Global Information Assurance Certification Paper

GIAC Firewall and Perimeter Protection
Practical Assignment for SNAP San Jose, May 8 – 13, 2000

Pamela Patterson
June 15, 2000

1. Egress Filtering

Egress filtering is the process of preventing certain types of network traffic from exiting your network. There is usually concern about preventing incoming traffic that is considered to be dangerous, but potentially dangerous outgoing traffic is usually viewed as "someone else's problem". The assumption has been that other networks will block any dangerous traffic attempting to enter their network or, if they don't, then any problems they have are their own fault. This attitude has been changed (at least in the security community) by the proliferation of attacks that use IP spoofing.

IP spoofing is when a computer sends out network traffic with a falsified source IP address. There is currently no way the destination computer can tell that the source address has been spoofed. The classic Smurf attack sends a very large number of ICMP Echo Requests (pings) to many different machines with the same spoofed source address. The actual machine at the spoofed address is swamped with millions of ICMP Echo Replies, and can be brought to a standstill.

The way to prevent this type of attack (and to minimize many other kinds of attacks) is to stop the spoofed packets. This cannot be done at the destination; it must be done at or near the source. Egress filtering is the process of preventing networks packets with illegitimate source addresses from leaving a network. For example, if a network has the address range 10.10.10.1 to 10.10.10.254, then a packet with a source address of 47.2.3.4 that is trying to get out of the network should be stopped at the boundary. This packet may be an honest mistake (a misconfigured machine or a faulty input filter) or it may be an attempt to spoof the IP address. Either way, the packet should not be allowed out of the subnet.

To configure egress filtering for Linux ipchains for the above subnet, we can use the following rule:

```
# ipchains –A output –s ! 10.10.10.0 /255.255.255.0 DENY –l
```

The parts of the filter are:

-A	add the rule
output	put the rule on the output chain
-s	filter on source address
! 10.10.10.0/255.255.255.0	addresses NOT in the subnet 10.10.10.0/24
DENY	drop the packet silently
-l	log the attempt

This filter should be applied at any Internet borders. There are two good ways to test the filter: misconfigure a machine (i.e. with an illegal network address) and see if you can connect out of your subnet, or procure a hacking tool which sends spoofed packets and try it out (spoof an address on another subnet of yours, not someone else's!) . Both of these methods can cause problems, disrupt the network and alarm the security people, so it is highly advisable to get written permission to do this from the owner of the subnet, and to do it outside of business hours.

The arguments in favour of doing this are:

a) it is quite simple, requiring little time and no extra investment in most environments
b) it is an good way of discovering misconfigurations in your subnet
c) while no one has yet been sued by the victim of an attack for allowing spoofed traffic from compromised machines on their network, this will happen at some point. It is possible that unprotected networks will legally be viewed as attractive nuisances, like unfenced swimming pools. Egress filtering shows due diligence in attempting to prevent your resources from being used for illegal purposes
d) it is the neighborly thing to do; if more people did it, the Internet would be a safer place.

2. Firewall Violations

These detects are all on a Linux machine running ipchains. The detects all start with "input DENY eth0" since they are all detects on incoming packets on my cable modem.

a) May 25 16:20:20 CR609279-A kernel: Packet log:
input DENY eth0 PROTO=6 195.114.246.230:3718 24.112.186.14:21
L=48 S=0x00 I=43367 F=0x4000 T=106

Protocol	PROTO=6	TCP
Source Address	195.114.246.230	apollo-sb0229.multiweb.net, which belongs to the Dutch ISP Sonera
Source Port	3718	Ephemeral port
Destination Address	24.112.186.14	cr609279-a.rchrd1.on.wave.home.com (me)
Destination Port	21	FTP command port
Packet size	L=48	48 bytes
Type of Service	S=0x00	
IP ID number	I=43367	
Fragment info	F=0x4000	The "Don't Fragment" bit is set
Time to Live	T=106	

Firewall rule violated:
ipchains –A input –p tcp –d 24.112.186.14/255.255.254.0 –j DENY –l (the default DENY rule)

This is probably someone looking for FTP sites that can be used without the knowledge of the site owner.

b) May 25 21:32:38 CR186996-A kernel: Packet log:
input DENY eth0 PROTO=17 24.112.232.188:2301 255.255.255.255:2301
L=40 S=0x00 I=16840 F=0x0000 T=128

Protocol	PROTO=17	UDP
Source Address	24.112.232.188	cr116192-a.flfrd.on.wave.home.com
Source Port	2301	Ephemeral port
Destination Address	255.255.255.255	The entire Internet
Destination Port	2301	Detects were logged on ports from 1024 to 5000, in sequence.
Packet size	L=40	
Type of Service	S=0x00	
IP ID number	I=16840	
Fragment info	F=0x0000	
Time to Live	T=242	

Firewall rule violated: Don't allow incoming High Port to High Port UDP packets.
ipchains –A input –p udp –s 0/0:1024:65535 –d 0/0 1024:65535 –J DENY -l

This is a UDP port scan; it was with many similar packets with different destination ports. Detecting which ports are open is a necessary step if someone is planning an intrusion. I eventually turned off logging for broadcast UDP scans, because there was just too much of it.

c) May 29 04:50:24 CR609279-A kernel: Packet log:
input DENY eth0 PROTO=6 216.115.147.134:4963 24.112.186.14:53
L=60 S=0x00 I=22642 F=0x4000 T=47

Protocol	PROTO=6	TCP
Source Address	216.115.147.134	cr609279-a.rchrd1.on.wave.home.com (proxied)
Source Port	4963	Ephemeral port
Destination Address	24.112.186.14	cr609279-a.rchrd1.on.wave.home.com (me)
Destination Port	53	Nameserver port
Packet size	L=60	60 bytes
Type of Service	S=0x00	
IP ID number	I=22642	
Fragment info	F=0x4000	The "Don't Fragment" bit is set
Time to Live	T=47	

Firewall rule violated: Don't allow DNS zone transfers
ipchains –A input –p tcp –s 0/0 1024:65535 –d 24.112.186.14/255.255.254.0 53 –j DENY -l

This is me trying to do a DNS zone transfer.

d) May 30 19:32:14 CR609279-A kernel: Packet log:
input DENY eth0 PROTO=17 194.133.179.128:31338 24.112.186.14:31337
L=47 S=0x00 I=24531 F=0x0000 T=108

Protocol	PROTO=17	UDP
Source Address	194.133.179.128	Don't know who this is.
Source Port	31338	Ephemeral port
Destination Address	24.112.186.14	cr609279-a.rchrd1.on.wave.home.com (me)
Destination Port	31337	BackOrifice listening port
Packet size	L=47	47 bytes
Type of Service	S=0x00	
IP ID number	I=24531	
Fragment info	F=0x0000	
Time to Live	T=108	

Firewall rule violated: Don't allow incoming High Port to High Port UDP packets.
ipchains –A input –p udp –s 0/0:1024:65535 –d 24.112.186.14/255.255.254.0 1024:65535 –J DENY -l

This is a scan for the BackOrifice Windows trojan. This is no problem with Linux, but this box is dual-boot with Windows NT.

The block containing the IP address is registered to "EU-GLOBALONE-OTHER-970109" for "Provider Local Registry". Global One is a European ISP. The IP address itself does not resolve to a name with DNS.

e) Jun 11 18:06:04 CR609279-A kernel: Packet log:
input DENY eth0 PROTO=6 216.254.150.10:3265 24.112.186.14:27374
L=48 S=0x00 I=25930 F=0x4000 T=112

Protocol	PROTO=6	TCP
Source Address	216.254.150.10	dialin-150-10.tor.primus.ca
Source Port	3265	Ephemeral port
Destination Address	24.112.186.14	cr609279-a.rchrd1.on.wave.home.com (me)
Destination Port	27374	SubSeven listening port
Packet size	L=48	
Type of Service	S=0x00	
IP ID number	I=25930	
Fragment info	F=0x4000	The "Don't Fragment" bit is set
Time to Live	T=112	

Firewall rule violated:
ipchains –A input –p tcp –s 0/0:1024:65535 –d 24.112.186.14/255.255.254.0 1024:65535 –J DENY -l

This is a scan for the SubSeven Windows trojan from a dialup account at a Toronto ISP.

3. Defense in Depth

b) Resistance to DDoS attacks with Dual Connections to the Internet

Some resistance to DDoS attacks can be implemented with dual connections to the Internet, by setting up a failover scenario from one ISP to the other in case of attack. While the DDoS attack can come from anywhere, it is most effective if it originates from machines within the address space of the victim's ISP, since little (if any) filtering is done within this address space. Recent successful DDoS attacks have brought down the ISP's equipment more than the victim's equipment.

First, each connection must be with a different ISP (and it would be perfect if the ISPs each did egress filtering). Second, both ISPs must use the Border Gateway Protocol (BGP) for maintaining their routing. Third, one of the ISPs must allow you a foreign address space (i.e. not in their assigned block of addresses). Third, the ISPs should be as far apart as possible, network topology-wise (this may give a bit of a performance hit to access via the second ISP, but that is much better than no access at all).

Neither ISP should be in the business of providing consumer-level Internet services (DSL or cable) which are becoming a gold mine of easily hackable Win* boxes that are just DDoS attack clients waiting to be triggered.

The idea is this: set up a high priority route to your site with a large ISP and a lower priority route with another ISP. Set up two large capacity border routers (as large as you can afford), one on each ISP connection, and another router downstream in your DMZ, all as BGP peers, so that if one border router is unable to connect, traffic will be sent through the other router. Outgoing traffic will immediately start using the secondary route. Since the ISPs are running BGP, the failure of your main connection will cause incoming traffic to be routed to your secondary connection, probably within a few (5 –10) minutes. If the two ISPs are "far" enough apart, much of the DDoS traffic will not make it through, as it will be choked or stopped by filtering, or it will bring down equipment on the route, forcing the ISPs to take some action.

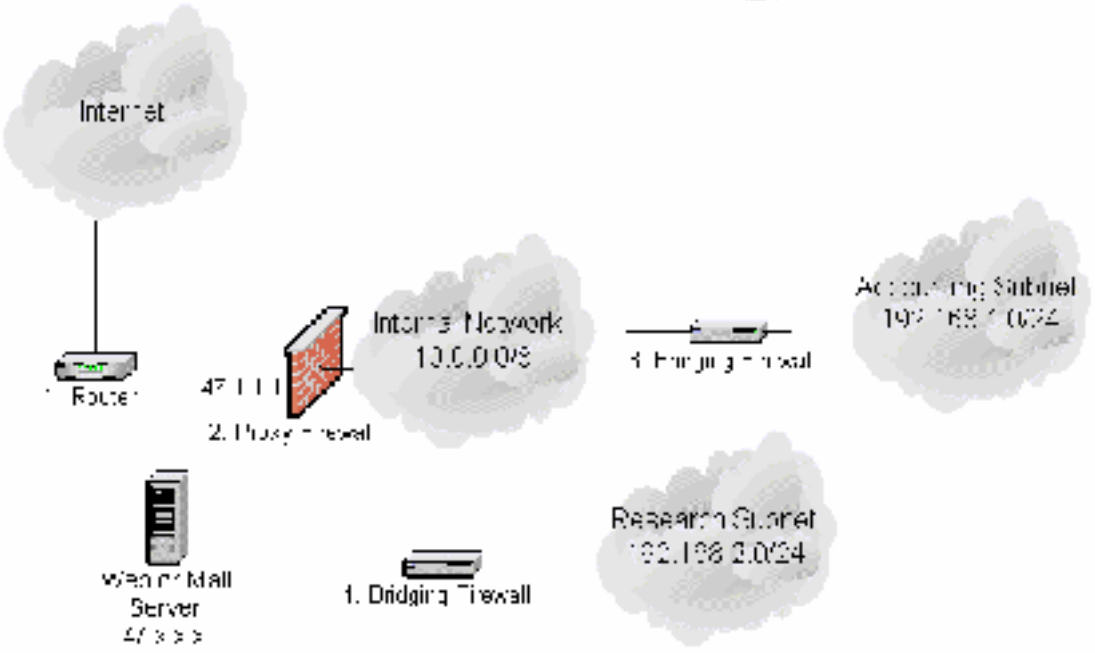
It may seem unfair to try and shove the DDoS traffic problems out to the ISPs, but it is their responsibility in fact, more than the responsibility of the victim.

b) Defense of Critical Subnets with Minimal Equipment

The most effective use of the existing equipment is to:

I. Use the router at the Internet boundary, with input and egress filtering protecting the DMZ
II. Put any web or mail servers in the DMZ behind the router
III. Put the proxy firewall between the DMZ and the internal network, with a real Internet address on the public side (47.1.1.1 in this example)
IV. Use a likely-looking RFC 1918 address range for the main internal network and then proxy (with NAT) all connections to the Internet
V. Use small RFC 1918 subnets (192.168.1.0 and 192.168.2.0 in this example) for the two secure subnets and use the bridging appliance-type firewalls to isolate them from the main internal network.

Access to the secure subnets will be blocked by the appliance-type firewalls. The default route within each secure subnet should point to its own bridging firewall. The proxy firewall must be set to proxy the subnet address ranges as well as the main internal network address if the secure subnets are to have Internet connectivity.



4. Sample Question

Design a way for a server pool (belonging to another company who provides web content to cell phone customers) to access an application server belonging to your company that provides various internal content (intranet web pages, LDAP directory information, Exchange and Unix e-mail) as HTML. The other company's servers and the application server can both do server-to-server https. The other company has controls in place to ensure that cell phone users can only retrieve data that belongs to them; your job is just to deliver the document when asked.

Answer

The first guess of putting the application server in the DMZ is a bad idea, since the application server would need to get web, LDAP results and mail from many internal machines through the firewall. Also, since the DMZ is more easily compromised than the internal network (one hopes), private e-mails would be susceptible to being sniffed, unencrypted, on the DMZ net. The second guess of putting the application server behind the firewall is better, but still requires opening multiple holes (in both the firewall and the border router) that "line up" and provide direct access to the application server from the Internet by, say, a port scanner.

The solution is to put a secure web proxy in the DMZ. The border router only allows incoming https from the external server pool to the web proxy. The web proxy does not have any confidential data on it; it just forwards the request. The firewall only allows incoming https from the web proxy to the application server. The application server can freely gather all its data safely within the corporate network. The web proxy can either pass the SSL through for the application server to validate (SSL tunneling), or negotiate separate SSL sessions with the outside server and internal application server (true SSL proxying). SSL tunneling will have better performance but SSL proxying will have better security.

This is pretty secure as long as:

a) the servers validate the SSL certificate(s) that they are presented with (this prevents IP spoofing attacks),
b) the web proxy is seriously hardened and is only running the web server (no mail or other services), so that no one can change the security settings,
c) some kind of intrusion detection is running in the DMZ and someone is paying attention to it,
d) the application server is hardened as well and is does not have IP forwarding enabled.

To set up this web proxy with Solaris and Apache mod-SSL see the following links:

http://www.sunworld.com/sunworldonline/common/security-faq.html (Solaris Security FAQ)
http://www.apache.org (Apache project)
http://www.modssl.org/ (SSL module for Apache)

If you don't want to build your own secure web server, use Stronghold http://www.c2.net/home.php3 (a commercial Apache derivative)

To tunnel SSL with iPlanet, see
http://developer.netscape.com:80/docs/manuals/proxy/adminux/encrypt.htm#1015838