Global Information Assurance Certification Paper

GIAC Level 2 GCFW Practical Assignment

(Assignment v1.5a, Amended 3/1/01)

"Secure Network Design, Configuration, Assessment, and Exploitation"

Submitted by

John A. McReynolds

10 April, 2001

GIAC Level 2 GCFW Practical Assignment (Assignment v1.5a, Amended 3/1/01)

GIAC Level 2 GCFW Practical Assignment (Assignment v1.5a, Amended 3/1/01)

Summary:
This paper is a practical presentation of a network security architecture as required for GCFW Certification by the SANS GIACTC.

Audience:
The intended audience is network and security administrators who are responsible for firewall implementation and administration, as well as the general security community who may be able to gain some knowledge from (or offer criticism toward) this paper.

Preface:
This project is one of this writer's many steps towards developing a useful Information Security skill set that will hopefully, one day, help this author provide some meaningful contribution to the security community. The credits for this writer's budding interest are belong to many other works and many other people, and include Stephen Northcutt, who wrote the inspirational words in the inner leaf of my copy of Network Intrusion Detection (2nd Ed) *"....security is a journey and not a destination. We need you to make the contributions to the community....."*.

Indeed, this author has found the field to be compelling and the tasks daunting. Were it not for the highly open and cooperative efforts of SANS, and the support of the large number of extraordinarily clever people in the security community, this writer would be ready to 'throw in the towel'. When one considers the premise that **security is a process and not a state**, it is easy to feel overwhelmed by an ever moving target. It is this writer's hope to develop the perspective, skill, and experience that is so necessary to stay one step ahead of the attacking community.

For now, I am merely a novice. Thank you SANS and the GIAC for planting the seed.

This paper consists of four requisite parts:

1) A design of a secure network for a fictitious business (GIAC Enterprises) that performs online sales and distribution to customers of a product (fortune cookie sayings), while also performing online business with suppliers, and partners. Additionally, this network architecture also includes proposed connectivity and access elements that will support another recently acquired enterprise.
2) An explanation and tutorial of the elements and configuration that enforce the security policy for this organization.
3) A planned, executed, and analyzed vulnerability assessment of the main firewall and perimeter security for the organization.
4) A three-phased attack upon another GIACTC GCFW practical consisting of:
    a. An attack against the firewall of the network described in that practical.
    b. A DoS (Denial of Service) attack against that network from 50 medium-to-high speed sites (Broadband or xDSL)
    c. A plan for an attack upon a host within that network, based upon researched vulnerabilities in that network's perimeter security systems.

SECTION 1-GIAC ENTERPRISES NETWORK ARCHITECTURE

This network architecture has been developed to support the business plan of GIAC Enterprises, a small Internet/E-Commerce start-up whose sole purpose in life is to make over $200 Million per year in revenue from online sales of fortune cookie sayings.
Functionally, this network is designed to provide GIAC Enterprises with secure facilities from which to:

1) Access the Internet from inside the network for such routine activities as:
    a. Conventional web-browsing (HTTP – Outbound)
    b. Conventional electronic mail transfer (SMTP – Outbound)
    c. Conventional electronic mail delivery (SMTP – Inbound)
    d. Conventional Domain Name Service resolution to support **a., b., and c.** (DNS – Outbound)

2) Host public web services for the advertisement and marketing of GIAC Enterprises products. At a minimum, services must include:
    a. Conventional web hosting (HTTP – Inbound)
    b. Basic DNS service to support **a.** (DNS – Inbound)

3) Host enhanced web services for the sales and delivery of GIAC Enterprises products. At a minimum, services must include:
    a. Capability to perform secure web transactions (HTTPS/SSL)
    b. Capability to securely deliver/transfer products to the customer (HTTP/HTTPS/SSL)
    c. Capability to securely accept electronic-format products from suppliers (HTTP/HTTPS/SSL)
    d. Capability to securely process financial transactions between GIAC Enterprises and its customers (HTTPS/SSL or similar)
    e. Capability to securely process financial transactions between GIAC Enterprises and its suppliers (HTTPS/SSL or similar)

4) Provide secure remote access (VPN services) to and from:
    a. Partners who perform value-added services
    b. Staff who work off-site (including sales staff/'road warriors')

5) Provide initial facilities for basic but secure network extensions to a recently acquired company, with provisions for long-term, dedicated-bandwidth connectivity.

This requirements list provides a foundation upon which to build a secure network for GIAC Enterprises. However, the service requirements also imply a potentially complex design. Complexity is not necessarily desirable, as has been stated in Murphy's Laws On Combat #26: "If it's too tough for the enemy to get in, you can't get out". Obviously, this doesn't quite hold true in the area of Network Security, but too many hurdles can render a network too cumbersome to use productively. Good network security requires a balance between security and usability. Simultaneously, however, it requires a thorough understanding of the threats and vulnerabilities, and an awareness of any known deficiencies in the architecture, in order that alternative defense measures may be taken in the event of a compromise. *Assume the enemy is smarter than you*, but set yourself up so that you can watch his activities.

The following items have been considered while developing this design:

1) Simple is better
    a. It is easier to manage
    b. It is easier to troubleshoot
    c. It is easier to recover from a disaster
    d. It is easier to secure

2) Defense-In-Depth / Layered Security
    a. Access Control at multiple points

b. Better isolation of business functions
c. Reduced vulnerability of 'crown jewels'

3) Scalability
   a. Networks grow as businesses grow
   b. Enhancements can be modular

4) Simplicity of Management
   a. Good graphical interfaces are easier to maintain and require less training

5) Other Defensive Measures
   a. Intrusion Detection Systems
   b. Secure, Centralized, Fault-Tolerant Accounting and Logging

6) Cost
   a. Current market effects have mandated moderate cost, with an initial security hardware budget of $20,000, including annual support contracts.
   b. Judicious use of moderate cost elements, which can later be reused in other areas of the network as the network expands and cash becomes available for higher performance (higher cost) equipment.
   c. Where applicable and easy to support, open source tools can and should be used.

In view of the above considerations, the following architecture has been developed:

1) A main perimeter structure consisting of:

a. A Cisco 3620 border/screening router running IOS v.12.0 (32 MB RAM)
This unit was selected for the following reasons (not necessarily in order of precedence):

i. Cisco IOS supports a very rich set of Access Control Lists (ACL's), both conventional and reflexive.
ii. This hardware platform supports a broad selection of interfaces, allowing for expansion to DS-3 capacity serial interfaces and 100 Mb Ethernet interfaces. Initially, the selected interfaces consist of a ***T-1 interface for the external link and a 10 Mb/s Ethernet internal interface*.**
iii. When the time comes to establish a multi-homed posture for network reliability, this platform provides a solid platform from which to run BGP-4.
*Remember-One of the 3 elements of security is **availability**.*
iv. The relatively high CPU power and memory capacity support large sets of ACL's, with a minimal impact upon performance.
v. IOS v.12.0 also provides support for simple Virtual Private Network (VPN) and Network Address Translation (NAT) services, although they are not used in the current implementation.
vi. The fundamentals of Cisco IOS configuration are well known, thus subsequent administrators are likely to have some fundamental skill in this area. This is important from a management standpoint.

b. **A main firewall unit running Checkpoint Firewall-1 (v.4.1/SP3) on top of a Sun Ultra-5 platform (Solaris 7/SunOS 5.7)** with external, internal and service network interfaces. This platform was selected for:

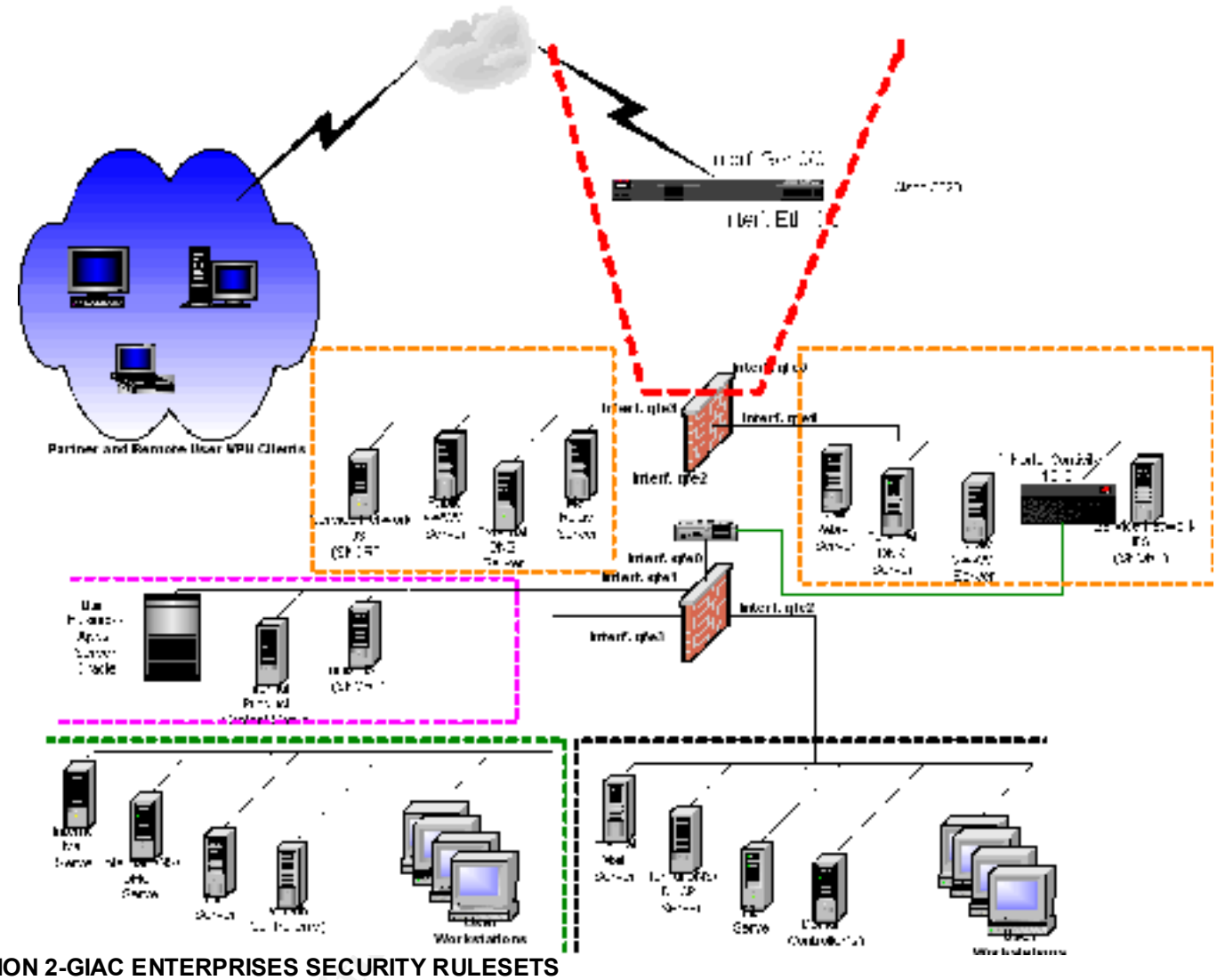
i. Ease of configuration-The Firewall-1 user interface is tops in its class.
ii. Extensibility – The Firewall-1 architecture readily supports centralized management of multiple firewalls, allowing consistent policy deployment. Additionally, management of firewall modules can be performed on a platform separate from the firewall itself.
iii. Scalability – Licenses can be purchased in support of 50-to-unlimited user increments.
iv. Reliability – Support for multiple units in a redundant, high-availability configuration, as funding becomes available for expansion. Again **availability** is a consideration.

v. Good cost/performance ratio - Sun Ultra-5 hardware is comparatively inexpensive, and can be reused when higher performance platforms become appropriate.
vi. In light of the above criteria, however, the selection was made with the caveats presented in the excellent paper about the Firewall-1 state table by Lance Spitzner (http://www.enteract.com/~lspitz), and testing provisions have been made accordingly, as detailed in Section 3- Assessment and Audit.

2) **An internal screened network** to provide additional layers of security (Defense in Depth) and isolation of business functions. This consists of the following architecture:
   a. A secondary firewall to provide an additional level of security between the main perimeter and main user network, while providing an additional subnet for more carefully controlled business application machines and product management machines. Again, an UltraSPARC/Solaris-based Checkpoint Firewall-1 was selected for the following reasons (in addition to those listed for the main firewall itself):
      i. A native User/Client/Application authentication gateway provides additional access control
      ii. Additional internal interfaces on this firewall provide support for a well segmented business applications network, thus providing comprehensive access control based upon 'need to know'.

3) **Virtual Private Network facilities** for partner and remote user access. This consists of:
   a. Nortel Contivity 1510 Extranet Access Switch. Deployed in a double-armed manner (one interface supports external encrypted and the other, internal, 'en clar' traffic), this unit was selected for the following reasons:
      i. Ease of management-The Contivity user interface is, like the Firewall-1 interface, tops in its class. Client software is also easy to manage. Additionally, the unit's native backup and recovery facilities provide for quick and simple restoration in the event of a failure.
      ii. Performance-The basic 1510 unit supports 100 concurrent IPSec connections, and supports 25-50 connections with almost no detectable latency. The VPN function should never be performed on the firewall itself, as this function creates a significant performance degradation, and also creates a single point of failure.
      iii. Flexibility – The unit is supported by an internal or external LDAP database, and has complete support for IPSec, L2TP, and PPTP protocols, while supporting highly configurable security profiles for groups and users. Clients can be standalone remote workstations or branch offices. Reconfiguration is quite simple, should the appliance need to be relocated on-the-fly.
      iv. Security – The unit supports its own ACL structure, which is, by default, a 'deny any' set of lists. An extensive set of logging and reporting facilities (including support for a remote syslog server) is another highly desirable feature.

   (Author's note: I do not work for Nortel – This product is simply one of the most easily managed units that I have worked with over several years. The only limitation is lack of support for multi-processor clients.)
4) Due to the recent acquisition of another small E-commerce startup, plans have been made to install a private, 256Kbps Frame-Relay link to the acquired company's network. Until the installation of this link is complete, VPN services are being utilized from that company to GIAC Enterprises. Currently, users from that network are authorized to gain VPN access using a profile for Business users, as described later in the VPN configuration section. Once the Frame-Relay circuit has been established, this too will be encrypted, due to potential vulnerabilities of some Frame-Relay switches. The dedicated circuit is intended to relieve the primary network T-1 Internet circuit of any unnecessary load, thus allowing it to be better utilized for customer traffic.



SECTION 2-GIAC ENTERPRISES SECURITY RULESETS

Due to the complexity of the GIAC network, a comprehensive rule set has been defined to support the GIAC Security Policy (The actual policy document is not included here, but rule sets are based upon the connectivity requirements outlined in Section 1). The fundamentals include protecting against the 'SANS Top Ten' vulnerability list, (references may be found at the end of this paper) but also some additional items referenced in Hacking Exposed, 2nd Edition by Joel Scambray, et al., and some fundamentals that are part of the GIAC/GCFW Curriculum.
Although not detailed in this paper due to its scope, the following items are covered as part of the site's security procedures (detailed in the site's security policy document)

1) Operating systems on all core systems must be 'hardened' (NT registry auditing and UNIX/NT service disablement, etc.)
2) Use SSH for all management, where applicable
3) Use and enforce strong passwords
4) Perform frequent backups and initial system snapshots
5) Perform regular system log audits
6) Perform system file integrity tracking via Tripwire or other file integrity checker

A) Border router core security policy enforcement with ACL's

The fundamentals of Cisco IOS Access Control Lists (ACL's) are well documented in various texts, and these are implemented in this network design. The ACL's used here are mostly *Extended Access Lists*, which in IOS are referenced by identifier numbers 100-199, and 2000-2699. The Extended ACL's provide packet filtering based upon source and destination address and port, either specifically or by range, and also support filtering by protocol (TCP, UDP, ICMP, and other specified IP protocols such as AH and ESP, protocols 51 and 50, respectively). However, being primarily 'stateless', they cannot intelligently filter based upon information gained from all elements of a connection. In some implementations, 'reflexive' ACL's can be used to gain some information about the state of a connection and allow appropriate returning packets to pass. This design does not utilize those features, for performance reasons (reflexive ACL's require some additional CPU processing and memory overhead). Instead, the primary firewall (FW-1) performs stateful inspection of packets that have been allowed through the screening router.
Additionally, IOS Standard Access Lists (ACL numbers 1-99 and 1300-1999) are included, to provide security for the remote management console of the router.

There are two caveats to the use of Cisco IOS ACL's:

1) They cannot intelligently process fragments. They may screen initial fragments that contain the IP and subsequent protocol headers, but subsequent fragments identified only by fragment ID can pass unchecked. This could (and often does) present a security hazard. The stateful firewall, however, does perform fragment checking prior to passing them on to the destination host. This is shown in Section 3, Assessment and Audit
2) In IOS v11.2, Extended Access List number 199 DOES NOT FILTER TRAFFIC. This is a known bug with Cisco ACL's and has been reportedly fixed in v12.0 and later (this author discovered and reported that issue to Cisco, who heretofore had been blissfully unaware of the problem). However, as a conservative precaution, Extended ACL 199 is not used in this project (nor in any operational network under this author's administration).

Beyond the subject of Cisco ACL's lie some of the other top issues related to the protection of a network behind routers:

1) Packets destined for any IP broadcast address should be denied, whether it be the common class-C broadcast address of 'x.y.z.255', or the common BSD-UNIX/Nortel-Bay networks default broadcast address of 'x.y.z.0'. In a subnetted environment, the all-ones host address within a subnet comprises the broadcast address for that subnet (i.e. in subnet 208.190.65.64 with a mask of 255.255.255.224, the broadcast address is 208.190.65.**95**). The reasoning behind this is related to the 'Smurf' type of Denial of Service attack, where a packet with a forged source address of a target is sent to an 'amplifying' network's broadcast address. By the rules according to IP, all hosts on that subnet should reply to that packet. Done enough times toward enough 'amplifying' networks, just a few forged packets could bring a 'victim network' to its knees under a flood of replies from the intermediate networks, who's hosts are merely doing what they've been programmed to do. Therefore, in ALL CISCO SCREENING ROUTER DEPLOYMENTS, any configuration MUST include the command on all interfaces:

   no ip directed broadcast

2) Cisco ACL's, while filtering packets in a desirable fashion, will also politely tell the offending host that their packets were blocked using the ICMP message 'Port Unreachable' (Type 3/Code 3), 'Host Unreachable' (Type 3/Code 1), or even more politely, 'Administratively Prohibited' (ICMP Type 3/Code 13). Usually, it is not desirable to explain this to an attacker, who can then infer that we are using ACL's to control traffic.
   Therefore, in ALL CISCO SCREENING ROUTER DEPLOYMENTS, any configuration MUST include the command

   no ip unreachables

3) IP Source routing must be prevented. The ramifications of allowing source-routed packets into a network are many, foremost of which is the ability of a malicious source to cause the victim routers and hosts to send packets where they were not intended, thus setting up the

possibility of a routing 'black-hole' or worse, a man-in-the-middle attack. Therefore, in ALL CISCO DEPLOYMENTS, any configuration MUST include the command

```
no ip source route
```

4) Any services offered by the Cisco router (including echo, chargen, and finger) could be exploited and should be dropped. . Therefore, in ALL CISCO DEPLOYMENTS, any configuration MUST include the commands

```
no ip tcp small services
no ip udp small services
no service finger
no ip http
```

5) By default, Cisco's IOS does not protect all passwords used for access to the router. Therefore, it is VITAL that a method for encrypting all router passwords be used. Therefore, in ALL CISCO DEPLOYMENTS, any configuration MUST include the command:
   `service password-encryption`
   This command will provide a *modest* measure of encryption for system passwords by using a simple algorithm. HOWEVER, if the router's configuration file were compromised, passwords using this encryption scheme could be easily cracked. In this instance, *some* protection is better than none at all.
   Additionally, Cisco IOS provides two methods of defining the higher-privileged 'enable' password: One using no base encryption (**`enable <password-string>`**), the other using an encryption algorithm (**`enable secret <password-string>`**) **.** *ONLY THE LATTER METHOD SHOULD BE USED IN LIVE DEPLOYMENTS.*

6) As is the case in most all SNMP managed network equipment, default community strings often come preset from the factory. Usually, the default Read-Only Community name is "public", and the default Read-Write Community name is "private". It is always good practice to change these to some other name, in order that unauthorized management via SNMP will be less likely to occur. Therefore, in ALL CISCO DEPLOYMENTS, any configuration MUST include the commands:

```
snmp-server community <new-community-string1> ro  ! Read-Only
snmp-server community <new-community-string2> rw  ! Read-Only
```

   Additionally, Cisco supports the use of ACL's to specify which hosts are authorized to manage the device. This is also good practice.

The ACL segments detailed below provide protection based upon the "SANS Top Ten".
An **important note** pertains to the requirement that 'returns' from connections initiated internally be allowed through. There are 3 ways to do this:

A) Allow all TCP and UDP ports with destination ports greater than 1023 bound for the internal network through.
B) Allow all UDP packets destined for ports greater than 1023 and "Established" (**`established`**) TCP connections through. (This method was selected with careful thought and a keen awareness of the ramifications. The details are explained below) This method allows all UDP packets with destination ports greater than 1023 And TCP packets with the 'ACK' bit set to pass through.
C) "Reflexively" determine which connections are returns, and allow those based upon the simple state table that Cisco supports with the '**`reflect`**' option to an extended ACL. This method is described extraordinarily well in Scott Winter's GCFW practical (links to his practical may be found at the end of this paper). However, this method does consume additional CPU power, and GIAC Enterprises intends to later deploy a

multihomed BGP-4 implementation for availability. Because BGP-4 also consumes a large amount of router resources, this option was ruled out. Further, for some, the principle of 'Let the router do the routing', precludes the extensive use of reflexive ACL's in some configurations.

The protocol-specific issues related to the use of FTP from an internally initiated connection need to be considered here as well. To be specific, when FTP is used in its traditional mode, 2 complete connections are established:

- The client initiated connection outbound to the server is from TCP >1023 to TCP 21. Subsequent packets from the server have the ACK bit set, and are passed by the (**established)** option.
- The server then initiates a connection to the client from TCP 20 to TCP >1023.

In this instance, the (**established)** or **'reflect'** options can't help us, as the second connection begins with a SYN ***from*** TCP port 20 of the server ***to*** a random 'ephemeral' port on the client. Thus all TCP ports > 1023 must be allowed through for this to be supported. The use of FTP PASV mode alleviates this problem by means of the server allowing the client to initiate the second connection as well, but there is no guarantee that all FTP connections will be set up as such. Documented policy *could* dictate that only PASV connections be used, but does not, in this case. Therefore, with this issue considered, the option selected was number 2 in the list above, whereby all ports above 1023 are allowed for UDP, and source-port 20 is allowed from outside hosts only to the 2 internal NAT addresses reserved for the two internal user networks. Careful blocking of other specific ephemeral ports (>1023) is performed in the access lists below to protect against attacks on services that reside in that range i.e. UNIX NFS, etc.), and then the ACL option 'established' is used to cover the remaining TCP connections. Finally, the stateful firewall is left to perform the work of sorting which connections belong to whom.

The actual ACL's are shown here, and in Section 3, Assessment and Audit, testing recommendations and traffic/log samples are provided to support them.

1) Router Control/Console Access
   Access to the router's management port (telnet virtual terminal) must be restricted to only those hosts that are authorized by the site security manager. The **standard** ACL segment below describes this, and how it is applied from the configuration prompt **config#**:

```
access-list 97 permit xxx.yyy.zzz.68

line vty 0 4
   access-class 97 in
```

2) Ingress Filtering
   Initial screening of traffic takes place with the use of these **extended** ACL's. The final ACL defined below is applied to the external serial interface from the configuration prompt **config#** using the command

```
interface Serial0/0
   ip access-group 197 in
```

   a. Prevent External Spoofing of Local Addresses and send to the log:

```
access-list 197 deny   ip xxx.yyy.zzz.0 0.0.0.255 any log
```

b. Private addresses *(RFC-1918) – These should NEVER be seen as source addresses on the Internet)*

```
access-list 197 deny   ip host 0.0.0.0 any log
access-list 197 deny   ip 10.0.0.0 0.255.255.255 any log
access-list 197 deny   ip 127.0.0.0 0.255.255.255 any log
access-list 197 deny   ip 172.16.0.0 0.15.255.255 any log
access-list 197 deny   ip 192.168.0.0 0.0.255.255 any log
access-list 197 deny   ip 224.0.0.0 31.255.255.255 any log
```

c. Deny ANY attempts to connect to any TCP or UDP port on our screening router, and log them:

```
access-list 197 deny   tcp any host xxx.yyy.zzz.3 log
access-list 197 deny   udp any host xxx.yyy.zzz.3 log
access-list 197 deny   tcp any host 63.239.32.29 log
access-list 197 deny   udp any host 63.239.32.29 log
```

d. PERMITS
These rules allow access from the Internet TO our web, email, and DNS servers, and the last three rules control IPSec and key exchange packets to our VPN device. Note that the VPN connections are logged, and TCP port 53 is not included-There are too many exploits of common BIND/NAMED implementations using TCP port 53. Because this site does not need to perform DNS zone transfers beyond its borders and because most (if not all) host lookups are not likely return records larger than 512 bytes, TCP port 53 can be blocked with little or no ill effect.

```
access-list 197 permit tcp any host xxx.yyy.zzz.200 eq www
access-list 197 permit tcp any host xxx.yyy.zzz.200 eq 443 !(HTTPS)
access-list 197 permit tcp any host xxx.yyy.zzz.230 eq www
access-list 197 permit tcp any host xxx.yyy.zzz.230 eq 443 !(HTTPS)
access-list 197 permit tcp any host xxx.yyy.zzz.222 eq smtp
access-list 197 permit tcp any host xxx.yyy.zzz.232 eq smtp
access-list 197 permit udp any host xxx.yyy.zzz.199 eq domain
access-list 197 permit udp any host xxx.yyy.zzz.229 eq domain
access-list 197 permit udp any host xxx.yyy.zzz.193 eq isakmp log
access-list 197 permit esp any host xxx.yyy.zzz.193 log
access-list 197 permit ahp any host xxx.yyy.zzz.193 log
```

Because of the decision to allow ephemeral ports (TCP/UDP > 1023) through, extremely careful filtering of vulnerable services must be performed. These are detailed below:

e. Although the final rule in this access list provides protection against many attempts to connect to ports below 1024, these rules provide protection against SNMP exploits and the UNIX 'r' services such as rlogin. This group also helps to clean up connections attempts to NetBIOS services that would otherwise clutter up the log in the Firewall.

```
access-list 197 deny   tcp any any eq sunrpc log
access-list 197 deny   udp any any eq sunrpc log
access-list 197 deny   tcp any any range 135 139
access-list 197 deny   udp any any range 135 netbios-ss
access-list 197 deny   tcp any any eq 445 log !NetBIOS under Win2000
access-list 197 deny   udp any any eq 445 log !NetBIOS under Win2000
access-list 197 deny   tcp any any eq finger log
access-list 197 deny   udp any any range 161 162 log ! SNMP
```

```
access-list 197 deny   tcp any any range 161 162 log ! SNMP
access-list 197 deny   tcp any any range exec 520 log
access-list 197 deny   udp any any range exec 520 log
```

f. This group blocks attempts to exploit other services, including the UNIX services NFS and Xwindows)

```
access-list 197 deny   tcp any any eq 1080 log
access-list 197 deny   udp any any eq 1080 log
access-list 197 deny   tcp any any eq 2049 log
access-list 197 deny   udp any any eq 2049 log
access-list 197 deny   tcp any any eq 4045 log
access-list 197 deny   udp any any eq 4045 log
access-list 197 deny   tcp any any range 6000 6255 log
access-list 197 deny   udp any any range 6000 6255 log
```

g. Router Visibility – Although these are not included in the final configuration (they are blocked by the first rules preventing any TCP or UDP traffic to either interface on the router), it is important to note that these are little-known ports that Cisco routers have for management purposes, and other networks may wish to include them in ACL's (Courtesy of Hacking Exposed, 2nd Ed.):

```
access-list 197 deny   tcp any any eq 1999 log
access-list 197 deny   tcp any any eq 2001 log
access-list 197 deny   tcp any any eq 4001 log
access-list 197 deny   tcp any any eq 6001 log
access-list 197 deny   tcp any any eq 9001 log
```

h. Firewall Visibility – Although these are not included in the final configuration (they are blocked by the last rule preventing any TCP or UDP traffic to any other host), it is important to note that these are little-known ports that Checkpoint Firewalls have for management purposes, and other networks may wish to include them in ACL's (Courtesy of Hacking Exposed and Lance Spitzner's Firewall-1 web page):

```
access-list 197 deny   tcp any any range 256-258 log
access-list 197 deny   udp any any range 256-258 log
```

i. ICMP traffic must also be carefully screened to limit visibility of our network and to limit the types of exploits that can be used against us. For troubleshooting purposes, some types of ICMP traffic need to be allowed, specifically ICMP echo-reply (Type 0/Code 0). This type of packet is allowed with the following caveat: It can (and has) been used to encapsulate messages to Trojan server agents that have been planted in unprotected machines. As a protective measure, a duplicate access list could be prepared that does NOT allow these packets through, and simply switching ACL's on the interface will provide a temporary allowance for these packets.

```
access-list 197 permit icmp any any echo-reply
access-list 197 permit icmp any any time-exceeded
access-list 197 permit icmp any any packet-too-big
```

Allow the remaining return ports to the mail servers, DNS servers, internal NAT addresses,

```
! Returns to Public Mail Servers
access-list 197 permit tcp any host xxx.yyy.zzz.222 established
access-list 197 permit tcp any host xxx.yyy.zzz.232 established
! Internal NAT address TCP returns
```

```
access-list 197 permit tcp any host xxx.yyy.zzz.65 established
access-list 197 permit tcp any host xxx.yyy.zzz.66 established
! Intern NAT FTP-Data connections - Firewall drops unmatched instances
access-list 197 permit tcp any eq ftp-data host xxx.yyy.zzz.65 gt 1023
access-list 197 permit tcp any eq ftp-data host xxx.yyy.zzz.66 gt 1023
! Internal NAT address UDP returns
access-list 197 permit udp any host xxx.yyy.zzz.65 gt 1023
access-list 197 permit udp any host xxx.yyy.zzz.66 gt 1023
! Public DNS server returns
access-list 197 permit udp any host xxx.yyy.zzz.199 gt 1023
access-list 197 permit udp any host xxx.yyy.zzz.229 gt 1023
```

DROP EVERYTHING ELSE and LOG!

```
access-list 197 deny ip any any log
```

3) Egress Filtering on the router's internal interface is also extremely desirable, as it prevents leakage of any private addresses (Yes, NAT has been known to not work properly), and to prevent the advertisement of SNMP management and the existence of NetBIOS, etc.
The router could still be managed by SNMP with a small change to allow specific source addresses for that protocol.
This ACL is applied to the INTERNAL router interface as follows:

```
interface Ethernet0/0
      ip access-group 107 in

access-list 107 deny   udp any any eq 135
access-list 107 deny   udp any any range netbios-ns netbios-ss
access-list 107 deny   tcp any any eq 135
access-list 107 deny   tcp any any range 137 139
access-list 107 deny   tcp any any eq 445
access-list 107 deny   udp any any eq 445
access-list 107 deny   udp any any range snmp snmptrap
access-list 107 deny   tcp any any range 161 162
access-list 107 permit ip 203.170.70.0 0.0.0.255 any
access-list 107 deny   ip any any log
```

B) THE FIREWALLS (CheckPoint Firewall-1 v4.1/SP3)

Checkpoint Firewall-1 was chosen for this architecture due to its ease of administration and scalability. However, it does have a significant history of exploits and vulnerabilities, as are well described by Lance Spitzner (see References) and also by Thomas Lopatic, John McDonald, and Dug Song at the BlackHat 2000 Briefings. (see References) These issues were handled admirably by Checkpoint, and the latest release with service packs has been shown to provide a very high level of security, while still maintaining the usability and scalability that is required of deployments such as this one.
As explained in Section 1, the architecture consists of 2 firewalls, one Primary, to provide protection for the Services networks and initial protection for the Internal networks, and one Internal firewall, which provides additional security for the internal networks, as well as providing Network Address Translation for the three internal networks, and access control to two of them, which are sensitive network areas:

1) The Product Network – This area provides all of the business and content-related systems required in support of E-Commerce operations. Access to this network must be strictly controlled, and the firewall provides controls based upon both IP address and upon identity, through the use of authentication rules.

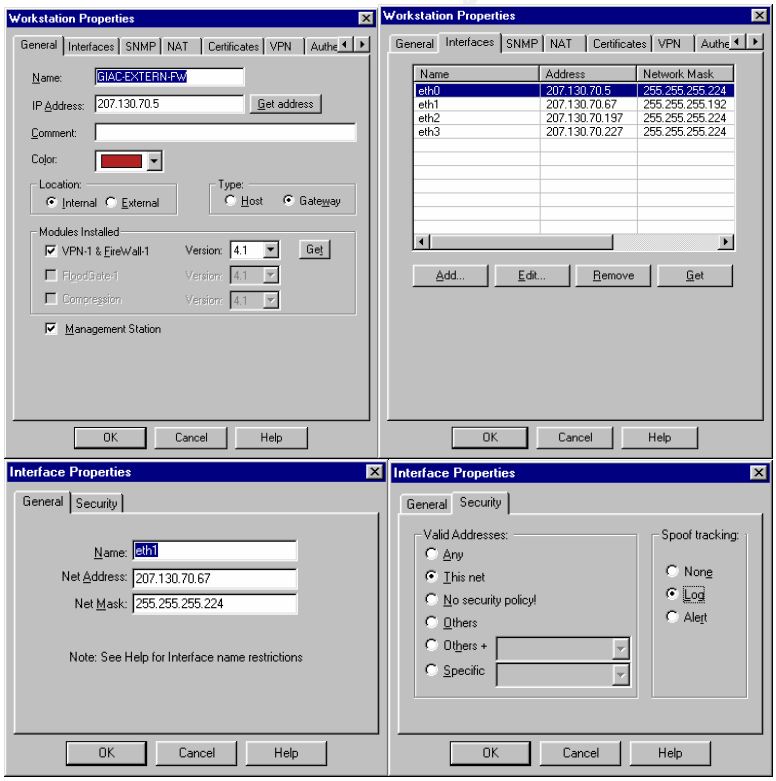
2) The Business Network – This area contains other business-related users and systems for sensitive areas such as Finance, Payroll, Product Development, and Human Resources

The firewalls have an extensive database of network ***objects*** (hosts, routers, networks, etc.) that are specifically defined for this network. Customized information is defined for each object and includes IP address of the object, the IP addresses of any additional interfaces, object type (workstation, gateway, etc.), and description. Additionally, an extensive database of network ***services*** is maintained. These describe well-known services such as HTTP, SMTP (mail) by protocol type (i.e. TCP or ESP) and port number, if applicable. Additional custom definitions can be created, and may include **groups** of objects or services, for convenience.

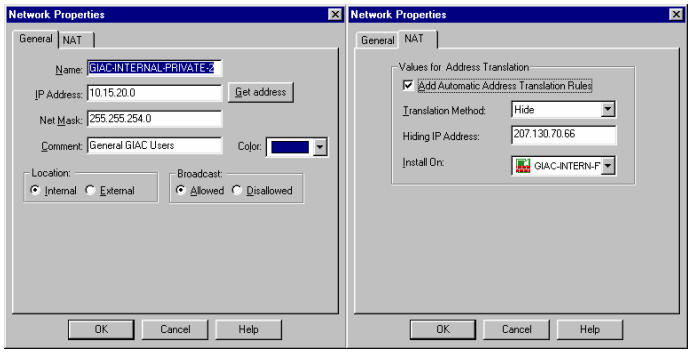
Policy enforcement is performed by creating rules which determine whether a packet is accepted or rejected based upon the **object** and **services** included in it. By maintaining a clear "state table" of all currently active, authorized connections, any anomalous packets bound for authorized hosts are denied, such as forged packets with the 'ACK' bit set. Additional inspection code determines the legitimacy of a packet based upon size and other criteria. An important feature of Firewall-1 is its ability to perform 'virtual fragment assembly', to determine the legitimacy of any packets that may have been broken up during transit. Packet fragments that are too small or out of order are dropped, as is shown in the Section 3 – Assessment and Audit.

As noted in Lance Spitzner's paper "Understanding the Firewall-1 State Table", some important issues have been resolved in the latest release and service pack, most notably the former tendency of Firewall-1 to allow lone 'ACK' packets to pass, without a corresponding 'SYN' packet, which is required to begin the TCP '3-Way Handshake', required of all TCP-based connections.

Definitions of Network objects are performed by defining the base IP address, subnet mask, location, and whether the IP broadcast address should be included as a valid address in the defined network. Although space does not permit the full graphical representation of all network objects defined for GIAC-Enterprises, an example is shown here, and the text-representation of the object definitions is provided in **Appendix –1.**



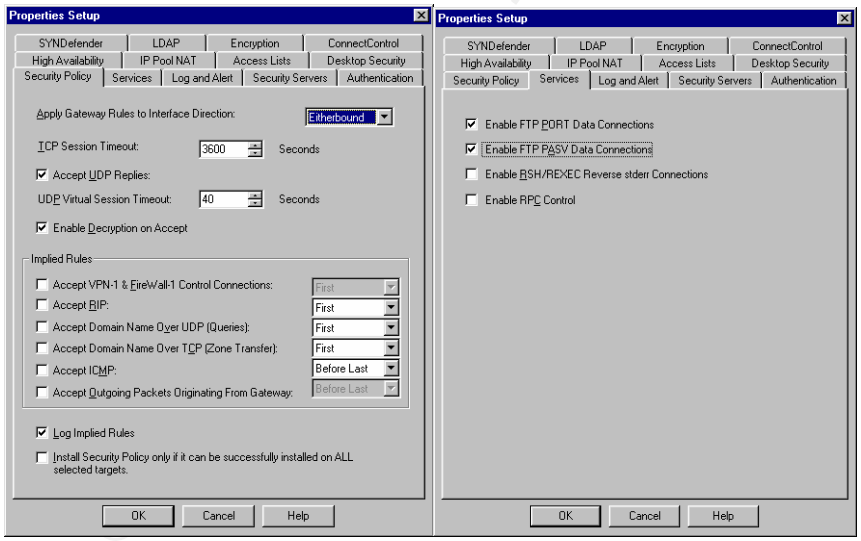
Additionally, Network address translation can be defined easily for Network objects. NAT parameters should include whether the 'valid' address is static (1 translated address per 1 valid address), or if the entire range of addresses to be translated are to be 'hidden' behind one address, as shown below. In the GIAC Enterprise network, a mix of static and 'hiding' translations is performed.



1) The Primary Firewall Rule Base

The Primary Firewall provides fundamental support of, and protection for, the connection requirements described in Section 1.

For each rule base, global properties are defined which control what services may be allowed by default, what any connection parameters may be, logging parameters, etc.
Note that all services within the "Implied Rules" box are deselected. Any requirements for these services are defined within the rule base itself, for better control and accounting. Far too many administrators overlook this section, inadvertently creating vulnerabilities that could be avoided.



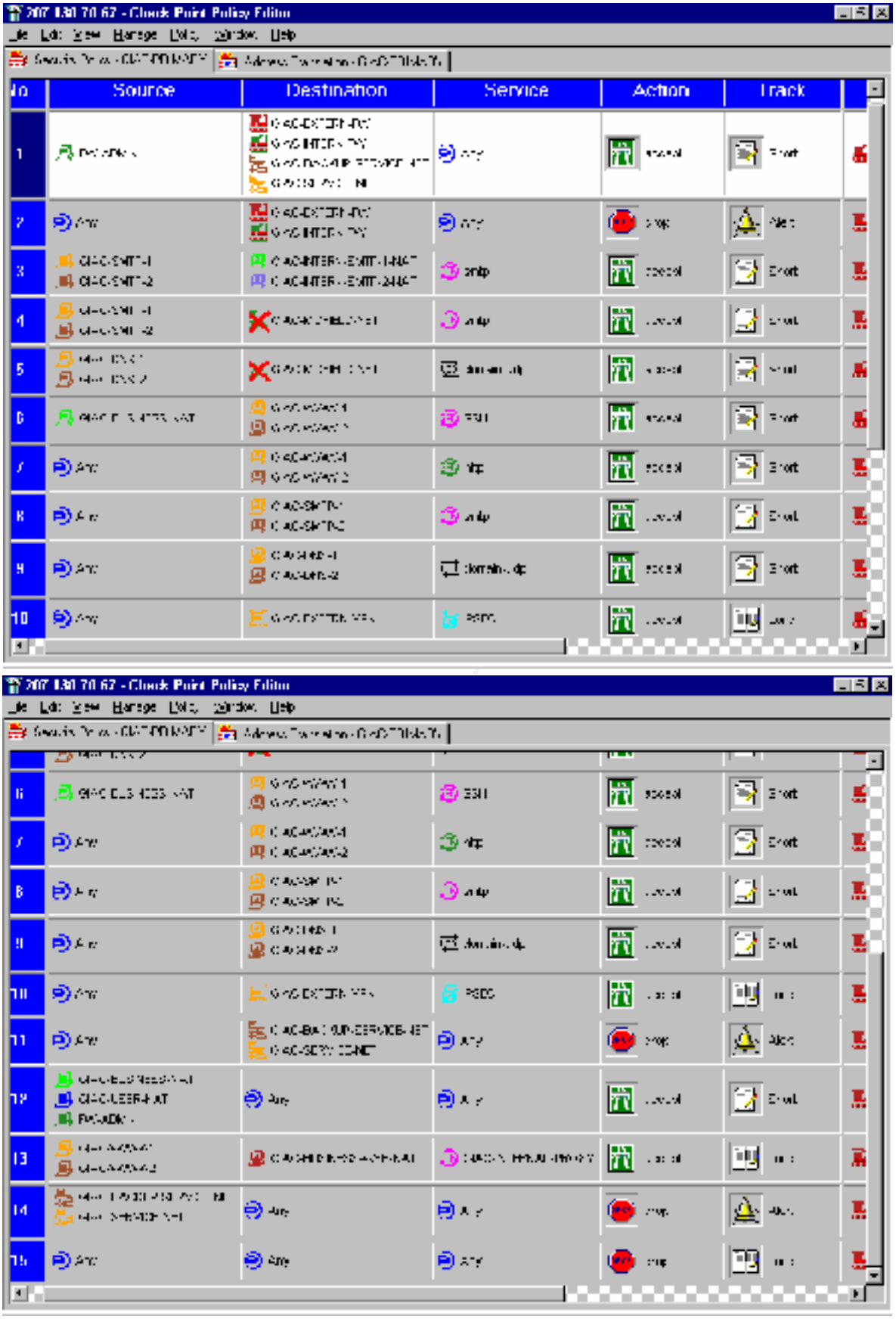
As shown in the accompanying graphical depiction of the Primary Firewall rule base, All of the conditions set forth in the connectivity requirements list in Section 1 are met.

Careful attention has been paid to limit visibility of internal network services by ensuring that internal SMTP and DNS requests are forwarded only to the SMTP and DNS servers in the services networks.

Additionally, access to and from the services networks has been limited only to those services which it has been set up to provide, specifically DNS (UDP only), SMTP, HTTP/HTTPS, and VPN services (AH, ESP, and UDP port 500 only).

Access to the services network from internal user network addresses (including VPN internal addresses) has also been restricted to only those required, namely HTTP/HTTPS, and for web administrators on the business subnet, SSH.
The rules are explained as follows, and testing recommendations and traffic/log samples are shown in Section 3, Architecture Audit:

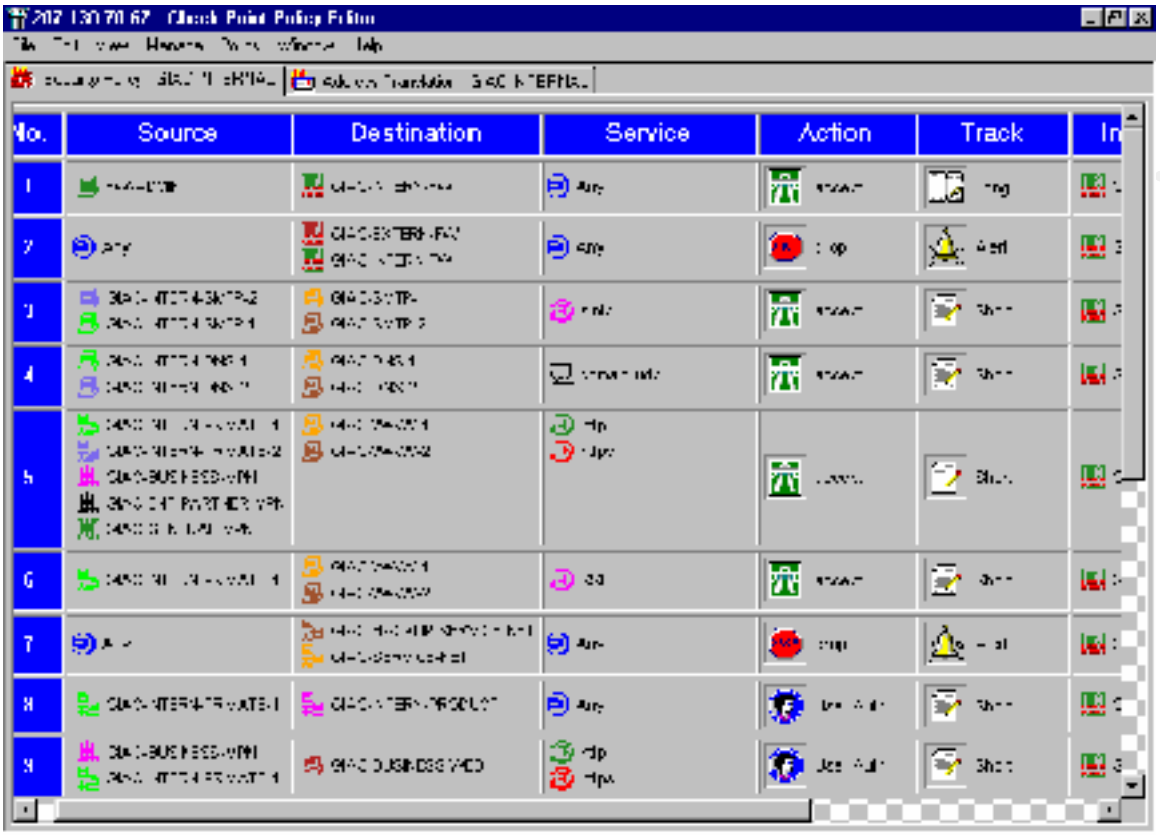
Rule 1 – Only FW-ADMIN may have full access to the services networks, firewall, and router.
Rule 2 – Any other access to the firewalls is denied
Rule 3 – Access from the public mail servers is allowed only for port 25 (SMTP) to the internal mail servers for inbound mail relay
Rule 4 – Access from the public mail servers to any non-internal address is permitted (note the red 'x' which 'negates' the internal midfield net). Because the routing and NAT tables in the firewall do not recognize the private internal addresses, packets for these destinations will no go anywhere. NAT rules are applied only to the internal firewall.
Rule 5 – Access from the public DNS servers is only permitted to the outside world for name lookups (again, note the 'negate' flag). There is no reason for the public DNS servers to initiate connections to the internal DNS servers. They only answer requests FROM the internal DNS servers, who use them as 'forwarders'.
Rule 6 – GIAC Business-subnet users have SSH access to the public web servers. Business needs dictates that this rule includes SSH for terminal access and file transfer, as the web site administrators reside in the business subnet.
Rule 7 –Anyone may have HTTP and HTTPS access to the public web servers.
Rule 8 – Anyone may access the public mail servers on TCP port 25. This rule covers both mail relay connections (outbound) from the internal mail servers, and inbound mail from the outside world.
Rule 9 – Anyone may access the public DNS servers on UDP port 53. Again, the internal servers are configured to use the public DNS servers as forwarders, and anyone in the outside world may use the limited zones listed on the public DNS servers. Again, no TCP port 53 allowed, due to the vulnerable nature of BIND.
Rule 10 – Anyone (authorized) may establish connections to the VPN gateway. Although conceivable, few exploits are known for the Contivity switch, except a little known DoS exploit.
Rule 11 - Any other access TO the services networks is denied.
Rule 12 – Any internal NAT-translated network users and the Firewall Admin may access the Internet. NOTE: This does NOT include VPN users. Access rules for VPN users are further defined on the internal firewall.
Rule 13 – The public web servers may access the internal web content server and business transaction server from their proxy addresses – The details of this functionality are unclear to this writer, and this specific rule will be modified once the final public -> private transaction process has been defined.
Rule 14 – **Any other access FROM hosts on the services networks is forbidden.**
Rule 15 – Any other access through this firewall is denied.

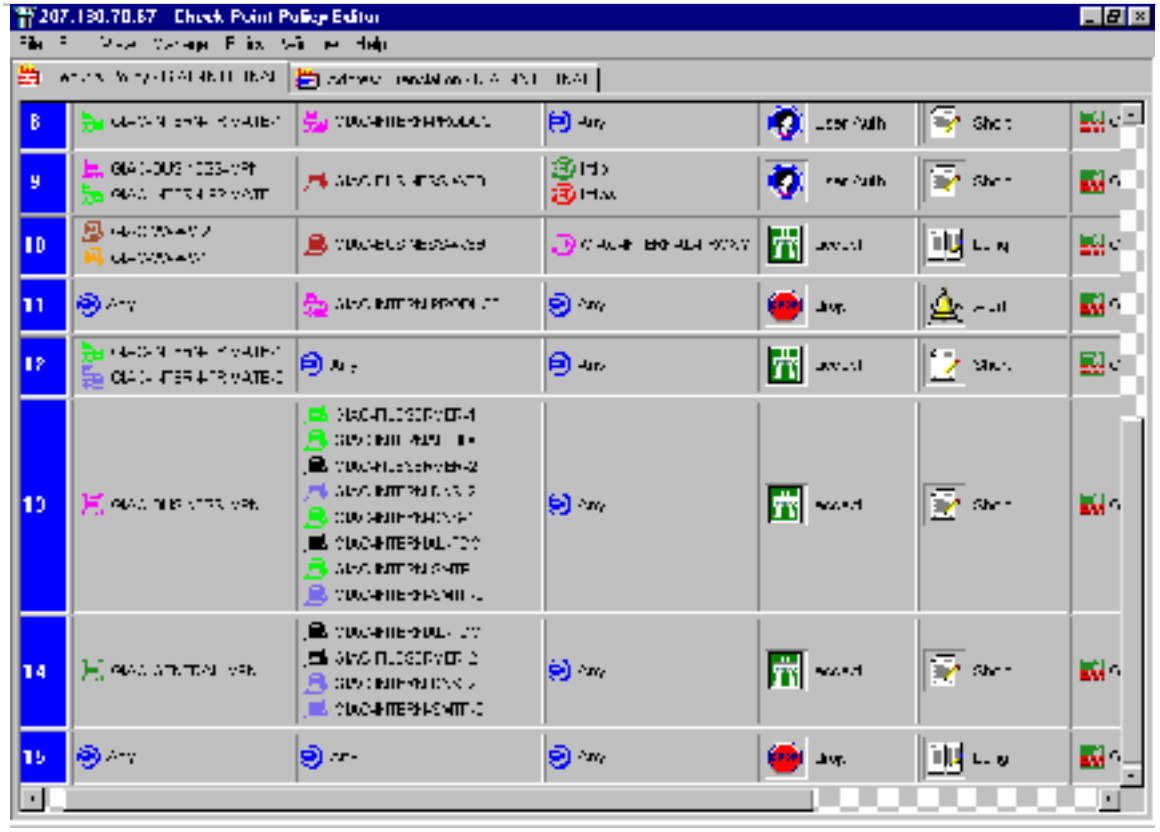


2) Internal (Business Function) Firewall Rule base

Once traffic has been cleaned up by the Primary firewall, the Internal firewall then takes up the process of directing and authorizing traffic to the various subnets, and forwarding any appropriate traffic to the Primary firewall, based upon rules 6, 7, 8, 9, and 12 of its rule base, and forwarding any authorized traffic to internal addresses based upon rules 3, and 13.

Rule 1 – FW-ADMIN may have full access to the Internal Firewall
Note that the FW-Admin host resides on the middlefield net, and thus does not require a rule permitting access to the Internet.
Rule 2 – Any other access to the Internal or Primary firewalls is denied.
Rule 3 – Internal mail servers may access the public mail servers for outbound relay.
Note that because NAT is performed on the Internal firewall, the translated address appears at the internal interface of the Primary firewall.
Rule 4 – Internal DNS servers may access the public DNS servers for unresolvable requests.
Rule 5 – Any internal hosts, including VPN, may access the public web servers for HTTP/HTTPS.
Rule 6 – Internal Business users may access the public web servers for administration using SSH. See Rule 6 on Primary firewall.
Rule 7 – Any other access from the internal network to the services networks is denied.
Rule 8 – GIAC Business users may access the Product and service management subnet.
Rule 9 – Access to the internal product web server is authorized for Business VPN users and GIAC Partners using HTTP and HTTPS.
Rule 10 - Access from the proxy ports on the Public web servers is allowed to the internal web server for transactions and authorized content retrieval.
Rule 11 – Any other access to the Product and service management network is denied.
Rule 12 – Internal users may access the Internet
Rule 13 – GIAC Business VPN users may access domain controllers, file servers, DNS servers, and mail servers on both of the internal user subnets.
Rule 14 – GIAC general VPN users may access the domain controller, file server, DNS server, and mail server on the general user subnet.
Rule 15 – All other access is denied.



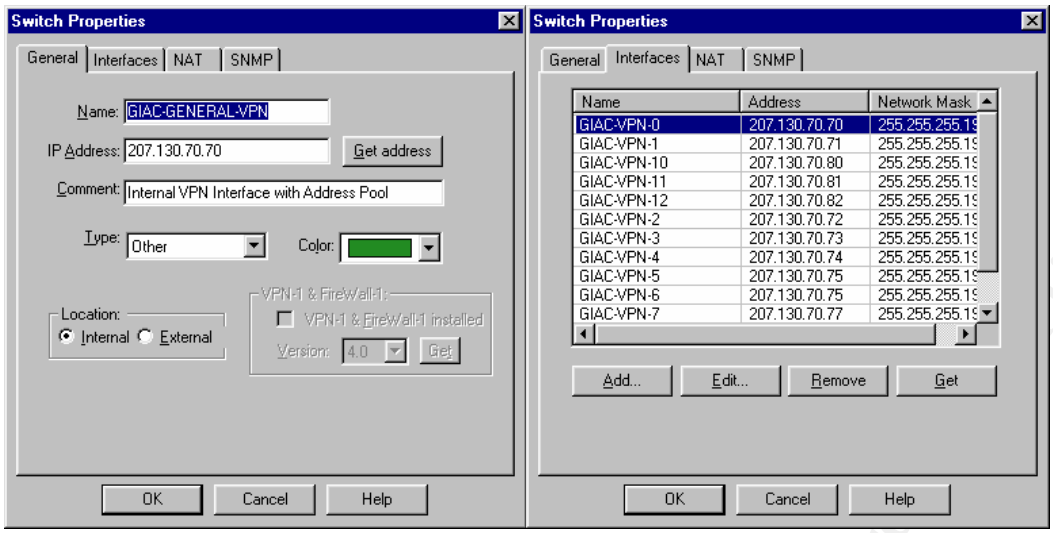


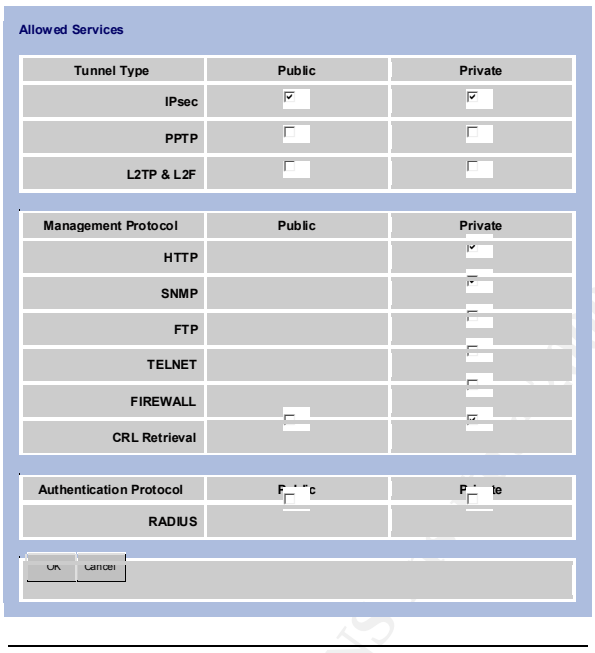
C) VPN Appliance (Contivity 1510) User Profile/Policy Base

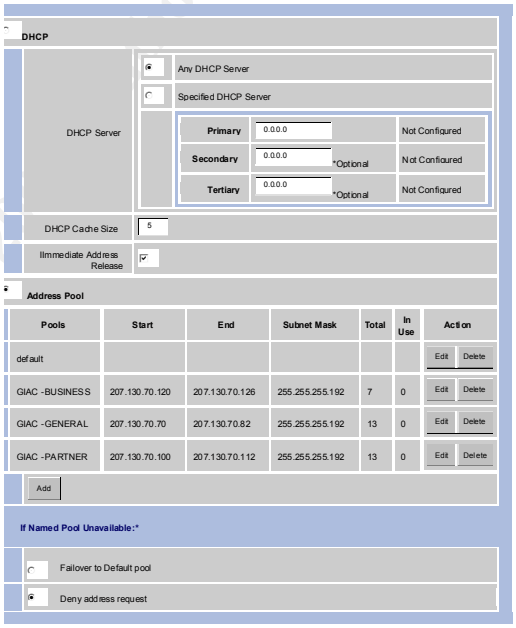
The Nortel Contivity 1510 Extranet Switch was chosen as a VPN appliance for this network due to its configurability and ease of administration. The feature set support by the unit is quite extensive, making it both highly interoperable and scalable.
For this network, the initial configuration consisted of having the unit use its own internal LDAP server for user and group profile storage and retrieval, and initial security parameters, shown below in graphical depictions, consisting of:

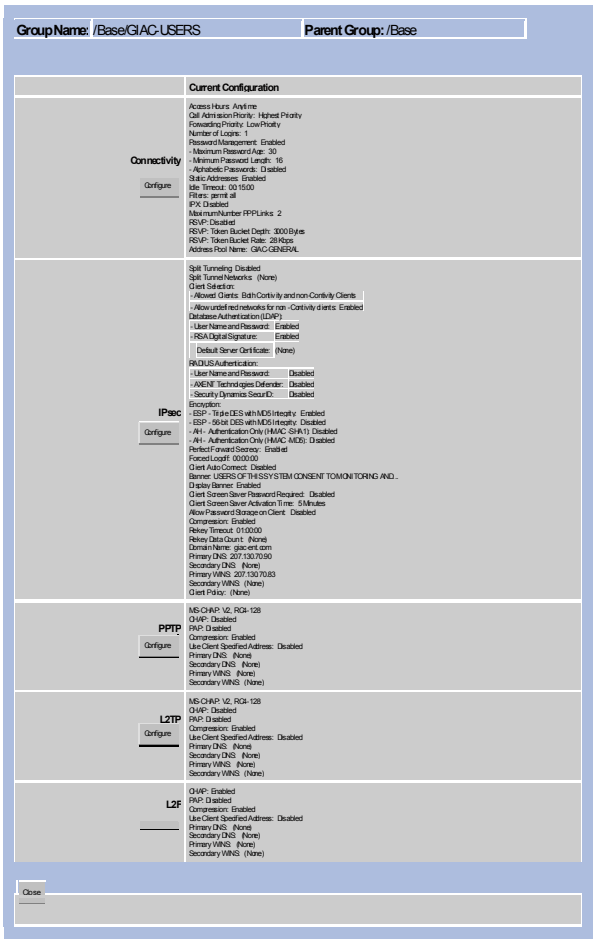
- Password management, supporting password aging and minimum length
- 1 hour mandatory rekey (the factory default is 8 Hours)
- Only IPSec is authorized as a connection protocol. PPTP, L2TP, and L2F are not authorized connection types, and the router and Primary firewall also have ACL's and rules that preclude the use of those other protocols.
- Only 3-DES IPSec encryption, to ensure data confidentiality. Simply using AH for identity and integrity would not be sufficient in a highly competitive environment.
- Configuration of specific address pools for authenticated clients to provide a token by which the firewall can provide access control based upon group and access need.
- Definition of groups with global parameters for that group, which dictate what services they can access, and what addresses they will be issued upon connection.
- Default supplied access lists were used to provide initial control over what services groups can gain access to. Specifically, the GIAC user groups are allowed access to any services, while the Partner group is allowed access only to HTTP and DNS. Again, the Internal firewall provides an additional level of access control based upon the IP address that a member of a group is issued.
- Split Tunneling (Split Horizon) is not allowed for any user, due to the potential security risks involved. Although unlikely, the possibility of the remote user becoming an unwitting gateway into the tunnel is a real one, and this administrator does not wish to undertake that risk. Providing VPN service to non-company entities is risky enough in itself, even with mandatory non-disclosures and acceptable-use policies.

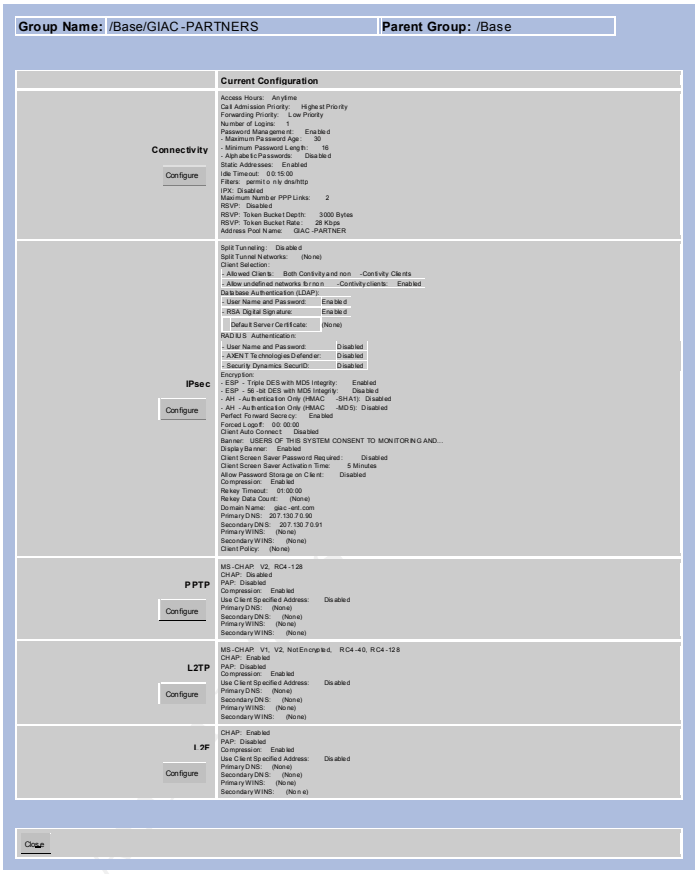
As shown below in a representative firewall object definition, a specific group of addresses is defined for the object "GIAC-GENERAL-VPN". This object has privileges defined under the group "GIAC-USERS" in the VPN device, AND under the rule base of the Firewall as GIAC-GENERAL-VPN.
If the status of a user changes, that user can merely be moved to a different group in the VPN device (in the event of a promotion), or removed entirely, without the need to change the firewall configuration.
Profiles for the other two groups (GIAC-PARTNERS and GIAC-BUSINESS) are also shown below.

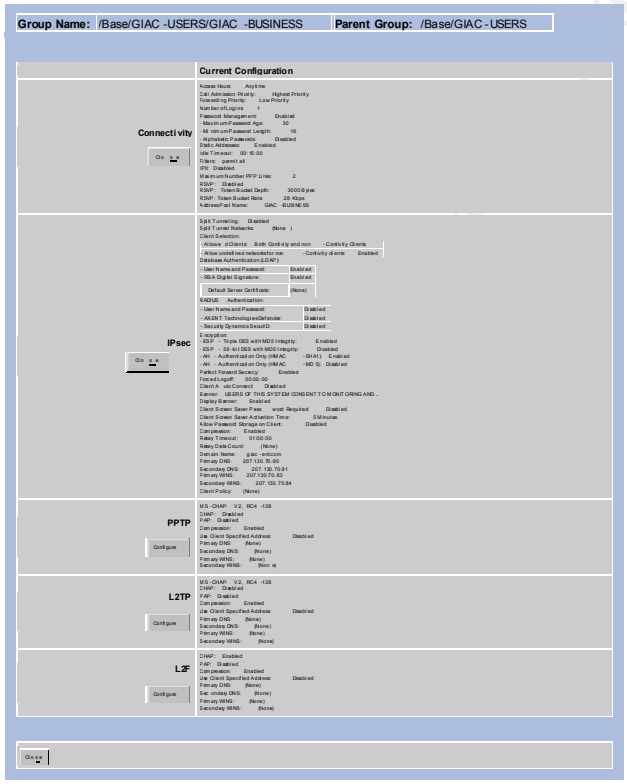












SECTION 3-GIAC ENTERPRISES PRIMARY FIREWALL VULNERABILITY ASSESSMENT AND SECURITY AUDIT

Note: This assessment was undertaken with an initially limited knowledge of tools, techniques, and exploits. All of the tests performed and data gathered were the result of a 'trial by fire', and credit goes to all parties out there who have provided the wealth of knowledge and code that provided a terrific introduction to the fascinating arena of vulnerability assessment and exploit.
These techniques, tools, and results were used on a test network that consisted of 'hot-spares':

- A Cisco 3620 router configured as described above
- A Checkpoint Firewall-1 package using a 30-day evaluation license, running on Linux 7.0
- A discarded Pentium laptop pressed into service as an assessment 'toolkit', running:
    a. Linux 7.0
    b. Assorted public-source assessment code, as described below.
- A Sun Ultra-5 / Solaris 7 workstation configured with up to 64 virtual interfaces to simulate the potential number of hosts on the GIAC Enterprises network segments.
- A Windows NT system to provide management of the firewall
- A suite of servers running on the Solaris unit, consisting of:
    a. JavaWeb server, to simulate the public web server
    b. A version of BIND native to the Solaris platform to simulate the public DNS server
    c. A version of Sendmail, also native to the Solaris platform to simulate the public SMTP server.

This test environment provided an outstanding opportunity to evaluate tools and techniques, and it is highly recommended that a similar environment be used by anyone who wishes to learn.

1) Assessment Plan

The assessment of the GIAC network requires a carefully planned set of objectives and procedures, in order to properly evaluate the security of the network. Although a complete set of test-plan documents is beyond the scope of this paper, sample elements are presented.

a) Rationale/Approach/Objective

The GIAC Enterprises network is, like most other e-commerce entities, potentially a choice target for nefarious entities (read-Crackers and Criminals). As a result of the noteworthy exploits of highly visible networks such as those maintained by Microsoft and Yahoo, the assessment must be undertaken with extreme thoroughness and deliberation.
The objectives are:

i. Assess the border router
- Ensure that the Access Control Lists in the router are properly blocking undesired traffic
- Ensure that the Access Control Lists in the router are properly passing desired traffic
- Ensure that unauthorized access to the router cannot easily be gained, based upon the access control lists defined.
- Ensure that no unauthorized traffic passes FROM the network through the border router

ii. Assess the Primary Firewall
- Ensure that only desired externally initiated traffic reaches only the services networks, specifically HTTP/HTTPS, SMTP, DNS (UDP only), and IPSec (IP protocols 50, 51 and UPD port 500)

- Ensure that NO externally initiated traffic appears on the middlefield or internal networks
- Ensure that only desired traffic initiated from the services network reaches the Internet, specifically SMTP and DNS (UDP) traffic
- Ensure that only desire traffic from the services network reaches the internal networks, specifically SMTP and proxied HTTP.
- Ensure that only desired traffic from the internal networks reaches the services network (HTTP/HTTPS for all, and SSH for business users).
- Ensure that only desired traffic from the internal networks reaches the Internet
- Ensure that any VPN-based traffic does not pass beyond the external interface of the Primary firewall
- Ensure that only desired VPN-based traffic reaches the services network, specifically HTTP/HTTPS

iii. Perform exploit tests against the public servers, router, and firewall.
- Ensure that they are appropriately 'hardened', and the router and firewall are as 'invisible' as possible.

iv. Assess the internal Firewall
The principles applied to the external firewall also apply to the Internal firewall. In addition, tests would be performed to ensure that:
- Authorized users gain appropriate access to resources on designated subnets
- Network Address translation is performed correctly and consistently
- Unauthorized users are denied access at the earliest possible point and at multiple levels
- Internal users gain successfully gain access to public web servers and the Internet

b) Tools/Techniques

To successfully perform this assessment and to provide meaningful results, a comprehensive set of tools must be employed to thoroughly evaluate vulnerabilities and performance. For this assessment, the following tools will be used, beyond those native to an operating system:

- NMAP – A highly useful and popular port scanning tool from www.insecure.org
- TCPDump – A vital tool for traffic analysis during assessments and "other instances" www.tcpdump.org, with credit to LBL
- WINDump – The Windows implementation of TCPDump, available from netgroup-serv.polito.it
- HPING2 – An extraordinarily useful packet crafting tool from Salvatore Sanfilippo.
- NESSUS – A very comprehensive vulnerability assessment tool from www.nessus.org
- Although not used, Netcat, by Hobbit, from http://www.l0pht.com is also a very popular tools for generating packets of a varying types and formats.
- Additional tools to be used, that are native to the operating systems:
    a. PING
    b. Traceroute
    c. SNOOP (Sun's implementation of TCPDump)

The techniques employed should include:

- Basic ICMP assessments to ensure that only desired ICMP traffic types are allowed
- Basic port mapping, using both TCP and UDP packets to check for device conformance to access rules defined, AND to check for **any**

ports which had not been previously considered and included in any defined access rules.

- Enhanced port scanning techniques, generating inappropriately formatted packets to ensure that devices respond appropriately (**or don't respond**) to anomalous traffic.
- Careful monitoring of test traffic both inside AND outside the device or network being tested. Traffic that isn't allowed through a device may still elicit a signature response from that device, thus providing a potential fingerprint for an attacker.
- Traffic generation for authorized services to ensure their appropriate behavior under normal conditions
- Traffic generation for authorized services to ensure their appropriate behavior under **abnormal** conditions.

c) Cost/Effort

A proper vulnerability assessment can be a highly involved undertaking. Therefore, to provide enough time to support a thorough analysis as described above, the following recommendation is made:

- Two personnel will be required to perform this assessment: One to configure and execute tests, and the other to check and tabulate results. Additionally, a 'second set of eyes' is often invaluable in detecting anomalous behavior, and to provide recommendations for additional tests or modifications of existing ones.
- It is expected that a total of 24 hours of test time will be required due to the extensive time require for certain tests, such as large-scale port scans. 4 hours of this time will be used during low-traffic hours in order to perform other tests that could affect network performance.
- Additionally, 6 hours will be required to properly generate a report detailing the results and findings of this assessment.
- At a flat rate of $100 per man-hour, the total cost of this assessment will be $6000.00. In light of GIAC Enterprises reliance upon this network for revenue generation, and in consideration of the evaluator's ability to provide supporting documentation for the procurement of an insurance policy protecting against loss due to theft and malicious activity, GIAC will find this to be a highly reasonable service.

d) Considerations/Risks

An assessment of this nature is, by its nature, potentially disruptive to network performance and availability. It is for this reason that testing will be performed as follows:

- Basic port scanning and access assessments will be performed during normal business hours. These tend to generate little traffic, and thus are not terribly disruptive. Additionally, any non-disruptive system vulnerability checks will be performed during this period.
- High intensity traffic analysis, including Denial of Service vulnerability testing and exploit-based testing will be performed during low-traffic periods.

Additionally, a comprehensive list of systems to be tested, and authorizations to perform all tests will be gained from the **senior executive** of GIAC Enterprises.

2) The Assessment Implementation

The assessment of the GIAC Enterprises network consisted of a number of tests, many of which are outlined briefly in the preceding section. Although the following section is by no means exhaustive, many of the procedures and results are included here.

Tools/Techniques

As described in the previous section, the tools utilized for the assessment consisted of NMAP, HPING2, NESSUS, TCPDump, and WinDump. Excerpts from the complete test plan are described and shown in the following sections, to validate the methods and results necessary for a network assessment.

i. Router Assessment – Spoof Checks

To perform spoof checks against the external router, a test suite was assembled using the following:

- An 'attacking' host outside the network running HPING2 with the following arguments:
  - HPING2 –A –p 80 –s 80 –a **xxx.yyy.zzz**.65 **xxx.yyy.zzz**.200
    This creates a string of TCP packets with the 'ACK' flag set, source starting port and destination ports set to 80 (HTTP), a destination address of the public web server, and a source address which is a legitimate internal address.
- The same attacking host running TCPdump to watch for return packets
- The firewall running its logging process

The router log correctly reported the dropped packets, as shown

```
Apr  7 19:32:49: %SEC-6-IPACCESSLOGP: list 198 denied tcp xxx.yyy.zzz.65(83) ->
xxx.yyy.zzz.200(80), 1 packet
Apr  7 19:32:50: %SEC-6-IPACCESSLOGP: list 198 denied tcp xxx.yyy.zzz.65(84) ->
xxx.yyy.zzz.200(80), 1 packet
```

Additionally, the firewall log showed no instance of these spoofed packets.

ii. Router Assessment – RFC-1918 Blocks

To perform tests validating the router's behavior when handling RFC-1918-style private addresses as specified in the ACL, a variation of the configuration used for Spoofed addresses was employed.

HPING2 –A –f –m 16 –p 80 –s 80 –a 192.168.0.40.130.70.200

In addition to creating TCP packets with the ACK flag set. These packets were also fragmented, resulting in some anomalous behavior.

Source showed:

```
20:15:51.360861 eth0 > 192.168.0.40 > xxx.yyy.zzz.200: (frag 90:4@16)
20:15:52.360241 eth0 > [|tcp] (frag 90:16@0+)
20:15:52.361099 eth0 > 192.168.0.40 > xxx.yyy.zzz.200: (frag 90:4@16)
20:15:53.360219 eth0 > [|tcp] (frag 90:16@0+)
20:15:53.361037 eth0 > 192.168.0.40 > xxx.yyy.zzz.200: (frag 90:4@16)
20:15:54.360220 eth0 > [|tcp] (frag 90:16@0+)
20:15:54.361074 eth0 > 192.168.0.40 > xxx.yyy.zzz.200: (frag 90:4@16)
```

Router Showed:

```
Apr  7 20:16:10: %SEC-6-IPACCESSLOGP: list 198 denied tcp 192.168.0.40(106) ->
xxx.yyy.zzz.200(80), 1 packet
Apr  7 20:16:11: %SEC-6-IPACCESSLOGP: list 198 denied tcp 192.168.0.40(107) ->
xxx.yyy.zzz.200(80), 1 packet
Apr  7 20:16:12: %SEC-6-IPACCESSLOGP: list 198 denied tcp 192.168.0.40(108) ->
xxx.yyy.zzz.200(80), 1 packet
```

Firewall log showed:

```
240 7Apr2001 20:15:58 GIAC-EXTERN-FW control ctl  lo inbound          Virtual
defragmentation error: Timeout (192.168.0.40 -> xxx.yyy.zzz.200 proto 6 id 90 len 0
offset 0) - 3 fragments dropped during the last 60 seconds
241 7Apr2001 20:16:22 GIAC-EXTERN-FW control ctl  lo inbound          Virtual
defragmentation error: Duplicate fragment (192.168.0.40 -> xxx.yyy.zzz.200 proto 6
id 90 len 24 offset 16) - 58 fragments dropped during the last 60 seconds
```

Note that this is desirable behavior, but may create a period of vulnerability while the firewall waits for additional fragments.
How the firewall behaves when attempting to process high volumes of incomplete fragments is worthy of consideration.

TCPDump on the firewall showed:

```
20:15:58.813070 eth0 < 192.168.0.40 > xxx.yyy.zzz.200: (frag 90:4@16)
20:15:59.808515 eth0 < 192.168.0.40 > xxx.yyy.zzz.200: (frag 90:4@16)
20:16:00.808711 eth0 < 192.168.0.40 > xxx.yyy.zzz.200: (frag 90:4@16)
20:16:01.808823 eth0 < 192.168.0.40 > xxx.yyy.zzz.200: (frag 90:4@16)
20:16:02.809005 eth0 < 192.168.0.40 > xxx.yyy.zzz.200: (frag 90:4@16)
```

Note that only the terminal fragment is received by eth0, thus demonstrating that the router did indeed drop the initial fragment based upon the complete IP header. However, there is significant cause for concern here, as the router ACL's clearly do not process packets solely based upon IP Address, but in the case of TCP, by initial flag (SYN, ACK, etc.) as well.

iii. Router and Firewall Assessment – Unauthorized Services

For this test, NMAP was used to perform a large-scale port scan for any available TCP-based services. Arguments to NMAP were:

nmap -sT -P0 -F --max_parallelism 100 -oN /tmp/nmap-rtr4 **xxx.yyy.zzz**.192/27

–sT attempts to establish a full TCP connection,
-P0 prevents initial ping scans of the hosts (thus saving time),
-F establishes 'Fast Mode, where only the ports known to NMAP are scanned, instead of all 65,535 possible ports. (That test was performed against the router itself, and took many hours)
--max_parallelism 100 attempts to support up to 100 concurrent connection attempts, again with intent of speeding the scan up.
-oN /tmp/nmap-rtr4 redirects NMAP's default output to file nmap-rtr4
- **xxx.yyy.zzz**.192/27 sets NMAP to scan all hosts in that subnet.

A sample of the output is shown below:

```
# Nmap (V. nmap) scan initiated 2.53 as: nmap -sT -P0 -F --max_parallelism 100 -oN
/tmp/nmap-rtr4 xxx.yyy.zzz.192/27
All 1062 scanned ports on  (xxx.yyy.zzz.192) are: filtered
All 1062 scanned ports on  (xxx.yyy.zzz.193) are: filtered
All 1062 scanned ports on  (xxx.yyy.zzz.194) are: filtered
All 1062 scanned ports on  (xxx.yyy.zzz.195) are: filtered
All 1062 scanned ports on  (xxx.yyy.zzz.196) are: filtered
All 1062 scanned ports on  (xxx.yyy.zzz.197) are: filtered
```

```
All 1062 scanned ports on  (xxx.yyy.zzz.198) are: filtered
All 1062 scanned ports on  (xxx.yyy.zzz.199) are: filtered
Interesting ports on  (xxx.yyy.zzz.200):
(The 1061 ports scanned but not shown below are in state: filtered)
Port       State       Service
80/tcp     open        http

All 1062 scanned ports on  (xxx.yyy.zzz.201) are: filtered
All 1062 scanned ports on  (xxx.yyy.zzz.202) are: filtered
All 1062 scanned ports on  (xxx.yyy.zzz.203) are: filtered
All 1062 scanned ports on  (xxx.yyy.zzz.204) are: filtered
All 1062 scanned ports on  (xxx.yyy.zzz.205) are: filtered
All 1062 scanned ports on  (xxx.yyy.zzz.206) are: filtered
```

As expected, no hosts responded to this port scan, except for TCP port 80, which does show up as OPEN on host **xxx.yyy.zzz**.200. This is correct, as we do wish to allow access to the public web server's HTTP port to anyone.
Note that TCP port 53 did not show up as OPEN for host **xxx.yyy.zzz**.199, whereas UDP port 53 does in a similar scan for UDP services.
This test correctly verifies the functionality of the router ACL's with respect to the services network.

Here, an attempt was made to connect to TCP port 25 on an unauthorized host using HPING2 as shown:

```
hping2  -S  -p 25 xxx.yyy.zzz.193
      (set the SYN flag for a packet bound for port 25 of xxx.yyy.zzz.193)

12:46:04.860285 eth0 > host.attacker.com.1215 > xxx.yyy.zzz.193.smtp: S
774223134:774223134(0) win 512
12:46:05.860265 eth0 > host.attacker.com.1216 > xxx.yyy.zzz.193.smtp: S
688933487:688933487(0) win 512
12:46:06.860208 eth0 > host.attacker.com.1217 > xxx.yyy.zzz.193.smtp: S
2089320994:2089320994(0) win 512

Apr  9 12:45:34: %SEC-6-IPACCESSLOGP: list 198 denied tcp host.attacker.com(1216) -
> xxx.yyy.zzz.193(25), 1 packet
Apr  9 12:45:35: %SEC-6-IPACCESSLOGP: list 198 denied tcp host.attacker.com(1217) -
> xxx.yyy.zzz.193(25), 1 packet
Apr  9 12:45:36: %SEC-6-IPACCESSLOGP: list 198 denied tcp host.attacker.com(1218) -
> xxx.yyy.zzz.193(25), 1 packet
```

In this case, no packets were seen via TCPDump running on the firewall. Thus, the router dropped the packets appropriately.

In the case of the Internal networks, no ports on any hosts should appear in state OPEN, as the firewall will only allow packets through that belong to an existing connection, as shown in the next test sample:

iv. Router and Firewall Assessment – Authorized Connections

For a connection to the public mail server, we would expect to see the packet accepted by both the router and the firewall, resulting in return packets to HPING2 as follows:

```
[root@localhost /tmp]# hping2  -S  -f -m 20 -p 25 xxx.yyy.zzz.222
```

```
eth0 default routing interface selected (according to /proc)
HPING xxx.yyy.zzz.222 (eth0 xxx.yyy.zzz.222): S set, 40 headers + 0 data bytes
46 bytes from xxx.yyy.zzz.222: flags=SA seq=0 ttl=252 id=32766 win=9112 rtt=6.6 ms
46 bytes from xxx.yyy.zzz.222: flags=SA seq=1 ttl=252 id=32767 win=9112 rtt=4.4 ms
46 bytes from xxx.yyy.zzz.222: flags=SA seq=2 ttl=252 id=32768 win=9112 rtt=6.2 ms
46 bytes from xxx.yyy.zzz.222: flags=SA seq=3 ttl=252 id=32769 win=9112 rtt=4.4 ms
```

When such a packet is allowed all of the way through both the router and the firewall, an instance of the connection should appear in the firewall's state table, as shown for an accepted connection to the public DNS server on UDP port 53.

```
                        ---- FW-1 CONNECTIONS TABLE ---

Src_IP      Src_Prt     Dst_IP      Dst_Prt     IP_prot     Kbuf  Type  Flags
      Timeout

host.attacker.com 2992  xxx.yyy.zzz.199   53    17    0     16386 ffffff00    40/40
host.attacker.com 2988  xxx.yyy.zzz.199   53    17    0     16386 ffffff00    36/40
host.attacker.com 2989  xxx.yyy.zzz.199   53    17    0     16386 ffffff00    37/40
```

Anomalous packets should never appear in the state table, which was NOT the case with previous versions of Firewall-1.

v. Router and Firewall Assessment - Anomalous Packets

For this test, a configuration using HPING2 as above, TCPDump, and use of the firewall logs was implemented. The arguments to HPING2 consisted of:

HPING2 –S –f –m 16 –p 80 **xxx.yyy.zzz**.200

This will send a series of 16 byte fragmented packets with the SYN flag set to TCP port 80 of the public web server.

Because the router ACL's are not configured to log legitimate WWW traffic, no entries relating this packet were found.
However, the firewall did correctly report the following:

```
215 7Apr2001 19:49:00 GIAC-EXTERN-FW log drop  eth0 inbound tcp host.attacker.com
GIAC-WWW-1 http 94  0 TCP packet too short
216 7Apr2001 19:53:04 GIAC-EXTERN-FW log drop  eth0 inbound tcp host.attacker.com
GIAC-WWW-1 http http  0 TCP packet too short
```

In another instance, packets were crafted similarly, except that the ACK flag was set, with the following firewall log output:

```
196 7Apr2001 19:48:13 GIAC-EXTERN-FW log drop  eth0 inbound tcp host.attacker.com
GIAC-WWW-1 http 141  0 unknown established TCP packet
197 7Apr2001 19:48:14 GIAC-EXTERN-FW log drop  eth0 inbound tcp host.attacker.com GIAC-WWW-1 http 142  0
unknown established TCP packet
```

This is clearly desirable behavior, as the firewall does indeed catch and drop fragments from non-SYN fragments destined for legitimate ports, which the router is incapable of doing.

In the case of UDP fragments, the previous behavior does not apply to the router.
Here, fragmented, oversized DNS packets are sent to an unauthorized host, and all fragments appear to have been caught by the router.

```
hping2  -2  -f -m 140 -d 250 -p 53 xxx.yyy.zzz.200
```

TCPDump Output from source host:

```
12:58:54.350133 eth0 > host.attacker.com.1043 > xxx.yyy.zzz.200.domain: 22616
updateDA [b2&3=0x5858] [22616a] [22616q] [22616n] [22616au] (132) (frag 79:140@0+)
12:58:54.351062 eth0 > host.attacker.com > xxx.yyy.zzz.200: (frag 79:118@136)
12:58:55.350131 eth0 > host.attacker.com.1044 > xxx.yyy.zzz.200.domain: 22616
updateDA [b2&3=0x5858] [22616a] [22616q] [22616n] [22616au] (132) (frag 79:140@0+)
12:58:55.351098 eth0 > host.attacker.com > xxx.yyy.zzz.200: (frag 79:118@136)
```

Log Output from router:

```
Apr  9 12:58:28: %SEC-6-IPACCESSLOGP: list 198 denied udp host.attacker.com(1049) -
> xxx.yyy.zzz.200(53), 1 packet
Apr  9 12:58:29: %SEC-6-IPACCESSLOGP: list 198 denied udp host.attacker.com(1050) -
> xxx.yyy.zzz.200(53), 1 packet
```

No errors appeared in the firewall log, and no instances of the packets appeared in the firewall state table, nor did any of the fragments appear in TCPDump output on the firewall itself.

In this test, HPING was used to generate spoofed source-address packets with ACK set, and fragmented them using the arguments:

HPING2 –A –f –p 80 –s 80 –a **xxx.yyy.zzz**.65 **xxx.yyy.zzz**.200

The router log correctly reported the dropped packets, as shown

```
Apr  7 19:34:49: %SEC-6-IPACCESSLOGP: list 198 denied tcp xxx.yyy.zzz.65(83) ->
xxx.yyy.zzz.200(80), 1 packet
Apr  7 19:34:50: %SEC-6-IPACCESSLOGP: list 198 denied tcp xxx.yyy.zzz.65(84) ->
xxx.yyy.zzz.200(80), 1 packet
```

However, the router did NOT drop all elements of the source packet, as shown by the firewall log:

```
125 7Apr2001 19:35:00 GIAC-EXTERN-FW control ctl  lo inbound          Virtual
defragmentation error: Duplicate fragment (xxx.yyy.zzz.65 -> xxx.yyy.zzz.200 proto
6 id 58 len 24 offset 16) - 12 fragments dropped during the last 60 seconds
126 7Apr2001 19:35:22 GIAC-EXTERN-FW control ctl  lo inbound          Virtual
defragmentation error: Timeout (xxx.yyy.zzz.65 -> xxx.yyy.zzz.200 proto 6 id 58 len
0 offset 0) - 1 fragments dropped during the last 60 seconds
```

This test validates the expected behavior of the router with respect to Spoofed source Packets, but adds two extra pieces of valuable information:

1 – The router only drops the initial fragment of a TCP packet containing the full TCP header
2 – The firewall correctly handles packets without their initial fragments.

vi. Additional Tests Performed

For educational purposes, some additional tests were performed that would also be run against a network in a formal assessment. With the use of the NESSUS vulnerability assessment tool,
the following tests were performed:

- Root Shell vulnerability scans against the SPARC station in the services network
- Web and CGI exploits against the public web server itself.
- SMTP vulnerability scans against the public mail server itself
- Denial of Service attacks against the screening router

Although the test results shown above do not include tedious, exhaustive output from all possible tests against all possible hosts, the tools and techniques would be applicable to any element of the network.

For comprehensive internal testing, the 'attacker' would be placed between the router and Primary firewall for a group of tests against the Services and Internal networks, and additional tests would be performed with the attacker in the 'middlefield' network, verifying the rule sets on both the Internal and External firewalls.
Additional tests would be performed from inside each of the private networks to verify access as authorized to each of the other subnets in the GIAC network.
Finally, tests would be performed as a VPN client in each of the VPN groups to verify appropriate access to the GIAC internal networks.

1) The Findings

The results of the tests performed indicate that largely, the network is secure.
As with any assessment, it must be noted that indications of current security levels cannot serve as a prediction of future security postures. Current threats, because they are known to reasonable degree, cannot be used wholly as a basis for the prediction of and protection against future threats.
With this caveat in mind, this analyst found the following:

a. Strengths

The network perimeter was found to be quite secure, and provided good protection against a variety of conventional and unconventional traffic types.
The multi-layered defense consisting of a comprehensive screening router Access Control List, followed by a Primary firewall prevented a great number of anomalous data packets through, while also significantly limiting entry points through which exploits could be launched. The use of the current version of CheckPoint Firewall-1 provides suitable protection against traffic types that were formerly not denied in previous versions of this product.
The use of a combination of the VPN client group address assignments, firewall access control based upon those addresses, and user authentication presents an acceptable means of controlling access to the critical business systems subnetwork.

b. Vulnerabilities

- The tendency of the screening router to incompletely drop fragmented packets is of some concern. Granted, this is a known behavior in Cisco IOS, and the firewall residing behind this router provides an additional measure of protection against such traffic.
However, it is possible that the handling of large numbers of such fragments may require the use of excessive processor cycles within the firewall, potentially resulting in a Denial or Degradation of Service through that device.
- The lack of a multi-homed posture (an alternate gateway to the Internet) could also result in a Denial of Service condition in the event of a link failure.
- The lack of redundancy with respect to the firewalls could also present a Denial of Service condition in the event of a failure.
- Based upon the rule sets provided to this analyst, the inability of the FW-ADMIN host to gain access to internal sub-networks could present a problem. It is assumed that the owner of that system has alternate systems that reside within one or more of those sub-networks, from which to perform any administrative task required in them.

- The current scheme for backups, logging, and Intrusion detection, whose data are transported via secondary interfaces on those machines residing on the services networks, is acceptable, despite the lack of documentation for them. **It is to be assumed that these servers do exist, and it is recommended that the network diagram and description be updated to reflect this**. Should one of those hosts be compromised, however, any data residing on the backup server or main IDS-system storage device would be considered suspect.
- The lack of time synchronization on network hosts and devices could present a problem when correlating access logs, traffic logs, and other data.

c. Recommendations

In light of these findings, the following recommendations are made:

- Investigate alternate routing/switching devices that better handle fragmented packets.
- Establish a multi-homed posture as soon as is practicable. This should use BGP-4 with strong authentication to peering networks to reduce the likelihood of malicious route manipulation.
- Create a load-balanced or simple redundant scheme for the Primary firewall, as a minimum. Redundancy of both firewalls would be a better solution, but it is understood that cost may be a factor in this instance.
- Establish a Network Time Protocol server to serve as a master time base for all hosts and network devices. This should use one of the GPS or Atomic Clock-based reference servers available on the Internet as its primary reference.
- Perform Full Daily backups on key servers and Intrusion detection devices residing on the services network, and perform periodic checks on the integrity of these backups to determine a suitable media rotation schedule.

SECTION 4-VULNERABILITY ASSESSMENT AND ATTACK UPON THE NETWORK DESIGN OF HEATHER BARD < http://www.sans.org/y2k/practical/Heather_Bard_GCFW.doc >

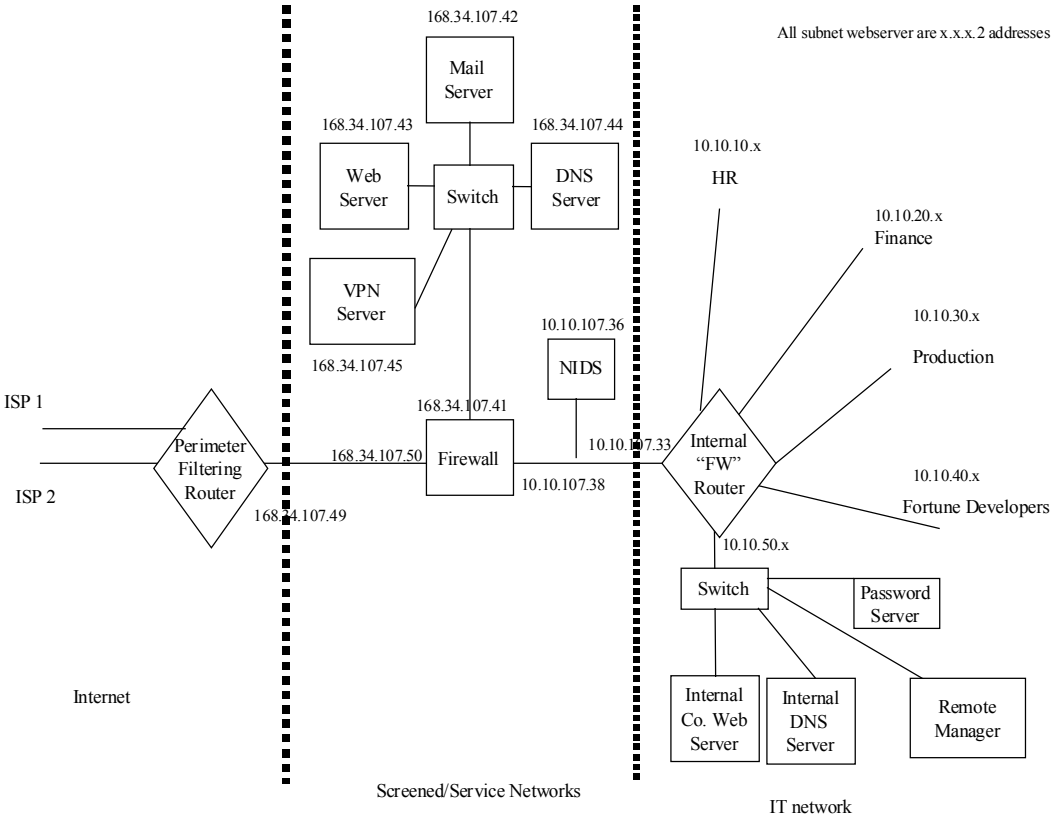
In order to exercise a successful attack upon a network, a large amount of preparatory work must be performed in order to determine where any vulnerabilities may lie.

The target network must be assessed using some simple 'discovery' techniques as follows:

1) Perform a simple 'ping' sweep to see if any hosts respond to ICMP echo request packets, thus advertising their existence. If ICMP echo-requests are blocked at the site, then other techniques must be used for initial scanning.
2) Perform a somewhat more subtle scan using TCP packets whose source port is 0 (Null) and whose flag bits are set to ACK. These are sent to a common port or range of ports to determine if any hosts are alive. If they are, they should respond with a RESET/ACK packet.
   A number of tools exist to perform such scans, including **nmap (www.insecure.org), netcat (by Hobbit),** a very impressive tool called **hping2**, which has the ability to craft packets in a variety of ways.
3) Alternatively, if this results in limited or no success due to a firewall or packet filter, another means of scanning beyond the packet filter is by using fragmented packets.
4) If a firewall or packet filter is suspected to be 'in the way', the type of firewall or filter can be determined by sending packets to known control ports on the device. These ports, ironically, are RARELY BLOCKED, and can give telltale clues about the device protecting the network. For example:
   a. TCP Ports 1999, 2001, 4001, 6001, and 9001 are used by Cisco routers, and will respond in some fashion.
   b. TCP ports 256-258 are used by CheckPoint's Firewall-1
   c. Other devices listen on ports 1080, 1745, and UDP port 161 (SNMP)

5) Once a few hosts have been discovered, additional discovery of operating system type and other open ports can be determined with a good degree of reliability by using **nmap** or **hping2.**

In the diagram below, which depicts the network design by Heather Bard from her GCFW practical of November 21, 2000 < http://www.sans.org/y2k/practical/Heather_Bard_GCFW.doc>, a number of theoretical attacks are to be performed. A good deal of research was performed to determine vulnerabilities of the router and firewall. In the 'real' world, the discovery process described above would have to be performed in order to determine the system types, and then vulnerability research would be performed prior to an actual exploit.



1) Attack on the Firewall

The objective in this instance is to either subvert or disable the site's firewall and/or border router. According to information found at www.securityfocus.com, the Raptor firewall used in this practical may have following vulnerabilities due to known bugs in certain versions as shown by the following postings to the BugTraq database:

a) It is possible to remotely lock Axent Raptor firewalls by sending them packets with malformed IP options fields. According to an advisory posted to bugtraq by the Purdue CERIAS labs, setting the SECURITY and TIMESTAMP IP options length to 0 can cause an infinite loop to occur within the code that handles the options (resulting in the software freezing). A consequence of this is a remote denial of service.

Axent has released a hotfix for this problem which is available at: ftp://ftp.raptor.com/pub/patches/V6.0/hotfixes.post602/dom/DOS-attack/ The 6.02 upgrade is available at:

ftp://ftp.raptor.com/pub/patches/V6.0/6.02Patch/dom/

< http://www.securityfocus.com/bid/736 >

b) Raptor Firewall is a product distributed and maintained by Axent Technologies, Inc. Raptor is an Enterprise-level firewall, providing a mixture of features and performance.

A problem in the software package could allow intruders access to private web resources. By using the nearest interface of the firewall as a proxy, it is possible to access a system connected to the other interface of the firewall within TCP ports 79-99, and 200-65535. The firewall will only permit connections to the other side on ports in this range, excluding port 80, and using HTTP. This affects firewall rules that permit HTTP traffic.

Therefore, it is possible for a malicious user to access internal web assets, and potentially gain access to sensitive information. It is also possible for an internal user to gain access to external web resources through the firewall, providing the resources are not running on the default port 80.

Attacker configures browser to use IP address of Raptor firewall as HTTP Proxy, then begins probing internal network.

The following workarounds are possible:

1. Use httpd.noproxy in the affected rule.

2. Downgrade to version 6.0.2

Additionally, patches are available:

Axent Raptor 6.5:

Axent hotfix SG6500-20000920-00 and SG6500-20001121-00
ftp://ftp.axent.com/pub/RaptorFirewall/Patches/6.50/Internal/http-int.zip

< http://www.securityfocus.com/bid/2517 >

Because it is unknown which version of the Raptor firewall the victim site is using, attempts to exploit these vulnerabilities would be attempted using the methods described above.

2) DDoS Attack

In this instance, the objective is to execute a Distributed Denial of Service attack via 50 xDSL/Broadband access connection sites. These are rapidly becoming the connection method of choice for small businesses, home offices, and 'recreational' Internet access.
Not surprisingly, a very large number of these connections are unprotected, and can be easily used to redirect data to one single site, thus potentially reducing its accessibility or blocking it altogether.
Some very simple techniques are:

3) Forging ICMP packets using the source address of the target site and sending them to many unwitting relay sites, which will then send reply packets of some kind to the target network or host. This is not a terribly effective means by itself, as the potential volume of packets send from the relay hosts is low. A means of improving this is to increase the size of the ICMP

packet sent, and running the attack from multiple processes to increase the volume of packets to be relayed.

4) Again, forging packets for relay as above, but using the broadcast address of the relay hosts to 'amplify' the response. The method here relies upon the attacker being able to find multiple sites and perform an ICMP subnet mask request for each network, and then calculating the broadcast address for that network. Subsequently, one packet sent to the broadcast address should elicit responses from every host that is alive on the network. This assumes, of course, that any intermediate routers or network devices allow such broadcast traffic. One by one, however, candidate sites can be discovered by first:
   i. Sending ICMP 'address mask request' (ICMP type 17/code 0) packets to a variety of Cable and xDSL sites whose address blocks are known. Then listen for replies using TCPDump to decode any replies to the netmask request.
   ii. Calculate the possible broadcast addresses for those networks
   iii. Sending ICMP or other packets to their broadcast addresses to determine which ones will provide multiple responses. This attack is known as the 'smurf' attack, and relies upon candidate 'amplifiers' in order to be successful. Current security practices are helping to reduce the incidence of such sites, however.

   Once a determination has been made about which sites can be used as Smurf amplifiers, begin the attack by sending packets whose source address has been forged to be that of the victim network (i.e., Heather Bard's). While this is underway, check to see if the site has been affected by the hail of replies.
5) Another method of DDoS is to craft packets that exploit vulnerabilities in specific vendor's implementations of TCP/IP, such as by sending malformed packets or packets with flags or options set that the host's IP stack doesn't know how to handle.
6) Ideally, a set of 'zombies' or Trojan servers such as those for Stacheldrat (ftp://ftp.technotronic.com/denial/stachel.tgz) would be sent, via 'spam' mail, to a large number of unprotected candidate sites, who may unwittingly open the attachment and implant the program onto their computer. Once this mail has been sent, scanning would begin to determine which sites had become servers, and then the process of beginning the DDoS attack would begin.

3) Host attack

Due to the number of exploits to Sendmail, this has been chosen as a possible host exploit to gain initial entry into the site. A quick telnet to port 25 of the mail server should elicit a reply with the version of mail daemon that is running, and from there, exploitation could begin. Note that her architecture does not include POP-3 mail retrieval from outside the network.

Once the version of SMTP server has been determined, a look through the FTP archives at www.technotronic.com for appropriate exploits would be performed.

a) If the daemon is Sendmail running on a UNIX server, then the version number would be used to select an exploit from ftp://ftp.technotronic.com/unix/sendmail/ . A terribly convenient list of exploits is available under the 'misc' subdirectory of that directory tree.

b) If the daemon is running on a Microsoft platform, the most likely exploit would be to perform a buffer overflow attack in order to gain the ability to execute commands remotely.

Alternatively, a means of having the mail server send one of its own files back to the attacker would be preferred.

A couple of objectives are considered here:

a) If possible, compromise the mail host by gaining a remote shell, and then use that point of entry to leapfrog into the network, again using SMTP-based messages that have been configured to appear as if they were originating from the mail server itself. If successful, attempt to include a Trojan attachment to create a back door into a host on the internal network.

b) Attempt to gain copies of critical system files that could then be used to prepare other exploits.

If exploitation of the mail server cannot be performed, then attempts would be made to exploit the web server. Again, determinations would be made about the version of web server that is running by connecting via telnet to port 80, and seeing what version of web server is running.
Once determined, an appropriate exploit would be chosen, likely 'directory traversal' attempts to gain system file information, or buffer overflow attempts to gain the ability to execute programs remotely.

NOTE: This writer has no experience with attacks such as those described here, and knows only of their existence through the various security alert newsletters such as those provided by CERT and SANS. Therefore, it will be understood if scoring of this section is thus jeopardized.

c) REFERENCES

The following sources were used in the preparation of this paper, with due and grateful credit to all indirect contributors. Without the resources detailed here, this paper would not have been possible.

BOOKS

Scambray, J; McClure, S; Kurtz, G. *Hacking Exposed – Network Security Secrets and Solutions*, 2nd Edition. Berkeley, California: Osborne/McGraw Hill, 2001

Northcutt, S; Cooper, M; Fearnow, M; Frederick, K. *Intrusion Signatures and Analysis.* Indianapolis, IN: New Riders Publishing, 2001

Northcutt, S; Novak, J; McClachlan, D. *Network Intrusion Detection, 2nd Edition.* Indianapolis, IN: New Riders Publishing, 2001

Novak, J, et al. *TCP/IP for Intrusion Detection and Firewalls.* The SANS Institute, 2000

Spitzner, L; Brenton, C; Winters, S; Northcutt, S. *Advanced Perimeter Protection and Defense in Depth.* The SANS Institute, 2000

RESOURCES AVAILABLE VIA THE WORLD WIDE WEB

Spitzner, L. *Understanding the Firewall-1 State Table.*
< http://www.enteract.com/~lspitz/fwtable.html >

Lopatic, T.; McDonald, J.; Song, D. *A Stateful Inspection of FireWall-1.*
< http://www.dataprotect.com/bh2000/ >

Winters, S. *Securing the Perimeter with Cisco IOS 12 Routers*
< http://www.sans.org/y2k/practical/Scott_Winters_gcfw.doc >

How To Eliminate The Ten Most Critical Internet Security Threats; The Experts' Consensus
< http://www.sans.org/topten.htm >

SOFTWARE TOOLS

TCPDump - < http://www.tcpdump.org/ >
Requires libpcap Packet Capture Library, also found at < http://www.tcpdump.org/ >

WINDump -< http://netgroup-serv.polito.it/windump/ >
Requires winpcap Packet Capture Driver found at: < http://netgroup-serv.polito.it/winpcap/ >

NMAP - < *www.insecure.org* >

NESSUS - < *www.nessus.org* >

HPING2 - < *http://www.kyuzz.org/antirez/hping/* >

FWTABLE – A Perl script used for the capture of the Firewall-1 state table. This tool was developed primarily by Lance Spitzner, and can be found at - < http://www.enteract.com/~lspitz/fwtable_1.1.txt >

APPENDIX – 1 Checkpoint Firewall-1 Network Object Definitions

The following network object definitions are included here to provide detail to the network design and security policy enforcement rules described for the GIAC Enterprises Network. They are excerpts from the actual "Inspect" code that is generated by Firewall on for each rule base.

The objects are divided here into two sections:

The Primary Firewall Objects
and
The Internal Firewall Objects

Although the actual firewall rule base code contains Network Address Translation information for any objects that have NAT definitions set, NAT details are only included for heading Internal Firewall Objects, as it is on that firewall that the NAT rules are applied.
Additionally, each set of rule base code refers only to those objects that are relevant to its own sets of rules.

The Primary Firewall Objects

ADDR_net(giac-extern-fw-net-if0, **xxx.yyy.zzz**.0, 255.255.255.224)
ADDR_net(giac-extern-fw-net-if2, **xxx.yyy.zzz**.192, 255.255.255.224)
ADDR_net(giac-extern-fw-net-if3, **xxx.yyy.zzz**.224, 255.255.255.224)
ADDR_host(fw-admin, **xxx.yyy.zzz**.68)
ADDR_hostif(fw-admin-if1, **xxx.yyy.zzz**.69)
ADDR_hostif(fw-admin-if2, 10.128.0.38)
ADDR_network(giac-midfield-net, **xxx.yyy.zzz**.64, 255.255.255.192)
ADDR_network(extern-dmz, **xxx.yyy.zzz**.0, 255.255.255.224)
ADDR_net(giac-intern-fw-net-if0, 10.15.21.0, 255.255.255.0)
ADDR_net(giac-intern-fw-net-if1, **xxx.yyy.zzz**.64, 255.255.255.192)
ADDR_net(giac-intern-fw-net-if2, 10.15.20.0, 255.255.255.0)
ADDR_gateway(giac-extern-fw, **xxx.yyy.zzz**.5)
ADDR_hostif(giac-extern-fw-if1, **xxx.yyy.zzz**.67)
ADDR_hostif(giac-extern-fw-if2, **xxx.yyy.zzz**.197)
ADDR_hostif(giac-extern-fw-if3, **xxx.yyy.zzz**.227)
ADDR_gateway(giac-intern-fw, **xxx.yyy.zzz**.69)
ADDR_hostif(giac-intern-fw-if0, 10.15.21.3)
ADDR_hostif(giac-intern-fw-if2, 10.15.20.3)
ADDR_network(giac-backup-service-net, **xxx.yyy.zzz**.224, 255.255.255.224)
ADDR_network(giac-service-net, **xxx.yyy.zzz**.192, 255.255.255.224)
ADDR_host(giac-smtp-1, **xxx.yyy.zzz**.222)
ADDR_host(giac-smtp-2, **xxx.yyy.zzz**.232)
ADDR_host(giac-intern-smtp-1-nat, **xxx.yyy.zzz**.93)
ADDR_host(giac-intern-smtp-2-nat, **xxx.yyy.zzz**.94)
ADDR_host(giac-dns-1, **xxx.yyy.zzz**.199)
ADDR_host(giac-dns-2, **xxx.yyy.zzz**.229)
ADDR_host(giac-business-nat, **xxx.yyy.zzz**.65)
ADDR_host(giac-www-1, **xxx.yyy.zzz**.200)
ADDR_host(giac-www-2, **xxx.yyy.zzz**.230)
ADDR_switch(giac-extern-vpn, **xxx.yyy.zzz**.193)
ADDR_host(giac-user-nat, **xxx.yyy.zzz**.66)
ADDR_host(giac-business-web-nat, **xxx.yyy.zzz**.89)
ADDR_host(giac-business-web, 10.10.20.254)
ADDR_host(giac-fileserver-1, 10.15.16.199)
ADDR_host(giac-fileserver-2, 10.15.20.199)

ADDR_host(giac-intern-dns-1, 10.15.16.250)
ADDR_host(giac-intern-dns-2, 10.15.20.250)
ADDR_host(giac-intern-smtp-1, 10.15.16.253)
ADDR_host(giac-intern-smtp-2, 10.15.20.253)
ADDR_host(giac-internal-bdc, 10.15.16.200)
ADDR_host(giac-internal-pdc, 10.15.20.200)
ADDR_network(giac-intern-private-1, 10.15.16.0, 255.255.255.0)
ADDR_network(giac-intern-private-2, 10.15.20.0, 255.255.254.0)

ADDR_valid(giac-business-web, **xxx.yyy.zzz**.89)
ADDR_valid(giac-fileserver-1, **xxx.yyy.zzz**.86)
ADDR_valid(giac-fileserver-2, **xxx.yyy.zzz**.85)
ADDR_valid(giac-intern-dns-1, **xxx.yyy.zzz**.90)
ADDR_valid(giac-intern-dns-2, **xxx.yyy.zzz**.91)
ADDR_valid(giac-intern-smtp-1, **xxx.yyy.zzz**.93)
ADDR_valid(giac-intern-smtp-2, **xxx.yyy.zzz**.94)
ADDR_valid(giac-internal-bdc, **xxx.yyy.zzz**.84)
ADDR_valid(giac-internal-pdc, **xxx.yyy.zzz**.83)
ADDR_valid(giac-intern-private-1, **xxx.yyy.zzz**.65)
ADDR_valid(giac-intern-private-2, **xxx.yyy.zzz**.66)

The Internal Firewall Objects

ADDR_net(giac-extern-fw-net-if0, **xxx.yyy.zzz**.0, 255.255.255.224)
ADDR_net(giac-extern-fw-net-if2, **xxx.yyy.zzz**.192, 255.255.255.224)
ADDR_net(giac-extern-fw-net-if3, **xxx.yyy.zzz**.224, 255.255.255.224)
ADDR_host(fw-admin, **xxx.yyy.zzz**.68)
ADDR_hostif(fw-admin-if1, **xxx.yyy.zzz**.69)
ADDR_hostif(fw-admin-if2, 10.128.0.38)
ADDR_network(giac-midfield-net, **xxx.yyy.zzz**.64, 255.255.255.192)
ADDR_network(extern-dmz, **xxx.yyy.zzz**.0, 255.255.255.224)
ADDR_net(giac-intern-fw-net-if0, 10.15.21.0, 255.255.255.0)
ADDR_net(giac-intern-fw-net-if1, **xxx.yyy.zzz**.64, 255.255.255.192)
ADDR_net(giac-intern-fw-net-if2, 10.15.20.0, 255.255.255.0)
ADDR_gateway(giac-intern-fw, **xxx.yyy.zzz**.69)
ADDR_hostif(giac-intern-fw-if0, 10.15.21.3)
ADDR_hostif(giac-intern-fw-if2, 10.15.20.3)
ADDR_gateway(giac-extern-fw, **xxx.yyy.zzz**.5)
ADDR_hostif(giac-extern-fw-if1, **xxx.yyy.zzz**.67)
ADDR_hostif(giac-extern-fw-if2, **xxx.yyy.zzz**.197)
ADDR_hostif(giac-extern-fw-if3, **xxx.yyy.zzz**.227)
ADDR_host(giac-intern-smtp-2, 10.15.20.253)
ADDR_host(giac-intern-smtp-1, 10.15.16.253)
ADDR_host(giac-smtp-1, **xxx.yyy.zzz**.222)
ADDR_host(giac-smtp-2, **xxx.yyy.zzz**.232)
ADDR_host(giac-intern-dns-1, 10.15.16.250)
ADDR_host(giac-intern-dns-2, 10.15.20.250)
ADDR_host(giac-dns-1, **xxx.yyy.zzz**.199)
ADDR_host(giac-dns-2, **xxx.yyy.zzz**.229)
ADDR_switch(giac-business-vpn, **xxx.yyy.zzz**.120)
ADDR_hostif(giac-business-vpn-if1, **xxx.yyy.zzz**.121)
ADDR_hostif(giac-business-vpn-if2, **xxx.yyy.zzz**.122)
ADDR_hostif(giac-business-vpn-if3, **xxx.yyy.zzz**.123)

ADDR_hostif(giac-business-vpn-if4, **xxx.yyy.zzz**.124)
ADDR_hostif(giac-business-vpn-if5, **xxx.yyy.zzz**.125)
ADDR_hostif(giac-business-vpn-if6, **xxx.yyy.zzz**.126)
ADDR_switch(giac-ent-partner-vpn, **xxx.yyy.zzz**.100)
ADDR_hostif(giac-ent-partner-vpn-if0, **xxx.yyy.zzz**.103)
ADDR_hostif(giac-ent-partner-vpn-if1, **xxx.yyy.zzz**.111)
ADDR_hostif(giac-ent-partner-vpn-if2, **xxx.yyy.zzz**.104)
ADDR_hostif(giac-ent-partner-vpn-if3, **xxx.yyy.zzz**.112)
ADDR_hostif(giac-ent-partner-vpn-if4, **xxx.yyy.zzz**.105)
ADDR_hostif(giac-ent-partner-vpn-if5, **xxx.yyy.zzz**.106)
ADDR_hostif(giac-ent-partner-vpn-if6, **xxx.yyy.zzz**.107)
ADDR_hostif(giac-ent-partner-vpn-if7, **xxx.yyy.zzz**.108)
ADDR_hostif(giac-ent-partner-vpn-if8, **xxx.yyy.zzz**.109)
ADDR_hostif(giac-ent-partner-vpn-if10, **xxx.yyy.zzz**.101)
ADDR_hostif(giac-ent-partner-vpn-if11, **xxx.yyy.zzz**.102)
ADDR_hostif(giac-ent-partner-vpn-if12, **xxx.yyy.zzz**.110)
ADDR_switch(giac-general-vpn, **xxx.yyy.zzz**.70)
ADDR_hostif(giac-general-vpn-if0, **xxx.yyy.zzz**.79)
ADDR_hostif(giac-general-vpn-if2, **xxx.yyy.zzz**.71)
ADDR_hostif(giac-general-vpn-if3, **xxx.yyy.zzz**.72)
ADDR_hostif(giac-general-vpn-if4, **xxx.yyy.zzz**.80)
ADDR_hostif(giac-general-vpn-if5, **xxx.yyy.zzz**.73)
ADDR_hostif(giac-general-vpn-if6, **xxx.yyy.zzz**.81)
ADDR_hostif(giac-general-vpn-if7, **xxx.yyy.zzz**.74)
ADDR_hostif(giac-general-vpn-if8, **xxx.yyy.zzz**.82)
ADDR_hostif(giac-general-vpn-if9, **xxx.yyy.zzz**.75)
ADDR_hostif(giac-general-vpn-if11, **xxx.yyy.zzz**.77)
ADDR_hostif(giac-general-vpn-if12, **xxx.yyy.zzz**.78)
ADDR_network(giac-intern-private-1, 10.15.16.0, 255.255.255.0)
ADDR_network(giac-intern-private-2, 10.15.20.0, 255.255.254.0)
ADDR_host(giac-www-1, **xxx.yyy.zzz**.200)
ADDR_host(giac-www-2, **xxx.yyy.zzz**.230)
ADDR_network(giac-backup-service-net, **xxx.yyy.zzz**.224, 255.255.255.224)
ADDR_network(giac-service-net, **xxx.yyy.zzz**.192, 255.255.255.224)
ADDR_network(giac-intern-product, 10.10.20.0, 255.255.255.0)
ADDR_host(giac-business-web, 10.10.20.254)
ADDR_host(giac-fileserver-1, 10.15.16.199)
ADDR_host(giac-internal-bdc, 10.15.16.200)
ADDR_host(giac-fileserver-2, 10.15.20.199)
ADDR_host(giac-internal-pdc, 10.15.20.200)

ADDR_valid(giac-business-web, **xxx.yyy.zzz**.89)
ADDR_valid(giac-fileserver-1, **xxx.yyy.zzz**.86)
ADDR_valid(giac-fileserver-2, **xxx.yyy.zzz**.85)
ADDR_valid(giac-intern-dns-1, **xxx.yyy.zzz**.90)
ADDR_valid(giac-intern-dns-2, **xxx.yyy.zzz**.91)
ADDR_valid(giac-intern-smtp-1, **xxx.yyy.zzz**.93)
ADDR_valid(giac-intern-smtp-2, **xxx.yyy.zzz**.94)
ADDR_valid(giac-internal-bdc, **xxx.yyy.zzz**.84)
ADDR_valid(giac-internal-pdc, **xxx.yyy.zzz**.83)
ADDR_valid(giac-intern-private-1, **xxx.yyy.zzz**.65)
ADDR_valid(giac-intern-private-2, **xxx.yyy.zzz**.66)
target_list30 = targets { giac-intern-fw };

Internal Firewall Address Translation Rules

```
// Address Translation Code
table_target_list30 = {
<TABLE_INDEX(1,1),
FWXT_SRC_STATIC, RANGE_MACHINE(giac-business-web), VALID_ADDR(giac-business-web),  0,
FWXT_EOX>,
<TABLE_INDEX(2,1),
FWXT_DST_STATIC, VALID_ADDR_RANGE_MACHINE(giac-business-web), giac-business-web,  0,
FWXT_EOX>,
<TABLE_INDEX(3,1),
FWXT_SRC_STATIC, RANGE_MACHINE(giac-fileserver-1), VALID_ADDR(giac-fileserver-1),  0,
FWXT_EOX>,
<TABLE_INDEX(4,1),
FWXT_DST_STATIC, VALID_ADDR_RANGE_MACHINE(giac-fileserver-1), giac-fileserver-1,  0,
FWXT_EOX>,
<TABLE_INDEX(5,1),
FWXT_SRC_STATIC, RANGE_MACHINE(giac-fileserver-2), VALID_ADDR(giac-fileserver-2),  0,
FWXT_EOX>,
<TABLE_INDEX(6,1),
FWXT_DST_STATIC, VALID_ADDR_RANGE_MACHINE(giac-fileserver-2), giac-fileserver-2,  0,
FWXT_EOX>,
<TABLE_INDEX(7,1),
FWXT_SRC_STATIC, RANGE_MACHINE(giac-intern-dns-1), VALID_ADDR(giac-intern-dns-1),  0,
FWXT_EOX>,
<TABLE_INDEX(8,1),
FWXT_DST_STATIC, VALID_ADDR_RANGE_MACHINE(giac-intern-dns-1), giac-intern-dns-1,  0,
FWXT_EOX>,
<TABLE_INDEX(9,1),
FWXT_SRC_STATIC, RANGE_MACHINE(giac-intern-dns-2), VALID_ADDR(giac-intern-dns-2),  0,
FWXT_EOX>,
<TABLE_INDEX(10,1),
FWXT_DST_STATIC, VALID_ADDR_RANGE_MACHINE(giac-intern-dns-2), giac-intern-dns-2,  0,
FWXT_EOX>,
<TABLE_INDEX(11,1),
FWXT_SRC_STATIC, RANGE_MACHINE(giac-intern-smtp-1), VALID_ADDR(giac-intern-smtp-1),  0,
FWXT_EOX>,
<TABLE_INDEX(12,1),
FWXT_DST_STATIC, VALID_ADDR_RANGE_MACHINE(giac-intern-smtp-1), giac-intern-smtp-1,  0,
FWXT_EOX>,
<TABLE_INDEX(13,1),
FWXT_SRC_STATIC, RANGE_MACHINE(giac-intern-smtp-2), VALID_ADDR(giac-intern-smtp-2),  0,
FWXT_EOX>,
<TABLE_INDEX(14,1),
FWXT_DST_STATIC, VALID_ADDR_RANGE_MACHINE(giac-intern-smtp-2), giac-intern-smtp-2,  0,
FWXT_EOX>,
<TABLE_INDEX(15,1),
FWXT_SRC_STATIC, RANGE_MACHINE(giac-internal-bdc), VALID_ADDR(giac-internal-bdc),  0,
FWXT_EOX>,
<TABLE_INDEX(16,1),
FWXT_DST_STATIC, VALID_ADDR_RANGE_MACHINE(giac-internal-bdc), giac-internal-bdc,  0,
FWXT_EOX>,
<TABLE_INDEX(17,1),
FWXT_SRC_STATIC, RANGE_MACHINE(giac-internal-pdc), VALID_ADDR(giac-internal-pdc),  0,
FWXT_EOX>,
<TABLE_INDEX(18,1),
```

```
FWXT_DST_STATIC, VALID_ADDR_RANGE_MACHINE(giac-internal-pdc), giac-internal-pdc,  0,
FWXT_EOX>,
<TABLE_INDEX(19,1),
FWXT_SRC_STATIC, RANGE_NETWORK(giac-intern-private-1), giac-intern-private-1,  0,
FWXT_DST_STATIC, RANGE_NETWORK(giac-intern-private-1), giac-intern-private-1,  0,
FWXT_EOX>,
<TABLE_INDEX(20,1),
FWXT_HIDE, RANGE_NETWORK(giac-intern-private-1), VALID_ADDR(giac-intern-private-1),  0,
FWXT_EOX>,
<TABLE_INDEX(21,1),
FWXT_SRC_STATIC, RANGE_NETWORK(giac-intern-private-2), giac-intern-private-2,  0,
FWXT_DST_STATIC, RANGE_NETWORK(giac-intern-private-2), giac-intern-private-2,  0,
FWXT_EOX>,
<TABLE_INDEX(22,1),
FWXT_HIDE, RANGE_NETWORK(giac-intern-private-2), VALID_ADDR(giac-intern-private-2),  0,
FWXT_EOX>
};
<> all@target_list30
   set r_xlate table_target_list30;


set r_xlate_pool 0;
service_list16 = { <80, 80>, <443, 443> };
service_list21 = { <259, 259>, <900, 900> };
ip_list1 = { <10.128.0.38, 10.128.0.38>, <xxx.yyy.zzz.64, xxx.yyy.zzz.127>, <xxx.yyy.zzz.192, xxx.yyy.zzz.255>
};
ip_list2 = { <xxx.yyy.zzz.1, xxx.yyy.zzz.30> };
ip_list3 = { <10.128.0.38, 10.128.0.38>, <xxx.yyy.zzz.65, xxx.yyy.zzz.126> };
ip_list4 = { <xxx.yyy.zzz.192, xxx.yyy.zzz.223> };
ip_list5 = { <xxx.yyy.zzz.224, xxx.yyy.zzz.255> };
ip_list6 = { <10.15.21.0, 10.15.21.255> };
ip_list7 = { <xxx.yyy.zzz.64, xxx.yyy.zzz.127> };
ip_list8 = { <10.15.20.0, 10.15.20.255> };
ip_list9 = { <10.128.0.38, 10.128.0.38>, <xxx.yyy.zzz.68, xxx.yyy.zzz.69> };
ip_list10 = { <10.15.20.3, 10.15.20.3>, <10.15.21.3, 10.15.21.3>, <xxx.yyy.zzz.69, xxx.yyy.zzz.69> };
ip_list11 = { <10.15.20.3, 10.15.20.3>, <10.15.21.3, 10.15.21.3>, <xxx.yyy.zzz.5, xxx.yyy.zzz.5>, <xxx.yyy.zzz.67,
xxx.yyy.zzz.67>, <xxx.yyy.zzz.69, xxx.yyy.zzz.69>, <xxx.yyy.zzz.197, xxx.yyy.zzz.197>, <xxx.yyy.zzz.227,
xxx.yyy.zzz.227> };
ip_list12 = { <10.15.16.253, 10.15.16.253>, <10.15.20.253, 10.15.20.253> };
ip_list13 = { <xxx.yyy.zzz.222, xxx.yyy.zzz.222>, <xxx.yyy.zzz.232, xxx.yyy.zzz.232> };
ip_list14 = { <10.15.16.250, 10.15.16.250>, <10.15.20.250, 10.15.20.250> };
ip_list15 = { <xxx.yyy.zzz.199, xxx.yyy.zzz.199>, <xxx.yyy.zzz.229, xxx.yyy.zzz.229> };
ip_list17 = { <10.15.16.0, 10.15.16.255>, <10.15.20.0, 10.15.21.255>, <xxx.yyy.zzz.70, xxx.yyy.zzz.75>,
<xxx.yyy.zzz.77, xxx.yyy.zzz.82>, <xxx.yyy.zzz.100, xxx.yyy.zzz.112>, <xxx.yyy.zzz.120, xxx.yyy.zzz.126> };
ip_list18 = { <xxx.yyy.zzz.200, xxx.yyy.zzz.200>, <xxx.yyy.zzz.230, xxx.yyy.zzz.230> };
ip_list19 = { <10.15.16.0, 10.15.16.255> };
ip_list20 = { <xxx.yyy.zzz.192, xxx.yyy.zzz.255> };
ip_list22 = { <10.10.20.1, 10.10.20.254> };
ip_list23 = { <10.10.20.254, 10.10.20.254>, <xxx.yyy.zzz.89, xxx.yyy.zzz.89> };
ip_list24 = { <10.10.20.0, 10.10.20.255> };
ip_list25 = { <10.15.16.0, 10.15.16.255>, <10.15.20.0, 10.15.21.255> };
ip_list26 = { <xxx.yyy.zzz.120, xxx.yyy.zzz.126> };
ip_list27 = { <10.15.16.199, 10.15.16.200>, <10.15.16.250, 10.15.16.250>, <10.15.16.253, 10.15.16.253>,
<10.15.20.199, 10.15.20.200>, <10.15.20.250, 10.15.20.250>, <10.15.20.253, 10.15.20.253>, <xxx.yyy.zzz.83,
xxx.yyy.zzz.86>, <xxx.yyy.zzz.90, xxx.yyy.zzz.91>, <xxx.yyy.zzz.93, xxx.yyy.zzz.94> };
ip_list28 = { <xxx.yyy.zzz.70, xxx.yyy.zzz.75>, <xxx.yyy.zzz.77, xxx.yyy.zzz.82> };
```

ip_list29 = { <10.15.20.199, 10.15.20.200>, <10.15.20.250, 10.15.20.250>, <10.15.20.253, 10.15.20.253>, <**xxx.yyy.zzz**.83, **xxx.yyy.zzz**.83>, <**xxx.yyy.zzz**.85, **xxx.yyy.zzz**.85>, <**xxx.yyy.zzz**.91, **xxx.yyy.zzz**.91>, <**xxx.yyy.zzz**.94, **xxx.yyy.zzz**.94> };