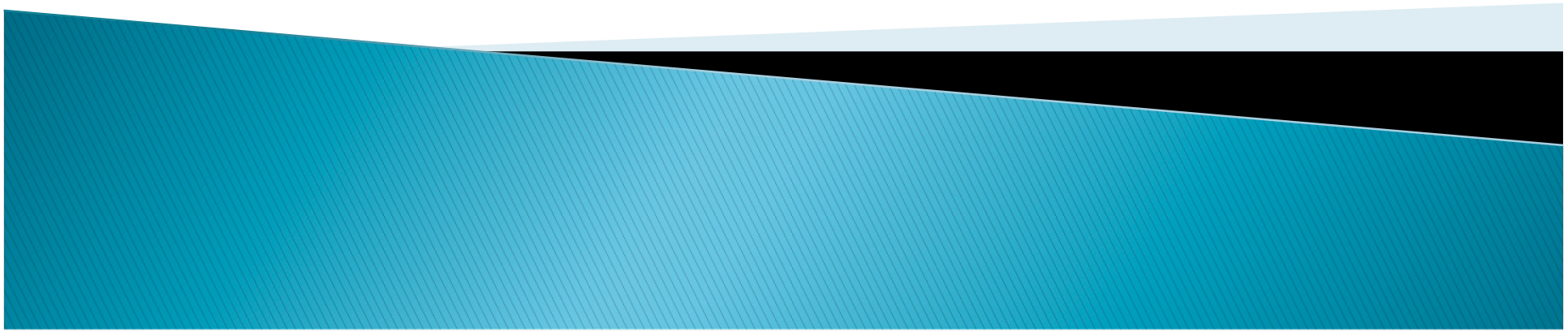
Malicious PDF files

Detecting and Analyzing

January 2010 Paweł Jacewicz – NASK/CERT Polska

Detection



Detection

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

HONEYSPIDER network

PDF file format

```
%PDF-1.1

1 0 obj
<<
 /Type /Catalog
 /Outlines 2 0 R
 /Pages 3 0 R
>>
endobj

2 0 obj
<<
 /Type /Outlines
 /Count 0
>>
endobj

3 0 obj
<<
 /Type /Pages
 /Kids [4 0 R]
 /Count 1
>>
endobj

… … …

xref
0 8
0000000000 65535 f
0000000009 00000 n
0000000074 00000 n
0000000120 00000 n
0000000179 00000 n
0000000300 00000 n
0000000384 00000 n

 trailer
<<
 /Size 7
 /Root 1 0 R
>>
startxref
408
%%EOF
```

Header

Objects

Reference table

Trailer

HONEYSPIDER network

PDF file format

```
%PDF-1.1
```

```
1 0 obj
<<
 /Type /Catalog
 /Outlines 2 0 R
 /Pages 3 0 R
>>
endobj
```

```
2 0 obj
<<
 /Type /Outlines
 /Count 0
>>
endobj
```

```
3 0 obj
<<
 /Type /Pages
 /Kids [4 0 R]
 /Count 1
>>
endobj

… … …
```

```
xref
0 8
0000000000 65535 f
0000000009 00000 n
0000000074 00000 n
0000000120 00000 n
0000000179 00000 n
0000000300 00000 n
0000000384 00000 n
```

```
 trailer
<<
 /Size 7
 /Root 1 0 R
>>
startxref
408
%%EOF
```

Header

Objects

Reference table

Trailer

HONEYSPIDER network

PDF file format

```
%PDF-1.1
```

```
1 0 obj
<<
 /Type /Catalog
 /Outlines 2 0 R
 /Pages 3 0 R
>>
endobj
```

```
2 0 obj
<<
 /Type /Outlines
 /Count 0
>>
endobj
```

```
3 0 obj
<<
 /Type /Pages
 /Kids [4 0 R]
 /Count 1
>>
endobj

... ... ...
```

```
xref
0 8
0000000000 65535 f
0000000009 00000 n
0000000074 00000 n
0000000120 00000 n
0000000179 00000 n
0000000300 00000 n
0000000384 00000 n
```

```
 trailer
<<
 /Size 7
 /Root 1 0 R
>>
startxref
408
%%EOF
```

Header

Objects

Reference table

Trailer

HONEYSPIDER network

PDF file format

```
%PDF-1.1
```

```
1 0 obj
<<
 /Type /Catalog
 /Outlines 2 0 R
 /Pages 3 0 R
>>
endobj
```

```
2 0 obj
<<
 /Type /Outlines
 /Count 0
>>
endobj
```

```
3 0 obj
<<
 /Type /Pages
 /Kids [4 0 R]
 /Count 1
>>
endobj

... ... ...
```

```
xref
0 8
0000000000 65535 f
0000000009 00000 n
0000000074 00000 n
0000000120 00000 n
0000000179 00000 n
0000000300 00000 n
0000000384 00000 n
```

```
 trailer
<<
 /Size 7
 /Root 1 0 R
>>
startxref
408
%%EOF
```

Header

Objects

Reference table

Trailer

HONEYSPIDER network

Obfuscation: Names

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

```
8 0 obj
<<
/Type /Action
/S /#55R#49
/U#52I (http://google.pl)
>>
endobj
```


HONEYSPIDER network

Obfuscation: Strings

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

```
8 0 obj
<<
/Type /Action
/S /URI
/URI (ht\
 tp\
 :/\
 /goog\
 le.\
 pl)
>>
endobj
```


HONEYSPIDER network

Obfuscation: Strings

```
8 0 obj
<<
/Type /Action
/S /URI
/URI (h\164\164p://go\157\147\154e.pl)
>>
endobj
```

Octal codes from ANSI table

HONEYSPIDER network

Obfuscation: Strings

```
8 0 obj
<<
/Type /Action
/S /URI
/URI <68 74 74          70 3A
2F2F    67              6F      6F
676C    65  2E   70     6C>
>>
endobj
```

Hexadecimal codes from ANSI table

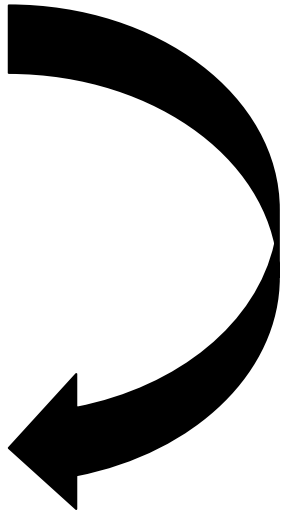
HONEYSPIDER network

Obfuscation: Streams

Filter	Description
ASCIIHexDecode	Decodes data represented by a string of ASCII hex characters
ASCII85Decode	Decodes data represented by a string in base-85 format
LZWDeode	Decompresses data saved in Lempel-Ziv-Welch format
FlateDecode	Decompresses data saved with zlib/deflate library
RunLengthDecode	Decompresses data saved in RLE format (byte oriented)
Crypt	Decodes encrypted data

Obfuscation: Streams

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

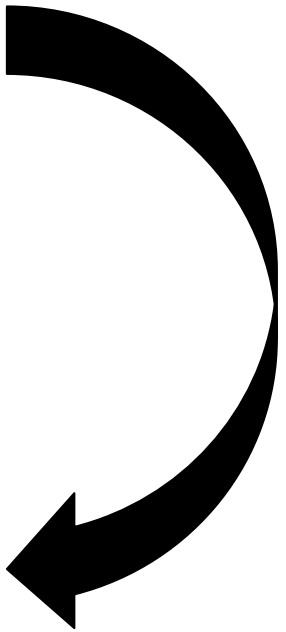


```
5 0 obj
<<
/Length 55
/Filter /ASCII85Decode
>>
stream
6<#'\7PQ#@1a#b0+>GQ(+?(u.+B2ko-qIocCi:G?DfTZ).9(%)78s~>
endstream
endobj
```


HONEYSPIDER network

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
```


HONEYSPIDER network

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%)**

HONEYSPIDER network

Detection: The HoneySpider

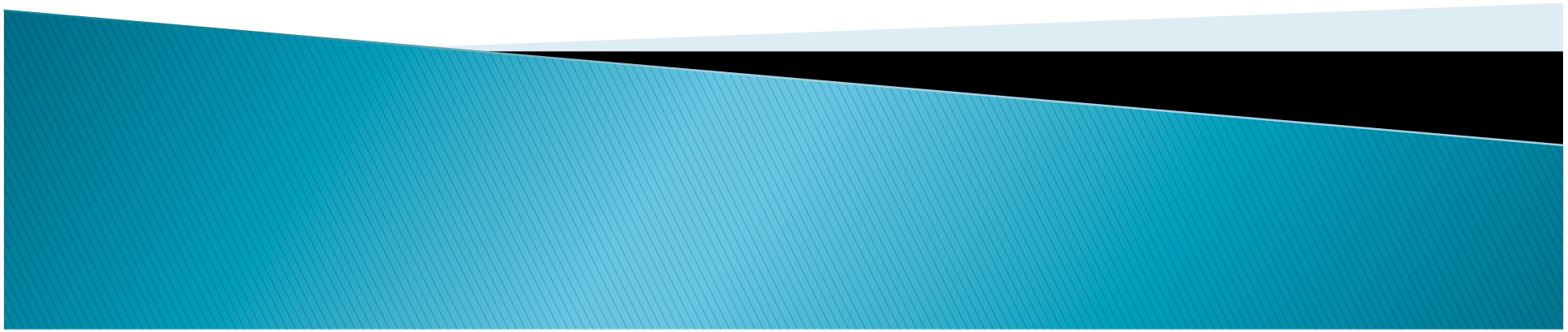
The HoneySpider High-Interaction Machine detects malicious behavior in the operating system.

When configured properly it can detect improper behavior of malicious PDF files.

HONEYSPIDER
network

Analysis

Tools and Conclusions



PDFiD

```
PDFiD 0.0.10 xSW.pdf
 PDF Header: %PDF-0.1
 obj                   25
 endobj                25
 stream                 2
 endstream              2
 xref                   2
 trailer                2
 startxref              2
 /Page                  1
 /Encrypt               0
 /ObjStm                1
 /JS                    1
 /JavaScript            1
 /AA                    1
 /OpenAction            0
 /AcroForm              0
 /JBIG2Decode           0
 /RichMedia             0
 /Colors > 2^24         0
```

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.

```
obj 9 0
 Type:
 Referencing:
 Contains stream

<</Filter /FlateDecode
/Length 2758
>>


 <<
   /Filter /FlateDecode

   /Length 2758

 >>


var keyStr = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
var ebal = eval("e"+"v"+"al");
function decode64(input) {
   var output = "";
   var chr1, chr2, chr3;
   var enc1, enc2, enc3, enc4;
   var i = 0;
   input = input.replace(/[^A-Za-z0-9\+\/\=]/g, "");
   do {
      enc1 = keyStr.indexOf(input.charAt(i++));
```


HONEYSPIDER network

Analysis

Malicious PDF file found on the web containing compressed JavaScript.

```
2 0 obj
<<
/OpenAction << /JS 9 0 R /S /JavaScript >>
/Type /Catalog
/Pages 3 0 R
>>
endobj
```

Two ways of infection:

- With *actions* invoking a JS function
- With metadata fields referring to a object containing JavaScript

```
9 0 obj
<<
/Type /Catalog
/Pages 1 0 R
/OpenAction [3 0 R /FitH null]
/PageLayout /OneColumn
/Names <</JavaScript 6 0 R>>
>>
endobj
```

Also almost always confusing object referencing.

HONEYSPIDER network

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.printd

HONEYSPIDER network

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

HONEYSPIDER network

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?

HONEYSPIDER
network