

SatOri Advisory

SATA200510

WinAsm Studio

Summary

A vulnerability has been discovered in WinAsm Studio, which can be exploited by malicious, anonymous individuals to compromise a vulnerable system.

The vulnerability is caused as a result of improper bounds checking when reading *.WAP files. This can be exploited to cause a stack-based buffer overflow by tricking a user into opening a maliciously constructed WinAsm project.

Successful exploitation of this vulnerability enables execution of arbitrary code.

Affected Versions

This vulnerability is confirmed in the following versions:

- WinAsm Studio 5.1.8.0

Other versions may also be affected.

Screen-dumps

The screenshot displays the WinAsm Studio interface. The assembly window shows the following code:

```
0040A9F5 MOV     DWORD PTR SS:[EBP-8], 0
0040A9FC LEA     EDI, [EBP-100]
0040AA02 > FF45 F8 INC     DWORD PTR SS:[EBP-8]
0040AA05 8085 AFFDFF LEA     EAX, [EBP-251]
0040AA08 50     PUSH   EAX
0040AA0C FF75 F8 PUSH   DWORD PTR SS:[EBP-8]
0040AA0F E8 3C870200 CALL   00433150
0040AA14 68 92814300 PUSH   OFFSET WinAsm.00438192
0040AA19 68 05010000 PUSH   105
0040AA1E FF75 FC PUSH   DWORD PTR SS:[EBP-4]
0040AA21 68 BF934300 PUSH   OFFSET WinAsm.004390BF
0040AA26 8085 AFFDFF LEA     EAX, [EBP-251]
0040AA2C 50     PUSH   EAX
0040AA30 68 DF8E4300 PUSH   OFFSET WinAsm.00438EDF
0040AA37 E8 DD8F0200 CALL   <JMP.&kernel32.GetPrivateProfileStringA
0040AA39 8BC0   OR     EAX, EAX
0040AA3B 8F84 F30100 JE     0040AC32
```

The registers window shows:

```
EAX 0012FBEB
ECX 1999999A
EDX 00000000
EBX 000C02AE
ESP 0012FBC4 PTR to ASCII "FILES"
EBP 0012FE3C
ESI 000C02E2
EDI 0012FD2F
EIP 0040AA32 WinAsm.0040AA32
C 0 ES 0023 32bit 0(FFFFFFFF)
P 0 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 0 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDF000(FFF)
T 0 GS 0000 NULL
O 0
0 0 LastErr 00000005 ERROR_ACCESS_DENIED
EFL 00000202 (NO, NB, NE, A, NS, PO, GE, G)
MM0 00C0 C0C0 00C0 C0C0
MM1 00C0 C0C0 00C0 C0C0
MM2 00C0 C0C0 00C0 C0C0
```

The stack dump window shows the following data:

```
Address Hex dump ASCII
00438000 11 58 45 00 1D 5E 45 00 4E.E.#e
00438008 21 5F 45 00 25 60 45 00 +.E.%.E
00438010 29 61 45 00 01 00 00 00 )a.e.0..
00438018 43 6F 64 65 48 69 2E 64 CodeHl.
00438020 6C 5C 00 4F 62 64 65 63 ll.Obje
00438028 74 73 46 6F 6E 74 00 20 tsFont.
00438030 2F 49 00 55 49 00 45 6E /I.UI.E
00438038 67 6C 69 73 68 20 28 4F glish (
00438040 66 66 69 63 69 61 6C 29 ffcial
00438048 00 01 0E 0A 6F 00 00 00 .0#.o..
00438050 00 95 70 00 00 67 00 2A .ou..A.
00438058 00 40 41 53 40 00 4C 49 .MRSH.L
00438060 4E 4B 00 52 43 00 41 42 NK.RC.F
00438068 43 44 45 46 47 48 49 4A CDEFGHI
00438070 4B 4C 4D 4E 4F 50 51 52 KLMNOPQ
00438078 53 54 55 56 57 58 59 5A STUVWXYZ
00438080 61 62 63 64 65 66 67 68 abcdefgh
00438088 69 6A 6B 6C 6D 6E 6F 70 ijklmnop
00438090 71 72 73 74 75 76 77 78 qrstuvw
00438098 79 7A 00 23 00 39 39 39 yz.#.99
004380A0 39 39 00 2E 2E 00 00 00 99.....
004380A8 00 00 49 4E 54 45 4C 4C ..INTEL
004380B0 49 53 45 4E 53 45 00 41 ISENSE.
004380B8 75 74 65 4E 6F 6D 70 6C utocmfc
004380C0 65 74 65 00 54 52 49 47 etc.TRI
004380C8 47 45 52 00 52 43 53 69 GER.RCS
004380D0 6C 65 6E 74 70 00 65 6C lent.Pe
004380D8 6C 65 73 54 6F 6F 6C 73 lesTool
004380E0 00 43 4F 4E 54 52 4F 4C .CONTRC
004380E8 53 00 01 41 75 74 6F 49 .S.9Autc
004380F0 6E 63 46 69 6C 65 56 65 ncFileL
004380F8 72 73 69 6F 6E 00 68 74 rson.h
00438100 74 70 3A 2F 2F 77 77 77 tp://ww
00438108 2E 77 69 6E 61 73 6D 2E .winash
```

The registers window shows the call to kernel32.GetPrivateProfileStringA:

```
Arg2
Arg1 => [LOCAL.2]
WinAsm.00433150
FileName = "C:\BACKUP\sharpe\Unencryp
Count = 261.
Buffer => [LOCAL.1]
Default = ""
Key
Section = "FILES"
KERNEL32.GetPrivateProfileStringA
```

The stack dump window shows the following data:

```
Section = "FILES"
Key = "4"
Default = ""
Buffer = "nine.inc"
Count = 261.
FileName = "C:\BACKUP\sharpe\Unencrypted\programming\ASM\Uu\In\vu\In.wap"
ASCII ".exe"
```

Annotations in the screenshot include:

- A red box around the buffer value "nine.inc" in the registers window.
- A red box around the call to kernel32.GetPrivateProfileStringA in the registers window.
- A red box around the stack dump showing return addresses being overwritten.
- A red box around the stack dump showing the return addresses being overwritten.
- A red box around the stack dump showing the return addresses being overwritten.

Resolution

There is currently no fix for this issue.

Timeline

- Vulnerability identified: 20.05.10
- Vendor informed: 27.05.10

- Vendor fix: *Currently unavailable*

References

- <http://blog.sat0ri.com/?p=512>
- <http://www.winasm.net/>