HOT FUZZING WITH ZZUF

Sam Hocevar (sam@zoy.org) — Hacker Space Festival — June 21st, 2008
OVERVIEW

- What’s fuzzing?
- Zzuf introduction
- Getting started
- Our way to a few 0-days
- What next?
What's Fuzzing?

• Feed a program with random data
  - White noise
  - Slightly modified input
  - Content-aware fuzzing

• Increasing use nowadays
  - Test suites
  - Attack tools
WHAT TO FUZZ?

- Complex data formats
  - Images, sound, videos...
  - Executables, bytecode

- Protocols
  - Network protocols
  - Databases

- Any user-provided data
**FUZZING RESULTS**

- **Bugs**
  - Exploitable bugs: good old buffer overflows
  - DoS: crashes, memory exhaustion, CPU bombs, deadlocks, data corruption...

- **Without even reading the code!**
ZZUF INTRODUCTION

• All-in-one fuzzing tool
  - Easy to use
  - Reproducible behaviour
  - Fuzzes everything on the fly

• Simple
  - No configuration file
  - No context-aware fuzzing
ZZUF ARCHITECTURE

input

application

output

input

zzuf

libzzuf

application

output
ZZUF INTERNALS

- Controlling zzuf binary
  - forks tested program
  - checks stdout, exit value, signals...

- LD_PRELOAD mechanism
  - intercepts file reading functions
    - open(), read(), fopen(), fread()...
  - also malloc() to check memory usage
Basic features

- Random seeds (-s)
- Fuzzing ratio (-r)
- Cherry-pick fuzzed data
  - Include/exclude file patterns (-I, -E)
  - Network (-n), standard input (-i)
  - Fuzz depending on byte offsets (-b)
  - Fuzz depending on byte values (-P)
OTHER FEATURES

- Parallel processing (-j)
- Detect stuck processes
  - Set maximum memory allocation (-M)
  - Set maximum running time (-T)
  - Set maximum stdout output (-B)
- See manual page for more
GETTING STARTED

- [http://libcaca.eu/wiki/zzuf](http://libcaca.eu/wiki/zzuf)
- **From Subversion:**
  - `svn co svn://svn.zoy.org/libcaca/zzuf/trunk`
  - `./bootstrap`
  - `./configure`
  - `make`
  - You’re done!
**FIRST STEPS**

- Standard utilities
  - `cat`, `more`
  - `grep`
  - `cp`, `dd`
- Network fuzzing
- Finding a real bug
Finding 0-Day Bugs

- objdump
- Image viewers
- MPlayer
- Firefox
- [your suggestion here]
CONCLUSIONS

- Fuzzing is cheap and easy
- It finds real, scary bugs
- Binary formats == easy targets
- Seldom used == seldom tested
- Warning: zzuf-proof != bug-free
**ZZUF'S FUTURE**

- Context-dependent fuzzing
  - ignore or recompute CRCs
  - divert the zlib library, too (for PNGs)

- Finish the Windows® port
  - help needed

- Attach to a debugger

THANKS!

Any questions?