Dan Guido – Fall 2010

FUZZING

Objective

1. What is Fuzzing?
   - Pros and Cons
   - What does Fuzzing Look Like?
   - Who Fuzzes?
2. Types of Fuzzing
3. Effective Fuzzing

What is Fuzzing?

Vuln Researcher’s Toolbox

- Source Code Review
  - Great coverage
  - Highly complex
  - Not always an option
- Binary Auditing
  - Decent coverage
  - Highly complex (specialized skillset)
  - Almost always an option

SNOW

- You just banged on the keyboard, right?
  - User: AAAAAAAAAAA
  - Password: ihateyoudanAAAAAAA….

- Reversing might have been too slow
- The attack surface was easily exposed
- You knew when exceptions occurred (crash)
- So you tested for boundary conditions

What is Fuzzing?

- "An automated method for discovering faults in software by providing unexpected input and monitoring for exceptions." – *Fuzzing*
- You already know how to fuzz!
  - We are going to automate your manual efforts

Fuzzing Goals

- Aims to be simple and effective
- Test exposed attack surface for boundary conditions
- Most effective against languages that use unmanaged memory and sometimes the web

Pros and Cons

- Availability – when can you not fuzz really?
- Reproducibility – write to target a protocol, fuzz every server that implements it
- Simplicity – doesn’t require strong knowledge of app internals, though it can help

Pros and Cons (cont)

- Coverage – how much did you really get?
  - Fuzzing shouldn't replace other types of testing
- Intelligence – fuzzing works best where vulns are caused by one vector by themselves
  - Multi-stage vulns aren't well suited for fuzzing

What does Fuzzing look like?

- cat /dev/urandom | nc –vv target port
  - Yes, this really works sometimes
- Model an ASCII network protocol
  - Find all verbs in FTP, test arg parsing on server
- Take a pcap, modify every byte, replay
  - Why not try every value for every byte?

Who took Statistics?

- How many test cases do you need to completely bruteforce one 4k file?
  - Bytes in a Kilobyte * 4
  - Possible ASCII characters
- (4 * (1024 ** 2)) ** 255 = LOL
- This is why we only test boundary conditions

Limitations Thought Exercise

- Access Control Flaws
- Poor Design Logic
- Backdoors
- Memory Corruption

Who Fuzzes?

- Software Vendors
  - In-house or out-sourced to security companies
  - MS SDL tools have been published
- Hackers
  - Fuzzers are an effective way to find oday
  - Laurent Gaffie and Charlie Miller's entire careers
- Academia
  - One area where academia isn't completely blind!

Types of Fuzzing

Random (dumbest)

- Fling totally random data at the target
  - cat /dev/urandom | nc –vv target port
  - Early environment variable fuzzers
- How do you identify the test case that crashed your app?

Mutational (dumb)

- Collect samples and modify
  - MiniFuzz, FileFuzz, SPIKEfile

- Things to remember:
  - Less setup time, but effectiveness depends on samples
  - Works best on file formats and simple clear text protocols

Generational (smart)

- Model what the application should process
  - Simple: auxiliary/fuzzers/ftp/client_ftp in Metasploit
  - Complex: Peach, Autodafe, SPIKE, Sulley
- Things to remember:
  - More setup time, but tends to yield better results
  - Works best with complex network protocols, APIs
- Dilemma: If you spend so much time modeling, why not just RE the target?

Evolutionary (smarter)

- Connect a smartfuzzer to a binary instrumentation framework for feedback
  - Modify inputs based on coverage
- Open area of research if you're interested!

Effective Fuzzing

Fuzzing Phases

- Approach depends on your objective
  - Are you going for coverage or exploits?
  - Type of app and format being fuzzed

1. Identify the Target
   - Want to re-use your fuzzer? Pick a common fileformat or a common library.
   - Look at past history of vulns in the target

Fuzzing Phases

2. Identify Inputs
   - Enumerate input vectors, not just the obvious ones
   - Filenames, reg keys, env variables, headers, etc

3. Generate Fuzz Data
   - Mutate or generate?

4. Execute Fuzz Data
   - Let's get cooking!

Fuzzing Phases

5. Monitor for Exceptions
   - What does an exception look like?
   - Ensure you can pinpoint the cause of a crash

6. Determine Exploitability
   - No one likes to talk about it, but this is the hardest
   - Usually a manual process, crash binning can help

Resources

- The course website
  - http://pentest.cryptocity.net/fuzzing
- Fuzzing: Brute Force Vulnerability Discovery
  - http://fuzzing.org