

The Art of File Format Fuzzing

iDEFENSE Labs

```

0002b510h: 3A 00 00 00 68 00 74 00 74 00 70 00 3A 00 2F 00 ; :...h.t.t.p.:./
0002b520h: 2F 00 77 00 77 00 77 00 2E 00 77 00 61 00 79 00 ; /.w.w.w...w.a.y.
0002b530h: 6E 00 65 00 67 00 72 00 65 00 74 00 7A 00 6B 00 ; n.e.g.r.e.t.z.k.
0002b540h: 19 00 6E 00 63 00 6E 00 00 00 1F 00 00 00 00 00 ; m./.....
0002b550h: 5A 3A 0E 00 00 00 43 00 61 00 6E 00 61 00 64 00 ; z:...a.n.a.d.
0002b560h: 61 00 00 00 1F 00 00 00 00 00 6E 00 ; a.....\:...O.n.
0002b570h: 74 00 61 00 72 00 69 00 6F 00 00 00 1F 00 5B 3A ; t.a.r.i.o.....[:
0002b580h: 10 00 00 00 57 00 39 00 47 00 20 00 39 00 57 00 ; ...W.9.G. .9.W.
0002b590h: 39 00 00 00 1F 00 59 3A 14 00 00 00 42 00 72 00 ; 9.....Y:...B.r.
0002b5a0h: 61 00 6E 00 74 00 66 00 6F 00 72 00 64 00 00 00 ; a.n.t.f.o.r.d...
0002b5b0h: 1F 00 5D 3A 20 00 00 00 31 00 32 00 33 00 20 00 ; ..]: ...1.2.3. .
0002b5c0h: 4D 00 61 00 69 00 6E 00 20 00 53 00 74 00 72 00 ; M.a.i.n. .S.t.r.
0002b5d0h: 65 00 65 00 74 00 00 00 03 00 71 3A 04 00 00 00 ; e.e.t.....q:...
0002b5e0h: 00 00 10 00 03 00 55 3A 04 00 00 00 00 00 00 00 ; .....U:.....
0002b5f0h: 1F 00 02 30 0A 00 00 00 53 00 4D 00 54 00 50 00 ; ...0....S.M.T.P.
0002b600h: 00 00 1F 00 03 30 24 00 00 00 77 00 61 00 79 00 ; .....0$.w.a.y.
0002b610h: 6E 00 65 00 40 00 67 00 72 00 65 00 74 00 7A 00 ; n.e.@.g.r.e.t.z.
0002b620h: 6B 00 79 00 2E 00 63 00 6F 00 6D 00 00 00 1F 10 ; k.y...c.o.m.....
0002b630h: 54 3A 01 00 00 00 0E 00 00 00 0A 00 00 00 53 00 ; T:.....S.
0002b640h: 4D 00 54 00 50 00 00 00 1F 10 56 3A 01 00 00 00 ; M.T.P....V:....

```

Michael Sutton – msutton@idefense.com

Adam Greene – agreene@idefense.com

- Who we are
- What you can expect from the presentation
- Agenda
 - Background
 - File format fuzzing
 1. Identifying targets
 2. Creating files
 3. Executing files
 4. Monitoring for exceptions
 5. Identifying vulnerabilities
 - Tool Demos
 - 0day Vulnerabilities
 - Conclusion

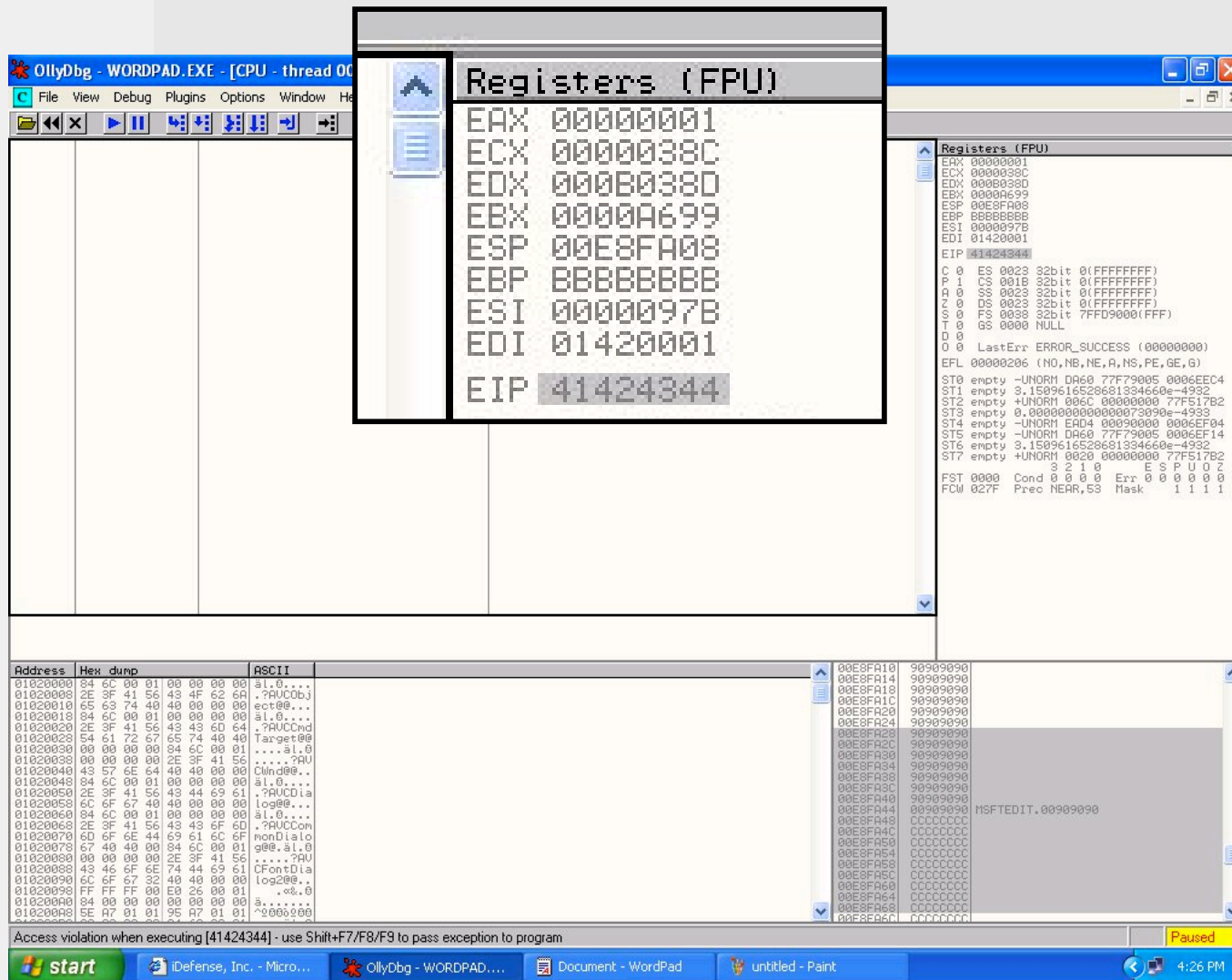
Background – What is file format fuzzing?

- File format → Protocol
 - Standardized means of communication
- Non-standard formats
 - Applications should be capable of dealing with anomalies
 - Input validation controls
 - Exception handlers
 - Error reporting
- What happens when controls aren't in place?
 - Buffer overflows
 - Integer overflows
 - Signedness issues
 - Invalid memory references
 - Infinite loops

Background – Historical vulnerabilities

- MS05-009 – Vulnerability in PNG Processing Could Allow Remote Code Execution
- MS05-002 - Vulnerability in Cursor and Icon Format Handling Could Allow Remote Code Execution
- MS04-041 - Vulnerability in WordPad Could Allow Code Execution
- MS04-028 - Buffer Overrun in JPEG Processing (GDI+) Could Allow Code Execution
- US-CERT TA04-217A – Multiple Vulnerabilities in libpng (Affecting Mozilla, Netscape, Firefox browsers)
- CAN-2004-1153 – Format String Vulnerabilities in Adobe Acrobat Reader

Background - MS04-041 MS Word Buffer Overflow



Registers (FPU)

EAX	00000001
ECX	0000038C
EDX	000B0380
EBX	0000A699
ESP	00E3FA08
EBP	BBBBBBBB
ESI	0000097B
EDI	01420001
EIP	41424344

Registers (FPU)

EAX	00000001
ECX	0000038C
EDX	000B0380
EBX	0000A699
ESP	00E3FA08
EBP	BBBBBBBB
ESI	0000097B
EDI	01420001
EIP	41424344
C 0	ES 0023 32bit 0(FFFFFFFF)
P 1	CS 001B 32bit 0(FFFFFFFF)
A 0	SS 0023 32bit 0(FFFFFFFF)
Z 0	DS 0023 32bit 0(FFFFFFFF)
S 0	FS 0038 32bit 7FFD9000(FFF)
T 0	GS 0000 NULL
D 0	
O 0	LastErr ERROR_SUCCESS (00000000)
EFL	00000206 (NO, NB, NE, A, NS, PE, GE, G)
ST0	empty -UNORM DA60 77F79005 0006EED4
ST1	empty 3.1509616528661334660e-4932
ST2	empty +UNORM 00C 00000000 77F517B2
ST3	empty 0.0000000000000073090e-4933
ST4	empty -UNORM EAD4 00090000 0006EF04
ST5	empty -UNORM DA60 77F79005 0006EF14
ST6	empty 3.1509616528661334660e-4932
ST7	empty +UNORM 0020 00000000 77F517B2
FST	0000 3 2 1 0 E S P U O Z D
FCW	027F Prec NEAR, S3 Mask 1 1 1 1 1

Address Hex dump ASCII

01020000	84 6C 00 01 00 00 00 00	al.0....
01020003	2E 3F 41 56 43 4F 62 60	?AU00j
01020010	65 63 74 40 40 00 00 00	ect00...
01020013	84 6C 00 01 00 00 00 00	al.0....
01020020	2E 3F 41 56 43 43 6D 64	?AUChd
01020023	64 61 72 67 65 74 40 40	Target00
01020030	00 00 00 00 84 6C 00 01	...al.0
01020033	00 00 00 00 2E 3F 41 56?AU
01020040	43 57 6E 64 40 40 00 00	CWind00..
01020043	84 6C 00 01 00 00 00 00	al.0....
01020050	2E 3F 41 56 43 44 69 61	?AUCLia
01020053	6C 6F 67 40 40 00 00 00	log00...
01020060	84 6C 00 01 00 00 00 00	al.0....
01020063	2E 3F 41 56 43 43 6F 6D	?AUCom
01020070	6D 6F 6E 44 69 61 6C 6F	nonDialo
01020073	67 40 40 00 84 6C 00 01	00al.0
01020080	00 00 00 00 2E 3F 41 56?AU
01020083	43 46 6F 6E 74 44 69 61	ConDila
01020090	6C 6F 67 32 40 40 00 00	log200..
01020093	FF FF FF 00 E0 26 00 01	..%.0
010200A0	84 00 00 00 00 00 00 00	ä.....
010200A3	6E A7 01 01 25 22 01 01	*000100

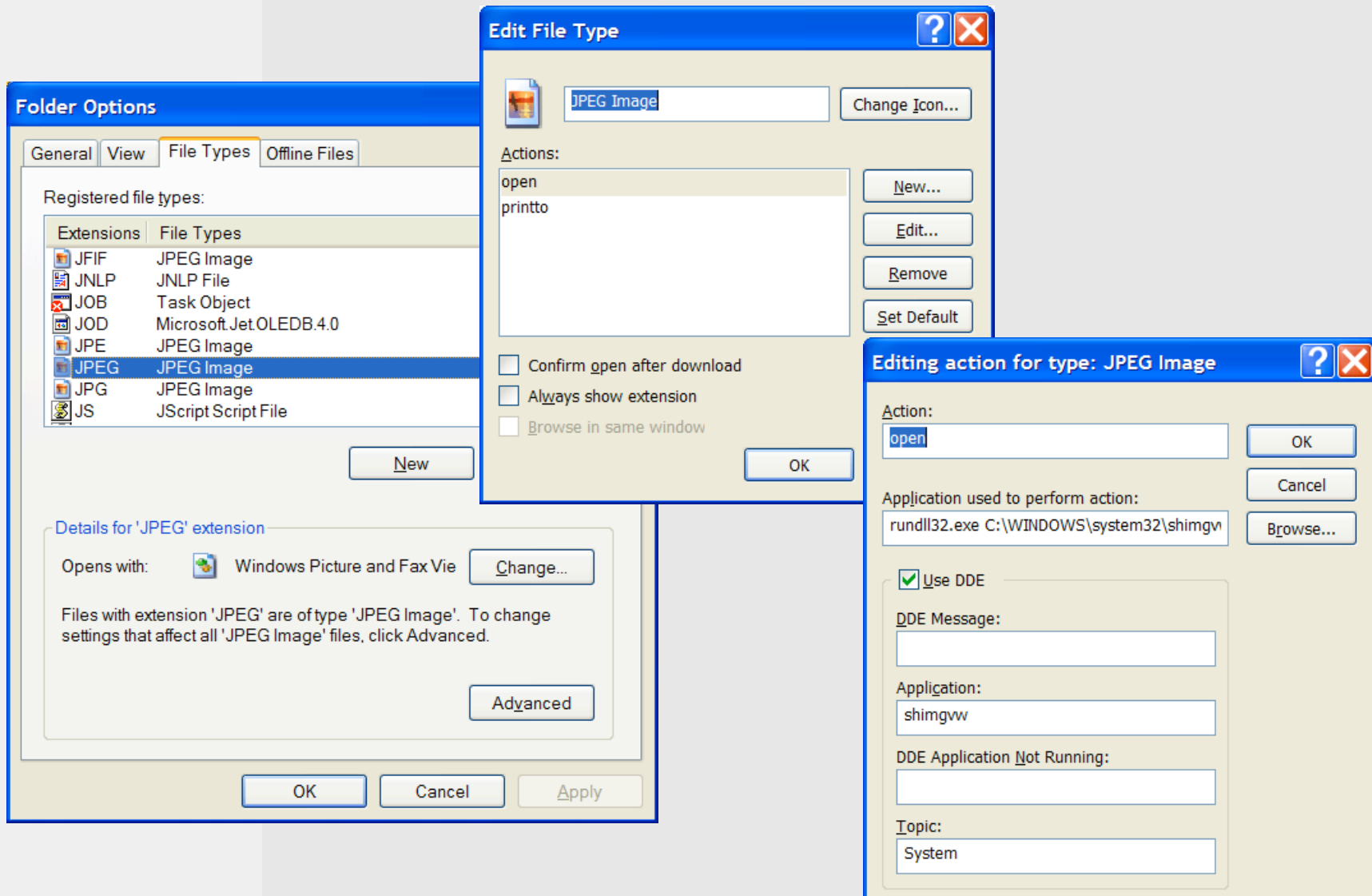
Access violation when executing [41424344] - use Shift+F7/F8/F9 to pass exception to program

start | iDefense, Inc. - Micro... | OllyDbg - WORDPAD... | Document - WordPad | untitled - Paint | 4:26 PM

- **Uneducated users**
 - Users are less likely to be wary of launching non-executable files from untrusted sources
- **Default configurations**
 - Applications designed for convenience allow processing of many untrusted files without user intervention
 - Many image files will be rendered in web browsers
- **Lack of layered security**
 - Complete network compromise can result from a single user's trusted actions (i.e. web browsing) using a 0day file format vulnerability

- **File types**
 - Binary
 - Formatted documents (doc, rtf, pdf, etc.)
 - Images (jpg, gif, png, etc.)
 - Media files (mpg, wav, avi, mov, mp3, etc.)
 - ASCII
 - XML
 - INI
- **Default applications**
 - Registered file types
 - Windows – Explorer & RegEdit
 - URI handlers
 - Windows - Explorer & RegEdit

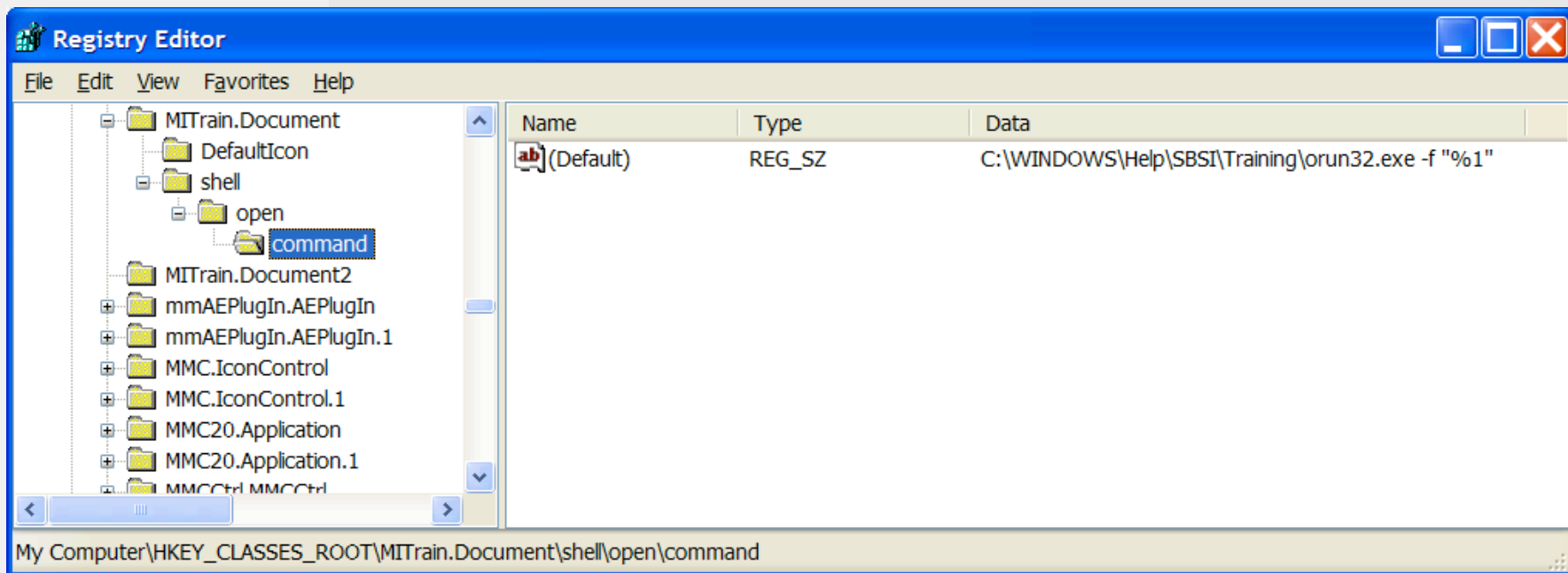
File Fuzzing – Registered file types



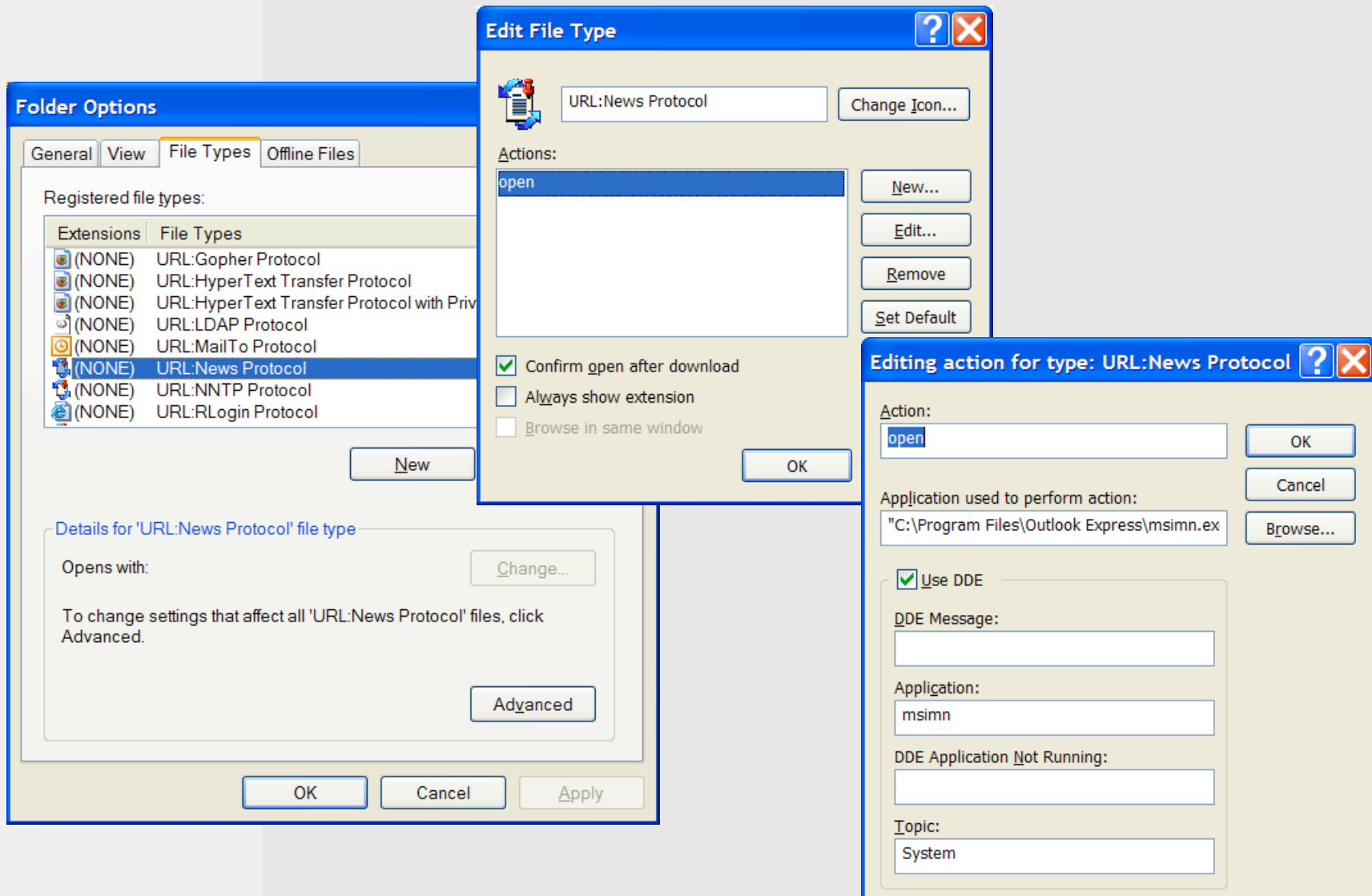
The image shows three overlapping Windows dialog boxes related to file type configuration:

- Folder Options:** The 'File Types' tab is active. It shows a list of registered file types. The 'JPEG' extension is selected, which is associated with the 'JPEG Image' file type. Below the list, it shows that files with the 'JPEG' extension are of type 'JPEG Image' and are opened with 'Windows Picture and Fax Viewer'. There are 'OK', 'Cancel', and 'Apply' buttons at the bottom.
- Edit File Type:** This dialog is for editing the 'JPEG Image' file type. It shows the name 'JPEG Image' and a 'Change Icon...' button. Under the 'Actions' section, 'open' and 'printto' are listed. There are buttons for 'New...', 'Edit...', 'Remove', and 'Set Default'. At the bottom, there are checkboxes for 'Confirm open after download', 'Always show extension', and 'Browse in same window', along with an 'OK' button.
- Editing action for type: JPEG Image:** This dialog is for configuring the 'open' action. The 'Action' is set to 'open'. The 'Application used to perform action' is 'rundll32.exe C:\WINDOWS\system32\shimgvw'. There are 'OK', 'Cancel', and 'Browse...' buttons. The 'Use DDE' checkbox is checked. Below it, there are fields for 'DDE Message:', 'Application:' (set to 'shimgvw'), 'DDE Application Not Running:', and 'Topic:' (set to 'System').

File Fuzzing – Registered file types



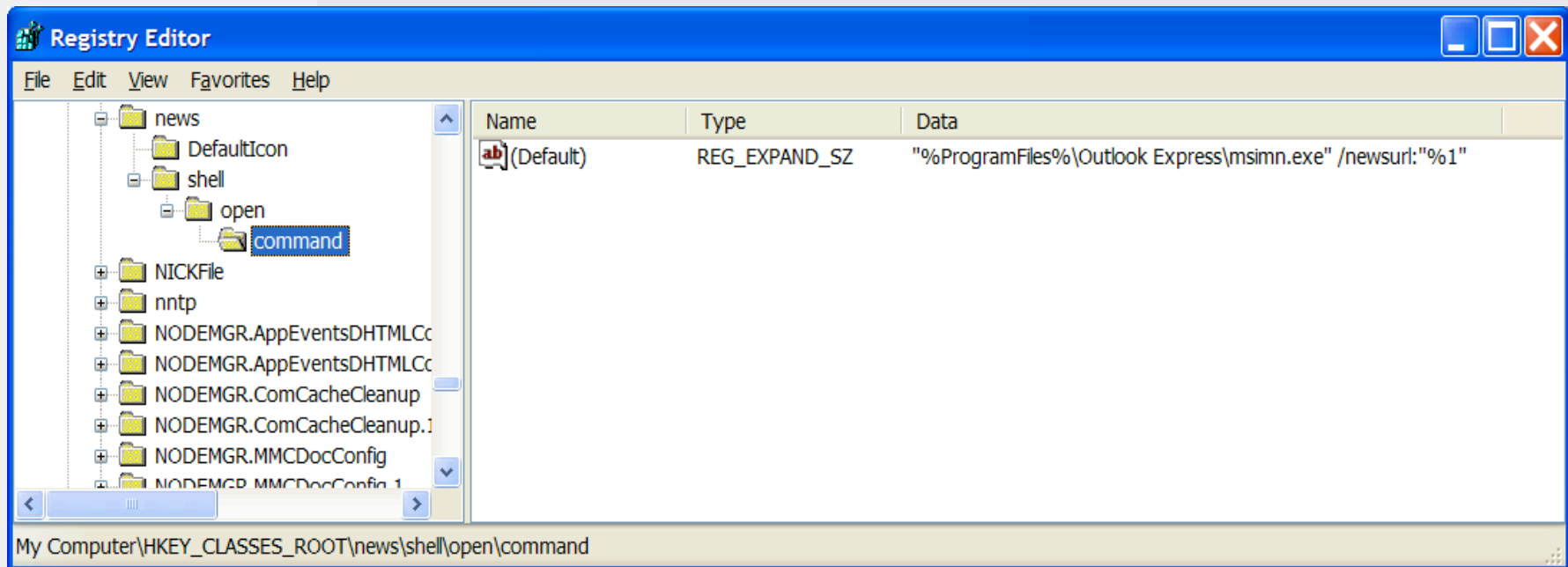
File Fuzzing – URI handlers



The image shows three overlapping Windows dialog boxes related to file type configuration:

- Folder Options:** The 'File Types' tab is active. The 'Registered file types' list includes 'URL:News Protocol' (extension: NONE). A 'New' button is visible at the bottom.
- Edit File Type:** This dialog is for 'URL:News Protocol'. It shows the 'Actions' list with 'open' selected. Checkboxes include 'Confirm open after download' (checked), 'Always show extension' (unchecked), and 'Browse in same window' (unchecked). Buttons for 'New...', 'Edit...', 'Remove', and 'Set Default' are present.
- Editing action for type: URL:News Protocol:** This dialog shows the 'Action' set to 'open' and the 'Application used to perform action' set to '"C:\Program Files\Outlook Express\msimn.exe"'. The 'Use DDE' checkbox is checked. Fields for 'DDE Message:', 'Application:', 'DDE Application Not Running:', and 'Topic:' are also visible.

File Fuzzing – URI handlers



- **Interesting Targets on Linux**
 - Antivirus products
 - Fuzzing Linux AV engines locally can lead to a remote vulnerability
 - Media Players
 - RealPlayer
 - Document Viewers
 - Adobe Acrobat Reader
 - Web Browsers
 - Think image formats

- Brute force – manipulating all bytes
 - Data types
 - Integers
 - (Un)signed byte
 - (Un)signed word
 - (Un)signed dword
 - ASCII
 - C-style strings
 - » ASCII string with a terminating NULL
 - XDR-style length tagged strings
 - » SUNRPC: ASCII string padded out to %4, 4 byte MSB length prepended
 - Other common length tagged strings
 - » 1 byte length prepended/appended
 - » 2 byte length prepended/appended

- Picking interesting values
 - Integers
 - Negative numbers (0xffffffff, 0x80000000, etc)
 - Large numbers (0x7fffffff, 0x20000000, etc)
 - Small values such as 0-10 (MS04-028)
 - Header values identifying the length of header/data segments
 - ASCII
 - Large strings / empty strings
 - Strings with “inaccurate” length tags
 - Long string, short tag
 - Short string, long tag
 - Strings with “accurate”, but long length tags (MS05-002, MS05-009, MS04-041)
 - Strings with format specifiers (CAN-2004-1153)

- **Why are these values so interesting?**
 - Decrementing small integers can cause them to wrap
 - Multiplying, adding, and incrementing large integers can cause them to wrap
 - Inconsistent methods for determining size can lead to overflows
 - Mixing up the true size of a string with the value the file has specified for it
 - Using user supplied data as a format string is obviously dangerous

- Brute force fuzzing pros/cons
 - Pros
 - No information about the file format is necessary
 - Automation of executing applications
 - Automation of detecting of exceptions
 - Cons
 - Difficult to identify/correct other dependent values (i.e. CRC-32 checksums)
 - Less efficient than intelligent fuzzing
 - Many false positives

- Intelligent fuzzing
 - Researching open file formats
 - Standards groups
 - ISO - <http://www.iso.org/>
 - W3C - <http://www.w3.org/>
 - Graphics (JPEG, PNG, SVG, etc.)
 - W3C - <http://www.w3.org/Graphics/>
 - Audio (MIDI, MP3, WAV, etc.)
 - MIDI - <http://www.midi.org/about-midi/specinfo.shtml>
 - Compressed/Archive (ZIP, TAR, RAR, etc.)
 - ZIP - <http://www.pkware.com/company/standards/appnote/appnote.txt>
 - Binary (a.out, ELF, COFF)
 - Microsoft – PE & COFF
<http://www.microsoft.com/whdc/system/platform/firmware/PECOFF.msp>

- Intelligent fuzzing (cont'd)
 - Researching proprietary file formats
 - Previous reverse engineering
 - Your good friend Google
 - File diffing
 - Headers vs. data
 - Header name/value pairs
 - Resources for multiple file format specs
 - <http://www.wotsit.org/>
 - <http://www.sonicspot.com/guide/fileformatlist.html>

- Intelligent fuzzing pros/cons
 - Pros
 - Can fuzz every field of the file properly
 - Can target “interesting” fields
 - Can ensure that lengths across blocks remain valid
 - Can ensure that CRC-32 values and other arbitrary calculations across blocks stay valid
 - Cons
 - The fuzz is only as complete as your file definition (fileSPIKE script)
 - You may need many different fileSPIKE scripts for one format to test out of order fields, files with different capabilities, etc
 - Constructing a thorough set of scripts can be time consuming

- Executing/processing files

- Continual execution
 - Scripting
 - GUI/console apps
- Timed termination
 - Windows
 - `taskkill /PID [PID]`
 - Windows API - i.e. `killProcess()`
 - *nix
 - `kill pid`
 - UNIX API – i.e. `kill()`

- **Browser Based File processing**
 - To test file processing code in browsers and ActiveX controls (images, media files, etc.)
 - Continual execution
 - META REFRESH cgi
 - Same method used in mangleme by lcamtuf
 - Timed termination
 - Not required

File Fuzzing – Monitoring for exceptions

- Identifying exception handlers
 - Function hooking
 - Debugging library/API
 - Linux ptrace
- Standard output/error
- Error logs
 - Microsoft event viewer
 - Application logs
- Application crash
 - Unhandled exceptions
- Return value

- **Stack overflows**
 - Microsoft Interactive Training Buffer Overflow
- **Heap overflows**
 - GNU Binutils readelf
- **Integer overflows**
 - Microsoft JPEG/GDI+ (MS04-028)
- **Format Strings**
 - Adobe Acrobat Reader (CAN-2004-1153)

Linux – SPIKEfile and notSPIKEfile

```
user@host$ cat png.spk
s_binary("89 50 4e 47 0d 0a 1a 0a"); // signature
s_block_start("len");
s_block_start("crc");
s_binary("49 48 44 52"); // IHDR
s_block_start("len");

s_int_variable(0x1,1); // width, 4bytes msb
s_int_variable(0x1,1); // height, 4bytes msb
s_int_variable(0x8,3); // depth 1 2 4 8 16

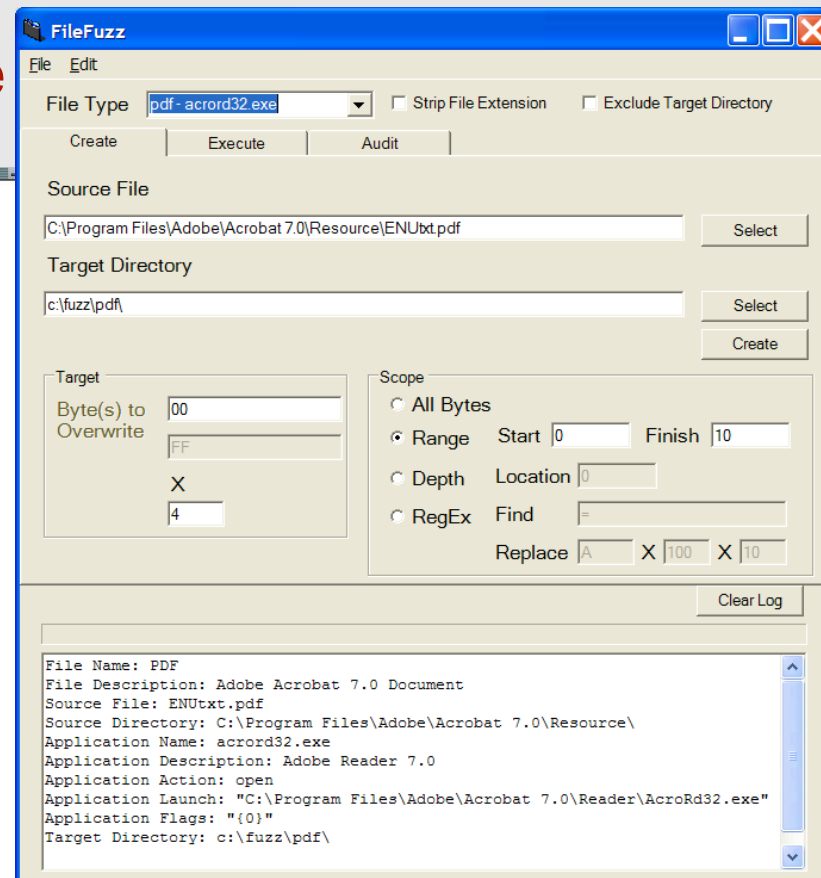
s_int_variable(0x3,3); // color type 0 2 3 4 6, 3 requires plte
s_int_variable(0x0,3); // only supported compress type
s_int_variable(0x0,3); // only supported filter method
s_int_variable(0x0,3); // 0, no interlace

s_block_end("len");
s_block_start("pncrc_word_bigendian("crc");
s_block_end("crc");
// END IHDR

//PLTE
s_block_start("len2");
s_block_start("crc2");
s_binary("50 4c 54 4b");
s_block_start("len2");
s_string_variable("AAABBBCCC"); // palette entries, must be 3 bytes
s_block_end("len2");
s_block_start("pncrc_word_bigendian("crc2");
s_block_end("crc2");
// END PLTE

//tRNS
s_block_start("len-trns"); // MS05-009
s_block_start("crc-trns");
s_binary("74 52 4e 53");
s_block_start("len-trns");
//
s_string_variable("ABC"); // palette entries, must be 3 bytes
// -----
s_block_end("len-trns");
s_block_start("pncrc_word_bigendian("crc-trns");
s_block_end("crc-trns");
//END tRNS

// IDAT
s_block_start("len3");
s_block_start("crc3");
s_binary("49 44 41 54");
s_block_start("len3");
//
s_binary("78 9c 63 60 00 00 02 00 01");
s_block_end("len3");
s_block_start("pncrc_word_bigendian("crc3");
s_block_end("crc3");
// END IDAT
// IEND
s_block_start("len4");
s_block_start("crc4");
s_binary("49 45 4e 44");
s_block_start("len4");
s_block_end("len4");
s_block_start("pncrc_word_bigendian("crc4");
s_block_end("crc4");
user@host$
```



Windows - fileFUZZ

Linux – SPIKEfile

- Simple adaptation of Immunity, Inc SPIKE
 - Modified to target files
 - Flexible execution and exception monitoring using ptrace
 - Multiple processes
 - CRC-32 over block support using
 - Takes .spk scripts as input

*Used to discover RealPlayer RealText Format String bug

Linux – notSPIKEfile

- Simple baseline fuzzer

- Requires a valid file to work from
- Flexible execution and exception monitoring using ptrace
- Multiple Processes

*Used to discover GNU Binutils readelf heap based integer overflow

Windows - FileFuzz

- **Simple baseline fuzzer**

- Requires a valid file to work from
- Flexible execution and exception monitoring
- Targets files with predefined handlers
- Can handle ASCII and binary files
- Has fancy GUI

*Used to discover Microsoft Windows Interactive Training heap based buffer overflow (MS05-031)

- Microsoft Interactive Training Buffer Overflow
 - CBO file parsing stack overflow
- RealPlayer RealText Format String
 - .rp file parsing format string
- Readelf Heap Overflow
 - GNU Binutils readelf heap based integer overflow

- Future trends and predictions
 - Attack
 - Further discovery tool automation
 - Increase in rate of vulnerability discovery
 - Defend
 - More file types blocked at network perimeter
 - File scanning utilities implement parsing functionality to identify non-standard file formats
 - File scanning utilities implement parsing functionality to identify malicious content (i.e. shellcode)

Questions?

