Malicious PDF files
Detecting and Analyzing

January 2010
Paweł Jacewicz – NASK/CERT Polska
Detection

> PDF file format
> Code obfuscation
> Exploits
> Why maliciously behaving PDFs are hard to detect?
PDF file format

Malicious PDF files – Detecting and Analyzing
PDF file format

Header

Objects

Reference table

Trailer

Malicious PDF files – Detecting and Analyzing
PDF file format

Header
Objects
Reference table
Trailer

Malicious PDF files – Detecting and Analyzing
Obfuscation: Names

8 0 obj
<<
/Type /Action
/S /URI
/URI (http://google.pl)
>>
endobj
Obfuscation: Strings

8 0 obj
<<
/Type /Action
/S /URI
/URI (http://google.pl)
>>
endobj

8 0 obj
<<
/Type /Action
/S /URI
/URI (http://google.pl)
>>
endobj

Malicious PDF files – Detecting and Analyzing
Obfuscation: Strings

8 0 obj
<
/Type /Action
/S /URI
/URI (https://go.pl)
>>
endobj

Octal codes from ANSI table
Obfuscation: Strings

8 0 obj
<<
/Type /Action
/S /URI
/URI <68 74 74 70 3A
2F2F 67 6F 6C>
>>
endobj

Hexadecimal codes from ANSI table
# Obfuscation: Streams

<table>
<thead>
<tr>
<th>Filter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCIIHexDecode</td>
<td>Decodes data represented by a string of ASCII hex characters</td>
</tr>
<tr>
<td>ASCII85Decode</td>
<td>Decodes data represented by a string in base-85 format</td>
</tr>
<tr>
<td>LZWDDecode</td>
<td>Decompresses data saved in Lempel–Ziv–Welch format</td>
</tr>
<tr>
<td>FlateDecode</td>
<td>Decompresses data saved with zlib/deflate library</td>
</tr>
<tr>
<td>RunLengthDecode</td>
<td>Decompresses data saved in RLE format (byte oriented)</td>
</tr>
<tr>
<td>Crypt</td>
<td>Decodes encrypted data</td>
</tr>
</tbody>
</table>
Obfuscation: Streams

5 0 obj
<<
/Length 42
>>
stream
BT /F1 24 Tf 100 700 Td (Hello world)Tj ET
endstream
endobj

5 0 obj
<<
/Length 55
/Filter /ASCII85Decode
>>
stream
6<#'7PQ#@1a#b0+>GQ(+?(u.+B2ko–qlocCi:G?DfTZ).9(%)78s~>
endstream
endobj
Obfuscation: Streams

stream
6<#\7PQ#@1a#b0+>GQ(+?(u.+B2ko-qIocCi:G?DfTZ).9(%)78s~>
endstream

5 0 obj
<<
/Length 168
/Filter [/ASCIIHexDecode /ASCII85Decode]
>>
stream
36 3C 23 27 5C 37 50 51 23 40 31 61 23 62 30 2B
3E 47 51 28 2B 3F 28 75 2E 2B 42 32 6B 6F 2D 71
49 6F 63 43 69 3A 47 3F 44 66 54 5A 29 2E 39 28
25 29 37 38 73 7E 3E>
endstream
endobj
Detection Rate

File xSW.pdf received on 2009.06.19 13:35:21 (UTC)
Current status: finished
Result: 1/41 (2.44%)

File xSW.pdf received on 2009.07.07 08:30:24 (UTC)
Current status: finished
Result: 3/41 (7.32%)

File xSW.pdf received on 2009.07.27 08:57:20 (UTC)
Current status: finished
Result: 4/41 (9.76%)

File xSW.pdf received on 2010.01.24 15:16:07 (UTC)
Current status: finished
Result: 5/41 (12.20%)
The HoneySpider High-Interaction Machine detects malicious behavior in the operating system.

When configured properly it can detect improper behavior of malicious PDF files.
Analysis
Tools and Conclusions
PDFiD

Simple string scanner. Generates statistics and can deobfuscate names in PDF dictionary.

Incorporated in VirusTotal.
PDF-Parser

Can parse and decompress objects.

Useful for extracting JavaScripts.
Malicious PDF file found on the web containing compressed JavaScript.

Two ways of infection:
- With actions invoking a JS function
- With metadata fields referring to an object containing JavaScript

Also almost always confusing object referencing.
Analysis

Decompressed stream contains JavaScript code exploiting the reader’s engine.

The code is usually further obfuscated and contains hidden eval functions.

The code contains function called by the /OpenAction
Analysis

Taking off another layer of obfuscation gives the final JS code exploiting vulnerabilities.

“Standard” exploitation using a heap-spray method and encoded shellcode.

Shellcode downloads loader which fetches the malware and infects the OS.
Exploits in the wild

- Collab.colectEmailInfo
- getAnnots
- JBIG2
- util.printf

/Colors > $2^{24}$
(associated with FlateDecode filter)

doc.media.newPlayer
util.printf

The latest...

...seen malicious PDFs are sooo much more advanced than the first ones...

- Malicious PDF files have become a container for malware
- Shellcode searches memory for loaded PDF document...
- ...and drops a benign one to fool the user
Conclusions

High popularity of the PDF format makes it a very “useful” attack vector.

The next emerging threat – more and more hacked websites contain malicious PDF files.

Extreme polymorphic capabilities of PDFs – the necessity of an advanced analysis tool.
The End...

Questions?