Global Information Assurance Certification Paper

DoS Attack "Smurfing" and Methods of Prevention

Ben Cameron
GSEC Certification
Version 1.3

Abstract

Some of the most interesting attacks done on networks today are those that are the hardest to track, and require very minimal effort on the part of the attacker. These types of attack can be done quickly, and with a very small pipeline to the internet, with a net result several dozen times the size of that pipeline lost for the attacked network. They're also very easy to prevent, if the proper measures are taken by both the victims and the networks where such attackers reside.

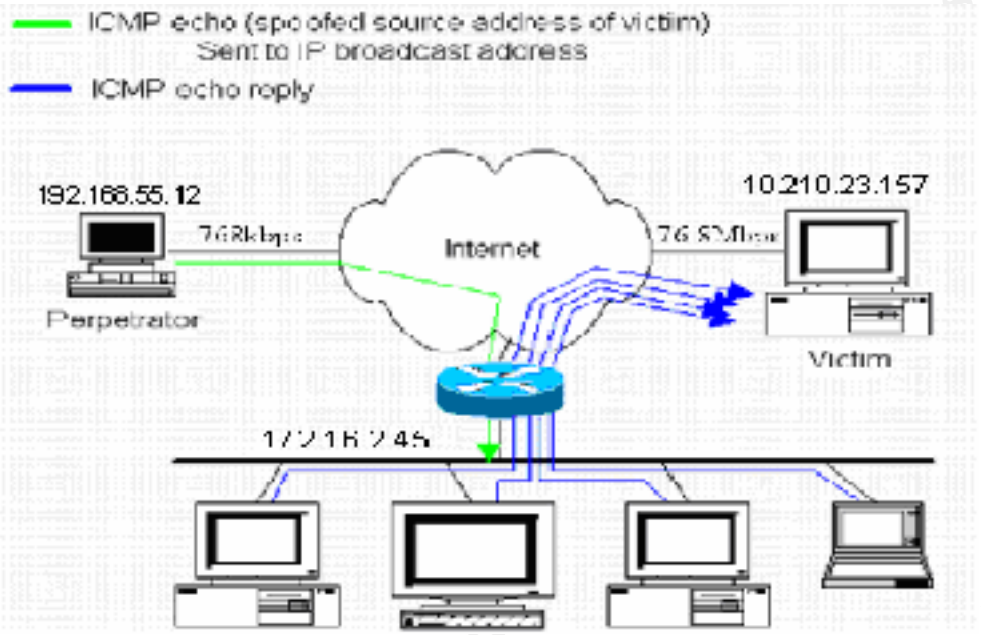
One of the latest in such attacks is called Smurfing, which comes from the program that is used to typically launch it. There are a variety of attacks that all operate in a similar manner, with different types of packets being used, and smurfing is the one that utilizes ICMP echo replies as its primary form of attack. In essence, the attacker uses an intermediary network to flood their victim with massive amounts of ICMP echo replies, through the use of a spoofed IP in the original ICMP echo request sent to this intermediary. This allows the attacker to both stay fairly anonymous, and to amplify the signal and vicousness of their attack up to and beyond 200 times, depending on the number of computers on a network.

Through reading this paper the reader should be able to get some idea of how the attack works, and the different methods of which to help prevent it. Also, even though prevention is the majority of securing a network, the reader will learn some interesting ways of tracking down an attacker who uses a spoofed IP in their attack.

Description of Attack

The newest such DoS attack is one called "Smurfing", which involves using ICMP echoes and echo replies combined with an intermediary network that allows IP broadcasting. It's an extremely damaging attack, as it's impossible to trace back by the victim, and impossible to stop on their end of the attack. The first thing that an attacker has to do is to figure out his intended target. Let's assume that for some reason or other he's already chosen it. Next he has to pick one or more "intermediary" networks, which are used to redirect and amplify the original stream of packets to the victim's network. The attacker essentially sends a constant stream of ICMP echo request packets to the broadcast address at the intermediary network with spoofed source IP fields, replacing the original IP with the IP of the victim. By doing this the intermediary gets hit with a lot of these echo requests, and if configured improperly will respond to each and every one of them. But it will respond to the victim instead of the attacker, and based on how many clients there are on the network being used to amplify the stream, it will flood the victim's network with thousands of packets per second. Let's say that the attacker sends a stream of 768kb/s (half of a T1 connection pipeline) to an intermediary network with a T3 connection that supports IP broadcasting. This intermediary acts like a sort of amplification device, taking the single ICMP echo stream, broadcasting it along layer 2 to any and all computers currently on the network (let's say 100), which in turn all reply with ICMP echo replies. Because the attacker spoofed the IP address in the original

ICMP echo packets sent to the intermediary router (replaced the original IP with the IP of the victim), all of these echo replies actually get routed to the victim. Even if the victim blocks all incoming echo replies without a concurrent outgoing echo request, the bandwidth of 768kb/s times 100 computers will equal 75.8Mbps, far more than a standard connection can handle.



[1]

Attacker: 192.168.55.12
Victim: 10.210.23.157
Intermediary: 172.16.2.45

Attacker sends spoofed ICMP echo packets with the reply IP 10.210.23.157 to the IP broadcast address at 172.16.2.45.

172.16.2.45 broadcasts to 100 computers, each replies to 10.210.23.157, continuously as the stream continues to come from the attacker.

768kb/s x 100 = 76.8Mbps

As you can see, it's a fairly straight-forward attack. The attacker uses someone else to do the actual attacking, and the victim is unable to see who it is that is initiating the attack. It's very hard to track down because of the spoofed IP, and if the attacker has comprimised a computer on the intermediary network then it is even harder to track down. Throughout this entire attack, the victim (10.210.23.157) only sees echo replies coming from the intermediary (172.16.2.45), so that the attacker is completely unknown

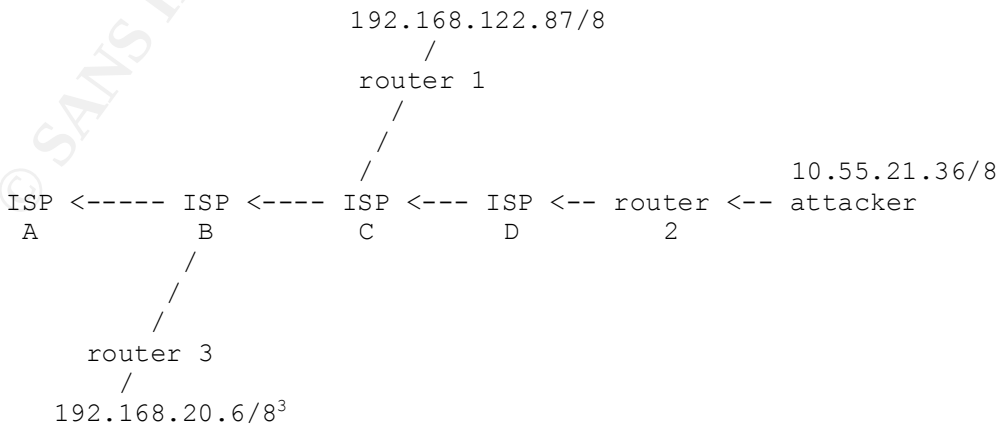
[1] http://www.pentics.net/denial-of-service/presentations/19971027_smurf_files/frame.htm

by the victim. Only the intermediary sees the packets coming directly from the attacker, and the IP source address is spoofed. The source ISP where the attacker is located will likely not notice anything, unless they are checking all of the packets leaving the network for some sort of a pattern.[2]

The only way for the victim to recover from this type of attack once it's begun is to either get the intermediary to stop sending the packets somehow, or to refuse all ICMP echo reply packets at their ISP's network level. Both of these take a lot of time to do, which is why it's so important to follow some basic preventative procedures. The three main players in this attack sequence are all vulnerable for different reasons, and some networks may think that because they're not vulnerable one way means that they won't be used in a smurf attack. The fact remains that each part of the sequence is vulnerable in a unique way, and to be truly protected a network needs to implement several changes.

Securing Originating ISP Network

The first part of this sequence is the attacker's ISP, or the connection that they're currently using to get onto the net. Most times this connection will be a cracked corporate account, or some sort of a hacked superuser connection on a UNIX box somewhere. The attacker will use this account to login to a machine on a network, or will dial into a network to gain access to their connection. After doing this the attacker can then spoof packets using a special program and send them out to the intermediary or "bounce" network. Most home ISPs don't scan for these types of packets, unless there's an unusually large number leaving the network. Even more numbers of corporate networks don't scan for them, leaving them open as a prime target for attackers to crack into and use. The easiest way for a network to protect themselves from being used as a source for smurfing is called Ingress Filtering. This type of filtering blocks outgoing packets that have a false source IP, or an IP that didn't originate on the network that it's trying to leave. Through ingress filtering, ISPs could block nearly 75% of all the spoofed IP attacks on the net, cutting down on the huge number of attacks such as smurfing or fraggle. Ingress filtering can work in the following way:



[2] http://www.pentics.net/denial-of-service/white-papers/smurf.cgi
[3] ftp://ftp.isi.edu/in-notes/rfc2267.txt

In this example, the attacker is behind router 2, and their ISP is D. Now, ISP D utilizes an input traffic filter in the input port of router 2, which only allows through data which originated from a source address in 10.55.21.35/8, effectively stopping any spoofed attacks from this network. It would use an IF/THEN system in the tables such as:

IF – Packet is from 10.55.21.35/8
THEN – Forward as normal

IF – Packet is from anywhere else
THEN – Deny all

This system would require a bit more processing power for the routers, as they would have to do a lot more checking for every single packet that tried to pass through, rather than just routing them. Also, the administrators for networks that use such techniques to prevent spoofing should log all of the dropped packets for further study, and determine if someone was trying to use the network for some sort of attack.

Securing the Intermediary (Bounce) Network

The next, and probably most important, place that needs to protect themselves against these types of attacks is the intermediary, or the bounce network. This is the tool that the attackers use during the course of the attack on the victim. Without these networks, the attacker isn't able to amplify the ICMP echo replies enough to overpower the bandwidth of the victim's network.

The first step in securing a network against becoming an intermediary is to disable IP broadcasting on the routers. This function isn't really that necessary, and certainly shouldn't be accessed by external sources. Also, ethernet NICs only listen to a couple of addresses during usage. The single MAC address that all devices listen to is called the "media broadcast" address, or FF:FF:FF:FF:FF:FF. If a NIC receives a packet sent to this address it will take that packet and process it immediately. It's important to stop routers from doing this, as it would allow an attacker to take advantage of a network. If you're using a Cisco router (or something similar) look for a command such as "no ip directed-broadcast". This will help to protect your network against broadcasts.

Under UNIX or Linux you can configure your machines to not respond to an ICMP echo sent to a broadcast response, which would filter out such bogus requests from the legitimate ICMP echo requests. This would be a good alternative to just shutting out all ICMP echo requests, which is another solution. Pretty much any computer with "ipfw" can configure themselves with rules such as:

```
ipfwadm -I -a deny -P icmp -D 192.168.0.255 -S 0/0 0 8
ipfwadm -I -a deny -P icmp -D 192.168.0.255 -S 0/0 0 8[4]
```

[4] Huegen

These will deny any incoming ICMP sent to the 0 and 255 addresses on your network (replace the 192.168.0 with your network address). This is probably the most effective method of preventing your network (or specific machines on it) from being used as a bouncing point in some sort of DoS attack.

There are a few different tools on the web that a company or individual can use to see if their network is up to date in terms of protection against this type of attack. One of these such sites is called www.netscan.org and offers scanning of individual networks. While scanning it checks to see if the network as a whole will allow IP broadcasting, and if a response to their echo is received.

IP	RESP	BRKN	DATE CHECKED
192.168.4.0	0	N	Sun Aug 19 06:24:32 2001
192.168.4.31	0	N	Sun Aug 19 06:24:32 2001
192.168.4.32	0	N	Sun Aug 19 06:24:32 2001
192.168.4.63	0	N	Sun Aug 19 06:24:32 2001
192.168.4.64	0	N	Sun Aug 19 06:24:32 2001
192.168.4.95	0	N	Sun Aug 19 06:24:32 2001
192.168.4.96	0	N	Sun Aug 19 06:24:32 2001
192.168.4.127	0	N	Sun Aug 19 06:24:32 2001
192.168.4.128	0	N	Sun Aug 19 06:24:32 2001
192.168.4.159	0	N	Sun Aug 19 06:24:32 2001
192.168.4.160	0	N	Sun Aug 19 06:24:32 2001
192.168.4.191	0	N	Sun Aug 19 06:24:32 2001
192.168.4.192	0	N	Sun Aug 19 06:24:32 2001
192.168.4.223	0	N	Sun Aug 19 06:24:32 2001
192.168.4.224	0	N	Sun Aug 19 06:24:32 2001
192.168.4.255	0	N	Sun Aug 19 06:24:32 2001[5]

This data shows that the current network (192.168.4.0) does not respond to the echo request sent to all of the comuters that have requested scans. As you can see, it gives you the date and time checked, and the different nodes on the network that were checked. This is a very useful tool for a quick check on your network to see just how vulnerable you are to being used as a intermediary. Another useful tool is located at www.powertech.no/smurf and actually analyzes as much of your network as it can.

Your probe of 192.168.4.0/24 yielded the following results:
Network: 192.168.4.0, Netmask: 255.255.255.0, Broadcast: 192.168.4.255, Responded: No (broadcast=0, network=0), Duplicates: 0

CONCLUSION: The network is not responding. Bad address or OK network.

id: 1025233
created: 1998-10-22 05:29:59 CET
updated: 2002-04-07 19:08:51 CET
network: 192.168.4.0/24
net-descr: not-analyzed

[5] http://www.netscan.org/netscan.cgi?net=192.168.4.144

last-probed: 2002-04-07 19:08:51 CET
responding: Yes
duplicates: 4 (highest seen, resets to 0 when 0 seen)
fixed: 2002-04-07 19:08:51

home-as: 1
as-descr: not-analyzed[6]

As you can see in this example, the network probed (192.168.4.0/24) showed as not responding on the IP broadcast address, which in this case is actually what you want. If it were to show some sort of response to the probe, then it probably means that your network allows directed broadcasting, and that you could be used for some sort of attack. This tool is very poweful for scanning entire networks for vulerabilities, and allowing administrators to know when they have successfully fixed the security holes in their system. It's also free and takes little time to use, so it's very easy for an administrator to use to check their network for any flaws.

Another problem that the intermediary could have is that a computer on their own network has been comprimised, and that is where the attack is actually being launched from. Although this does take a lot more work by the attacker on the front-end, it's much harder for the network administrators to prevent. Since everything on the network is behind the router already, you can't block any IP broadcast packets from coming into the network. The attacker is already inside, and simply has to flood the network with packets, and if it's a hubbed environment then the same effect will be reached as if the attacker had broken through from the outside. He still has to spoof the IP of his victim in order to get the computers on the network to respond appropriately, but as long as there isn't an IDS running somewhere on the network searching for such spoofed packets, they won't likely be noticed or stopped. One thing you could always do to prevent this is to program your router not to allow out any ICMP echo responses, which although would severely limit some of your network's services, would prevent you from being used as an intermediary in a smurf attack. You could also go to each machine on your network and configure each individual OS to not respond to an ICMP packet sent to a broadcast address, which would virtually eliminate all risk of being involved in an attack. Unfortunetly this isn't a very viable solution for networks with large amounts of computers, and servers that need to have very little downtime. There are a number of ways to do this in the different flavors of UNIX out there, most of which consist of a variant of "sysctl -w net.inet.ip.directed-broadcast=0" or something along those lines. Anything that tells the OS to either deny IP directed broadcasts, or sets it to off or 0 will do the trick. Even though this is the most basic level of the network that you can disable broadcasting on, and takes the most time, it's probably the best way to be sure that you've done the job right.

[6] http://www.powertech.no/smurf/probe.cgi?network=192.168.4.0&probe=PROBE

Securing the Victim's Network

The last part of the process that needs to be fixed is the the victim's network. This is probably the most difficult of the three to secure, as there's almost no way to prevent lots of packets from flooding the connection. Without the connection being secured properly, the network could be easily comprimised through any standard DoS attack, especially a smurf attack. With a 200-host LAN as the intermediary, there could easily be about 80Mbps of traffic hitting the victims target, that's upwards of 35Kpps (packets per second). This can all be done with only 400Kbps of original bandwidth, that's doable with a DSL or cable modem connection. Most corporate networks are connected at a rate of about 10Mbps with an Ethernet connection, which means that the huge flood of packets would quickly overcome the connection and effectively disable any web services that the company or individual was running. The best way to combat this is to be able to drop packets at the network border (where the network connects to the outside world), or at some point upstream in the ISP's main network. If the company is able to do this, most of its bandwidth would be freed up from the attack, and essential services such as web and FTP servers would continue to run until such time as the attack could be stopped and full bandwidth returned.

There are some brands of routers that allow packets to take several different paths through the router, which gives administrators a couple of options as to how to deal with such attacks. Cisco routers in specific has one called "process" switching, which is essentially used when a complex task is required during packet processing. This type of switching is probably the slowest available, but also allows for the most accuracy in routing. Another way for the routers to deal with packets is at the interrupt level, which doesn't require any processing time when compared to process switching. Up until the most recent versions of IOS, all access-list denies were computed during the process level, which obviously makes them much slower than normal routing. If the access-list does deny something such as an ICMP echo request, then the IOS requires that an unreachable response be sent back to the originating computer system, so as to inform them that the echo was blocked. But Cisco recently put out a code change in version 11.1CA and later of their IOS that drops such packets at the interrupt level. This is a very important change for Cisco to make to their routers, because it dramatically increases the number of packets that could be dropped during a smurfing attack. Only 2 packets per second will not be dropped, because these will be used to create the ICMP uncreachable reply to the originating host.

Probably the easiest way is for all of the ICMP echo reply packets to be filtered at the ingress interfaces on your routers, essentially dropping all of them before they can enter your network. The problem with this is that it can still fill up your bandwidth and disable or cripple many of your essential web services.

Another way of protecting bandwidth on a network is called Committed Access Rate (CAR). CAR works by limiting the amount of bandwidth allowed to be used by a certain type of packet, such as ICMP echo requests. If you want some packets to be let through, for legitimate purposes, but not all packets, then this would be an appropriate way to limit

this. Since many IRC servers have become the targets of such attacks, they have pretty much all implemented this type of technology. You can secure a single machine to not allow ICMP echo reply packets, but other machines on the same network may be attacked as well. To fix this, CAR could be instituted on the network, limiting all ICMP echo reply packets to 128Kbps bandwidth, and secure it that way.

```
! traffic we want to limit
access-list 102 permit icmp any any echo
access-list 102 permit icmp any any echo-reply
! interface configurations for borders
interface Serial3/0/0
 rate-limit input access-group 102 256000 8000 8000 conform-action transmit exceed-
action drop
```
[7]

As you can see in the above section of code, ICMP echo and echo reply bandwidth is limited to 256Kbps. If needed you could customize this code with multiple "rate-limit" commands for any other types of traffic, such as UDP pings or TCP SYN packets. So far CAR is only available on certain Cisco routers though, which is unfortunate for networks that don't use these model routers.

Securing the three different networks involved in a smurfing attack is probably the most important defense that can be taken against such attacks. Making sure that your routers are up to date, that IP broadcasting is disabled wherever possible, and setting up an ingress filtering system to prevent people on your network from even sending out spoofed IP packets. If every network out there put into practice all of these recommendations, then the amount of DoS attacks on the net would be reduced by at least half, probably more.

Once an attack has been committed and noticed by the network administration, it is important to try and identify the source of the attack and who has been committing it. Preventing an attack or being able to catch one and stop it while it's happening is very important, and is probably all that most admin can hope for. But being able to catch the person who's committing the attack is also important, and could stop further instances of this attack. Chances are if the person attacking your network does it once, they have some reason for it and will try to attack you again.

Tracking attacks can be difficult, especially in the case of smurfing because the source IP is spoofed (that's how the attack works). With cooperation from the different ISPs involved, and the intermediary, it is possible to track down an attacker such as this. Packet logging at the router level is certainly possible, and with most Cisco routers you can set them up to catch packets that are dropped for various reasons, such as because there is a stream of ICMP echo reply packets, and you want 2 every second to be logged for future study. The problem is that any logging that happens has to be done at the process level, which is obviously much slower. Because it's slower, a higher level attack at the rate of 35Kpps or so would overwhelm a router trying to sort and log the packets,

[7] http://www.pentics.net/denial-of-service/white-papers/smurf.cgi

whereas an attack of about 10Kpps would not overwhelm a router as quickly. Also, if an attack continues for very long then obviously whatever server is being used to generate the log files could be overloaded and crashed, which could be the goal of the attacker all along.

Probably the best method for logging on a router is ACL routing, which works better with Cisco routers. On all newer versions of Cisco routers a keyword was introduced that works with ACL logging to log all ICMP ping packets with input interface and a MAC address. You can enter it into the router like so:

access-list 101 permit icmp any any echo log-input

The problem with this is that you can only log a MAC address from a multi-access network, such as FDDI. Point-to-point connections won't be able to be logged with a MAC either, as it's not included with the information. If you were able to capture a packet off of a connection such as FDDI or Ethernet, here's an example of what it'd look like:

Sep 10 23:17:01 PDT: %SEC-6-IPACCESSLOGDP: list 101 permitted icmp 10.0.7.30 (FastEthernet1/0 0060.3e2f.6e41) -> 10.30.248.3 (8/0), 5 packets[8]

As you can see, in the section "(FastEthernet1/0 0060.3e2f.6e41)" the router was able to log much more info than in the following example, which is with a point-to-point network:

Sep 10 23:29:00 PDT: %SEC-6-IPACCESSLOGDP: list 101 permitted icmp 10.0.7.30 (BRI0 *PPP*) -> 10.0.19.242 (8/0), 1 packet[9]

You can certainly see the differences in the two log entries, with the second one only showing "*PPP*" in the place where there was some important data such as the MAC address in the first one. This could make a huge difference in trying to track down an attacker, as the MAC address could be circulated to suspected ISPs and eventually be tracked down successfully.

Using the first log entry (with the MAC address information) you can see that the packet came from a FastEthernet1/0 connection, and from MAC address 0060.3e2f.6e41.

Sep 10 23:17:01 PDT: %SEC-6-IPACCESSLOGDP: list 101 permitted icmp 10.0.7.30 (FastEthernet1/0 0060.3e2f.6e41) -> 10.30.248.3 (8/0), 5 packets[10]

You can also see that it was destined for the IP address 10.30.248.3. You would need to then use the command "show ip arp" in order to determine the IP address for the MAC

[8] Huegen
[9] Huegen
[10] Huegen

address of the originating host, and use this information to determine if it was in fact the original source computer (attacker) or was just an intermediary used to attack you. If the latter is the case, then you can use this information to enlist their help in tracking down the original attacker, in a sort of hop-by-hop fashion of tracking. Using "show ip arp" can be a very useful method in tracking down an IP address, as seen in this example:

```
netlab#show ip arp 0060.3e2f.6e41
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  10.0.183.65             32   0060.3e2f.6e41  ARPA   FastEthernet1/0
```
[11]

You can see in this response from the "show ip arp" command that the IP address for the MAC that was logged is 10.0.183.65, which gives you one hop to go backwards to in the tracking process. And although it is fairly simple for the victims to track backwards to the intermediary, because the IP address of the echo reply is a true address and not a spoofed one, it is harder for the intermediary to trace backwards to the true source of the attack, since the packets all contain spoofed IPs.

Conclusion

Smurfing can be a very dangerous and damaging attack for a company or individual. Having your network's bandwidth used up for a long period of time not only slows it down and prevents you from accessing the web, but if you're a company that lives off of web sales or web access, then this could be financially damaging to the company as well. Most times a company is attacked because they did something that annoyed the attacker, such as banning them from a gaming site or from an IRC channel (if the company runs some IRC servers). This is most often the reasoning behind such an attack.

Most companies these days have disabled direct broadcasting, since it's not a widely utilized tool and is prone to all sorts of attacks. But not many at all have decided to institute ingress filtering yet, or have even checked their outgoing packet traffic for ICMP type packet patterns. Even less have their routers setup with proper ACLs and log packets properly as they come in, dropping all but a few every second if a stream of certain types of packets comes in through the router. So although a network may indeed be secure from having someone break into it and use the router to broadcast packets across the network, most aren't secure as a base of attack or a victim. It's important that as a network administrator, you look at an issue from all different angles, especially if you have a fairly large network which could be both a target and an accessory to an attack. Just being safe from one thing doesn't make you safe from another, so you need to work through the attack and how it works, and at each step check to make sure that your network is secure. There's three important questions that you should ask yourself when securing your network:

[11] Huegen

1. Are you allowing broadcasting? If so is it necessary for your business, and perhaps should you disable it?
2. Do you filter outgoing packets and drop all traffic with a spoofed source IP?
3. Have you and your ISP worked out some method of dropping large amounts of packets from your pipeline if you were to come under attack? If so, how effective is it, and does it log packets?

After going through your network according to the above questions, and securing any problems that you find, you can be sure that if an attack occurs, or someone tries to use you for an attack, not only will it be near to impossible for them to do so, but you will also be alert for such an event and you will be logging it. This gives you the ability to fight back against the attacker and track them back to their source. This is probably the most important part of all, since unless you stop the attacker at the source it's likely he'll just attack again and again.

Referenced Documents

"Defining Strategies to Protect Against TCP SYN Denial of Service Attacks". Cisco Systems Inc. 2001. URL: http://cio.cisco.com/warp/public/707/4.html.

"Defining Strategies to Protect Against UDP Diagnostic Port Denial of Service Attacks." Cisco Systems Inc. 2001. URL: http://cio.cisco.com/warp/public/707/3.html.

"Flood Attacks." Asta Networks, Inc. 2002. URL: http://www.astanetworks.com/resources/types/flood_attacks.html.

Homelien, Oystein. "Smurf Amplifier Registry." 9 Apr. 2002. URL: http://www.powertech.no/smurf/.

Huegen, Craig A. "The Latest In Denial of Service Attacks: "Smurfing" Description and Information to Minimize Effects." 8 Feb. 2000. URL: http://www.pentics.net/denial-of-service/white-papers/smurf.cgi.

"IP Addressing Commands. Cisco Systems Inc. 2001." URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/cs/csprtn1/csipadr.htm#xtocid748113.

"NetQuake Protocol problem resulting in smurf like effect." Q. 1998. URL: http://www.attrition.org/security/denial/w/nquake-s.dos.html.

"Netscan.org." netscan.org. 2001. URL: http://www.netscan.org/.

"Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing." P. Ferguson. Jan. 1998. URL: ftp://ftp.isi.edu/in-notes/rfc2267.txt.

"The Latest In Denial Of Service Attacks: 'Smurfing'." Craig A. Huegen. 2000. URL: http://www.pentics.net/denial-of-service/presentations/19971027_smurf_files/frame.htm