Michael Sutton Adam Greene

The Art of File Format Fuzzing

In September 2004, much hype was made of a buffer overflow vulnerability that existed in the Microsoft engine responsible for processing JPEG files. While the resulting vulnerability itself was nothing new, the fact that a vulnerability could be caused by a non-executable file commonly traversing public and private networks was reason for concern. File format vulnerabilities are emerging as more and more frequent attack vector. These attacks take advantage of the fact that an exploit can be carried within non-executable files that were previously considered to be innocuous. As a result, firewalls and border routers rarely prevent the files from entering a network when included as email attachments or downloaded from the Internet.

As with most vulnerabilities, discovering file format attacks tends to be more art than science. We will present various techniques that utilize file format fuzzing that range from pure brute force fuzzing to intelligent fuzzing that requires an understanding of the targeted file formats. We will present a methodology for approaching this type of research and address issues such as automating the process. Techniques will be discussed to address challenges such as attacking proprietary file formats, overcoming exception handling and reducing false positives. The presentation will include demonstrations of fuzzing tools designed for both the *nix and Windows platforms that will be released at the conference and the disclosure of vulnerabilities discovered during the course of our research.

Michael Sutton is a Director for iDEFENSE, a security intelligence company located in Reston, VA. He heads iDEFENSE Labs and the Vulnerability Aggregation Team (VAT). iDEFENSE Labs is the research and development arm of the company, which is responsible for discovering original security vulnerabilities in hardware and software implementations, while VAT focuses on researching publicly known vulnerabilities. His other responsibilities include developing tools and methodologies to further vulnerability research, and managing the iDEFENSE Vulnerability Contributor Program (VCP).

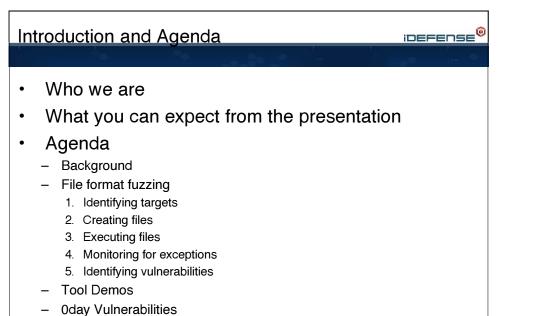
Prior to joining iDEFENSE, Michael established the Information Systems Assurance and Advisory Services (ISAAS) practice for Ernst & Young in Bermuda. He is a frequent presenter at information security conferences.

Michael obtained his Certified Information Systems Auditor (CISA) designation in 1998 and is a member of Information Systems Audit and Control Association (ISACA). He has completed a Master of Science in Information Systems Technology degree at George Washington University, has a Bachelor of Commerce degree from the University of Alberta and is a Chartered Accountant. Outside of the office, he is a Sergeant with the Fairfax Volunteer Fire Department.

Adam Greene is a Security Engineer for iDEFENSE, a security intelligence company located in Reston, VA. His responsibilities at iDEFENSE include researching original vulnerabilities and developing exploit code as well as verifying and analyzing submissions to the iDEFENSE Vulnerability Contributor Program.

His interests in computer security lie mainly in reliable exploitation methods, fuzzing, and UNIX based system auditing and exploit development. In his time away from computers he has been known to enjoy tea and foosball with strange old women.

																		inee	
L.	16													T:				IDEF	
											þ.,			11.64m				1. A.	1.600
0002b510h:	3A	0.0	0.0	00	68	00	74	00	74	0.0	70	0.0	3A	00	2F	0.0		:h.t.t.p	. /.
0002b520h:	2F	00	77	00	77	00	77		2E	00	77	00	61	00	79	00		/.w.w.ww	
0002b530h:	6E	00	65	00	67	00	72	00	65	0.0	74	00	7A	00	6B	00	;	n.e.g.r.e.t	.z.k.
0002b540h:	'9	٩F	ΡE	Δo	63	ര	6E	0.0	e	0	-0	pr	n¢	Эţ	1	D C	7	7.1 m.C	
0002b550h:	5A	3A	0 E	00	00	00	43	00	61	00	6E	00	61	00	64	00	ż	Z: Z .a.n	a.d.
0002b560h:	61	00	00	00	1F	00	Ð	37.	1.0	N٤	5	<u>e</u> e	lbs	00	6E	00	;	a\:	.0.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	$\stackrel{*}{_{\prime}}$	9	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	$\stackrel{*}{_{\prime}}$]:1.2	.3
0002b5c0h:	4D	00	61	00	69	00	6E	00	20	00	53	00	74	00	72	00	;	M.a.i.nS	t.r.
0002b5d0h:	65	00	65	00	74	00	00	00	03	00	71	3A	04	00	00	00	$\stackrel{*}{_{\prime}}$	e.e.tq	
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	;	OS.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	ż	k.yc.o.m	
0002b630h:	54	3A	01	00	00	00	0E	00	00	00	0A	00	00	00	53	00	;	Т:	S.
0002b640h:	4D	00	54	00	50	00	00	00	1F	10	56	3A	01	00	00	00	;	M.T.PV	
			1	Mic	hae	el S	utto	on -	- m	sut	ton	@ic	defe	ens	e.c	om			
						-													
				Ada	am	Gre	en	e –	ag	ree	ene	۳IC	iete	ense	9.CC	m			



- 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

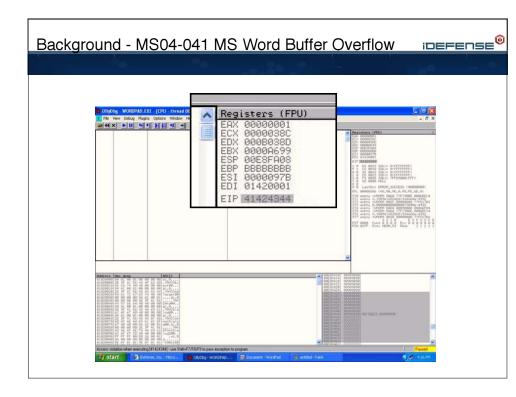
Background – Historical vulnerabilities

idefense[®]

<u>idefense[®]</u>

- 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

Backg	roun	d	-	M	S	04	ŀ-()4	1	M	S	V	Vc	oro	d I	Βι	Jft	fe	er Overflow	i⊏		inse [©]
<u>@</u> u	JltraEdit-32	- [C:	\w2	ksp3	i pl	ехр	loit.v	vri*]														
diam'r.	Eile Edit ;	-	-	_					ioļum	n M	acro	Ad	/ance	ed y	Mindo	ow t	<u>t</u> elp					
+	• • 🗅	õ	ď		(3	Q.	#	W	þ [102 011		-	*		Ê				<u>- M</u>	49 49	
	w2ksp3_pl_ex	ploit.	wri*																			
000	000a80h:	00	03	00	00	34	03	00	00	02	00	00	03	00	00	11	03	;	4			
000	000a90h:	00	00	03	00	FF	40	31	00	15	12	90	01	00	00	25	73	;	ÿ01⊐%s			
																			∜s∜ New RomanÌÌÌ			
																			ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ			
																			ììììììììììììììììììì			
																			11111111111111111			
																			ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ			
																			1111111111111111			
																			11111111111111111			
																			ììììììììììììììììììì			
																			11111111111111111			
000	000b30h:	AA	AA	AA	AA	BB	BB	BB	BB	44	43	42	41	FF	FF	FF	FF	2	<pre>Para ***********************************</pre>			
000	000b40h:	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	;				
000	000b50h:	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	;				
000	000b60h:	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	;	000000000000000000000000000000000000000			
000	000b70h:	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	;	000000000000000000000000000000000000000			
000	000b80h:	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	;	000000000000000000000000000000000000000			
000	000b90h:	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	;	000000000000000000000000000000000000000			
000	000ba0h:	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	;	000000000000000000000000000000000000000			
000	000bb0h:	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	;	000000000000000000000000000000000000000			
000	000bc0h:	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	;	000000000000000000000000000000000000000			
000	000bd0h:	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	;	000000000000000000000000000000000000000			
000	000be0h:	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	;	000000000000000000000000000000000000000			
000	000bf0h:	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	;	000000000000000000000000000000000000000			
000	000c00h:	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	;	000000000000000000000000000000000000000			
000	000c10h:	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	;	000000000000000000000000000000000000000			



BLACK HAT FINGS

digital self defense

Background - What's the risk?

<u>idefense[©]</u>

<u>idefense[®]</u>

- 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 Oday file format vulnerability

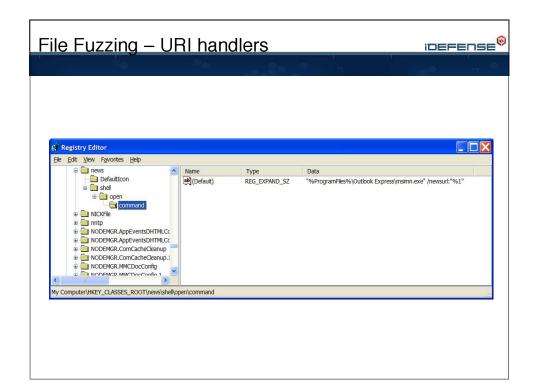
File Fuzzing – Identifying targets

- · 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

e Fuzzing – Reg	istered me type		FEN
	Edit File Type	2 🔀	
Folder Options	JPEG Izmage	Change Jcon	
General View File Types Offine Files	Actions:		
Registered file types:	open	New	
Extensions File Types	printto	Edit	
JFIF JPEG Image			
JNLP JNLP File		Bernove	
JOD Microsoft Jet OLEDB.4.0		Set Default	
JPE JPEG Image		Editing action for type: JPEG Image	?
JPEG JPEG Image	Confirm open after download	Editing action for type. JPEG image	الكا
JS JScript Script File	Always show extension Browse in same window	Action:	
		apen	OK
New	ОК		Cancel
and the second		Application used to perform action: rundll32.exe C:\WINDOWS\system32\shimgv	
Details for 'JPEG' extension		rundii32.exe C:\wuvbOw5\system32\snimgv	Browse
Opens with: Mindows Picture and Fax V	fie <u>Change</u>	Use DDE	
Files with extension 'JPEG' are of type 'JPEG Imag	e' To change	DDE Message:	
settings that affect all 'JPEG Image' files, click Adv		Eve messign.	
	200 million - 100 million -		
	Adyanced	Appligation: shimguw	
		DDE Application Not Running:	
ОК Са	Apply Apply		
		Topic:	
		System	

=ile	e Fuzzing – R	egistere	d file ty	pes idef	Ense
	eg <mark>istry Editor</mark> <u>E</u> dit <u>V</u> iew F <u>a</u> vorites <u>H</u> elp				
	MITrain.Document	Name	Type	Data	117
<		(Default)	REG_SZ	C:\WINDOWS\Help\SBSI\Training\orun32.exe -f "%	61"
My Co	omputer\HKEY_CLASSES_ROOT\MITrain.	Document\shell\open\col	mmand		ii.

e Fuzzing – URI h	analoro		FENS
			1,000
	Edit File Type	? 🗙	
older Options	URL:News Protocol	Change Jcon	
General View File Types Offine Files	Actions:		
Registered file types:	open	Bew	
Extensions File Types		Edit	
(NONE) URL:Gopher Protocol		Remove	
(NONE) URL:HyperText Transfer Protocol (NONE) URL:HyperText Transfer Protocol with Priv			
(NONE) URLIDAP Protocol		Set Default	
(NONE) URLMailTo Protocol (NONE) URL:News Protocol	Confirm gpen after download	Editing action for type: URL:News Pro	tocol ? 🔀
(NONE) URL:NNTP Protocol	Always show extension		
(NONE) URL:RLogin Protocol	Browse in same window	Action:	ОК
New	0		UK
		Application used to perform action:	Cancel
Details for 'URL News Protocol' file type		*C:\Program Files\Outlook Express\msimn.ex	Browse
Opens with:	Change		
		₩ Use DDE	
To change settings that affect all 'URL:News Protocol' fil Advanced.	es, click	DDE Message:	
Aurantes.			
		Application:	
L	Adyanced	msimn	
		DDE Application Not Running:	
OK Cancel	Apply		
	GPT-7	Topic:	
		System	



File Fuzzing – Identifying targets on Linux

Interesting Targets on Linux

- Antivirus products
 - · Fuzzing Linux AV engines locally can lead to a remote vulnerability

<u>idefense[©]</u>

<u>idefense[®]</u>

- Media Players
 - RealPlayer
- Document Viewers
 - Adobe Acrobat Reader
- Web Browsers
 - Think image formats

File Fuzzing – Creating files

• 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

File Fuzzing – Creating files

- Picking interesting values
 - Integers
 - Negative numbers (0xffffffff, 0x8000000, 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)

File Fuzzing – Creating files

<u>idefense[®]</u>

<u>idefense[®]</u>

- 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

File Fuzzing – Creating files

<u>idefense[©]</u>

<u>idefense[©]</u>

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

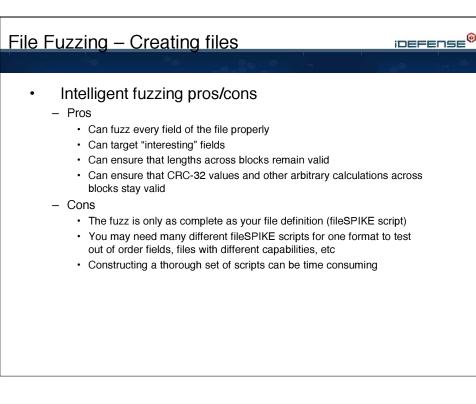
File Fuzzing – Creating files

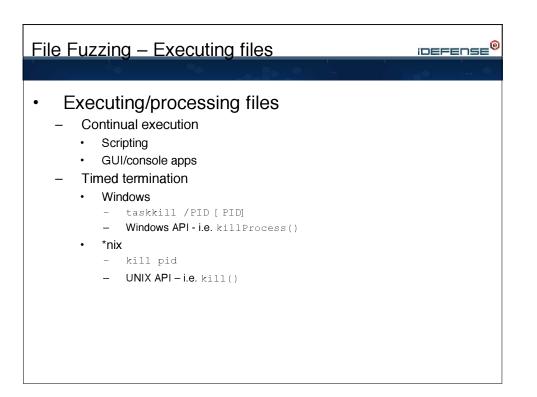
Intelligent fuzzing

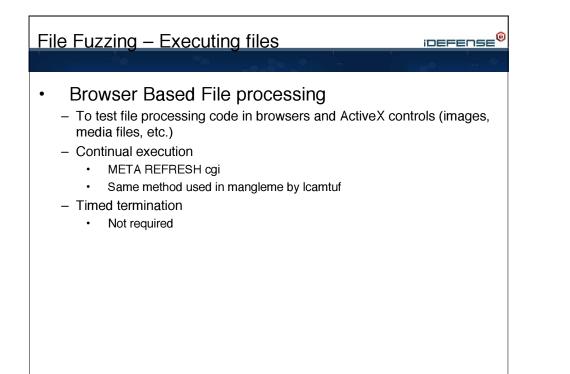
•

- 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 <u>http://www.midi.org/about-midi/specinfo.shtml</u>
 - 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.mspx

File Fuzzing – Creating files idefense[®] Intelligent fuzzing (cont'd) ٠ 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 •







File Fuzzing – Monitoring for exceptions

<u>idefense</u>®

idefense

- 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

File Fuzzing – Identifying exploitable vulns

- 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)

Automation - Tools	idefense
inux – SPIKEfile and notSPIKEfile	FileForz
<pre>Address of the second sec</pre>	Source File C Program Files/Adde/Acrobs/ 20/Resource/EH4.btt.pdf Target Directory C houges Files/Adde/Acrobs/ 20/Resource/EH4.btt.pdf Target Dyres/to 00 Committe Target X Counter C Range Start 0 Files/to 0 Fil
<pre>// How Text == = // / How Text == // / How Text == // / HOW == // / / HOW == // / / HOW == // / / / HOW == // / / / / / / / / / / / / / / / /</pre>	Replace X W X W X W Carlos
/ LNC / LN	Applieston Heme: scrottl.see Applieston Berrigtion Abdie Bader 7.0 Applieston Artioni open Applieston Lemnh: "CityTorgum Tile#Add0e(Acrobat 7.0\Beader\AcroBd12.eae* Applieston Flags" "(0)" Target Disectory: cit/sar/pdf)
// 100 _ 100 _ 101	Windows - fileFUZZ

Automation - Tools <u>idefense[®]</u> 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

Automation - Tools

the second second

idefense[®]

<u>idefense[®]</u>

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

Automation - Tools

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)

Oday Vulnerabilities

idefense[©]

<u>idefense[©]</u>

- 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

Conclusion

· 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 nonstandard file formats
 - File scanning utilities implement parsing functionality to identify malicious content (i.e. shellcode)

