remain to continue the infiltration and propagation process.

Combined with the infiltration phase, propagation can threaten a denial-of-service by itself[9]. If the automation causes networks to slow down due to file transfers (the copying of the DDoS code), then legitimate users will suffer because of the traffic.

2.3 Distributed Technology

Most DDoS architectures have three tiers: the individual that initiates and controls the attack, or the attacker; the processes that coordinate the hundreds or thousands of machines, or the masters; and the hundreds or thousands of zombie processes on machines that deliver the payloads to the victim, or the agents.

This is how it works: the malevolent individual – the attacker – signals the masters. The attacker has cried havoc. Either the attacker writes a script to automate the command-line operation of the masters, or the attacker has a program that sends packets to the masters telling the masters to begin an assault. The attacker tells the masters the address of the victim, the type of the attack (if there is the option), and the length of time for which the assault will last.

The masters, having received orders from the attacker, begin communicating with all of the agents. They tell the agents the address, type, and length of time. So far, this implies a few things: 1) each time an agent is successfully installed on a compromised machine, it tells the master, so that the master knows of its existence; 2) each time a master is successfully installed the master must tell the attacker or the attacker must have some other way of knowing this; 3) from 1 and 2: the attacker has a list of the masters, the masters know the address of the attacker, the masters have lists of the agents, and the agents know the address of their particular master.

Finally, after the agents receive their orders, they let slip the dogs of war. Each agent unleashes one form of packet storm or another. The payload spreads out through the Internet and converges on the unsuspecting victim. DDoS attacks have been known to last for hours and even days.

Think of all that capital spiraling down the drain. Think of every business and every web-shopper that tried to connect to the site, gave up, and turned to a competitor that was, at the very least, decent enough to give its clients the time of day. Sure, that isn't the whole picture. But the clients don't know that.

If the DDoS framework relies on a chain of command, then why not try to break that chain? Indeed, early forms of DDoS code were susceptible to interference. The first DDoS programs to receive widespread use were trin00 and Tribe Flood Network. In a trin00 network, the attacker sends commands via TCP/IP to the masters, and the masters send commands via UDP/IP to the agents. Trin00 networks use unusual ports to conduct the communication, and a system administrator with a keen eye can point out these strange ports[10]. TFN requires command-line access to the masters from the master, but the masters can communicate to the agents via ICMP[11]. Neither trin00 nor TFN use any type of encryption, so an eavesdropper can learn the format of the commands easily. Then the eavesdropper (probably the concerned administrator) can hijack the network.

Stacheldraht was the first DDoS technology to use encrypted commands. With stacheldraht, the attacker sends commands via TCP/IP, and the master uses TCP/IP with its agents to receive the agent locations but sends commands to the agents via ICMP[12]. Stacheldraht's use of encryption makes it difficult to hijack the network, but not impossible. A brute force attack can crack the encryption. Stacheldraht does not hide its ports, and a system administrator can find these open channels.

TFN2K requires command-line access to the masters, but the masters have a very elusive method of contacting the agents. The masters randomly choose from TCP/IP,

ICMP, or UDP/IP (or any combination of the three), to send signals to the agents[13]. Its communication is very difficult to detect.

Tracing the source of a DDoS comes with many difficulties. Finding all of the agents does not necessarily mean that the attacker will be found, because the agents know only of the masters. Finding all of the parts of the network may depend on the cooperation of other system administrators and may depend on the ability to find and decrypt the address of the master or the attacker.

2.4 Reflection, Amplification, and IP Spoofing

Before moving on to the discussion of the different payloads a DDoS attack may deliver, one should understand the techniques an attack may use. These techniques make some attacks possible that otherwise would not be. These techniques are: IP spoofing, reflection, and amplification.

IP spoofing is quite simple. An IP packet has a source address and a destination address. If a packet leaves a machine with a different source address than that of the machine, then its address was spoofed. Any machine that receives a spoofed packet will not know the identity of the true sender. If the receiving machine should respond to the packet, the response will go to the spoofed address. There is no current way to prevent the spoofing of an IP address. However, there are ways to prevent a spoofed packet from reaching the Internet. These ways will be discussed later.

Reflection is possible only because of IP spoofing. Reflection works like so: a packet leaves a machine for a certain destination with the spoofed address of the intended target machine. The packet arrives at its destination. The destination machine does not know from where the packet came, because of the fake source address. Nevertheless, the destination machine will respond. The response then arrives at the fake source address, the victim. The art of reflection is true to its name: packets are bounced – or, *reflected* – off of an intermediary[15]. If each of the thousand agents in a DDoS attack reflects its payload off of multiple networks, then the victim may never be able to find all of the agents, no matter how hard the victim tries. Because reflection relies on IP spoofing, if IP spoofing is prevented on the Internet, then so is reflection.

Amplification is also quite simple. It can happen in one of two ways: if one packet is sent to a machine and more than one packet is returned; and if a request of a certain size is sent to a machine and a response of a larger size than the request is returned.

If, perhaps, reflection and amplification are combined, then untraceable payloads can land on a victim in massive volumes.

2.5 Denial-of-Service Payloads

2.5.1 IP vulnerabilities

IP, or Internet Protocol, provides a means of standardized transport through the Internet. Data are encapsulated in an IP packet and sent from relay to relay; IP packets reach a router, and based on the packet's destination, they are sent along their way. IP packets have a source address and a destination address to make the communication process possible. Furthermore, if a datum exceeds the maximum size of a single IP packet, the datum is split among multiple IP packets. Thus IP has a method of conveying how many packets belong to a single datum and in what order they belong.

Fragmentation: this attack involves breaking a large datum into many smaller pieces. All of the pieces are sent into the Internet except for one. A machine that receives these pieces must wait for every piece in order to restore the original datum. By neglecting a single piece, the receiving machine will not ever be able to restore the original datum. Resources on the receiving machine are tied up and wasted as it continues to wait for a piece of the datum that will never arrive.

IP Spoofing: discussed earlier in section 2.4. Involves the falsification of the source address of an IP packet.

2.5.2 ICMP vulnerabilities

ICMP, or the Internet Control Message Protocol, exists to facilitate the maintenance of the Internet. It has many different messages to do this. Some of the different messages involve routing directions. If a router forwards a packet in a certain direction, and the receiving machine knows that the target cannot be reached on such a course, the receiving machine will send an ICMP message to the router saying so. If the receiving machine had such knowledge, it might tell the router in what direction to send future packets. Roughly speaking, there are two basic types of ICMP messages: pings and re-directs. Often pings are used to establish the existence of a target. A well-known ping is the ECHO_REQUEST. When a machine receives an ECHO_REQUEST, it will respond with an ECHO_REPLY. A TIME_EXCEEDED message responds to a packet that has traversed the Internet for longer than its lifespan. PORT_UNREACHABLE messages respond to UDP/IP packets sent to inoperative ports.

Ping of Death: this attack is only effective when ICMP is implemented poorly. An over-sized ping is sent to the target. The target will not handle the size correctly (assuming, of course, that the target has a poor implementation). When the packet is stored in memory, it will overwrite information, perhaps important information, and the system will crash[4]. This happens because the message is allotted a certain amount of space: the maximum size of the ping. If the ping is too large and the logic does not recognize this, then the ping message will fill up the allotted space and even more space.

Smurf: this is a well-known reflector and amplifier attack. The attacker sends an IP broadcasted ping to a network. Every machine on the network will respond to this broadcast and send a reply ping. Each of the replies will head to a victim, because the source address was spoofed[3][5]. The victim then receives a swarm of ping replies that it did not expect.

Black Holes: Because routers use ICMP as a way of maintaining accurate relay tables, if the tables are corrupted, a victim may be starved of messages destined for it. If a router does not protect its tables with authentication, for example, then it is vulnerable to such an attack. Once an attacker learns the location of all of the routers through which a site may be accessed, the attacker can falsify the routing tables at all of the access points. Then, when a legitimate client attempts to send data to the site, the client will not be able to establish contact. Every packet destined for the site reached one of those access points and was sent in the wrong direction.

2.5.3 TCP/IP vulnerabilities

IP does not provide a means of ensuring that data arrived at its destination and arrived there intact. This makes IP unreliable as a transport mechanism. Since practicality

dictates that sometimes reliability is necessary, TCP appeared. TCP, or Transmission Control Protocol, is a reliable transport protocol that provides sequence numbers for packet ordering, ACK messages for acknowledging message receipts, congestion windows for flow control, and NACK messages for the occasional problem, should one arise. TCP thusly ensures that both parties involved in communication know the state of the data transfer. TCP can be seen as a finite state machine, with states according to the characteristics of the connection[1]. When a connection is first opened, a three-step handshake must complete so that both parties are in agreement that the connection is opened and so that both parties know what the sequence numbers are. The party that prompts for a connection sends a SYN message. The receiver then enters the SYN_RECEIVED state, and sends a SYN_ACK as it enters the SYN_SENT state. The prompting party then responds with an ACK message. The receiver enters the ESTABLISHED state and the connection is ready for data transfer. During the course of the connection, the congestion window adjusts itself according to the number of ACK messages received (in older implementations). The integrity of the connection is dependent upon the correct procession of the sequence numbers. If packets arrive out of order or in duplicate, a party knows that something could be wrong.

SYN Flood: the consumption of a machine's resources due to an excessive number of half-open connections. When a machine receives a SYN, it immediately allocates a TCB, or transmission control block, to the connection. Then the machine sends its SYN_ACK. If the other party does not respond with an ACK, the connection cannot enter the ESTABLISHED state. Most implementations, after a certain amount of time, terminate the half-open connections. However, an implementation must allow enough time to pass to account for traffic delays, otherwise legitimate connection attempts will fail. No matter how quickly the half-open connection expires, it is always possible to start enough half-open connections to starve the machine of its resources[2].

Established Connection Exploitation: Even if the attacker responds to all of the SYN_ACK messages with ACK messages, the attacker may still choose to leave the connection hang indefinitely. TCP implementations robust against SYN floods may not prevent excessive numbers of ESTABLISHED-state connections. This, too, will use up the system's resources[1].

RST Storm: When a TCP SYN packet arrives for an unused port, an RST packet, or Reset packet, will be returned. An RST storm is a reflector attack in which the RST packets are targeted for the victim.

Sequence Number Guessing: not a denial-of-service attack, but pertinent to the next attack that will be mentioned. As was mentioned, TCP relies on the sequence numbers of a connection to correctly delegate the connection. Older implementations used initial sequence numbers that were relatively easy to predict. An attacker could use this knowledge to spoof packets with forged sequence numbers. The receiving machine would believe that the packet came from a legitimate client due to the address and to the sequence number. The attacker could send arbitrary or malicious data to the receiver by this means.

ACK Splitting: an attack based on the illegitimate expansion of the congestion window. Some TCP implementations, particularly older ones, open the congestion window based on the number of ACK messages received. Because an attacker might be able to guess the sequence numbers in a connection, an attacker could flood a machine

with ACK messages – one per byte, perhaps. The receiver will count these ACK messages, and, believing the messages to be real, will open the congestion window. The client on the other end of the connection sees that the window opened and sends more data accordingly. This can be done many times, and will overwhelm the receiver with data from legitimate clients because of the illegitimate ACK messages. Or it may simply flood a network with traffic[14].

Essentially every configuration of a TCP packet has been used in a denial-of-service attack. The effectiveness of any configuration depends solely upon the strength of the TCP implementation. If there are flaws or oversights, then a particular configuration may cause a great deal of harm. But if there are no flaws and only minute oversights, then a certain packet configuration may not have any adverse effects useful to a denial-of-service attack.

2.5.4 UDP/IP vulnerabilities

UDP, or User Datagram Protocol, is an unreliable transport mechanism. UDP packets, like TCP packets, lie inside IP packets. Unlike TCP packets, UDP has no way of keeping track of the order of packets, and so it cannot protect against packet loss. UDP is particularly useful for certain streaming applications where a lost packet will not cause significant problems. Because of its nature, UDP does not have any acknowledgment mechanism. UDP does not have ACK messages like TCP. It means that UDP has fewer options for use as a reflector, and it also means that a sender can transmit unlimited amounts of data without ever waiting for a single reply from the receiver. UDP has a “chargen” service. Chargen, or character generator, returns a packet of arbitrary characters when prompted. Chargen is used for the maintenance of a network. UDP also has echo services, like ICMP.

Fraggle: essentially a remake of the Smurf program[5]. This is a reflector and amplifier attack. An IP broadcast is sent to a network. The UDP information specifies a request for an echo or to the “chargen” service. Each machine will respond with packets of characters, but the source address is spoofed, so the responses land on an unsuspecting victim[5].

Garbage Flood: UDP, requiring no ACK messages for its packets, is the perfect mechanism to use to hurl unlimited amounts of meaningless data at a target. Although such packets may not necessarily place computationally intensive loads on a target, if many machines send masses of garbage at a single target, then at the very least the target’s network will be flooded with traffic[6].

ICMP: Port Unreachable: Since ICMP is used to reply to UDP packets arriving on unused ports (the PORT_UNREACHABLE flag), it is possible for UDP packets to initiate a reflector attack with ICMP packets bombarding the target[15].

SNMP Exploitation: SNMP, or the Simple Network Management Protocol, uses a well-known UDP port for a request/reply service. UDP can be used to make request to SNMP in a reflector attack with SNMP’s UDP replies bombarding the target[15].

DNS Exploitation: see the following section on DNS vulnerabilities. UDP is used to make large numbers of resolution requests in a reflector attack in which the replies are sent to the victim.

2.5.5 DNS vulnerabilities

DNS, or Domain Name Server, is critical to the operation of the Internet in its current condition. A simplified explanation of its use: when a user types `www.somesite.com`, the computer needs to know the IP address (for discussion, let the address be `127.0.0.1`). The computer cannot pull this number out of the air, so it sends a name-resolution-request to the DNS. The DNS sees the name `www.somesite.com` and either returns `127.0.0.1`, or it makes a recursive query. In other words, if it does not know the IP address, it passes the request on to another DNS. If `www.somesite.com` exists then the computer will eventually get its answer: `127.0.0.1`. If the DNS had to make a recursive query, the resolution-reply will be larger than the resolution-request. If the computer could not have a DNS to which it could send its queries for addresses, then an IP packet could never reach its destination, and the computer would, more or less, be cut off from the Internet.

DNS Bombardment: an amplification attack. An attacker floods all of the name servers to which a machine or network makes its requests with recursive queries. Because the recursive queries require the help of other name servers, such a flood can disable a name server. If all of the name servers to which a machine or network makes its requests are disabled, then that machine, or any machine on that network, will be unable to communicate with the rest of the Internet.

Resolution Flood: a reflector attack that involves sending queries to a name server for which the replies are sent to the victim. The requests are UDP-based, and so require little effort for the attacker to produce. The more name servers that are abused, the more replies that will land on the victim. The victim will receive a flood of unexpected DNS replies[7]. If the spoofed requests in this attack require recursive searches, then the attack becomes an amplification attack as well, and the victim will suffer an even larger assault[8].

2.6 Defenses

Many of the following protections against denial-of-service are preventative measures. This means that while the measures will not stop a packet storm during an attack, they will make it harder for an attacker to set up and launch a DDoS attack in the future. Ways exist to directly protect machines from DDoS, such as filtering techniques and congestion control algorithms. Some ways help more than others, but if several methods are wisely combined, a machine might become the least appealing DDoS target an attacker could choose.

2.6.1 IP defenses

Fragmentation: this has become less and less of a threat as of late. Machines can be configured to wait only a short amount of time for the missing piece of a message before dropping all of the data. This way, machines free the resources wasted in fragmentation attacks quickly. The time the machine waits before dropping the data is the window in which such a fragmentation attack will work. Thus the fragmentation attack is still possible.

IP Spoofing: there are several ways to stop falsely addressed packets. The most effective way is egress filtering. When a packet leaves a machine and reaches the edge router, the router checks the source address. If the address does not fall within a valid range (i.e., the subnet mask or perhaps any of the addresses on the network) then the

packet is dropped[23]. CISCO routers provide another protection: the routers check a packet whenever they receive one. If the packet's source address is feasible based upon the interface on which the packet was received, then it is relayed. If the packet could not have possibly come from that interface, it is discarded[5]. As for straightforward ingress filtering, the destination network has several options. The receiver should not allow IP broadcasts. Allowing IP broadcasts gives an attacker a tool for reflector and amplification attacks such as the smurf attack[3]. The receiver should not allow packets to enter the network that have source addresses that originate from within the network. Under no legitimate circumstance would such a packet need to enter the network, and so these packets can only mean trouble. Lastly, the receiver might create a list of trusted addresses from which it will accept packets. However, this provides very little protection, as a spoofed packet might pretend to originate from a trusted address.

Tracing the packets that arrive in a DDoS attack offers little return in many cases. If the packets have spoofed IP addresses, then the only way to track the packet is to contact the administrators of all the possible routers closest to the receiver and hope they have made logs. Then, from the closest administrators, all of the administrators of the possible routers closest to that first set of routers must be contacted; and so on and so forth. This is a painstaking process that does not always end fruitfully. If the DDoS packets had valid source addresses, then the receiver will know all of the agents' addresses. In reality, DDoS attacks can involve so many agents that the packets have valid addresses. When it comes time to trace the attack, there are simply too many agents to proceed with the trace. There are, however, tools to aid in tracing an attack. These tools will be discussed at a later point.

IP Attack	Defense
Fragmentation	Timed packet dropping
Address Spoofing	Egress filtering, Deny incoming IP Broadcasts

2.6.2 ICMP defenses

Ping of Death: such an attack depends entirely on a poor implementation of ICMP. By writing an implementation that not only checks the size of the packet but also handles oversized packets properly, a machine can become totally impervious to the attack.

Smurf: once again, this is a reflector and amplifier attack. By preventing IP spoofing, the reflection is prevented. By denying IP broadcasts, the amplification is prevented[5]. However, if the reflection and amplification occurs, what happens when the storm of replies arrives at the target? A receiving machine may choose to shut down its ICMP services completely. Doing this curbs functionality, as ICMP exists to aid in the process of information exchange through the Internet. If the receiver filters out only ICMP's ping messages, then perhaps functionality suffers only an acceptable loss. It is, of course, at the discretion of the operators of a receiving machine. If the receiver filters against ping messages, then the smurf attack loses its punch.

Black Holes: protecting valid routes can prevent black holes. A machine must use some sort of measure to prevent any given person from telling it where to send its

packets. Passwords require little overhead, but give little security. Checksums that incorporate time require moderate overhead, but give moderate security and protection against the re-use of obsolete redirections. Encryption requires the greatest overhead, but also gives the greatest protection[19].

ICMP Attack	Defense
Ping of Death	Proper ICMP implementations
Smurf	Deny incoming IP Broadcasts, Shutdown ICMP services
Black Hole	Protect the routing tables

2.6.3 TCP/IP defenses

SYN Flood: preventing this attack has given headaches to many a person. It seems that since the only reason the attack works is because TCP allocates resources immediately, the only solution is to use a TCP that does not allocate resources immediately[16]. A machine receives a SYN, computes the sequence number and processes any necessary logic, then sends its SYN-ACK. It does not set resources aside for the connection until a valid ACK arrives, completing the three-step handshake.

Established Connection Exploitation: a simple way of stopping too many established connections is to limit the number of connections any process may have. If a process must not have a limit, then each connection should undergo authentication. A challenge-response system often solves the problem. The machine that receives the SYN replies with a SYN-ACK that has a puzzle with it. The returned ACK must have the correct response. Naturally, the answer should be computationally difficult for unauthorized users.

RST Storms: filtering against RST packets creates more problems. If a machine refuses to accept RST's, then the states it maintains for its connections will eventually linger. The accumulation of such states will consume the machine's resources. By denying the RST packets, a machine has effectively given itself a maximum time to operate correctly before it consumes its own resources. However, if a cooperative effort prevented IP spoofing, then RST packets could not be reflected off of unwitting parties.

Sequence Number Guessing: the use of more complex algorithms for determining the starting sequence numbers of a connection can impair the ability of an attacker to guess the numbers. Allotting blocks of sequence numbers to each connection offers a solution if the blocks are determined on a number of factors. If the TCP implementation wisely chooses its factors and how the factors will be used, then allotting such blocks can make a sequence number attack very difficult[17].

ACK Splitting: the effectiveness of this attack comes from TCP's counting the number of ACK messages to determine the congestion window. The solution, then, would be to cease counting the number of ACK messages. Instead, the congestion window should adjust according the number of bytes that been acknowledge in those ACK messages. Thus, a thousand ACK messages for one byte apiece adjust the congestion window by the same amount that a single ACK for a thousand bytes adjusts it[18].

TCP Attack	Defense
SYN Flood	Stateless 3-step handshake
EST Exploitation	Limit the maximum connections, Authentication
RST Storm	None
Sequence Number Guessing	High complexity sequence algorithms
ACK Splitting	Count ACK's by byte

2.6.4 UDP/IP defenses

Fraggle: Shutting down any access to UDP diagnostic ports from outside the network will prevent an attacker from using a network as a reflector and amplifier. This prevents fraggle attacks launched at others. Direct protection comes at the cost of denying all UDP services altogether. This way, any UDP packets received will be dropped and a DDoS attack will fail. However, denying UDP is not always feasible or practical. Lastly, the prevention of IP spoofing would be the prevention of fraggle attacks, because the attacker would not be able to bounce the UDP packets off of other networks.

Garbage Floods: As from fraggle protection, UDP services may be denied, stopping all incoming UDP packets[6]. If this is not possible, the use of congestion control techniques may mitigate the effects of an attack. See the later section on Congestion Control.

ICMP: Port Unreachable: Stopping the influx of the port-unreachable messages requires stopping the ping messages from ICMP. ICMP exists to help a working Internet, and if the ping messages can be filtered without a severe loss in functionality, then doing so will stop port-unreachable floods. IP spoofing makes this possible, and so preventing IP spoofing will prevent these attacks.

SNMP Exploitation: Restricting the use of SNMP to local users will help prevent SNMP's use as a reflector. This prevents a network from abuse as a reflector, not from an attack.

DNS Exploitation: See the following section on name server defenses.

UDP Attack	Defense
Fraggle	Deny certain UDP ports, Deny all UDP
Garbage Flood	Deny UDP, Congestion control
ICMP: Port Unreachable	Deny ICMP
SNMP Exploitation	Restrict SNMP to local users

2.6.5 DNS Defenses

DNS Bombardment: Restricting recursive queries to local users prevents such attacks from the outside[15]. Such a restriction makes it much harder to bring down a name server with numbers of requests.

Resolution Flood: Because this attack is made possible solely by the spoofing of IP addresses, preventing the spoofing would prevent the attack.

Black Holes: The best way to stop the formation of black holes is to protect the routing tables. Routing packets can have several security mechanisms: passwords,

checksums, and encryption[19]. Passwords have low overhead and low security, checksums have moderate overhead and moderate security, and encryption has the highest overhead with the highest security.

DNS Attack	Defense
DNS Bombardment	Restrict recursive queries to local users
Resolution Flood	None
Black Holes	Protect the routing tables

2.6.6 Generalized Defenses: Tracing the Attacks

Tracing a DDoS attack without any aid can turn out to be nearly impossible. At every hop through the Internet, a packet could have come from a number of routers. A victim of a DDoS attack has little or no hope of tracing an attack if the routers did not perform adequate logging. Furthermore, once the victim attempts to perform the tracing beyond the scope of the victim's local network, the assistance of the administrators of the other networks is required. If the addresses of the packets were spoofed, then who knows from where the packets really came. Even if the addresses were not spoofed, current DDoS attacks use so many agents in the assault that the sheer number of different addresses can make a trace infeasible. Reflector attacks, which exist because of IP spoofing, hide the locations of the agents, because a trace will likely end with the reflectors. Altogether, tracing without help is hopeless cause.

There are products out there that, if cooperatively deployed, can aid the tracing process. However, let the emphasis lie on the *cooperative* deployment of the products. If these tools happen to reside on every router traversed in an attack, then the attack could possibly be traced. Some such tools send ICMP messages to the target of the packets with a low probability. If an attack occurs, the probability goes up that the ICMP messages will be dispatched. When the attack is over, every router traverse in the assault might have sent an ICMP message. The victim can then piece together the source[22].

2.6.7 Generalized Defenses: Flow Control

This technique concerns cooperative efforts between networks and service providers. Flow control facilities reside on the routers and monitor the packets they relay. The facilities have a variety of uses: some simply detect DDoS attacks and report the occurrences, while other detect the attacks and attempt to piece together the source of the attack. These flow control utilities are proprietary and are effective only when put to use cooperatively. Most of the utilities perform scanning on the packets by either signature-based criteria or anomaly-based criteria. Signature scanning looks for telltale signatures that indicate a packet's illegitimate or malevolent use. Anomaly scanning looks for subtle or overt changes in traffic, referenced against previous traffic histories and known attack-traffic characteristics[22].

2.6.8 Generalized Defenses: Congestion Control

Congestion control involves grouping all incoming packets by a class or by a flow. TCP/IP, UDP/IP, and ICMP are classes. Classes can also come from distinguishing packets based on services (e.g., telnet, ftp, streaming video, etc.). Distinction based on

flows is generally done by separate the packets for each connection. Once all of the incoming packets have been grouped, the groups can be acted upon.

When traffic fills the network, important services should have priority for bandwidth over unimportant services. When services contend for bandwidth, which will receive it: a file server hosting data for clients, or an employee's personal ftp server attempting to host data for the employee's buddies? Naturally, a business would want to place priorities on the flows across its network so that important services are never starved of bandwidth.

This idea extends to DDoS. Think of the attack as an unimportant flow. Congestion control will forcibly hold the attack packets while the important services tend to clients. This way, a DDoS attack harms a service less. There are several mechanisms for performing this kind of flow control: Random Early Detection (RED) and its relatives, priority queues, RSVP, and others[20], including more cooperative methods such as Dynamic Resource Pricing[16].

2.6.9 TCP Wrappers

When a TCP/IP packet arrives, it is classified according to its service (e.g., ftp, telnet, etc.). This is the idea of a wrapper. Once the packet is wrapped according to its service, the machine may decide whether to drop the packet. This is a further use of ingress filtering to mitigate the impact of DDoS. However, wrapper utilities are usually the first things to be disabled when an attacker infiltrates a machine, as wrappers are well known.

2.6.10 DDoS Defense Table

Attack	Defense
IP	
Fragmentation	Timed packet dropping
Address Spoofing	Egress filtering, Deny incoming IP Broadcasts
ICMP	
Ping of Death	Proper ICMP implementations
Smurf	Deny incoming IP Broadcasts, Shutdown ICMP services
Black Hole	Protect the routing tables
TCP	
SYN Flood	Stateless 3-step handshake
EST Exploitation	Limit the maximum connections, Authentication
RST Storm	None
Sequence Number Guessing	High complexity sequence algorithms
ACK Splitting	Count ACK's by byte
UDP	
Fraggle	Deny certain UDP ports, Deny all UDP
Garbage Flood	Deny UDP, Congestion control
ICMP: Port Unreachable	Deny ICMP

3.0 The Aftermath

What next? The idea is to decide what types of attack does an administrator of a network or an e-business feel are the most prevalent and dangerous threat. Once that has been decided, the next step is to erect the castle walls. What defenses will offer little protection, and what defenses will limit functionality too severely?

DoS is no laughing matter. To stress the damage a DDoS attack can cause, it is helpful to think of time a server is down as customers lost. If a DDoS attack lasts for an entire day, or if it takes a day to recover from an attack, then how many clients could not make their transactions? How many prospective clients found the e-business dissatisfying and went with a competitor?

Perhaps, in the long run, the best defenses against DoS are preventative measures. IP address spoofing plagues the Internet so long as networks and service providers fail to filter outgoing packets based on the feasibility of the source address. The prevention of IP spoofing would smother the prevalence of reflection attacks that bounce packets with bad addresses. Preventative measures may seem inadequate, but for the time being, such measures can have the most effect.

4.0 References

1. "CERT Advisory CA-2000-21 Denial-of-Service Vulnerabilities in TCP/IP Stacks", Shawn Hernan, November 30, 2000, CERT, <http://www.cert.org/advisories/CA-2000-21.html>
2. "CERT Advisory CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks", September 19, 1996, CERT, <http://www.cert.org/advisories/CA-1996-21.html>
3. CERT Advisory CA-1998-01 Smurf IP Denial-of-Service Attacks", January 5, 1998, CERT, <http://www.cert.org/advisories/CA-1998-01.html>
4. "CERT Advisory CA-1996-26 Denial-of-Service Attack via ping", December 18, 1996, CERT, <http://www.cert.org/advisories/CA-1996-26.html>
5. "The Latest in Denial of Service Attacks: 'Smurfing' Description and Information to Minimize Effects", Craig Huegen, February 8, 2000, Pentics, <http://pentics.net/denial-of-service/white-papers/smurf.cgi>
6. "CERT Advisory CA-1996-01 UDP Port Denial-of-Service Attack", February 8, 1996, CERT, <http://www.cert.org/advisories/CA-1996-01.html>
7. "CERT Incident Note IN-2000-04", Kevin Houle, Friday, April 28, 2000, http://www.cert.org/incident_notes/IN-2000-04.html
8. "J-063: Domain Name System (DNS) Denial of Service (DoS) Attacks", September 1, 1999, AusCERT, <http://ciac.llnl.gov/ciac/bulletins/j-063.shtml>
9. "Trends in Denial of Service Attack Technology", Kevin Houle and George Weaver, October 2001, CERT, http://www.cert.org/archive/pdf/DoS_trends.pdf
10. "The DoS Project's 'trinoo' distributed denial of service attack tool", David Dittrich, October 21, 1999, University of Washington, <http://staff.washington.edu/dittrich/misc/trinoo.analysis>
11. "The 'Tribe Flood Network' distributed denial of service attack tool", David Dittrich, October 21, 1999, University of Washington, <http://staff.washington.edu/dittrich/misc/tfn.analysis>
12. "The 'stacheldraht' distributed denial of service attack tool", David Dittrich, December 31, 1999, University of Washington, <http://staff.washington.edu/dittrich/misc/stacheldraht.analysis>
13. "Axent releases a full TFN2K Analysis", Jason Barlow, August 3, 2000, SecuriTeam, <http://www.securiteam.com/securitynews/5YP0G000FS.html>
14. "TCP Congestion Control with a Misbehaving Receiver", Stefan Savage, Neal Cardwell, David Wetherall, Tom Anderson, 1999, University of Washington, <http://www.cs.washington.edu/homes/savage/papers/CCR99.pdf>
15. "An Analysis of Using Reflectors for Distributed Denial-of-Service Attacks", Vern Paxson, June 26, 2001, University of Leeds, <http://www.icir.org/vern/papers/reflectors.CCR.01/reflectors.html>
16. "Mitigating Distributed Denial of Service Attacks with Dynamic Resource Pricing", David Mankins, Rajesh Krishnan, Ceilyn Boyd, John Zao, Michael Frentz, 2001, ACSAC, <http://www.acsac.org/2001/papers/75.pdf>
17. "Defending Against Sequence Number Attacks [RFC 1948]", S. Bellovin, May 1996, Network Working Group, <http://www.faqs.org>
18. "TCP Congestion Control with Appropriate Byte Counting", Mark Allman, November 2001, IETF, <http://www.ietf.org/internet-drafts/draft-allman-tcp-abc-02.txt>

19. "Site Security Handbook [RFC 2196]", B. Fraser, September 1997, Network Working Group, <http://www.faqs.org>
20. "Cisco Quality of Service and DDOS", David Moore and Holly Xiao, July 25, 2001, The MITRE Corporation, http://www.mitre.org/support/papers/tech_papers_01/moore_cisco/moore_cisco.pdf
21. "Dynamics of Early Random Detection", Dong Lin and Robert Morris, 1997, Harvard University, <http://www.pdos.lcs.mit.edu/~rtm/papers/fred.pdf>
22. "DoS DEFENSE", Shon Harris, September 2001, Information Security Magazine, <http://www.infosecuritymag.com/articles/september01/cover.shtml>
23. "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing [RFC 2267]", P. Ferguson and D. Senie, January 1998, Network Working Group, <http://www.faqs.org>