Protecting Web Servers from Distributed Denial of Service Attacks

Frank Kargl
Department of Multimedia Computing
University of Ulm
Germany
frank.kargl@

Joern Maier
Department of Multimedia Computing
University of Ulm
Germany
joern.maier@informatik.uni-ulm.de

Michael Weber
Department of Multimedia Computing
University of Ulm
Germany
weber@

ABSTRACT

Recently many prominent web sites face so called Distributed Denial of Service Attacks (DDoS). While former security threats could be faced by a tight security policy and active measures like using firewalls, vendor patches etc. these DDoS are new in such way that there is no completely satisfying protection yet. In this paper we categorize different forms of attacks and give an overview over the most common DDoS tools. Furthermore we present a solution based on Class Based Routing mechanisms in the Linux kernel that will prevent the most severe impacts of DDoS on clusters of web servers with a prepended load balancing server. The goal is to keep the web servers under attack responding to the normal client requests. Some performance tests and a comparison to other approaches conclude our paper.

Categories and Subject Descriptors

C.2 [**Computer-Networks**]: General

General Terms

Security, Performance, Measurement

Keywords

Distributed Denial of Service Attacks, DDoS, Web Server Security, Class Based Routing, Linux

1. INTRODUCTION

1.1 Motivation

Security threats are as old as the Internet itself. In fact the first connection between computers in the ARPAnet between SRI and UCLA resulted in a crash of the receiving system due to some bugs in the communication software: a classical Denial-of-Service attack[1].

Another prominent story which 'DoS'-ed hundreds of machines is the Internet Worm[2][3][4]. But it was at the beginning of 2000 when a complete new quality of DoS attacks started to be used widely. So called Distributed Denial of Service attacks stroke a huge number of prominent web sites including Ebay, Amazon or Buy.com[5].

It is quite evident that as the attacking tools become more and more available in the cracking community (we intentionally avoid the term hacker here) the threats to services in the Internet become stronger. As the amount of monetary transactions that are handled over the Internet increases, this state cannot be accepted. Unfortunately we will see that there is no complete solution for this kind of attacks yet. The W3 Security FAQ[6] states that protection from attack tools like TFN, TFN2K or stacheldraht is an active field of current security research.

1.2 Attack scenario

Before going more into details on DoS attacks we will first give an overview over the systems that are typically involved in DoS attacks. As you can see in figure 1.2 big web sites usually use more than one system running their web server. The clients access these servers via a load balancing server which redirects the HTTP requests to one of the servers. Todays web servers don't work as stand alone systems but need the support of a number of backend systems (like database- or file-servers) to fulfill their tasks. The whole LAN network where the site is hosted is typically protected by a firewall system. On the way the IP datagrams have to pass a number of routers. On each of these systems there is at least the hardware, the operating system and (as part of the OS) a TCP/IP protocol stack that can fall victim to attacks like the ones we will describe in the next chapter. For some attacks the crackers use other hosts in the Internet as relays.

2. CLASSIFICATION OF DOS AND DDOS ATTACKS

2.1 General attack classification

Before discussing details of DDoS attacks and defense mechanisms it is useful to give a coarse grain overview and classification over security threats in general. After this we

WWW10, May 1-5, 2001, Hong Kong.
ACM 1-58113-348-0/01/0005.

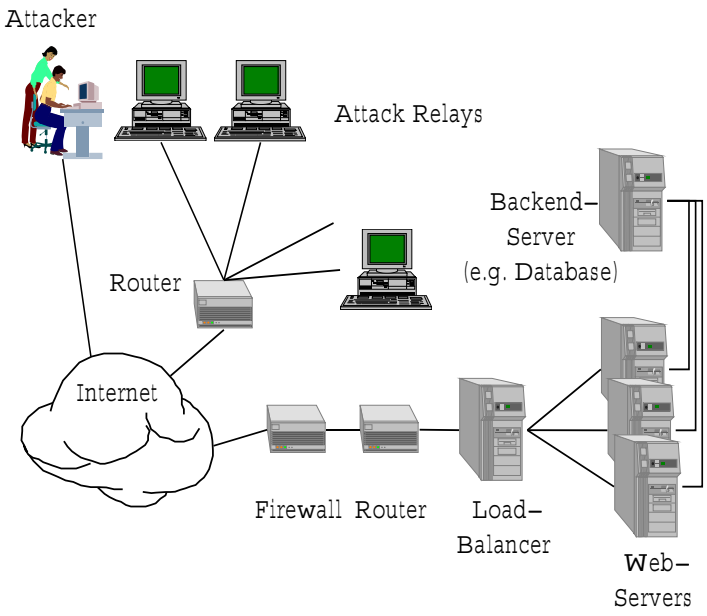

Figure 1: Overview of attack scenario.

will provide definitions for DoS and DDoS attacks and a more specific classification scheme for DoS attacks.

A possible classification of IT attacks according to the intention of the cracker could be[7]:

Denial of Service (DoS) The main goal of the attack is the disruption of service. This can be reached by a variety of ways as we will see later.

Intrusion Here the intention is simply to get access to a system and to circumvent certain barriers. People with such an intention meet the classic image of the old style hackers. Normally they try to reach their goal without doing severe damage and they inform the system administrator of the bugs found in the system.¡/li¿

Information Theft Main goal of this kind of attacks is access to otherwise restricted, sensitive information.

Modification Here the attacker actively tries to alter information. This kind of motivation is increasing lately as you can see from the enormous number of hacked and altered websites[8].

2.2 Definitions for DoS and DDoS

In the rest of the paper we will focus on attack forms which are motivated by the Denial-of-Service intention. The WWW Security FAQ[6] describes a DoS attack as:

> ... an attack designed to render a computer or network incapable of providing normal services. The most common DoS attacks will target the computer's network bandwidth or connectivity. Bandwidth attacks flood the network with such a high volume of traffic, that all available network resources are consumed and legitimate user requests can not get through. Connectivity attacks flood a computer with such a high volume of connection requests, that all available operating system resources are consumed, and the computer can no longer process legitimate user requests.

J.D. Howard defines DoS as[9]:

> If computer hardware, software, and data are not kept available, productivity can be degraded, even if nothing has been damaged. Denial-of-service can be conceived to include both intentional and unintentional assaults on a system's availability. The most comprehensive perspective would be that regardless of the cause, if a service is supposed to be available and it is not, then service has been denied.
>
> An attack, however, is an intentional act. A denial-of-service attack, therefore, is considered to take place only when access to a computer or network resource is intentionally blocked or degraded as a result of malicious action taken by another user. These attacks don't necessarily damage data directly, or permanently (although they could), but they intentionally compromise the availability of the resources.

The kind of DoS attacks that we are interested in is usually carried out via networks and one of the main targets of such attacks are high profile websites like the ones named in the introduction. As these sites usually have a lot of hardware at their disposal, they are much harder to attack than ordinary hosts connected to the Internet. The web sites are normally composed of several web-servers with a load balancing system in front and they have multi megabit network connections.

Consequently attackers have discovered new ways of bringing these systems to its knees. They don't use single hosts for their attacks but they also cluster several dozens or even hundreds of computers to do a coordinated strike. The WWW Security FAQ[6] on Distributed Denial of Service (DDoS) attacks, as these form is called:

> 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. Typically a DDoS master program is installed on one computer using a stolen account. The master program, at a designated time, then communicates to any number of "agent" programs, installed on computers anywhere on the Internet. The agents, when they receive the command, initiate the attack. Using client/server technology, the master program can initiate hundreds or even thousands of agent programs within seconds.

The aggregated bandwidth of this large number of agent programs is probably bigger than any website's uplink capacity. In fact one can even speculate on the effects of an attack to major IP routers or global exchange points which could render large parts of the Internet inaccessible. [10], [11] and [12] offer more information on the related technologies. We will also give some examples of common tools and attacks in one of the next sections.

2.3 DoS attack classification

DoS and DDoS attacks usually use a limited number of well known attacks with names like smurf, teardrop or SYN-Flood. Some tools use combinations and DDoS tools like TFN launch coordinated attacks from a big number of hosts. Again we will try to provide a classification in categories according to specified criteria. In subsection *2.3.4* we will give examples of common DoS attacks and try to categorize them.

2.3.1 System attacked

Our first category is the system under attack. According to the scenario in section 1.2 we can identify a number of attack points that would render the web server useless to normal clients.

First of all we could attack the clients themselves, which is kind of useless as a general DoS attack, because there are just too many of them. A first reasonable attack point for a cracker wanting to launch a DoS is the router that connects the site hosting the webserver to its ISP. This would effectively cut off all access to the website.

Another possible target is the firewall system, although firewalls usually tend to be (or at least should be) quite immune to direct attacks. On the other hand, the firewall is usually a bottleneck as all in- and outbound traffic needs to pass through it. An attack form with a very high load might stop the firewall from working properly. As an example: during our experiments some misconfigured SYN-flood attack overloaded our university's firewall and blocked most in- and outgoing traffic.

The load-balancer is another attack target. As more and more manufacturers of such systems will become aware of this threat there is a good chance that this system will be one of the main points of defending web sites from DoS attacks.

Next the individual webservers can be attacked. As there are usually multiple servers, the effort to overload them all is usually higher than to attack a single bottleneck like the load-balancer. On the other hand, most webservers use of-the-shelf hard- and software and are thus especially vulnerable to known DoS attacks.

Finally there may be supporting services like database servers etc. As foreign access to these servers should be blocked at the firewall, the chance of a direct DoS attack on one of this systems is fairly small. Nevertheless a tricky attack on the webserver may cause a flood of subsequent requests to e.g. a database system so that normal requests aren't processed anymore.

2.3.2 Part of the system attacked

Attack forms can be further divided by the part of the system that is attacked. Some attacks may affect the hardware of a given system although such incidents are very rare. Theoretically, a network card or CPU could fail to work due to some data in a network packet it receives or processes. On the other hand, a simple overload of a 10 MBit/s Ethernet link is an attack based on the limitation of the hardware.

Attacks targeting the operating system or the TCP/IP stack of a host or router are more likely. Many attacks of this type are known, some are bugs that can be fixed, some are fundamental limitations of a protocol specification etc.

On the application level there are web servers, database servers, CGI scripts etc. These could be attacked as well.

2.3.3 Bug or Overload

In general one has to distinguish whether a DoS is a cause of a specific bug or just an overload of components that function according to their specification. Although bugs are often more severe in their effects, most of the time the vendors quickly provide fixes. All the administrators have to do is to apply them to their system in order to avoid further attacks. Attacks that are based on an overload are typically harder to cope with. Of course you can buy new hardware, but as long as an attacker finds enough resources to use as relays in the Internet he will always bring your system to a halt. Changing the specification or protocols in order to fix the hole that allows the DoS is nearly impossible as this would often mean changing the software in millions of computers worldwide.

2.3.4 Examples

The following examples give a little insight in different DoS attack forms. They show that our categorization is suitable for distinguishing common DoS attacks.

Cisco 7xxx routers with IOS/700 Software Version 4.1(1) or 4.1(2) have a bug where a long password given at a telnet login session leads to a buffer overflow and a subsequent reboot. According to our criteria the attacked system is a router, the part of the system under attack is the operating system and it is a bug which has been fixed in subsequent releases[13].

Jolt2 is an attack targeting most Microsoft Windows systems (Win 98, NT4 SP5 and SP6, Win 2000). Jolt sends a continuous stream of ICMP ECHO_REPLY fragments with

specially tuned fragmentation parameters to the attacked host. The exact cause of the following action is not known as the code of Microsoft's Operating Systems isn't freely available. What can be observed is that the consumption of CPU and memory resources raises to 100% which renders the system is unusable. This may be used as an attack on webservers, load balancers or even firewalls, if they run one of the named operating systems. The attacks aim at the network stack of the operating system and the reaction is caused by a bug. Fixes are available from Microsoft[14].

Microsoft's Internet Information Server versions 4.0 and 5.0 suffered from a so called server URL parsing bug. The decoding of escape sequences in URL strings was implemented very inefficiently. Submitting long strings with large amounts of escape characters effectively stopped the web server from working for a significant time. This attack is an attack against the webserver application. It is based on an implementation bug. A fix is available from Microsoft[15].

Smurf attacks use so-called amplifier sites in order to multiply the amount of traffic that hits the destination. These attacks send ICMP_ECHO_REQUEST packets with a spoofed sender address to one or several subnet broadcast addresses. The packets are received and replied by as many stations as are connected to the subnet. The replies are sent directly to the spoofed address of the attack destination. Normally this leads to congestion in the target's local network connection. Often even the lines of the ISP to which the target is connected become overloaded. This attack class strikes all kinds of targets, no matter if routers or hosts. Most of the time it is used against webservers. The effects of the attack typically put a huge stress on the network and the system (both software and hardware) that has to parse and discard the incoming packets. It is not based on a bug, although many people today think that subnets that allow a broadcast ping are a faulty configuration[16][17][18].

The main principle of SYN flood attacks is to generate many half-open TCP connections by sending SYN packets to the target without replying to the following SYN-ACK packet. Most of the time the SYN packets also have spoofed source addresses of non-existent or currently inactive hosts. As today's hosts can typically handle only a limited number of half-open connections per port and discard new connection requests as long as the so called backlog queue is full, this effectively stops ordinary clients from connecting to the (web-) server. This attack form falls in the category of attacks to webservers or load-balancers which have a weakness in their TCP implementations. Part of the problem is the unlucky connection establishment procedure of the TCP protocol so it is not totally clear, weather it is an implementation bug or design flaw[19][20][21].

There had been a number of attacks on apache webservers like Apache MIME flooding[22] or Apache Sioux Attack[23]. With specially formatted HTTP requests it was possible to make the webserver use up huge amounts of memory. As soon as the system started swapping out server processes or used up the whole system memory, the server was effectively dead. These attacks both target a specific webserver software on the webserver systems. They exploit implementation bugs.

This concludes our list of DoS examples. Of course this is only a small selection, online archives[24] provide huge lists of bugs and exploits in former and current software versions. Many of these exploits were packaged into attack tools and some of them are used by DDoS tools in order to attack a site from many locations at once.

2.4 From DoS to DDoS

Major Internet websites like amazon or Yahoo tend to have Internet connections with very large bandwidth and server farms with lots of components. Furthermore they are typically protected by firewall systems that block the known attacks that are based on malformed packets like jolt2 does. In the second half of 1999 DDoS tools matured to a point where a wide spread use was foreseeable. In November the CERT/CC invited a number of accredited security experts to a workshop on distributed-systems intruder tools[12]. Their fears about large-scale attacks were proved soon later in February 2000 when major Internet sites where under attack[5]. There are currently a few popular DDoS attack tools, which we will describe in the following sections: Trinoo, Tribe Flood Network (TFN), it's successor TFN2K and a tool called 'stacheldraht'.

The architecture of these tools is very similar and in fact some tools are modifications of others. The actual attack is carried out by so called daemons. A number of daemons is controlled by a handler and finally these handlers are activated by the attacker using client tools. Figure 2.4 displays the architecture of such a system.

The intrusion into computers onto which handlers and daemons are to be installed usually follows a simple pattern[25]:

1. A stolen account is set up as a repository for precompiled versions of scanning tools, attack (i.e. buffer overrun exploit) tools, root kits and sniffers, DDoS handler and daemon programs, lists of vulnerable and previously compromised hosts, etc.

2. A scan is performed on large ranges of network blocks to identify potential targets. Targets would include systems running various services known to have remotely exploitable buffer overflow security bugs, such as wu-ftpd, RPC services for "cmsd", "statd", "ttdbserverd", "amd", etc.

3. A list of vulnerable systems is then used to create a script that performs the exploit, sets up a command shell running under the root account that listens on a TCP port and connects to this port to confirm the success of the exploit.

4. From this list of compromised systems, subsets with the desired architecture are chosen for the Trinoo network. Pre-compiled binaries of the DDoS daemons and handlers programs are created and stored on a stolen account somewhere on the Internet.

5. A script is then run which takes this list of "owned" systems and produces yet another script to automate the installation process, running each installation in the background for maximum multitasking. The result of this automation is the ability for attackers to set up the denial of service network in a very short time frame and on widely dispersed systems whose true owners often don't even realize the attack.

6. Optionally, a "root kit" is installed on the system to hide the presence of programs, files, and network con-

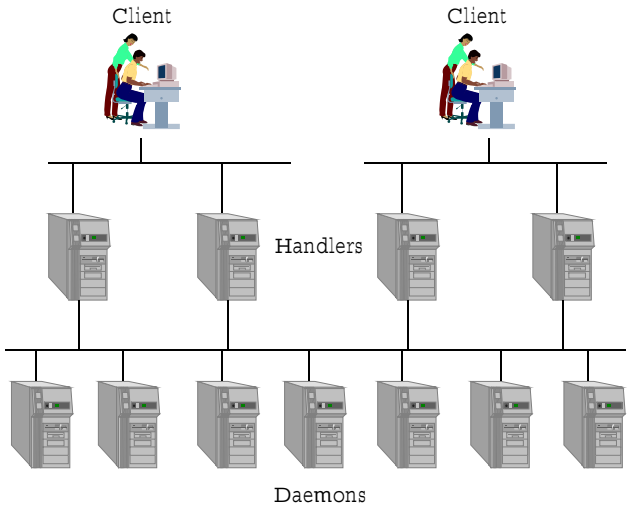

Figure 2: Architecture of DDoS tools

nections. This is more important on the handler system, since these systems are the keys to the DDoS attack network.

2.4.1 Trinoo

Trinoo was the first DDoS Tool widely known. It uses UDP flooding as attack strategy. Access to the handlers is simply done via a remote TCP connection to the master host. The master communicates with the daemons using plain UDP packets[25].

2.4.2 Tribe Flood Network

TFN was written in 1999 by someone using the pseudonym 'Mixter'. In addition to Trinoo's UDP flooding it allows also TCP SYN and ICMP flood as well as smurf attacks. Handlers are accessed using standard TCP connections like telnet or ssh. Other alternatives are ICMP tunneling tools like LOKI[26][27]. Communication between the handler and the daemons is accomplished with ICMP_ECHO_REPLY packets which are harder to detect than UDP packets and can often pass firewall systems[28].

2.4.3 TFN2K

TFN2K is the successor to TFN and was also written by Mixter. It incorporates a number of improvements like encrypted communication between the components which makes it much harder to detect TFN2K by scanning the network. Handlers and daemons can now communicate using either ICMP, UDP or TCP. The protocol can change for each command and is usually selected randomly. There is one additional attack form called TARGA attack. TARGA works by sending malformed IP packets known to slow down or hangup many TCP/IP network stacks. Another option is the so called MIX attack which mixes UDP, SYN and ICMP_ECHO_REPLY flooding[29].

2.4.4 stacheldraht

stacheldraht is supposed to be based on early TFN versions and is the effort to eliminate some of its weak points. Communication between handlers and daemons is done via ICMP or TCP. Remote control of a stacheldraht network is accomplished using a simple client that uses symmetric key encryption for communication between itself and the handler. Similar to TFN, stacheldraht provides three attack forms: ICMP, UDP and TCP SYN flooding. A characteristic feature of stacheldraht is its ability to perform daemon updates automatically[30].

2.4.5 What will be next

The future will surely bring more elaborate DDoS tools which will simply improve the given features by better encryption and more attack forms. It can also be expected that future tools will have better and easier (graphical) user interfaces allowing an application even by novice users. Eventually we will even see an integration of the distribution phase of handler and daemon programs into the user interface. Such a development will surely increase the number of attacks by an order of magnitude.

Even more frightening might be a scenario where the tools don't stop at DDoS attacks but automate the process of intruding hosts with daemons which again spread themselves to other hosts. This way automated hacking into hundreds of thousands of computers might be possible.

3. PROTECTION FROM DDOS

3.1 General Protection

To provide protection from DoS or DDoS attacks, basic security measures are mandatory. If a running system is hacked into, no more network attacks are necessary, since local attacks (like processes consuming lots of memory or CPU time, or simply shutting down the system) are far more effective. A set of firewalls should be used to separate the interior net (and probably a demilitarized zone) from the Internet. Intrusion Detection Systems should be used to notify the system administrators of unusual activities.

The firewall rules should include some sanity checks for source and destination addresses: Packets arriving from the

Internet must not have a source address originating from the interior net, and vice versa. By rejecting packets from the interior net with a non-local source address, packet spoofing becomes impossible. This technique is known as ingress and egress filtering[31]. Even if a host is invaded by a hacker, these rules make it impossible to use that host as a platform for further attacks requiring spoofed packets (e. g. SYN-Floods[20], Smurf-Attacks[15][17], etc.).

As explained in section 2, there are several well-known kinds of DoS attacks that exploit implementation bugs and for many of them there are known countermeasures. Of course, it is highly recommended to take these measures by installing the appropriate security patches, enabling SYN cookies[32], etc.

In contrast to attacks focusing on implementation or protocol errors, it is rather difficult to defend against DoS or DDoS attacks which overload the systems network connection or local resources. These attacks usually put a heavy load on the target by making regular requests very rapidly. It is hard to distinguish if a web server is stormed by thousands of clients, or if there is a DoS attack in progress.

A simple way to force the problem of heavy load is to use a server farm together with a load balancer. This will help against small attacks, but not against a DDoS started from several hundred hosts. Furthermore, increasing the number of servers is rather expensive.

Many experts think that the only durable solution to this problem is to globally improve the security on all hosts in the Internet to take attackers the possibility to use other hosts for running daemons. This includes securing the local operating system, applying ingress- and egress filters and protecting sites with firewall systems[12][33][34].

We don't think that the global state of Internet security will become any better in the foreseeable future. While many companies will start improving their security, taking measures like the ones detailed above, the growth of the Internet will simply absorb these effects. Many new companies and even more individuals with permanent connections will storm the net and many of them won't have the time, budget or will to secure their systems. Some people think about incentives to force people to care for security. One example of such an action is the abolishment of flatrates and the network wide use of usage-based fees[33]. We don't think that users and industry will follow these propositions, so solutions to protect the Internet sites under attack need to be found. There are already a small number of propositions how to deal with certain attack forms.

A simple response to SYN floods is the "first packet reject" method: The first packet sent by every host is discarded. This renders the common SYN flood attack ineffective, but all normal connection requests are delayed to the first retransmit of the SYN packet. Thus, all requests will be considerably slower. On the other hand, the SYN flood attack may be simply adapted by sending each SYN packet twice.

Another technique is the so called moving target defense. Here the host under attack changes its IP address to avoid being attacked. The problem here is that the legitimate users of the system need to be informed that the IP address has changed which usually is done by updating the DNS system. Due to caching it can take up to a number of days until all clients are informed of the update. This is clearly unacceptable as the effects are as severe as any DoS attack can be. Furthermore attackers only need to incorporate DNS lookups into their tools in order to evade this protection.

We developed a solution that tries to enhance the protection from DoS and DDoS attacks to webservers. It uses a combination of publically available protection tools and augments them with a load monitoring tool that stops clients (or attackers) from consuming too much bandwidth.

3.2 Our DDoS Protection Environment

Our system consists of several webservers that are accessed via a load balancing tool. All systems are running Linux and all other software is either freely available or self-written. Figure 3.2 shows an overview of the components. We use a number of readily available security options in the kernel to provide basic protection. The Linux Virtual Server is used for high speed load balancing. ipchains firewall mechanisms protect the load balancer from unwanted traffic. The web servers are running the popular apache server software. Our Traffic Shaping Monitor scans the network and reports unusual activity to the central monitor on the load balancer. This component can manipulate traffic using either the Class Based Queuing or ipchains filtering of the kernel.

3.2.1 Linux Kernel

We have carefully chosen Linux Kernel Version 2.2.16 as base for all our systems as this is known to be immune to most poisoned traffic attacks like teardrop or TARGA. The backlog queue of the system defaults to 128 entries and tcp_syn_cookies is enabled. This makes the system very robust against SYN flood attacks.

3.2.2 Linux Virtual Server

The load balancer we use is the Linux Virtual Server (LVS)[35]. LVS inserts itself directly into the kernel which provides a maximum performance again stabilizing the system against overload attacks. LVS has two load balancing algorithms: round robin and least connection. We are using 'least connection' as this provides generally a fairer load distribution between the webservers. There are three different modes to access the webservers.

Network address translation (NAT) transcripts every incoming packet and changes the destination IP from the load balancer's to the web server's IP. All outgoing traffic is transcripted alike. As all in- and outgoing traffic has to pass the load balancer, this is not an ideal solution for our purposes, as the load balancer may easily become a bottle neck.

Direct Routing Request Dispatching (DR) changes the layer 2 MAC addresses of incoming packets to the MAC address of the web server and forwards the packets. Web servers may answer directly, bypassing the load balancer. All Webservers must reside in one IP subnet for this to work.

IP Tunneling (IPIP) is a solution where incoming packets are wrapped in an IPIP encapsulation and are tunneled to the webserver. At the tunnel end packets get unwrapped and are delivered to the webserver application.

We have chosen to use IPIP mode as this has no restrictions on subnets and showed to work very efficiently.

3.2.3 ipchains Firewall

All systems protect themselves from unauthorized access by filtering incoming packets according to a number of security rules. In brief the rules state that only port 80 is

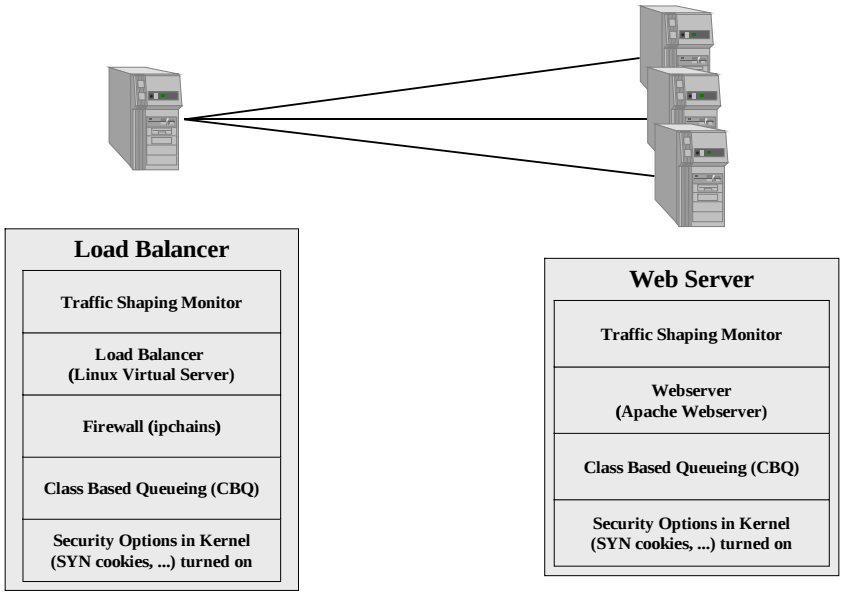

Figure 3: Overview of protection environment

reachable directly and only ICMP host unreachable messages are accepted. Another set of rules allows communication between the load balancer and the webservers as well as access to some important local services (DNS server etc.). This configuration may later be modified dynamically by the Traffic Shaping Monitor to totally block all traffic from attacking hosts. All these measures provide a pretty stable environment which should block all common attacks to the systems and leave only the web server reachable. The only two potential security holes that are not covered here are bugs in the webserver (or CGI scripts etc.) and overload attacks which generate a large amount of HTTP traffic. The first aspect is not covered here and needs to be considered by a careful webmaster who regularly checks security forums for bug alerts in the apache webserver and who carefully checks all active components deployed (like CGI scripts, servlets etc.)

3.2.4 *Class Based Queuing and the Traffic Monitor*

Class Based Queuing (CBQ) is a function of the Linux kernel. It allows the setup of different traffic queues and of rules that determine what packets to put in what queue. Furthermore you can assign a certain amount of the available bandwidth to each of the queues. If a queue is full packets get discarded. There are different queuing disciplines from which we have chosen *Stochastic Fairness Queuing (SFQ)* because it consumes only few memory and computing power. On the other side it is not fully deterministic in what packets end up in what queues. For our purposes this is no problem. Other available disciplines include Token Bucket Filtering or Random Early Detect. For more information refer to[36].

In our system there is a configurable number of input queues on the load balancer and output queues on the webservers. There is a default input queue which can consume as much bandwidth as available. If the Traffic Monitor in the load balancer detects a possible DoS attack it gradually slows down all incoming traffic from the origination IP address by assigning it to more and more slower queues with e.g. 1000 kBit/s, 600 kBit/s, 300 kBit/s and finally to a queue with only 100 kBit/s. If even this does not stop the attack, the IP address is blocked in a firewall list for a configurable amount of time. At the same time it directs the webservers to slow down the outgoing traffic to the attacker's IP address gradually.

The Traffic Monitor consists of a manager and a monitor program. The monitor is running on the load balancer and all webservers. It is implemented as 3 separate threads. Thread 1 monitors the network for packets destined to or originating from the virtual web servers address. The source IP address, the length and the time of occurrence are noted in a hashtable (with the IP address as key). Thread 2 checks the hash table every 3 seconds. If it finds that a certain IP address is emitting or receiving too many traffic or if the packet/size ratio falls under a certain amount it marks the IP address as a potential attacker. Thread 3 is a server thread which listens for commands from the manager. The manager polls all the monitors in regular intervals for a list of potential attackers. It analyzes the supplied data and decides whether a potential attacker is indeed categorized as malicious. In this case the manager instructs the monitors to downgrade the attackers IP address to a lower queue or block it at all. After a certain interval of normal activity, IPs can be upgraded to better queues.

The manager sorts the IPs in one of several classes based on the data it receives from the monitors. Class 1 IPs have produced too much traffic with very small packets. As this is very likely a DoS attack, new packets from that source are totally blocked at the ipchains filter. Class 2 IPs use too much bandwidth over a long time. They get downgraded into a lower queue. Class 3 IPs use too much bandwidth

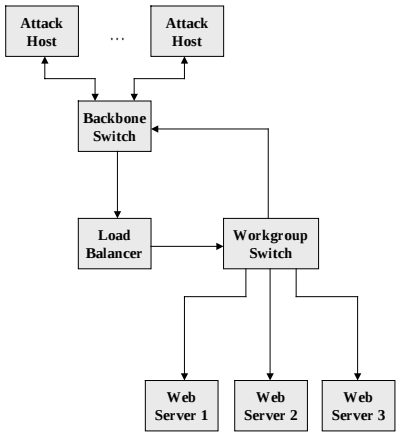

Figure 4: Test scenario

but only for a short time. This may as well be a peak from an ordinary client. These IPs are now under suspicion. A timer is started and if the behavior of this IP persists it is put into class 2. Finally there is class 4. IPs in this class don't produce or consume much bandwidth but they send lots of very small packets. Again there is a timer and if this behavior is shown for a certain amount of time, a ipchains filter blocks traffic from that IP.

All filters and queues have associated expiration timers. After expiration the filter is deleted and the queue is upgraded to the next higher class.

4. PERFORMANCE TESTS

We have conducted a number of experiments where we targeted our webserver system with different DDoS attacks. The testing environment is as shown in figure 4. All systems used run SuSE Linux (Kernel 2.2.16) on a 800 MHz Athlon CPU and 256MByte RAM. All network connections are switched and full-duplex 100 Mbit/s Ethernet (100 Base TX). The LBS consists of three web-servers and one load balancer.

For our test we use 8 attack hosts and one 'normal' client which has a normal browsing behavior. With this test we want to show that the 'normal' client is not effected by the attack but can access the website without disturbance. As you can see, the attack hosts can in theory produce about 8 times the input capacity of the load balancer. So even a higher number of attack hosts would not be able to put more stress on the load balancer.

Normal attacks like TCP SYN floods did not show any significant degradation of system performance but where handled effectively by the Linux SYN Cookie mechanism. UDP floods were blocked by the ipchains filters. Of course it is still possible to overload the network connection at some point in front of our system. The normal consequence would be to install the filters as early in the traffic flow as possible blocking everything except HTTP traffic (tcp/80).

We use the following tools for attacks. http_load written by Jef Poskanzer[37] which runs multiple HTTP fetches in parallel to test the throughput of a web server. It runs as a single process so it doesn't take to much CPU time. In our test the program reads 100 URLs from a file and tries to fetch them from the webserver randomly over a specified period of time. http_load uses 64 threads in parallel which try to fetch as many pages as possible in 210 seconds of time. We use three different URL sets: set one consists of 100 static html pages, set two consists of 100 URLs which access a CGI script with different parameters. Finally set 3 consists of 34 URLs accessing the CGI script and 66 static html pages.

The other test tools are TFN2K (see section *2.4.3*) and SYN Flooder from a hacker called Zakath. SYN Flooder performs a heavy SYN-flood attack. We slightly modified it so we can do SYN-flood attacks with randomly generated IP addresses.

For simulating an ordinary web client, we decided to run http_load with a single thread and the mixed URL database.

As a reference we ran this tool several times without any attack to the web site. Then we ran the tool while the system was under attack with and without the traffic shaping monitor tool active. In a first run, all attack hosts use the same attack at a time. We tested the following cases

- http-attack using http_load and static html database
- http-attack using http_load and the CGI database
- http-attack using http_load and the mixed database
- SYN - flooding attack using TFN2K
- ICMP - flooding attack using TFN2K
- TARGA-flooding attack using TFN2K
- mixed attack (SYN-ICMP-TARGA) using TFN2K
- SYN-flooding attack with single IP address using SYN Flooder
- SYN-flooding attack with spoofed IP addresses using SYN Flooder

Next we ran mixed attacks, where

- 3 computers running a http-attack with the mixed database and 5 computers doing a TFN2K mixed attack
- 5 computers running a http-attack with the mixed database and 3 computers doing a TFN2K mixed attack
- 3 computers running a http-attack with the mixed database and 5 computers doing a TFN2K SYN-flood attack
- 5 computers running a http-attack with the mixed database and 3 computers doing a mixed TFN2K SYN-flood attack

Traffic measuring was done by inserting a hub at different locations in the test network. There we connected a notebook running the Linux network analyzers ethereal and tcpdump. Before starting the tests we did some measurements to ensure that the system is really capable of snooping a fully loaded 100 MBit/s Ethernet. We had to reduce the snap length to 64 byte to achieve this, but this has no negative influence on the measurements as this is enough to

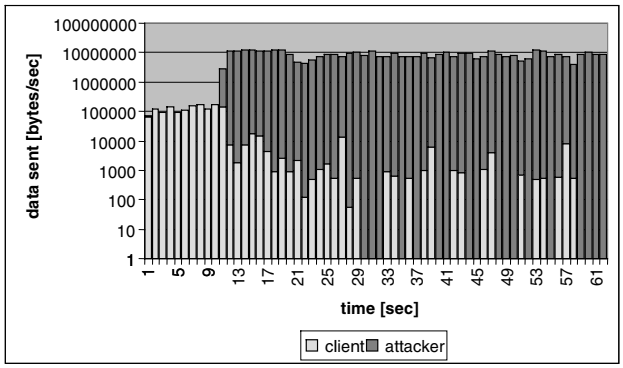

Figure 5: Test 1 without traffic monitor

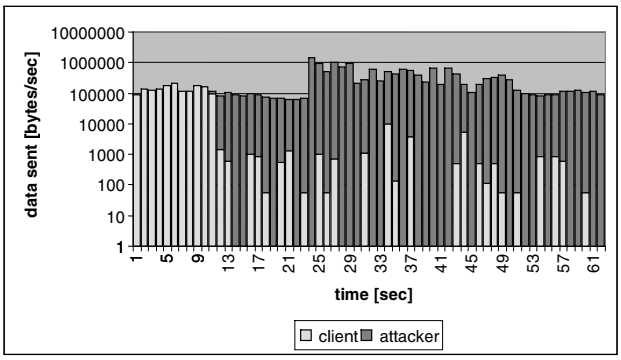

Figure 7: Test 2 without traffic monitor

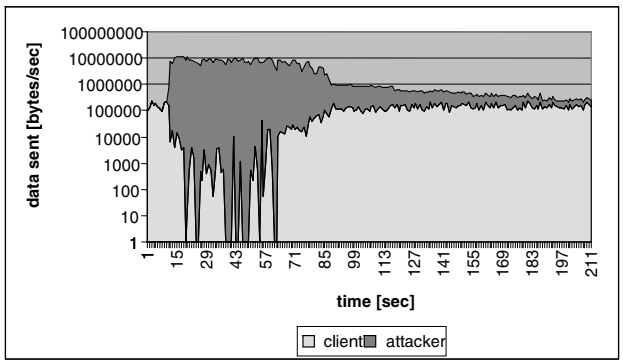

Figure 6: Test 1 with traffic monitor

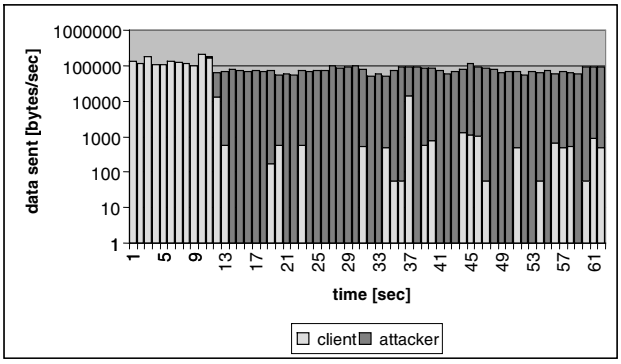

Figure 8: Test 2 with traffic monitor

determine the source and destination IP as well as the size of a packet.

The sniffer is inserted at 3 positions:

BP1 (break point 1): Between the backbone switch and the load balancer. Here all incoming traffic can be gathered. This is important because not all traffic is passed to the webservers.

BP2 (break point 2): Between the load balancer and the workgroup switch. Here all data can be gathered that is forwarded to the webservers. This way we can determine how much traffic has already been blocked by the load balancer.

BP3 (break point 3): Between the backbone and the workgroup switch. Here all outgoing traffic can be measured. This is all the traffic produced by the webservers.

Due to limited space, we will present here the results of only two of the tests, namely the http-attacks with a static and a CGI database. These tests show how our traffic monitor reacts to http overload attacks under different conditions.

4.1 Test 1: http-attack using http_load and static html database

Figure 4 shows the result at BP3 without the traffic shaping monitor. The client http_load started before the attack took place. 10 seconds later started to collect data from the webservers, the eight computers running http_load with 64 threads in parallel started to attack the system. The client's performance is constantly decreasing, finally only sporadic packets are sent. Obviously the system is nearly unreachable for a normal user. Using the traffic shaping monitor we get a different result as shown in figure 4. Again the attack starts after 10 seconds. The system realized that it was under attack and degraded the attackers hosts in bandwidth restricted queues. So the attackers get only a limited amount of incoming and outgoing bandwidth and the normal client can access the webserver in an almost normal manner.

4.2 Test 2: http-attack using http_load and CGI-database

In this test, we don't serve static html documents to the client but instead they are generated via a CGI script. Figures 4.1 and 4.1 show the system under attack at BP3 without and with the traffic shaping monitor. As we can see, the overall traffic is much lower than in the previous test scenario. The reason for this is that the CGI scripts are very CPU intensive. As a single attacker produced only around 10000 Bytes/s of traffic and this is less than the threshold at which the traffic shaping monitor classified a host as an attacker, the attackers aren't recognized by the traffic shaping monitor and no countermeasures are taken.

It is pretty obvious, why our traffic shaping monitor is of no use in this scenario. As there is no sensor for CPU load in the system and the used bandwidth is well within the allowed range, the traffic monitor can't react accordingly. Again this shows a general problem of DoS defense. You can only react to attacks that you can distinguish from normal behavior. In this special case one could install an additional CPU monitor on the web servers that monitors the execution of CGI scripts. The monitor needs a way to relate the CPU time used by a CGI script to the IP address of the client that triggered the script. It could then transmit the calculated resource usage to the central load monitor which could in turn activate countermeasures.

5. RELATED WORK

Most manufacturers have realized the importance of defense measures against DoS attacks. Since firewalls, routers, and switches are highly specialized devices, highly optimized defenses may be implemented at several points. Almost all firewalls are able to hold half-open states for several thousand connections; incoming packets are checked against signatures of known DoS attacks. Firewalls are even capable of stateful processing of packets: This allows dropping of packets which make no sense in the current state of a TCP connection.

ArrowPoint[38] is producing switches with load-balancing capabilities. These switches have been extended to provide some security against DoS attacks. Fragmented packets or packets being too short are dropped. Further, some checks on the IP addresses are done (e. g. source and destination address must be different, loopback or multicast addresses are not allowed). TCP connections are parsed, too. The 3-way handshake must be completed within 16 seconds, otherwise these packets are dropped. Another useful feature is the possibility to block packets directed to special ports, or originating from certain IP address ranges. Network address translation (NAT) is provided, too.

F5 is the manufacturer of BigIP, a load balancer switch with several security-relevant features. First, BigIP provides a monitor tool to watch the network traffic. Attack attempts will be noted in a log file. BigIP provides port mapping (i.e. packets for a special port are transparently redirected to a different host) and NAT. Further, packet filtering is provided. Packets belonging to several kinds of attack (e.g. Teardrop, Land, Ping of Death) are recognized and will be discarded[39].

Both manufacturers provide firewall load balancing: The traffic is equally shared on several firewalls; the switch makes sure that all packets belonging to the same stream will be routed through the same firewall. This allows both load balancing and the possibility to set up a redundant backup firewall.

6. CONCLUSION

As we have shown, DDoS attacks are a substantial threat to todays Internet infrastructure. We don't believe that a global security standard can be reached in a foreseeable time that would prevent attackers from finding and using lots of relay hosts. Our solution to the problem of handling massive http overload requests is based on class based routing and active traffic monitoring. Attacking hosts are detected and assigned to low bandwidth queues or even blocked completely. Tests have shown that systems protected in such a way can deliver regular service to their clients while under attack. We think that automatic traffic monitoring and automatic traffic shaping are promising ways of dealing with high bandwidth DoS attacks and should be implemented in future commercial products.

7. ADDITIONAL AUTHORS

Additional authors: Stefan Schlott
email: stefan.schlott@informatik.uni-ulm.de

8. REFERENCES

[1] K. Hafner, M.Lyon. *Where Wizards Stay Up Late.* Simon & Schuster, New York, 1996.

[2] E.H. Spafford. *The internet worm program: An analysis.* Purdue Technical Report CSD-TR-823, Deoartment of Computer Sciences Purdue University, West Lafayette, IN. 1988.

[3] D. Seeley. *A tour of the worm.* Department of Computer Science, University of Utah, 1988.

[4] M. Eichin, J. Rochlis. *With microscope and tweezers: An analysis of the internet virus of november 1988.* Massachusetts Institute of Technology, 1988.

[5] M. Williams. *Ebay, amazon, buy.com hit by attacks*, 02/09/00. IDG News Service, 02/09/00, http://www.nwfusion.com/news/2000/0209attack.html - visited 18.10.2000.

[6] L. Stein. *The world wide web security faq, version 2.0.1.* http://www.w3.org/Security/Faq/ - visited 04.10.2000.

[7] S.M Bellovin, W.R. Cheswick. *Firewalls and Internet Security.* Addison Wesley Longman, 1994.

[8] *Attrition mirrored sites.* http://Attrition.org/mirror/attrition/ - visited 03.11.2000.

[9] Dr. J.D. Howard. *An analysis of security incidents on the internet 1989 - 1995.* Carnegie Mellon University, Carnegie Institute of Technology, http://www.cert.org/research/JHThesis/ - visited 02.11.2000.

[10] J. Elliot. *Distributed denial of service attacks and the zombie ant effect.* IT Professional, Mar./Apr. 2000, pp55-57.

[11] K.T. Fithen. em Internet Denial of Service Attacks and the Federal Response. Testimony before the Subcommittee on Crime of the House Committee on the Judiciary and the Subcommittee on Criminal Justice Oversight of the Senate Committee on the

Judiciary, February 29, 2000, http://www.cert.org/congressional_testimony/Fithen_testimony_Feb29.html - visited 10.11.2000.

[12] *Results of the Distributed-Systems Intruder Tools Workshop* Pittsburgh, Pensilvania USA, November 2-4 1999, CERT Coordination Center, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, http://www.cert.org/reports/dsit_workshop.pdf - visited 12.11.2000.

[13] *Field Notice: 7xx Router Password Buffer Overflow* Revision 1: December 15 1997, http://www.cisco.com/warp/public/770/pwbuf-pub.shtml - visited 18.10.2000.

[14] *Microsoft Security Bulletin (MS00-029): Patch available for 'IP Fragment Reassembly' Vulnerability.* May 19, 2000, http://www.microsoft.com/technet/security/bulletin/ms00-029.asp - visited 18.10.2000.

[15] *Microsoft Security Bulletin (MS00-23): Patch available for 'Myriad Escaped Characters' Vulnerability.* April 12, 2000, http://www.microsoft.com/technet/security/bulletin/ms00-023.asp - visited 18.10.2000.

[16] K. Wooding. *Magnification Attacks - Smurf, Fraggle, and Others.* http://www.codetalker.com/whitepapers/dos-smurf.html - visited 19.10.2000.

[17] C.A. Huegen. *The Latest in Denial of Service Attacks: 'Smurfing'; Description and Information to Minimize Effects.* http://www.pentics.net/denial-of-service/white-papers/smurf.cgi - visited 19.10.2000.

[18] *CERT Advisory CA-98.01 'smurf' IP Denial-of-Service-Attacks.* January 5, 1998, http://www.cert.org/advisories/CA-1998-01.html - visited 23.10.2000.

[19] daemon9. *route infinity, TCP SYN Flooding Attacks.* Phrack magazine, Vol. 7, Issue 48, File 13 of 18, July 1996.

[20] C.L.Schuba et.al. *Analysis of a Denial of Service Attack on TCP.* Coast Laboratory, Department of Computer Science, Purdue University.

[21] *CERT Advisory CA-96.21, TCP SYN Flooding and IP Spoofing Attacks.* September 19, 1996, http://www.cert.org/advisories/CA-1996-21.html - visited 23.10.2000.

[22] *Web servers / possible DOS Attack / mime header flooding (archive).* http://www.securityfocus.com/archive/1/{10516—10520—10521—10525—10526} - visited 23.10.2000.

[23] *YA Apache DoS attack (archive).* http://www.securityfocus.com/archive/1/10228 - visited 23.10.2000.

[24] *Rootshell.com.* http://www.rootshell.com/ - visited 08.02.2001.

[25] D. Dittrich. *The DoS Project's "trinoo" distributed denial of service attack tool.* October 21, 1999, http://staff.washington.edu/dittrich/misc/trinoo.analysis.txt - visited 13.11.2000.

[26] *Project Loki.* Phrack Magazine, Volume Seven, Issue Forty-Nine, File 06 of 16, http://www.phrack.com/search.phtml?view&article=p49-6 - visited 23.10.2000.

[27] *L O K I 2 (the implementation).* Phrack Magazine Volume 7, Issue 51 September 01, 1997, article 06 of 17, http://www.phrack.com/search.phtml?view&article=p51-6 - visisted 23.10.2000.

[28] D. Dittrich. *The 'Tribe Flood Network' distributed denial of service attack tool.* October 21, 1999, http://staff.washington.edu/dittrich/misc/tfn.analysis.txt - visited 13.11.2000.

[29] J. Barlow, W. Thrower. *TFN2K - An Analysis.* AXENT Security Team, February 10, 2000 (Updated March 7, 2000) Revision: 1.3, http://packetstorm.securify.com/distributed/TFN2k_Analysis-1.3.txt - visited 13.11.2000.

[30] D. Dittrich. *The 'stacheldraht' distributed denial of service attack tool.* December 31, 1999, http://staff.washington.edu/dittrich/misc/tfn.analysis.txt - visited 13.11.2000.

[31] P. Ferguson, D. Senie. *RFC 2267, Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing.* Cisco Systems Inc., BlazeNet Inc., January 1998.

[32] D.J. Bernstein. *SYN Cookies.* ftp://koobera.math.uic.edu/syncookies.html - visited 13.11.2000.

[33] X. Geng, A.B. Whinston. *Defeating Distributed Denial of Service Attacks.* IEEE IT-Pro, July/Aug. 2000.

[34] *Submissions to the Paketstorm DDOS paper constest.* http://packetstorm.securify.com/papers/contest/ - visited 13.11.2000.

[35] *Linux Virtual Server.* http://www.linuxvirtualserver.org/ - visisted 13.11.2000.

[36] *Linux Advanced Routing HOWTO.* http://www.linuxdoc.org/ - visited 14.02.2001.

[37] Jef Poskanzer. *http_load.* http://www.acme.com/software/ - visited 10.02.2001.

[38] Arrowpoint. *Whitepaper: Web Site Security and Denial of Service Protection.* http://www.arrowpoint.com/solutions/white_papers/printer/Web_Site_Security.html - visited 12.11.2000.

[39] F5. *Whitepaper: A Defense To Denial of Service Attacks and Other Cyber Threats.* http://secure.f5.com/solutions/whitepapers/defense.html - visited 12.11.2000.