Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Egress Filtering: Why and How

Why Use Egress Filtering

So much attention is given to protecting hosts and network perimeters from external attack, that many times ingress filtering is the only direction even considered when building a defense. But a well-rounded defense looks at traffic going both ways. For starters, there are some dangers that ingress filtering alone cannot address:

- No matter how tough the firewall, there is always the risk of unforeseen vulnerabilities, failures, etc. that could allow a host to be compromised. From a compromised host, attacks could be waged against other sites.

- A user already inside the network (an employee who wants to do a little hacking of his own, a consultant who gains physical access, etc.) might use network resources to attack other sites. To the degree that it is possible, the firewall policy should include measures to control and detect such infractions.

In addition to these risks, using egress filtering can make the job of stopping network information probes easier. It is difficult to predetermine all the ways an attacker may try to gather information about the network, so it is wise (and easy) to also include filtering rules to stop the answers from being sent.

In the case of attacks from inside (by either physical access or a compromised system), if the attacker does nothing to hide his source address, eventually you will probably hear about it and be able to track it down. But if they spoof the source address, the only way to reliably handle the situation is to filter spoofed packets leaving your network. Fortunately, this is quite simple to do.

How to Use Egress Filtering

To cover the above-mentioned issues, the two main types of traffic that an egress filter should look for are: packets whose source address is not a legal address in the internal network, and responses to requests that are not necessary for outsiders according to your security policy (for example, “destination unreachable” messages that can be used to map a network by silence).

For my example syntax, I will use ipchains and a simple network similar to the one I am tasked with securing, which is a class C address space with no subnetting – we’ll call it 10.10.10.0. The external router is connected to eth0, and eth1 goes to the internal network. Imagine these commands as being in an appropriate order in complete chains that include ingress filtering as well, and have defaults as follows:

```
ipchains –P input DENY
ipchains –P output ACCEPT
```

Be sure that any catch-alls “apply”s at the end of the input chain don’t include packets coming into eth1. For catching spoofed source addresses, the following syntax would be used, which accepts packets coming into the internal network interface card that have an legitimate (internal network) source IP:

```
ipchains –A input –i eth1 –s 10.10.10.0/24 –j ACCEPT
```

The default deny in the input chain will catch packets with any other source address.

As an example of an outgoing response to stop, here is the syntax to deny ICMP “destination unreachable” messages. It is put in the output chain for the interface card to the router so that answers from the firewall itself are also checked:

```
ipchains –A output –i eth0 –p icmp –icmp-type destination-unreachable
```

To test the first rule, the most direct method is to use a tool that can generate packets with spoofed source addresses, if available. An alternative would be to put a machine on the internal network just for testing that has an IP address that is not part of the normal address space, which may require taking the

© SANS Institute 2000 - 2005 Author retains full rights.

firewall off-line and having an isolated test network. Monitor the packet traffic on both sides of the firewall and verify that the offending packets were stopped. Be sure also to test valid packets, also, to make sure they get through.

To test the second rule, use a host outside the firewall, either on the internet or on the DMZ if you have one. If you have ingress rules covering all known offending inquiries, comment out one of those rules from your script, reconstruct the chain, and test using that request. Don't forget to re-insert the rule afterward and reconstruct the chain again.

Firewall Policy Violations

Preface

The small company I work for has no firewall yet (setting one up is my job), and since I work only two days a week and am busy with missionary work the rest of the time, there was no time to set up a test network to attack (although I would have liked to do that, as I would have learned a lot). So I looked through some of the recent detects on the GIAC web site and selected a few that I thought represented a variety of types of attacks. I will show the detect log entries as they appeared on the web site, explain the type of attack and how to read the logs, and give a typical example rule that would have caught the violation. Although the logs are from different firewall devices and software (routers, ipchains, and commercial firewall software), for simplicity I will give all the rule examples in ipchains syntax.

Violation #1: NetBios Scan

Date Time Rule that detected it Protocol Source IP Source port Dest. IP Dest. port

```
Jun 6 13:22:36 fwall 15 deny: UDP from 209.179.192.145.137 to mysubnet.1.137
Jun 6 13:22:38 fwall 15 deny: UDP from 10.1.1.2.137 to mysubnet.1.137
Jun 6 13:22:38 fwall 15 deny: UDP from 209.179.192.145.137 to mysubnet.1.137
Jun 6 13:22:38 fwall 15 deny: UDP from 209.179.192.145.137 to mysubnet.1.137
Jun 6 13:22:39 fwall 15 deny: UDP from 10.1.1.2.137 to mysubnet.1.137
Jun 6 13:22:55 fwall 15 deny: UDP from 209.179.192.145.137 to mysubnet.2.137
Jun 6 13:22:57 fwall 15 deny: UDP from 10.1.1.2.137 to mysubnet.2.137
Jun 6 13:22:57 fwall 15 deny: UDP from 209.179.192.145.137 to mysubnet.2.137
Jun 6 13:22:57 fwall 15 deny: UDP from 209.179.192.145.137 to mysubnet.2.137
Jun 6 13:23:03 fwall 15 deny: UDP from 209.179.192.145.137 to mysubnet.3.137
Jun 6 13:23:05 fwall 15 deny: UDP from 209.179.192.145.137 to mysubnet.3.137
```

These logs represent a typical UDP scan that is looking for Windows NT/9x hosts using the NetBios UDP port 137. Most of the scans come from host 209.179.192.145 (or a mystery host using that as a spoofed source address), also using port 137 as the source, but alternately a second source address (10.1.1.2, a typical *internal* address) was also used to probe the same machines, perhaps as a different tactic to try to fool the firewall into thinking it was normal NetBios traffic. The destination addresses were probed incrementally starting with mysubnet.1, and each host was probed several times each. Presumably this pattern continued until the hacker gave up, realizing that either a firewall was quietly catching his attacks (the truth) or there weren't any Windows machines there.

An ipchains rule that would catch this scan is:

```
ipchains –A input –i EXT_NIC –dport 137:139 –j DENY -l
```

If this scan had not been caught, the attacker may have been able to determine the IP addresses of Windows machines on the network, and then searched for Windows vulnerabilities to exploit.

© SANS Institute 2000 - 2005 Author retains full rights.

Violation #2: Port Scan

Date Time Action Direction Protocol Source IP Source port Dest. IP Dest. port

```
Jun 05 2000 21:30:30: Deny inbound UDP from 216.53.10.1/6801 to x.x.x.31/1051
Jun 05 2000 21:32:30: Deny inbound UDP from 169.132.184.211/6801 to
      x.x.x.31/1052
Jun 05 2000 21:34:30: Deny inbound UDP from 216.53.10.1/6801 to x.x.x.31/1053
Jun 05 2000 21:36:31: Deny inbound UDP from 169.132.184.211/6801 to
      x.x.x.31/1054
Jun 05 2000 21:38:31: Deny inbound UDP from 216.53.10.1/6801 to x.x.x.31/1055
Jun 05 2000 21:40:31: Deny inbound UDP from 169.132.184.211/6801 to
      x.x.x.31/1056
Jun 05 2000 21:42:31: Deny inbound UDP from 216.53.10.1/6801 to x.x.x.31/1057
Jun 05 2000 21:44:31: Deny inbound UDP from 169.132.184.211/6801 to
      x.x.x.31/1058
Jun 05 2000 21:46:31: Deny inbound UDP from 216.53.10.1/6801 to x.x.x.31/1059
Jun 06 05:55:14: Deny inbound UDP from 216.53.10.1/6801 to x.x.x.31/1303
Jun 06 05:59:06: Deny inbound UDP from 169.132.184.211/6801 to x.x.x.31/1304
Jun 06 06:01:08: Deny inbound UDP from 216.53.10.1/6801 to x.x.x.31/1305
```

This seems to be a semi-random UDP port scan of a particular host (x.x.x.31) masquerading as coming from two different sources, presumably to fool firewall software looking for repeating patterns. The packets are clearly manufactured, because the source port, although an ephemeral port, is always the same. The rules that would catch this scan are a set that would block all UDP to any ports that are not needed on that machine. For example, if the machine in question was a DNS server, port 53 would be needed for requests, so in that case, the rule would look like:

```
ipchains –A input –p udp –d x.x.x.31/0 !53 –j DENY -l
```

Similar rules could be written for TCP ports, other hosts, etc. The logs would clearly show that a scan was attempted, although in this case, since it is obvious that at least one of the source addresses was spoofed, probably both of them were, so catching the perpetrator would be difficult.

If this scan had not been caught, any ports that might be opened by the target host to access other hosts as a client (i.e. ephemeral ports) might be attacked.

Violation #3: Looking for Back Doors

Date Time Rule that detected it Source location
Source IP Source port Dest. location Dest. IP Dest. port

```
Jun 6 18:40:55 : Deny inbound udp src outside:
                    63.22.106.113/1037 dst lb-dmz:x.x.x.13/31337
Jun 6 18:40:55 : Deny inbound udp src outside:
                    63.22.106.113/1037 dst lb-dmz:x.x.x.86/31337
Jun 6 18:40:55 : Deny inbound udp src outside:
                    63.22.106.113/1037 dst lb-dmz:x.x.x.104/31337
Jun 6 18:40:55 : Deny inbound udp src outside:
                    63.22.106.113/1037 dst lb-dmz:x.x.x.121/31337
Jun 6 18:40:55 : Deny inbound udp src outside:
                    63.22.106.113/1037 dst lb-dmz:x.x.x.139/31337
Jun 6 18:40:56 : Deny inbound udp src outside:
                    63.22.106.113/1037 dst lb-dmz:x.x.x.156/31337
Jun 6 18:40:56 : Deny inbound udp src outside:
                    63.22.106.113/1037 dst lb-dmz:x.x.x.174/31337
Jun 6 18:40:56 : Deny inbound udp src outside:
                    63.22.106.113/1037 dst lb-dmz:x.x.x.210/31337
```

© SANS Institute 2000 - 2005 Author retains full rights.

This attacker is looking for Windows machines that have the Trojan backdoor Back Orifice installed, which uses TCP port 31337 to allow entry. Although the destination IP addresses are not all the hosts in the Class C space, the person who submitted these logs noted that not all the hosts that actually exist were probed, so it is likely that rather than the attacker knowing the IP addresses of the hosts on the network and methodically probing only them, he was probably doing a random low-and-slow probe to avoid being noticed.

To block this scan at the firewall, you could try anticipating all the high-numbered ports that are used by the various backdoors, but a simpler solution is to simply shut down all packets destined for ephemeral ports, unless there is some (very unusual, non-standard) legitimite application that uses them.

```
  ipchains –A input –i EXT_NIC –dport !1:1023 –j DENY -l
```

If this scan had not been blocked, nothing particularly terrible would happen as long as no backdoors were installed on any of the PCs. But who wants to guarantee that? :-o

Violation #4: Syn-Fin Scan

Date Time Host Program[Rule#?] Attack Type Source IP Source port Dest. IP Dest. port

```
Jun 6 02:37:55 hosth snort[256]: SCAN-SYN FIN: 63.226.11.117:53 -> a.b.c.19:53
Jun 6 02:37:55 hosth snort[256]: SCAN-SYN FIN: 63.226.11.117:53 -> a.b.c.32:53
Jun 6 02:37:55 hosth snort[256]: SCAN-SYN FIN: 63.226.11.117:53 -> a.b.c.33:53
Jun 6 02:37:56 hosth /kernel: Connection attempt to TCP a.b.c.62:53 from
      63.226.11.117:53
Jun 6 02:37:56 hosth snort[256]: SCAN-SYN FIN: 63.226.11.117:53 -> a.b.c.51:53
Jun 6 02:38:01 hosth snort[256]: spp_portscan: portscan status from
      63.226.11.117: 17 connections across 17 hosts: TCP(17), UDP(0) STEALTH
Jun 6 02:38:01 hosth snort[256]: SCAN-SYN FIN: 63.226.11.117:53 -> a.b.d.52:53
Jun 6 02:38:06 hosth snort[256]: SCAN-SYN FIN: 63.226.11.117:53 -> a.b.e.79:53
Jun 6 02:38:07 hosth snort[256]: spp_portscan: portscan status from
      63.226.11.117: 10 connections across 10 hosts: TCP(10), UDP(0) STEALTH
Jun 6 02:38:07 hosth snort[256]: SCAN-SYN FIN: 63.226.11.117:53 -> a.b.e.91:53
Jun 6 02:38:26 hosth snort[256]: spp_portscan: End of portscan from
      63.226.11.117
```

From the fact that the source and destination port were both 53, it seems that the attacker was hoping to find a host running DNS that had a vulnerability he could exploit, perhaps involving zone transfers (which come from 53). But the main point about these detects is that the packet had both the SYN and FIN flags set. The SYN flag is used to indicate a request to begin a connection session, but a three-way handshake is required before the connection is considered opened. The FIN flag should only be used to request termination of an active, fully open session, so the use of those flags together is a violation of the TCP standard. Some intrusion detection systems don't properly process packets with both those flags set, so the attacker can try to hide his tracks that way.

Ipchains seems to also be a system that cannot detect this flag combination specifically, but it is the sort of low-level trick that is best caught at the router anyway, unless you've bought one of those high-powered commercial firewalls. I can't seem to lay my hands on the syntax for an ACL entry on a Cisco router (not yet owning such a router, and only having downloaded small parts of the documentation off the Cisco web site), but the key is specifying the individual flags set on the packet in question – if both SYN and FIN are set, no matter what else is in the packet or where it came from, you want to dump it.

If this is not caught, it will be a way for the hacker to scan the system without being detected if the IDS won't pick it up either. There very well may be a DNS out there that has a hole…

Violation #5: Echo Request Network Mapping

Date Time Host Rule# Action Protocol Source IP Dest. IP Message Type

```
Jun 4 16:41:47 fwall 11 deny: icmp from 38.27.213.54
  to xxx.xxx.xxx.255 type Echo Request
Jun 4 16:41:47 fwall 12 deny: icmp from 38.27.213.54
  to xxx.xxx.xxx.0 type Echo Request
Jun 4 16:41:48 fwall 11 deny: icmp from 38.27.213.54
  to xxx.xxx.xxx.255 type Echo Request
Jun 4 16:41:48 fwall 12 deny: icmp from 38.27.213.54
  to xxx.xxx.xxx.0 type Echo Request
```

This attacker seems to want to know what IP addresses are real hosts on this Class C network. He is sending ICMP echo request packets (i.e. ping) to two addresses: one with the host section all zeros and the other all ones. The all zeros address is typically used to refer to the network as a whole (in filtering rules, documentation, etc.), but I don't really know what it would actually do as the destination address of a packet. But all ones in the broadcast address for the network, a popular destination for hackers mapping a network using ICMP. (Note: ICMP has no such thing as ports, so there is no port numbers listed in the detects.)

The command to catch all pings from the Internet is:

```
ipchains –A input –i EXT_NIC –p icmp –icmp-type echo-request
```

Or if you only want to stop these specific ones (especially the one destined for the broadcast address is a good one to filter), you would use the following:

```
ipchains –A input –i EXT_NIC xxx.xxx.xxx.0 –p icmp –icmp-type echo-request
ipchains –A input –i EXT_NIC xxx.xxx.xxx.255 –p icmp –icmp-type echo-request
```

If the ones to the broadcast address are left unfiltered, and the answers going back out are also unfiltered (see the above paper on Egress Filtering), each host that has ping enabled (which is typically everybody, since it is often used for network troubleshooting) would reply to the request. With one simple command the attacker would get a very nice list of all the hosts that are awake and responding to pings.

Violation #6: Heinz 57 Mixture

I know you asked for five violations, but I'll throw in one more for extra credit, or just for amusement, anyway. This list of logs was detected during a single day on a home PC connected to a cable modem, proving that it's not just Amazon.com and Bank of America that get attacked!

Date Time (I don't know) Messsage Type? Protocol ?? Source IP Source port Dest. IP Dest. port

```
Jun 3 00:34:01 cc1014244-a kernel: securityalert: tcp if=ef0
  from 24.3.6.190:4421 to 24.3.21.199 on unserved port 27374
Jun 3 00:41:57 cc1014244-a kernel: securityalert: tcp if=ef0
  from 24.3.6.190:2649 to 24.3.21.199 on unserved port 27374
Jun 3 00:57:49 cc1014244-a kernel: securityalert: tcp if=ef0
  from 24.3.6.190:2086 to 24.3.21.199 on unserved port 27374
Jun 3 01:51:17 cc1014244-a kernel: securityalert: tcp if=ef0
  from 24.3.9.15:1567 to 24.3.21.199 on unserved port 27374
Jun 3 05:39:48 cc1014244-a kernel: securityalert: udp if=ef0
  from 62.125.37.168:60000 to 24.3.21.199 on unserved port 2140
Jun 3 11:11:06 cc1014244-a kernel: securityalert: tcp if=ef0
  from 210.140.231.147:109 to 24.3.21.199 on unserved port 109
Jun 3 11:33:22 cc1014244-a kernel: securityalert: tcp if=ef0
  from 208.191.77.182:2982 to 24.3.21.199 on unserved port 8080
Jun 3 11:33:23 cc1014244-a kernel: securityalert: tcp if=ef0
  from 208.191.77.182:2982 to 24.3.21.199 on unserved port 8080
Jun 3 14:25:20 cc1014244-a kernel: securityalert: tcp if=ef0
  from 24.6.159.44:1242 to 24.3.21.199 on unserved port 27374
```

© SANS Institute 2000 - 2005 Author retains full rights.

```
Jun 3 14:47:54 cc1014244-a kernel: securityalert: udp if=ef0
  from 24.15.67.242:137 to 24.3.21.199 on unserved port 137
```

The first four detects, coming from various ephemeral ports to port 27374, look like a check for a Trojan called SubSeven 2.1. Then someone else checked port 2140 once, perhaps looking for Deep Throat or The Invasor. Next came a packet, again from someone new, to port 109, which is the designated port for accessing a POP2 server (not likely to have much luck there with a PC). The next two are hits from somebody else on Wingate (port 8080), then a different person looking for SubSeven 2.1, and finally somebody checking the NetBios UDP port 137. It's a busy day on the Internet!

To block all this stuff, you would filter requests to any port you're not needing to answer on, which for a typical PC might be everything if, for TCP, your firewall software can differentiate between incoming requests for a connection and incoming answers to a request you made. Basically, any incoming SYN packet could be suspect. (Obviously ipchains is not the protocol you would be using on a Windows PC, but this is just an example.) The –y option specifies that we're only looking for packets with the SYN flag set:

```
ipchains –A input –i EXT_NIC –p tcp –y -dport !1:1023 –j DENY -l
```

For UDP, you can't be so clever (since it is a stateless protocol), but unless the cable modem ISP needs NetBios broadcasts for something (maybe to assign you an IP address? I have never used cable modems, as they don't have them in Japan yet to my knowledge), you could just block any traffic to NetBios ports using the example given for Violation #1.

If this traffic was not filtered, eventually you might end up with a Trojan you're not aware of and then someone might find it. Or they might find an ephemeral port open for a current connection you have to a web site and manage to exploit it. Best to be on the safe side!

Defense in Depth Architecture – DDOS Resistance

The keys to being resistant to distributed denial of service attacks is two-fold:

- To protect the site from being victimized by a flood of packets from many sites at once
- To prevent the site from being used as one of the "attacking" sites

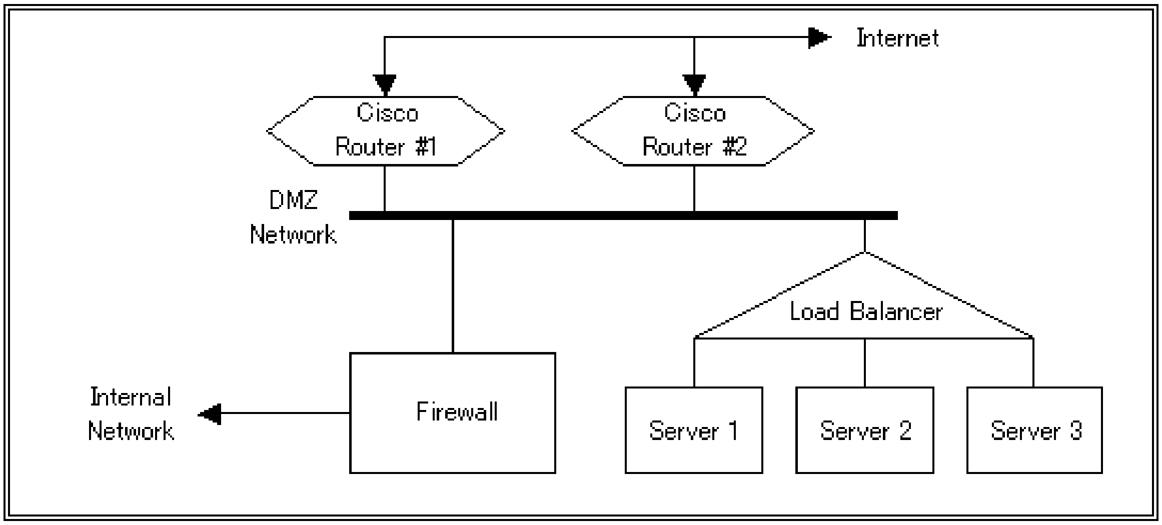
I haven't yet had a chance to study various commercial firewall products in detail, but I know that ipchains does not look robust enough to keep a flood of packets from overwhelming it to the point where it doesn't let legitimate traffic through. But that's more the job of routers, anyway, rather than firewall hosts. The day before writing this paper, I got approval to purchase a Cisco router for my company's network (our current router is not very robust for security, and I wanted a second router in order to construct a screened subnet architecture). In my brief first look at a little of the documentation for the IOS software, I noticed that they have a special feature called "TCP Intercept" that is specifically designed to control the most typical type of DOS attacks - that of a flood of SYN packets to a server with unreachable return addresses, which leaves the server in the awkward position of having to maintain a ton of half-open connections until it can handle no more and ends up refusing legitimate requests. The TCP Intercept feature uses an extended access list to maintain a list of the servers, and rather than sending the SYN packet on, tries to finish establishing the connection as a proxy on behalf of the server. If the connection is established (i.e. its return SYN/ACK is ACKed), it makes a connection with the server and "knits" the two connections together. Meanwhile, it has settings to control the number of half-open connections it allows and the timeouts on those connections, going into what is called "aggressive" mode when thresholds are surpassed (so that it doesn't allow itself to get overwhelmed, either, when a DOS attack is launched).

Based on what I have read, I would say that Cisco routers would be good choices for this network. I note that this configuration is to have two connections to the Internet. It would certainly be a possibility to simply use one router with two interfaces, as Cisco routers can also do load balancing, but if we are really

© SANS Institute 2000 - 2005 Author retains full rights.

paranoid about DDOS attacks, maybe two routers would be even better. I also think it is safe to assume that if this company is worried about distributed DOS attacks, they must be a site that gets a lot of traffic and therefore has a high volume setup for servicing the public (if they were just an ordinary, small site with one server, attackers would not need to employ lots of attacking sites in a coordinated effort in order to bring it down). So although it is not specified in the assignment, I will draw the site as having three servers with a load balancer.

Dual Internet Connection Topography for DDOS Resistance



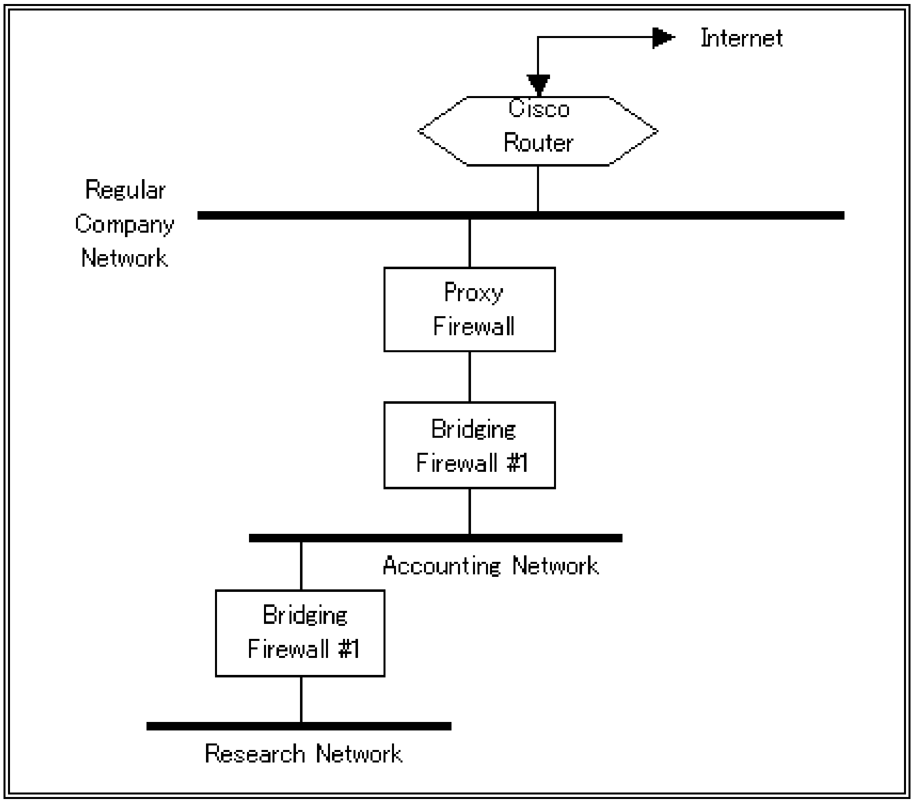
Defense in Depth Architecture – Internal Subnetworks

I see two possibilities for what is meant by the question, and I feel the need to address both. The statement, "…for the most effective protection," could be interpreted to mean specifically what it talked about in the problem, that is, the two critical subnetworks. The second way of looking at it is the broader view, assuming that there is more to the company than those two subnetworks, and that they want decent protection from the Internet as well. There might even be a web and/or DNS server that, since it must be more exposed than the network containing the computers for marketing, production, support, etc., should be on a separate, more DMZ-type network. Performance also is not mentioned as a concern, but in the second way of thinking, would also be considered. I will approach these two ways of thinking individually.

The Approach of Focusing Only on the Stated Need

First let's look at how to make accounting and research as safe as possible, without much regard for other aspects. There is no mention of encryption in the scheme, so I will not use it in my design, but I think it would be good to mention that using encryption, as if the connection to each department was a VPN, would provide even more insulation. Now, getting into the topography discussion, it does not say which of the two departments is more critical, but we can make one of them even more secure without lowering the safety of the other by layering them. For the sake of example say that research is more sensitive. A no-holds-barred, "tighten research as much as possible and accounting almost as much" solution might look like the following:

Single Need Focused Topography

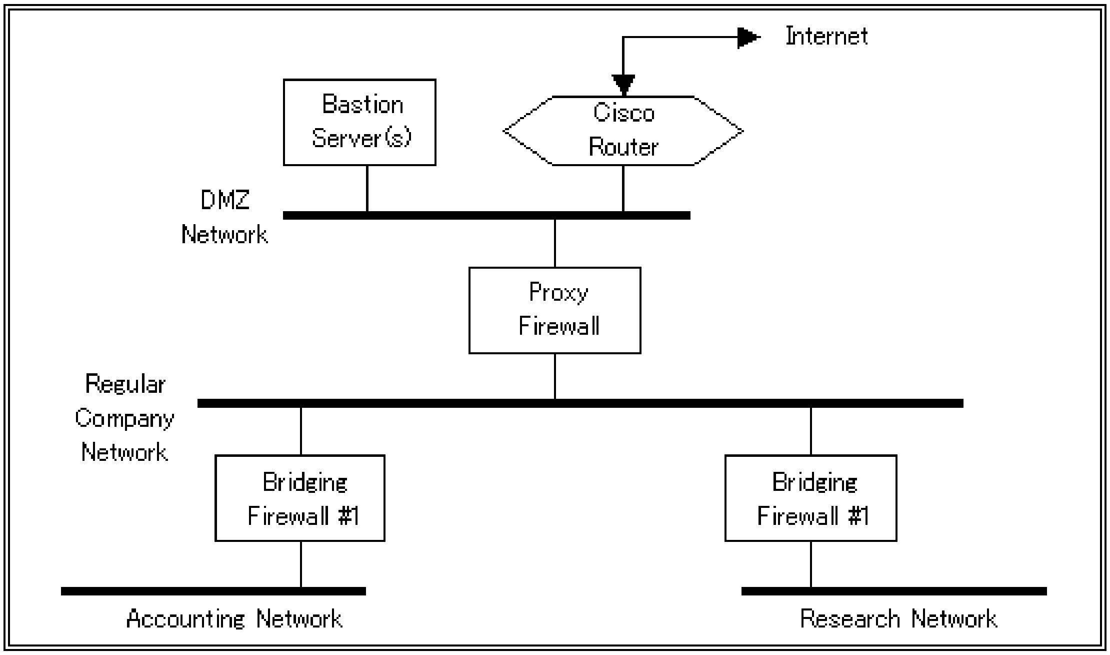


The Well-Rounded Approach

Although other needs are not mentioned in the assignment, I prefer, rather than the above approach, to assume that this is a normal company. That means that there are other company functions besides research and accounting that don't want to get trashed by hackers. It also means that even from the vantage point of research, the marketing department is more "trusted" than badguy.com. Also I assume that everyone, including the two sensitive departments, wants to access the Internet with decent performance. I also suspect that the company is likely to have a web server and a DNS server, and maybe some other stuff that should have a little distance from even such "unimportant" parts of the company such as marketing and production. Considering those things, the above solution looks a bit extreme. I think the guy who ordered the equipment before he left may have been thinking this way, too, because the equipment list seems to lend itself well to a solution that considers a little bit of everything. With the router and the proxy firewall, they can start with a nice, sensible company firewall that has a place for hardened external servers. Then, the bridging firewalls can be used to separate each of the sensitive departments from the rest, resulting in three layers of protection (router, proxy, appliance firewall) between each sensitive department and the big, bad world. The network traffic is completely separated, so no one can sniff where they're not supposed to without breaking through a firewall. Network Address Translation can be used to insulate even the network structures from each other. In the end, I seems to me that even though the statement, "…this equipment… cannot be sent back," sounds like I'm supposed to wish I had something else to work with, I'm actually pretty content with what the guy ordered. Here is the way it would look:

Well-Rounded Approach Topography

© SANS Institute 2000 - 2005 Author retains full rights.



My Test

You work for a 20-person company that currently runs on a single network, with two servers, shared printers, etc., connected to the Internet by a consumer-type router that has a little filtering capability but would definitely not be considered a firewall solution. One server is a fairly new NT-based machine that you use for the important company info (product source code, accounting, etc.), the other is an old but perfectly functional Linux server that currently serves as their web server, DNS, and mail server.

The company is now getting concerned about both security and Internet services, and has assigned you the job of figuring out what best to do (since you took the GIAC course and are now the resident security expert!). They want a real firewall, and they want to add ftp and expand their web site to include functions that will require cgi scripts. They are also concerned about their e-mail privacy, not wanting their e-mail mailboxes too exposed. But they have a limited budget of $2000 to work with. Here are the prices of some items you might consider:

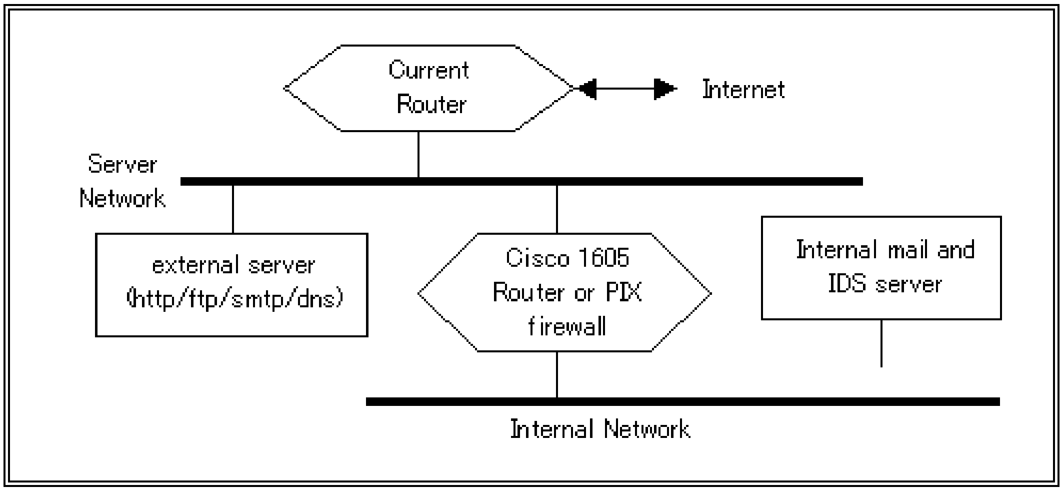
Cisco 802 router with one Ethernet port, one ISDN port, IOS software	$700
Cisco 1605 router with two separately filterable Ethernet ports and card slot for add-ons, IOS software	$1100
Add-on card for Cisco #2 with an ISDN port	$400
Basic Intel box with no software, adequate to run as a Linux server or as a firewall with ipchains or other open-source software	$500
A little faster and bigger Intel box with Windows NT	$1300
Hubs, Ethernet cards for Intel boxes	negligible for the sake of this exercise
Cisco PIX Firewall 506, 2 Ethernet ports	$1500

© SANS Institute 2000 - 2005 Author retains full rights.

Here are a few possible solutions, as I see it. There may very well be other good solutions. In my diagrams I don't show the rest of the network (NT server, client PCs, printers), but they would be on the internal network.

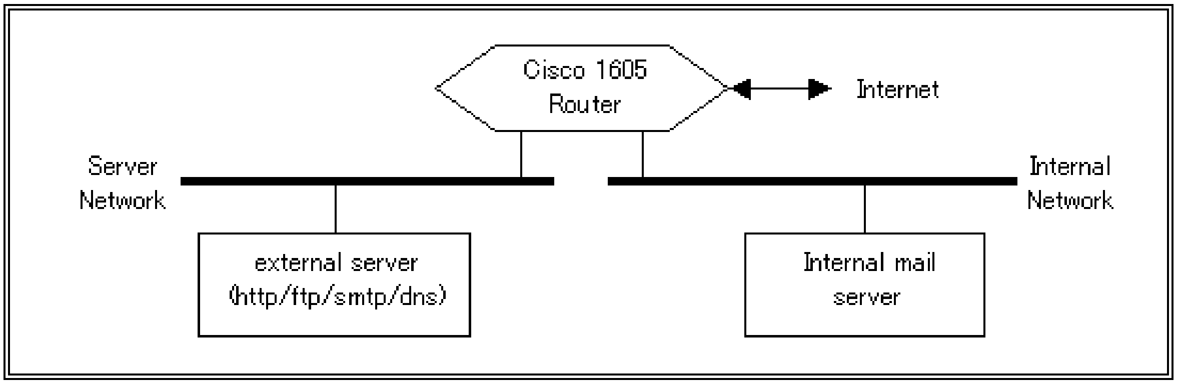
Textbook Screened Subnet Architecture

The first idea is to purchase the Cisco 1605 or PIX firewall and one more Linux server (total $2100 or $2500) and set up a standard screened subnet architecture, using the new servers to cover the external server jobs, and just store-and-forward the mail to the existing server, which would reside on the internal net to protect the mailboxes, and also do intrusion detection.


Current Router
Internet
Server Network
external server (http/ftp/smtp/dns)
Cisco 1605 Router or PIX firewall
Internal mail and IDS server
Internal Network

Modified Screened Subnet Architecture

A variation on the first idea is to get both the Cisco 1605 router and the ISDN add-on card, and put the current router to rest (it could serve as a backup in case of a hardware failure of the Cisco). With the two interfaces, the basic functions of a screened subnet can still be achieved, and the more powerful filtering capability of the Cisco can be applied to the outer interface as well.

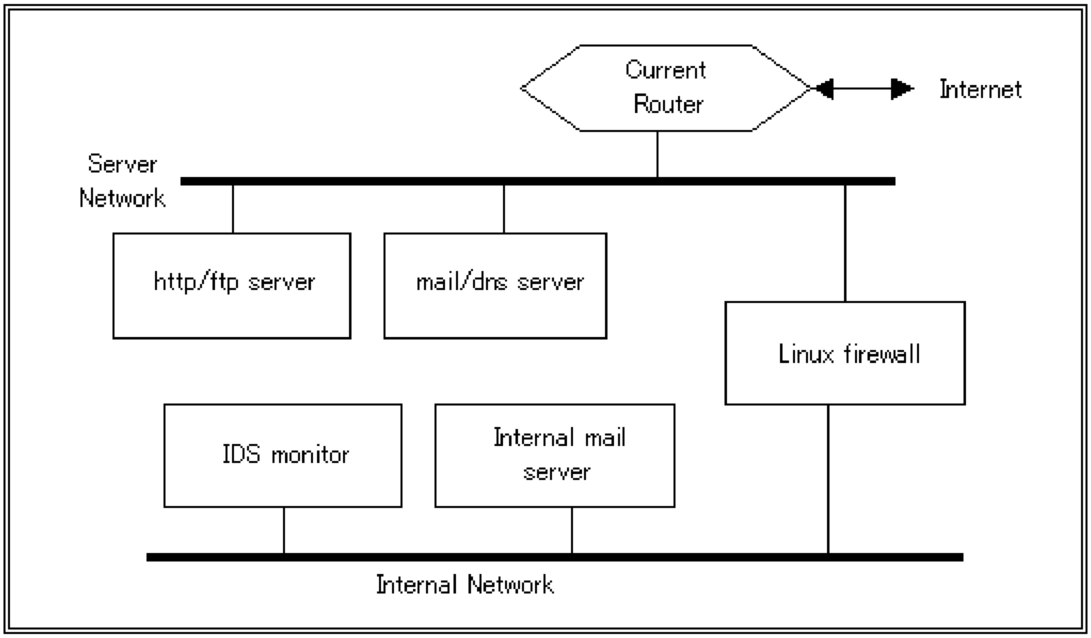


Open-Source Linux Firewall Solution

Another alternative is to use a dual-hosted Linux server for the firewall with the current router, selecting software such as ipchains packet filtering or squid proxy software, or a combination of both types. That would be less secure at the perimeter, but allow purchase of up to three more Linux servers (or an NT server, if they are so inclined, but that's not what I would do), to allow a lot more separation for the

© SANS Institute 2000 - 2005 Author retains full rights.

different servers, and even a dedicated box for intrusion detection.



References: www.techadvice.com/help/Products/F/firewall.htm (for info and prices on commercial software), www.tribecaexpress.com/Cisco (for info and prices on Cisco products), local computer shops (for prices on PCs).

© SANS Institute 2000 - 2005

© SANS Institute 2000 - 2005 Author retains full rights.