The Art of File Format Fuzzing

iDEFENSE Labs

Michael Sutton – msutton@idefense.com
Adam Greene – agreene@idefense.com
Introduction and Agenda

• 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
Background – What’s the risk?

- **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 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
File Fuzzing – Registered file types

Folder Options

File Types

Registered file types

Extensions | File Types
---|---
JPEG | JPEG Image
JIF | JPEG Image
JNLP | JNLP File
JOB | Task Object
JOD | Microsoft Jet OLEDB 4.0
JPE | JPEG Image
JPEG | JPEG Image
JPG | JPEG Image
JS | JavaScript File

Details for JPEG extension

Opens with: Windows Picture and Fax View

Files with extension JPEG are of type JPEG Image. To change settings that affect all JPEG Image files, click Advanced.

Edit File Type

JPEG Image

Actions:
open
printto

New
Edit
Remove
Set Default

Browse in same window

Editing action for type: JPEG Image

Application used to perform action:
rundll32.exe C:\WINDOWS\system32\shimgv.dll

Advanced

DDE Message:

Application:
shimgv.dll

DDE Application: Not Running:

Topic:
System
File Fuzzing – Registered file types
File Fuzzing – URI handlers
File Fuzzing – URI handlers

Registry Editor

My Computer\HKEY_CLASSES_ROOT\news\shell\open\command

Name | Type         | Data                                               
-----|--------------|----------------------------------------------------
(Null) | REG_EXPAND_SZ | "%ProgramFiles%\Outlook Express\msmnr.exe" /newsurl:%1"
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
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, 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)
File Fuzzing – Creating files

- 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

- 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
      - W3C - [http://www.w3.org/](http://www.w3.org/)
    - Graphics (JPEG, PNG, SVG, etc.)
      - W3C - [http://www.w3.org/Graphics/](http://www.w3.org/Graphics/)
    - Audio (MIDI, MP3, WAV, etc.)
    - Compressed/Archive (ZIP, TAR, RAR, etc.)
    - Binary (a.out, ELF, COFF)
      - Microsoft – PE & COFF
File Fuzzing – Creating files

• 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
File Fuzzing – Creating files

- **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
File Fuzzing – Executing files

- 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()
File Fuzzing – Executing files

• 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
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

Linux – SPIKEfile and notSPIKEfile

Windows - fileFUZZ
Automation - Tools

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 – not SPIKE file

- 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)
0day Vulnerabilities

• **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 non-standard file formats
    - File scanning utilities implement parsing functionality to identify malicious content (i.e. shellcode)