

[UNIX] XPDF Multiple Buffer Overflow Vulnerabilities (JPXStream.cc, Stream.cc)

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2005-12/msg00110.html>

- *From:* SecuriTeam <support@xxxxxxxxxxxxxx>
 - *Date:* 28 Dec 2005 12:18:13 +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>

XPDF Multiple Buffer Overflow Vulnerabilities (JPXStream.cc, Stream.cc)

SUMMARY

<<http://www.foolabs.com/xpdf/>> Xpdf is "an open source PDF viewer for the X Window System and Motif. Xpdf runs on practically any Unix-like operating system. Xpdf can decode LZW and read encrypted PDFs. The official version obeys the DRM of PDF files, which may prevent copying, printing, or converting some PDF files. There are patches which will ignore these DRM restrictions".

Multiple buffer overflow vulnerabilities discovered in Xpdf, which could be exploited by malicious remote attackers to execute arbitrary commands and thus take complete control over vulnerable system.

DETAILS

Vulnerable Systems:

* Xpdf version 3.01 and prior

Exploitation of those vulnerabilities could result in arbitrary code execution with privileges of the xpdf process. Currently, exploitation resulting in code execution is theoretical and dependent on the process

memory layout. A typical exploitation attempt would require an attacker to supply a malicious pdf to the victim. The victim would need to open the corrupt pdf file in xpdf.

DCTStream::Baseline Heap Overflow Vulnerability:

The vulnerability specifically exists due to insufficient input validation in the DCT stream parsing code. The DCTStream::readBaselineSOF function from xpdf/Stream.cc takes the value of numComps from user-controllable data from within the PDF file. The numComps value is used in a loop to copy data from the file into a preallocated buffer in the heap, shown as follows:

```
GBool DCTStream::readBaselineSOF() {  
..  
numComps = str->getChar();  
..  
for (i = 0; i < numComps; ++i) {  
compInfo[i].id = str->getChar();  
c = str->getChar();  
compInfo[i].hSample = (c >> 4) & 0x0f;  
compInfo[i].vSample = c & 0x0f;  
compInfo[i].quantTable = str->getChar();  
}  
..
```

Overly large values supplied to numComps will result in corruption of heap memory resulting in a DoS condition, potentially resulting in arbitrary code execution.

CVE Information:

<<http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2005-3191>>
CAN-2005-3191

DCTStream::Progressive Heap Overflow:

The vulnerability specifically exists due to insufficient input validation in the DCT stream parsing code. The DCTStream::readProgressiveSOF function from xpdf/Stream.cc takes the value of numComps from user-controllable data from within the PDF file. The numComps value is used in a loop to copy data from the file into a pre-allocated buffer in the heap as shown below.

```
GBool DCTStream::readProgressiveSOF() {  
..  
numComps = str->getChar();  
..  
for (i = 0; i < numComps; ++i) {  
compInfo[i].id = str->getChar();  
c = str->getChar();  
compInfo[i].hSample = (c >> 4) & 0x0f;  
compInfo[i].vSample = c & 0x0f;  
compInfo[i].quantTable = str->getChar();  
}
```

```
}  
..
```

Overly large values supplied to numComps result in corruption of heap memory, resulting in a DoS condition, potentially resulting in arbitrary code execution.

CVE Information:

<http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2005-3191>
CAN-2005-3191

StreamPredictor Heap Overflow Vulnerability:

The vulnerability specifically exists due to insufficient input validation in the Predictor stream parsing code. The StreamPredictor::StreamPredictor function from xpdf/Stream.cc takes the value of numComps from user-controllable data from within the PDF file. The numComps value is used in a series of calculations within the StreamPredictor function. Using specially crafted values, a call to gmalloc can be forced to allocate the minimum number of bytes, which may later be overrun with user-supplied data from the PDF file leading to corruption of heap memory that might result in a DoS condition or arbitrary code execution.

CVE Information:

<http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2005-3192>
CAN-2005-3192

JPX Stream Reader Heap Overflow Vulnerability:

Local exploitation of a heap-based buffer overflow vulnerability in xpdf, as included by multiple vendor's software distributions, could allow attackers to cause a denial of service (DoS) condition, potentially resulting in arbitrary code execution. The vulnerability specifically exists due to insufficient input validation in the JPX Stream parsing code for decoding embedded JPEG 2000 images. The JPXStream::readCodestream function from xpdf/JPXStream.cc takes the value of nXTiles and nYTiles from user-controllable data from within the PDF file. The nXTiles and nYTiles values are then used in a gmallocn() call as shown below.

```
GBool JPXStream::readCodestream(Guint len) {  
..  
switch (segType) {  
case 0x4f: // SOC – start of codestream  
// marker only  
break;  
case 0x51: // SIZ – image and tile size  
if (!readUWord(&capabilities) ||  
!readULong(&img.xSize) ||  
!readULong(&img.ySize) ||  
!readULong(&img.xOffset) ||  
!readULong(&img.yOffset) ||  
!readULong(&img.xTileSize) ||  
!readULong(&img.yTileSize) ||
```

```
!readULong(&img.xTileOffset) ||
!readULong(&img.yTileOffset) ||
!readUWord(&img.nComps) {
error(getPos(), "Error in JPX SIZ marker segment");
return gFalse;
}
..
img.nXTiles = (img.xSize - img.xTileOffset + img.xTileSize - 1) /
img.xTileSize;
img.nYTiles = (img.ySize - img.yTileOffset + img.yTileSize - 1) /
img.yTileSize;

img.tiles = (JPXTile *)gmallocn(img.nXTiles * img.nYTiles,
sizeof(JPXTile));
```

The values are used again later in JPEG format parsing code to copy data from the file into a pre-allocated buffer in the heap. Overly large values supplied to nXTiles and nYTiles result in corruption of heap memory, which results in a DoS condition. This could result in arbitrary code execution.

CVE Information:

<<http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2005-3193>>
CAN-2005-3193

Disclosure Timeline:

- * 13.10.05 – Initial vendor notification
- * 19.10.05 – Initial vendor response
- * 05.12.05 – Coordinated public disclosure

ADDITIONAL INFORMATION

The information has been provided by iDefense.com.

The original articles can be found at:

<<http://www.odefense.com/application/poi/display?id=342&type=vulnerabilities>>
<http://www.odefense.com/application/poi/display?id=342&type=vulnerabilities>

<<http://www.odefense.com/application/poi/display?id=343&type=vulnerabilities>>
<http://www.odefense.com/application/poi/display?id=343&type=vulnerabilities>

<<http://www.odefense.com/application/poi/display?id=344&type=vulnerabilities>>
<http://www.odefense.com/application/poi/display?id=343&type=vulnerabilities>

<<http://www.odefense.com/application/poi/display?id=345&type=vulnerabilities>>
<http://www.odefense.com/application/poi/display?id=343&type=vulnerabilities>

=====

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.

-
- Prev by Date: [*\[NEWS\] Panda Antivirus ZOO Library Heap Overflow*](#)
 - Next by Date: [*\[NEWS\] Mac OS X KHTMLParser DoS*](#)
 - Previous by thread: [*\[NEWS\] Panda Antivirus ZOO Library Heap Overflow*](#)
 - Next by thread: [*\[NEWS\] Mac OS X KHTMLParser DoS*](#)
 - Index(es):
 - ◆ [*Date*](#)
 - ◆ [*Thread*](#)