Bug Huntin' With Media Formats

A Quick and Dirty Guide to Finding Media Bugs

By xort

BHWMF::Overview

- What Defines a Media Format?
- Who's Having Problems
- Studying A Format for Common Issues
- Common Obstacles to Overcome
- Tools
- Creating Tools For the Job
- Analyzing data
- Examples

Bug Huntin' With Media Formats

What Defines a Media Format? – Tons of stuff! – lose term

(This presentation is focused mostly around applications that provide users with graphical content)

Some Are Relatively simple (BMP, ICO, GIF, JPG, JIF, JFIF, JP2, JPX, J2K, J2C, PNG, TIFF, FPX, PCD, DIB, …. The list goes on and on)

- **These can often be fuzzed with little programming**
- **A great deal of fuzzers are already out there that you can take advantage of (don't re-invent the wheel unless you suspect it's a shitty wheel)**

Some Are a Bit More Complicated (PDF, PPT, etc…)

- **These often serve as container encapsulation formats for collections of smaller/more simple formats like above**

Bug Huntin' With Media Formats

BHWMF::Who's Having Problems

Microsoft – 2009 score: 22

Malformed AVI Header Vulnerability - CVE-2009-1545
AVI Integer Overflow Vulnerability – CVE-2009-1546
Embedded OpenType Font Heap Overflow Vulnerability - CVE-2009-0231
Embedded OpenType Font Integer Overflow Vulnerability – CVE-2009-0232
DirectX NULL Byte Overwrite Vulnerability - CVE-2009-1537
DirectX Pointer Validation Vulnerability - CVE-2009-1538
DirectX Size Validation Vulnerability – CVE-2009-1539
MJPEG Decompression Vulnerability – CVE-2009-0084
Legacy File Format Vulnerability - CVE-2009-0220
Integer Overflow Vulnerability - CVE-2009-0221
Legacy File Format Vulnerability - CVE-2009-0222
Legacy File Format Vulnerability - CVE-2009-0223
Memory Corruption Vulnerability - CVE-2009-0224
PP7 Memory Corruption Vulnerability - CVE-2009-0225
Legacy File Format Vulnerability - CVE-2009-0226
Legacy File Format Vulnerability - CVE-2009-0227
Memory Corruption Vulnerability - CVE-2009-0556
PP7 Memory Corruption Vulnerability - CVE-2009-1128
PP7 Memory Corruption Vulnerability - CVE-2009-1129
Heap Corruption Vulnerability - CVE-2009-1130
Data Out of Bounds Vulnerability - CVE-2009-1131
Legacy File Format Vulnerability - CVE-2009-1137

BHWMF::Who's Having Problems

Adobe – 2009 score: 40

SEC RESEARCHERS SMELL BLOOD IN THE WATER…

CVE-2009-2186, CVE-2009-2028, CVE-2009-1870, CVE-2009-1869, CVE-2009-1868, CVE-2009-1867, CVE-2009-1866, CVE-2009-1865, CVE-2009-1864, CVE-2009-1863, CVE-2009-1862, CVE-2009-1861, CVE-2009-1860, CVE-2009-1859, CVE-2009-1858, CVE-2009-1857, CVE-2009-1856, CVE-2009-1855, CVE-2009-1599, CVE-2009-1493, CVE-2009-1492, CVE-2009-1365, CVE-2009-1062, CVE-2009-1061, CVE-2009-0928, CVE-2009-0927, CVE-2009-0889, CVE-2009-0888, CVE-2009-0658, CVE-2009-0522, CVE-2009-0521, CVE-2009-0520, CVE-2009-0519, CVE-2009-0512, CVE-2009-0511, CVE-2009-0510, CVE-2009-0509, CVE-2009-0198, CVE-2009-0193, CVE-2009-0114

BHWMF::Who's Having Problems

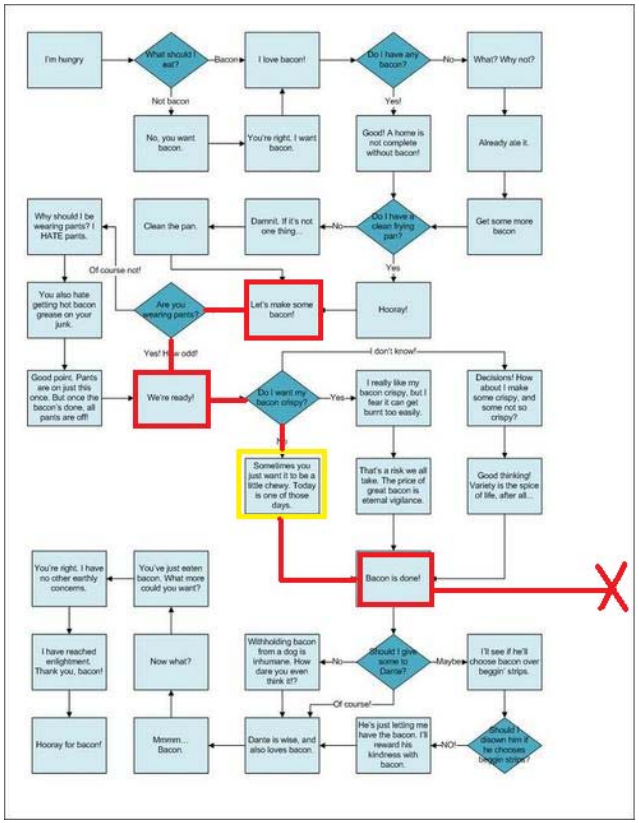
Adobe – 2009 score: 41

Another Bad Apple…

CVE-2009-0001, CVE-2009-0002, CVE-2009-0003, CVE-2009-0004, CVE-2009-0006, CVE-2009-0007, CVE-2009-0008, CVE-2009-0009, CVE-2009-0010, CVE-2009-0040, CVE-2009-0145, CVE-2009-0154, CVE-2009-0155, CVE-2009-0160, CVE-2009-0165, CVE-2009-0185, CVE-2009-0188, CVE-2009-0519, CVE-2009-0520, CVE-2009-0946, CVE-2009-0950, CVE-2009-0951, CVE-2009-0952, CVE-2009-0953, CVE-2009-0955, CVE-2009-0957, CVE-2009-0959, CVE-2009-1693, CVE-2009-1703, CVE-2009-1704, CVE-2009-1705, CVE-2009-1709, CVE-2009-1719, CVE-2009-1720, CVE-2009-1721, CVE-2009-1722, CVE-2009-1726, CVE-2009-1727, CVE-2009-1728, CVE-2009-2188, CVE-2009-2468

Bug Huntin' With Media Formats

…Studying A Format for Common Issues…

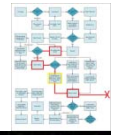


BHWMF::Common Issues > BMP Header (win)

offset	size	description
0	2	signature, must be 4D42 hex
2	4	size of BMP file in bytes (unreliable)
6	2	reserved, must be zero
8	2	reserved, must be zero
10	4	offset to start of image data in bytes
14	4	size of BITMAPINFOHEADER structure, must be 40
18	4	image width in pixels
22	4	image height in pixels
26	2	number of planes in the image, must be 1
28	2	number of bits per pixel (1, 4, 8, or 24)
30	4	compression type (0=none, 1=RLE-8, 2=RLE-4)
34	4	size of image data in bytes (including padding)
38	4	horizontal resolution in pixels per meter (unreliable)
42	4	vertical resolution in pixels per meter (unreliable)
46	4	number of colors in image, or zero
50	4	number of important colors, or zero

A typical memory allocation Issue

BHWMF::Common Issues > BMP Header (win)



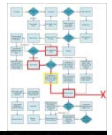
offset	size	description
0	2	signature, must be 4D42 hex
2	4	size of BMP file in bytes (unreliable)
6	2	reserved, must be zero
8	2	reserved, must be zero
10	4	offset to start of image data in bytes
14	4	size of BITMAPINFOHEADER structure, must be 40
18	4	image width in pixels
22	4	image height in pixels
26	2	number of planes in the image, must be 1
28	2	number of bits per pixel (1, 4, 8, or 24)
30	4	compression type (0=none, 1=RLE-8, 2=RLE-4)
34	4	size of image data in bytes (including padding)
38	4	horizontal resolution in pixels per meter (unreliable)
42	4	vertical resolution in pixels per meter (unreliable)
46	4	number of colors in image, or zero
50	4	number of important colors, or zero

TotalSize

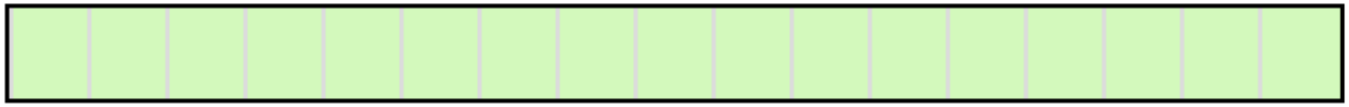
Width

Height

BHWMF::Common Issues > Memory Abuse



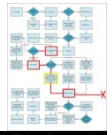
malloc (Height x Width) to create a storage buffer



TotalSize is now used in a memcpy operation to copy data from the file in memory to the allocated memory space

Eventually, we write past the allocated buffer – Causing memory corruption.

BHWMF::Common Issues > Index Abuse (JBIG2)



The original JBIG2 stream's 5th bit is modified from 0x00 to 0x40 indicating a large page association inwhich an index will be used

00 00 00 01	00	00 ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ??

Now we can load data from our base pointer + stored index offset (aka base[index])

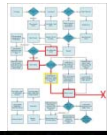
00 00 00 01	40	00	44 44 44 44	?? ?? ?? ?? ?? ?? ?? ??

In the case of the JBIG2 exploit, this allowed us (after some heap spraying to load any values we wanted in memory that led to further exploitation and finally full compromise of the target program

MOV DWORD PTR DS:[EDX+EBP*4],ECX

(EDX & EBX are 100% under our control – controlled write to anywhere in memory)

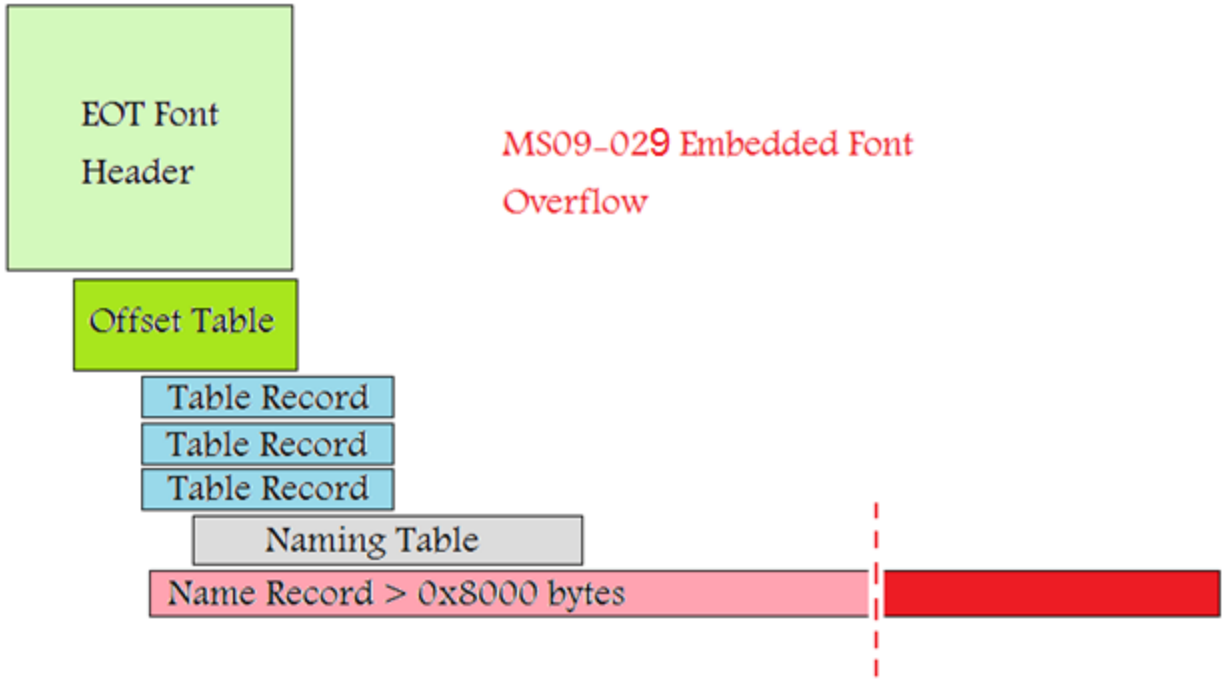
BHWMF::Common Issues > Text Segments



Many formats allocate space within specific segments to store data

-Author
-Creation Date
-Last Modified
-OS Fingerprints
-Format string bugs can exist here!
-These bugs can go unnoticed Years

<3 heap overflows
<3 buffer overflows



BHWMF::Common Issues > Looking to RFC's

RFC's are your MOFO friend!!!

Read between the lines for text like:

"should"

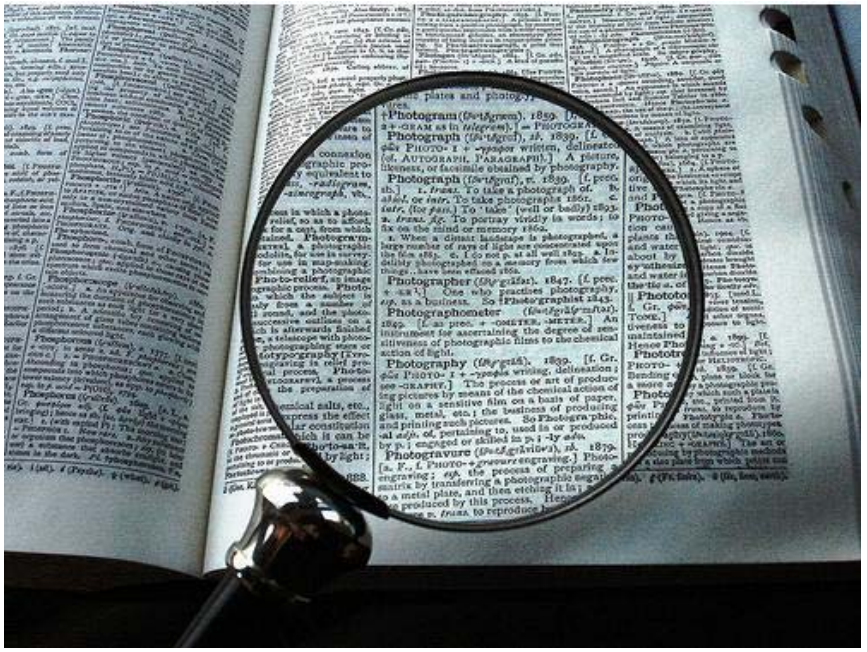
"should always"

"never"

"undefined results"

"must be"

"don't"



ALWAYS try exactly the opposite

BE OCD! Read more – Find More!

Bug Huntin' With Media Formats

Common Obstacles

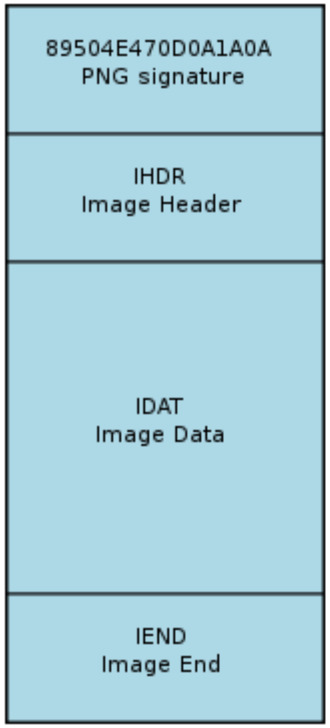
BHWMF::Common Obstacles > Basics

Compression Routines – Most media formats, these days, utilize compression algorithms. Often these must be satisfied to get any real kind of fuzzing accomplished.

Don't Ignore them completely ! sometimes this is where the vulnerabilities are!

Checksums & Indexes – If you do not account for specific fail-checking algorithms in a format, then you **WILL NOT** find the bugs beyond these sort of checks.

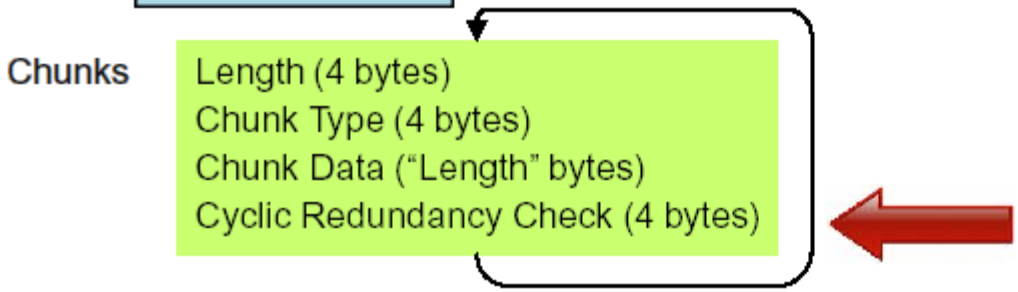
BHWMF::Common Obstacles > checksums



PNG File is a great example of a container format.

The Minimum amount of 'chunks' are shown to the left. Each chunk is made up of a structure shown below

Before sending our fuzzed files to the target application, we must first update the CRC values in any modified/fuzzed chunk



BHWMF::Common Obstacles > Index Tables

```
%PDF-1.1
```
Header

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

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

<some sections removed>

```
xref
0 8
0000000000 65535 f
0000000012 00000 n
0000000089 00000 n
0000000145 00000 n
0000000214 00000 n
0000000381 00000 n
0000000485 00000 n
0000000518 00000 n
```
Cross Reference

```
trailer
<<
 /Size 8
 /Root 1 0 R
>>
startxref
642
%%EOF
```
Trailer

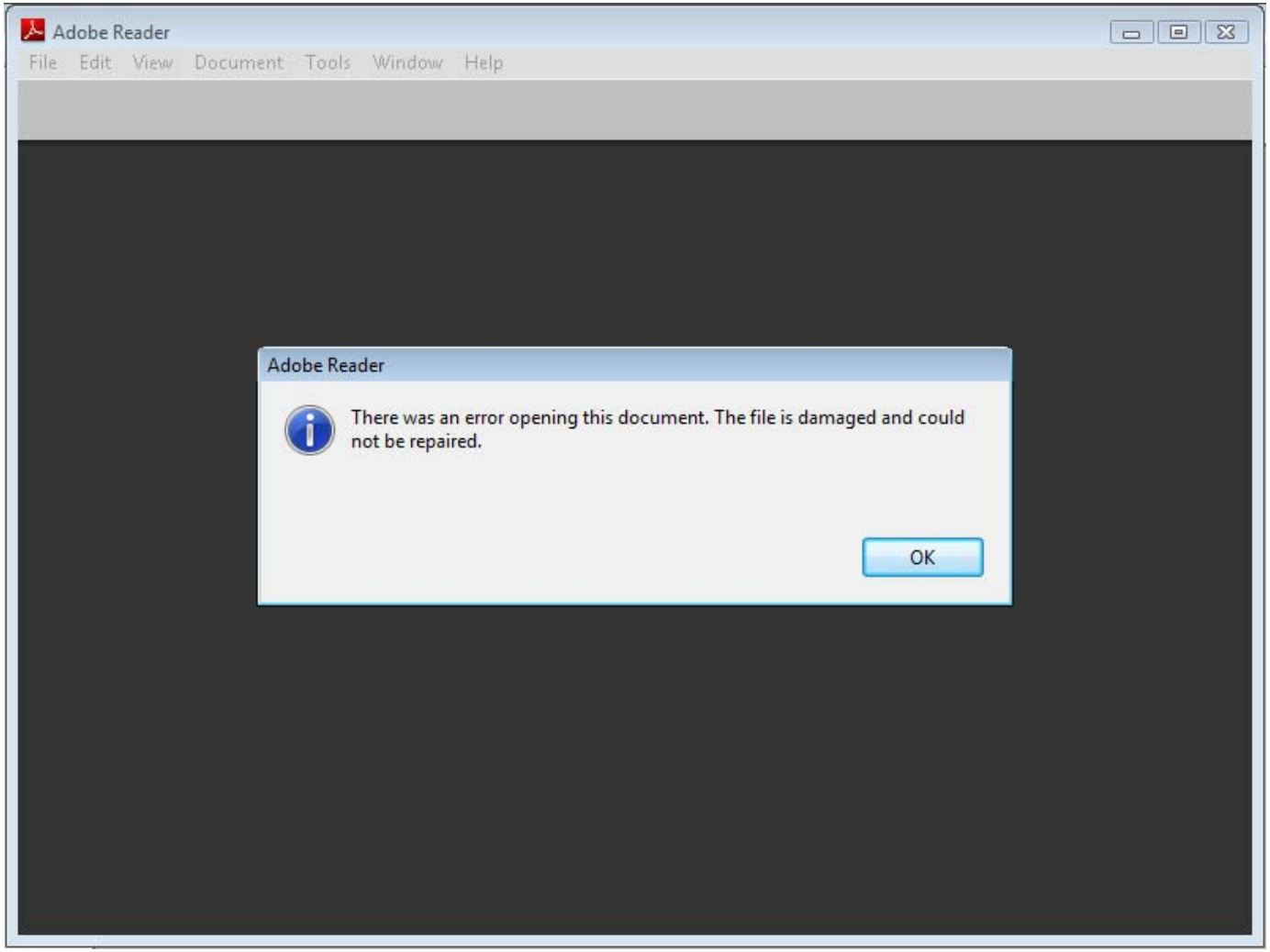
The PDF File format

1.PDF Header

2.The Trailer (with a pointer to xref)

3.The XREF table maintains offsets to the individual stream/object sections

BHWMF::Common Obstacles > Errors



Error MsgBoxes Cause problems and can make your fuzzer hang!

We can defeat them by either restarting the whole process, or sending 'Clicks' to the OK buttons…

BHWMF::Common Obstacles

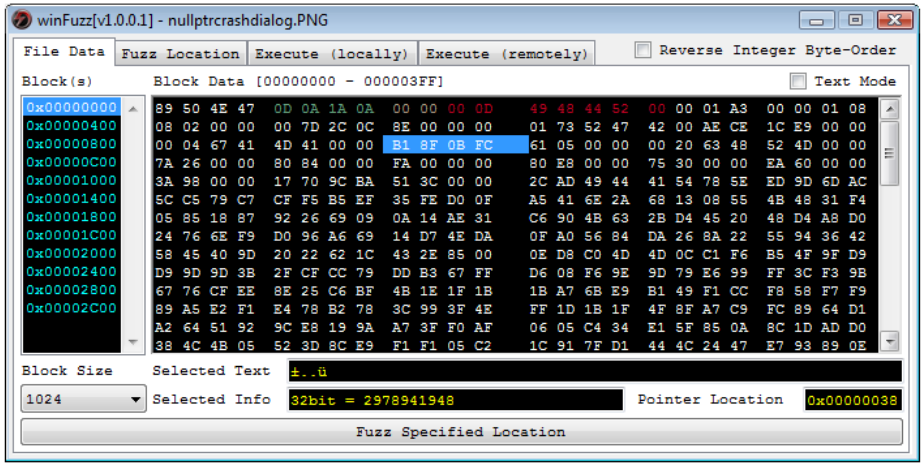
```
// Close Adobe Error MsgBox

  while (1) {
   Sleep (10);
   HWND hWnd = FindWindow(L"#32770",  L"Adobe Acrobat");
                if(hWnd) {
                  HWND hWnd2 = FindWindowEx(hWnd, 0, L"GroupBox", 0);

                  HWND hWnd3 = FindWindowEx(hWnd2, 0, L"Button", 0);

                  if  (hWnd3 != NULL) {
                                HWND i = SetActiveWindow(hWnd3);
                                if (i) {
                                              SendMessage(hWnd3, BM_CLICK, 0, 0);
                                }
                  }

   }
   hWnd = 0;
  }
```

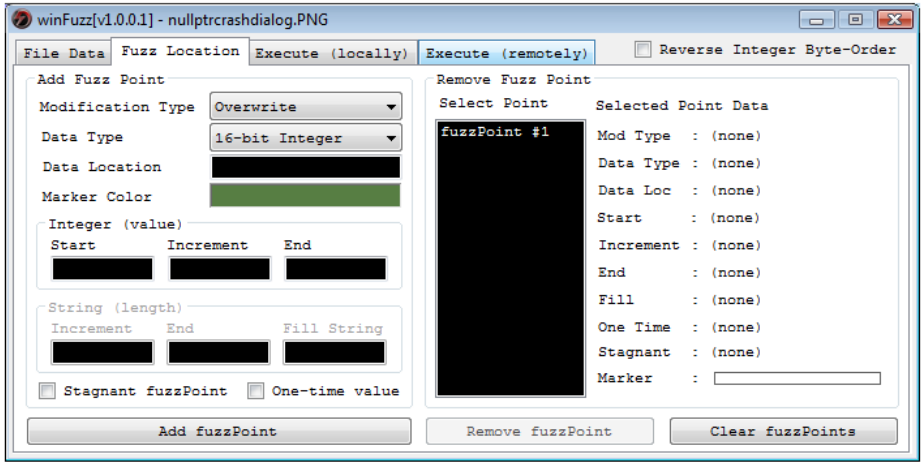
Bug Huntin' With Media Formats

Tools – taking advantage of what is already out there

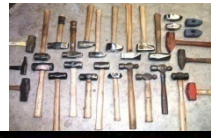
BHWMF::Existing Tools > winfuzz

- **Winfuzz by v9**
- **Allows for quick and Easy location based fuzzing**
- **Neat remote fuzzing option for bringing several computers together on one pc**





BHWMF::Existing Tools > Peachfuzz



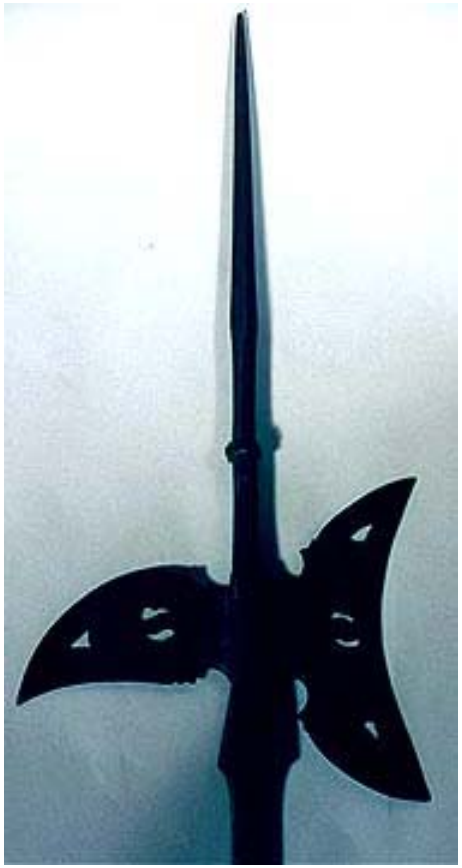
- **PeachFuzz v2**

- **Awesome versatile framework**

V2 comes with nice documentation and examples (finally!)

BHWMF::Existing Tools > SpikeFuzz

- **SpikeFuzz by Immunitysec**
- **One of the First Real Fuzzing Frameworks**
- **Nice Library base that will help tackle almost any format**



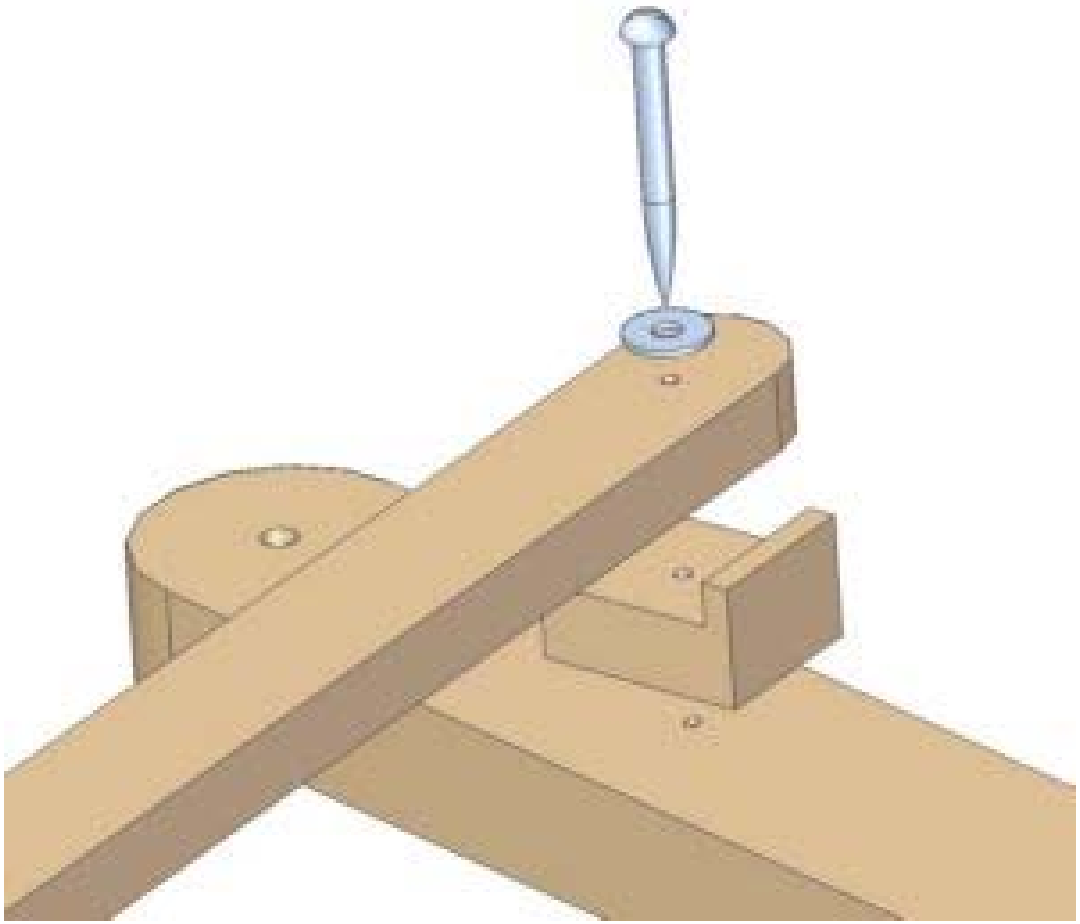
BHWMF::Existing Tools > others…

- **Hundreds of other fuzzers out there!**
- **Check some of them out:**
- **http://www.krakowlabs.com/lof.html**

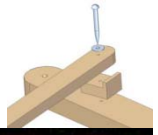
  (list of public fuzzers)

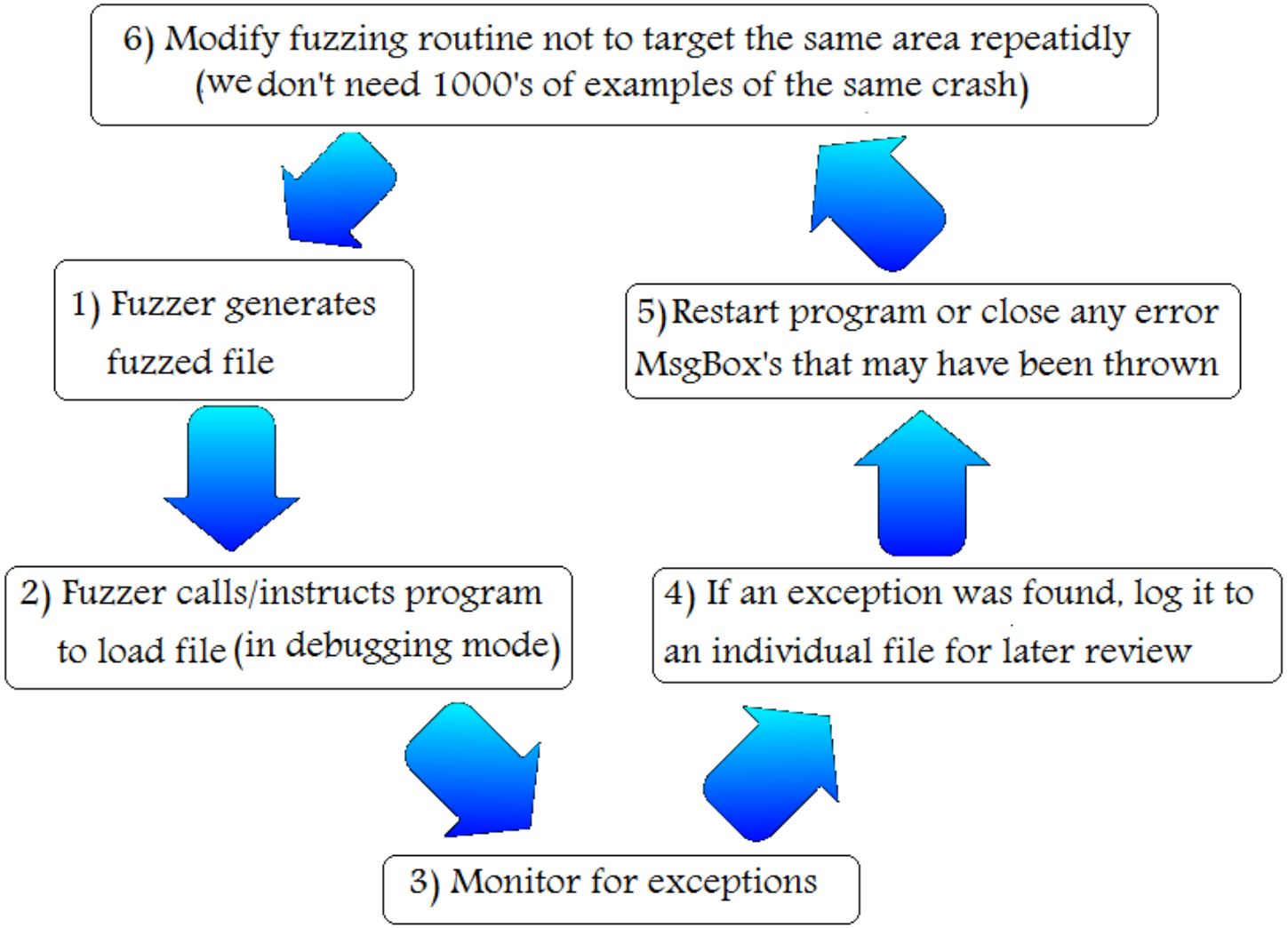
Bug Huntin' With Media Formats

- **Creating Tools For the Job**

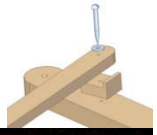


BHWMF::Tools > Basic Fuzzing Loop





BHWMF::Tools > Fuzzing Techniques

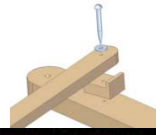


Random (blind) Fuzzing

- Modifying randomly selected areas of a file
- The beginning of headers and embedded structures will produce more results
- Bit Bashing *(ex: 0x41 to 0x61)*
- String Injection *(ex: 41414141414141…)*
- Winfuzz is great for this

Inject in more then one location! Lots more bugs will be found this way. First write enables an option, second write gives it a bad value you can catch!

BHWMF::Tools > Fuzzing Techniques



Formatted Fudging

Learn the basic layout of a files format and abide by the “rules”. Checksums must be met.

You will find more than blind fuzzing

Requires at least some programming knowledge

Try inserting data that appears similar to what might occur in the file. (Better to reference existing files for visual aid first)

BHWMF::Tools > Fuzzing Techniques

Existing Template Modification

Have your fuzzer load an existing file and recognize its internal structures

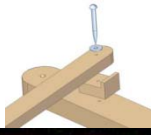
Modify one or more of these existing structures

Account for checksums or other checks

Really good technique to find those deep/hard to find bugs

Not TOO time consuming on the fuzzing side – Researching your findings can be testing though.

BHWMF::Tools > Fuzzing Techniques



Full Blown Fuzzer

Meet all of RFC's expectations

Can take weeks of programming to implement

Cheat: steal code / modify it where you can
(or trick redsand into writing it)

You will have a much greater understanding of what/where the bugs you find are - and more importantly - exactly how to exploit them.

BHWMF::Tools > Fuzzing Techniques

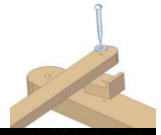
Tips on what data to fuzz with?

numbers like: "0x10000", "0x100000", "0x99999999", "65535", "65536", "65537", "16777215", "16777216", "16777217", "-268435455" "4294967294", "4294967295", "4294967296", "357913942", "-357913942", "536870912", "-536870912", "0", "-0", "1", "-1", "32767", "-32768", "2147483647", "-2147483647", "2147483648", "-2147483648", "1.79769313486231E+308", "3.39519326559384E-313", "99999999999", "-99999999999", "0x100", "0x1000",

Strings like: “A”x100, “A”x1000, “A”x10000, "%s%p%x%d", "%.1024d", "%.1025d", "%.2048d", "%.2049d", "%.4096d", "%.4097d", "%99999999999s", "%08x", "%%20n", "%%20p", "%%20s", "%%20d", "%%20x",

These values will help weed out boundary issues, buffer/heap overflows, and format string vulnerabilities

BHWMF::Tools > Fuzzing Techniques



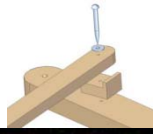
Controlling the target – the DDE / OLE way

Some applications like Adobe Acrobat/Reader can be simply controlled through transport control mediums such as OLE/DDE.

We can create basic loops that loop through DDE controls like such:

```
acrobat.Exec('[DocClose("%s")]' % doc)

acrobat.Exec('[FileOpen("%s")]' % doc)

acrobat.Exec('[DocOpen("%s")]' % doc)

acrobat.Exec('[DocGoTo("%s", %d)]' % (doc, page))
```

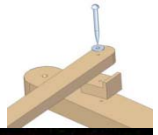
BHWMF::Tools > PeachFuzz Example



PeachFuzz GUI fuzzing example (1/4 – Intro code)

```
<?xml version="1.0" encoding="utf-8"?>
  <Peach xmlns="http://phed.org/2008/Peach" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
        xsi:schemaLocation="http://phed.org/2008/Peach ../peach.xsd" version="1.0"
        author="Michael Eddington">

      <!-- Import defaults for Peach instance -->
      <Include ns="default" src="file:defaults.xml" />

     <!-- Define our file format DDL -->
     <DataModel name="FileData">
           <String value="Hello World!" />
     </DataModel>
```

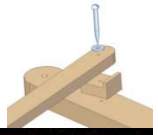
BHWMF::Tools > PeachFuzz Example



PeachFuzz GUI fuzzing example (2/4 – Defining Publishes Actions)

```
<!-- Define a simple state machine that will write the file and then launch a
     program using the FileWriterLauncher publisher -->

<StateModel name="State" initialState="Initial">
    <State name="Initial">
        <Action type="open" />

        <!-- Write out contents of file -->
        <Action name="WriteFile" type="output">
            <DataModel ref="FileData" />
        </Action>

        <!-- Close file -->
        <Action type="close" />

        <!-- Launch the file consumer -->
        <Action type="call" method="notepad.exe"/>
    </State>
</StateModel>
```

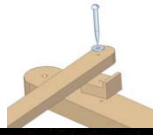
BHWMF::Tools > PeachFuzz Example



PeachFuzz GUI fuzzing example (3/4 – monitor code)

```
<!-- Setup a local agent that will monitor for faults -->
<Agent name="LocalAgent">
    <Monitor class="debugger.WindowsDebugEngine">

        <!-- The command line to run. Notice the filename provided matched up to
             what is provided below in the Publisher configuration -->
        <Param name="CommandLine" value="c:\windows\system32\notepad.exe fuzzedfile.txt" />

        <!-- This parameter will cause the debugger to wait for an action-call in the state
             model with a method="notepad.exe" before running program. -->
        <Param name="StartOnCall" value="notepad.exe" />
    </Monitor>

    <!-- Enable heap debugging on our process as well. -->
    <Monitor class="process.PageHeap">
        <Param name="Executable" value="notepad.exe"/>
    </Monitor>
</Agent>
```

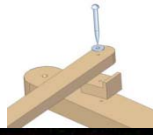
BHWMF::Tools > PeachFuzz Example



PeachFuzz GUI fuzzing example (4/4 – recording results)

```
<Test name="TheTest">
      <Agent ref="LocalAgent" />
      <StateModel ref="State"/>

      <!-- Configure our publisher with correct filename to write too -->
      <Publisher class="file.FileWriterLauncherGui">
            <Param name="fileName" value="fuzzedfile.txt" />
            <Param name="windowName" value="Notepad" />
            <Param name="debugger" value="true"/>
       </Publisher>
</Test>

<Run name="DefaultRun">
      <Test ref="TheTest" />
      <Logger class="logger.Filesystem">
           <Param name="path" value="c:\peach\logtest" />
       </Logger>
</Run>
</Peach>
<!-- end -->
```

BHWMF::Tools > Fuzzing Libraries!!!



One Last Important note….

- **Never write a fuzzer for a specific program.**
- **Try writing library functions that return your fuzzed data**
- **This will allow you to be flexible and fuzz multiple applications**
- **Also, allows you to hit things you wouldn't be able to before**

(i.e. call a generate_jpeg() subfunction and then use that in your generate_pdf() code!)

Bug Huntin' With Media Formats

- **Analyzing Data**

BHWMF::Analyzing Data

UNPATCHED PDF vulnerability (Stack Overflow in /B bit rate option)

Found by simply overwriting a newline with a digit – Changing 16 to 168

C:\...Settings\ruckuss\Desktop\pdf-sounds\findings\trash\on_open_aiff_pdfa2_1031 - makes adobe exit.pdf

```
/Type /XObject
>>
stream
H%(02)(08)0(00)(00)(00)(00)(01)
endstream
endobj
25 0 obj
<</B 168/E /Signed

/Filter /FlateDecode
/Length 54390
/R 5000
/Type /Sound
>>
stream
H%(0C)<{TTu(02)€□¿ûœ{çÎÌ□□AÁÇ(1E)SS(11)-(1F)Hé-□²(07)(03)+3(
"(04)W(13)5!m□ùX%°2ÑÝ'73sßißz¾s¾óýó(01)€l»#(04)(à(05)A□(00
(06)ƒÁÈ@-¨(0E)□D‡ÑNTƒêÑ''l(03),ÈC
ð€å`(0B)x
(10)H(07)ù(13)‡A(1E)¸
²Ñ~Ô  † np(0B)òÀûäêA(11)Tê,°% `ip(0C)¸  ¶,T(10)‡Þu,ð(1
;Œ](02)qÎVÇk(05),Ì°(01)(06)(12)¹»EìÆÓ°-@s(0E)Z¶=(1B)¶R"¼(10)
```

Ln: 1 Col: 1/9 Ch: 1/9 EOL: CR | 1252 | Mixed

C:\Documents and Settings\ruckuss\Desktop\pdf-sounds\findings\trash\on_open_aiff_pdfa.pdf

```
/Type /XObject
>>
stream
H%(02)(08)0(00)(00)(00)(00)(01)
endstream
endobj
25 0 obj
<</B 16
/E /Signed
/Filter /FlateDecode
/Length 54390
/R 5000
/Type /Sound
>>
stream
H%(0C)<{TTu(02)€□¿ûœ{çÎÌ□□AÁÇ(1E)SS(11)-(1F)Hé-□²(07)(03)+3(
"(04)W(13)5!m□ùX%°2ÑÝ'73sßißz¾s¾óýó(01)€l»#(04)(à(05)A□(00
(06)ƒÁÈ@-¨(0E)□D‡ÑNTƒêÑ''l(03),ÈC
ð€å`(0B)x
(10)H(07)ù(13)‡A(1E)¸
²Ñ~Ô  † np(0B)òÀûäêA(11)Tê,°% `ip(0C)¸  ¶,T(10)‡Þu,ð(1
;Œ](02)qÎVÇk(05),Ì°(01)(06)(12)¹»EìÆÓ°-@s(0E)Z¶=(1B)¶R"¼(10)
```

Ln: 186 Col: 9/9 Ch: 7/7 EOL: CR | 1252 | Mixed

BHWMF::Analyzing Data

Another vulnerability found within 10 mins of fuzzing (Some appear to be not useful but just need about a weeks worth of tinkerin'!)

- Sometimes it takes minutes to find a bug but weeks to develop an exploit for! :P

C:\...ttings\ruckuss\Desktop\pdf-sounds\findings\trash\on_open_aiff_pdfa - FEEEFEEE crash on esi DEC.pdf

```
71 0 obj
<</N3GS0 72 0 R
>>
endobj
72 0 obj
<</Type /ExtGState
/TR2 /Default
/ca 1.0
/CA 1.0
>>
endobj
73 0 obj
<</Length 49
>>
stream
 q /RelativeColorimetric ri /NxGS0 gs /NxX0 Do Q
endstream
endobj
xref
0 74
0000000000 65535 f
0000000000 00000 f
```

Ln: 1548 Col: 1/19 Ch: 1/19 EOL: CRLF | 1252 | Mixed

C:\Documents and Settings\ruckuss\Desktop\pdf-sounds\findings\trash\on_open_aiff_pdfa.pdf

```
71 0 obj
<</NxGS0 72 0 R
>>
endobj
72 0 obj
<</Type /ExtGState
/TR2 /Default
/ca 1.0
/CA 1.0
>>
endobj
73 0 obj
<</Length 49
>>
stream
 q /RelativeColorimetric ri /NxGS0 gs /NxX0 Do Q
endstream
endobj
xref
0 74
0000000000 65535 f
0000000000 00000 f
```

Ln: 1526 Col: 7/7 Ch: 7/7 EOL: CR | 1252 | Mixed

BHWMF::EOT