Advanced Custom Network Protocol Fuzzing

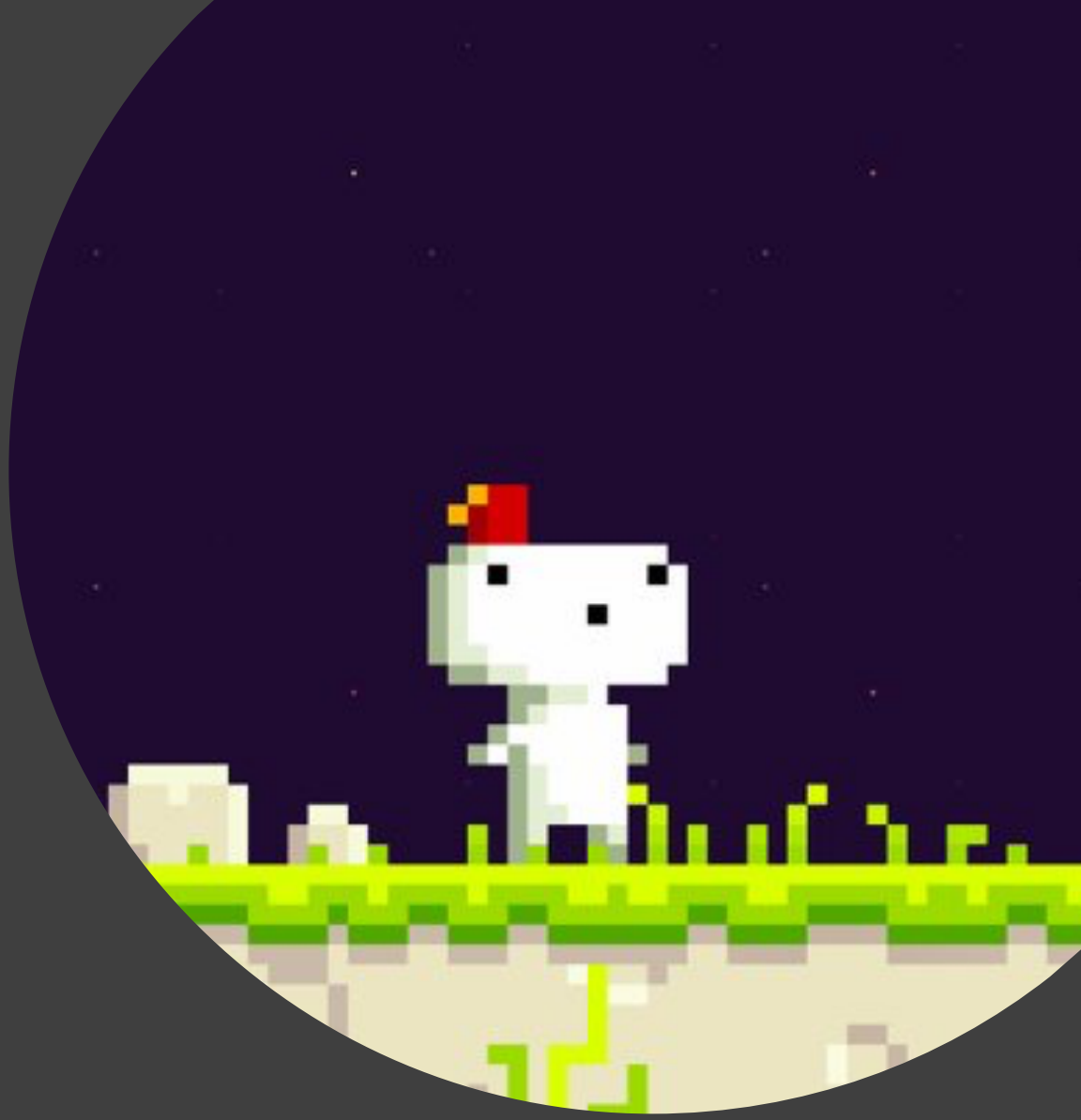
Joshua Pereyda

Intro – Course Goals

- Learn About Fuzzers
- Write your own protocol specification in boofuzz
- Use your fuzzer to find bugs
- ???
- Profit

Intro – Me

- Joshua Pereyda
  - Software Engineer in Security
  - Experience fuzzing professionally
  - Maintains boofuzz
  - Oracle Cloud Infrastructure



@jtpereyda

Outline

- Lecture: Fuzzing Intro
- Lecture: Basic Techniques with boofuzz
- Exercise 1 – Target Practice
- Exercise 2 – Target Practice 2
- Exercise 3 – On Your Own
- Lecture: Reverse Engineering & Advanced Topics

Outline

- Lecture: Fuzzing Intro
- Lecture: Basic Techniques with boofuzz
- Exercise 1 – Target Practice
- Exercise 2 – Target Practice 2
- Exercise 3 – On Your Own
- Lecture: Reverse Engineering & Advanced Topics

Intro – Outline

- Fuzzing & Fuzzing Techniques
- Current Tools

Intro – Outline

- Fuzzing & Fuzzing Techniques
- Current Tools

Security Research

- Vulnerability Research – Look for problems
- Exploit Development – Exploit problem to impact security

Fuzzing

- Vulnerability research tool
- Generate invalid/unexpected inputs
- Detect failure (typically segfault/crash/etc.)

Fuzzing vs. Scanning

Scanning

- Automated
- Easily deployed vs. new targets
- Look for known issues
- Non-destructive
- Operational activity
- Low to no instrumentation
- Run in prod ☺
- Easy to scale

Fuzzing

- Automated
- New targets require engineering/config
- Look for unknown issues (0-days)
- Destructive
- Development activity
- More instrumentation => More value
- Run in prod ☹
- Scalable but with more effort

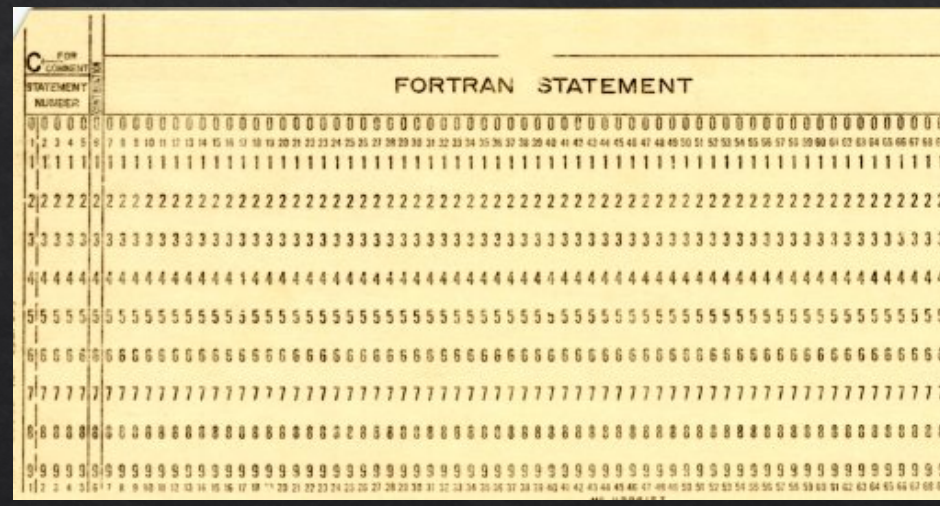
Origins: 1988

- Developed by Dr. Barton Miller at UW Madison as a class project
- Fed random input to Unix utilities
  - Crashed over 25% of tested commands, including emacs, vi, make, telnet, csh, ftp, lex, telnet, …
  - Miller was surprised at the number of failures found
- Key development: Use randomness to violate assumptions.

```
curl 'fj48914309qr2p3 rim e/resd,fa;wf.,4vpl6v3/5p.vl;ul.6.ty[5p16[4[1]4\5][13]5123'
```

Origins

- 1950s – Punch cards from trash bin fed into program to evaluate robustness



- 1988 – First use of term “fuzzing”

Origins: 1988

- Developed by Dr. Barton Miller at UW Madison as a class project
- Fed random input to Unix utilities
  - Crashed over 25% of tested commands, including emacs, vi, make, telnet, csh, ftp, lex, telnet, …
  - Miller was surprised at the number of failures found
- Key development: Use randomness to violate assumptions.

```
curl 'fj48914309qr2p3 rim e/resd,fa;wf.,4vpl6v3/5p.vl;ul.6.ty[5p16[4[1]4\5][13]5123'
```

Command Line to Network Protocols: ~2000

- PROTOS project, University of Oulu, Finland
- Found vulnerabilities in 40 of 49 products
- 14 vulnerable to remote code execution (RCE)
- Used a structured, generational, or “smart“ fuzzing approach
- Procedural, not random

Generational ("Smart") Fuzzing

IPv4 Header Format

0								1								2								3							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Version				IHL				DSCP						ECN		Total Length															
Identification																Flags			Fragment Offset											
Time To Live								Protocol								Header Checksum															
Source IP Address																															
Destination IP Address																															
Options (if IHL > 5)																															

Protocol Fuzz Test Case Example

- -> Anomalous Message
- <- Response (optional)
- Health Check
  - -> Request
  - <- Response
  - If no response:
    - Log failure
    - Restart target

Mutational Fuzzing

- Start with a valid data sample
- Mutate valid sample to get test cases

Guided Fuzzing

- Gray/White Box
- Utilize a metric to decide which mutations to emphasize
- Common metric: Code coverage
- Concolic (concrete + symbolic) execution: Symbolic execution guides test case generation

Instrumentation

- Ping/response check
- Detect process crash
- Memory leaks
- Side effects
- Functional misbehaviors

Review: Fuzzing Techniques

- Random vs. Mutational vs. Generational
- Black box vs. White box
- Guided vs. Unguided
- Instrumentation Options
- Target type

Fuzzing Strengths

- Find bugs quickly
- Ensure robust public interfaces
- Quicker than manual testing
- Existing tools

Fuzzing Weaknesses

- Some code paths are hard to find
- Checksums can kill mutational fuzzing
- Logic bugs often involve bizarre combinations of functionality that generational fuzzers can miss
- Exciting research: Generational and guided fuzzing, e.g. “AFLSmart”

Intro – Outline

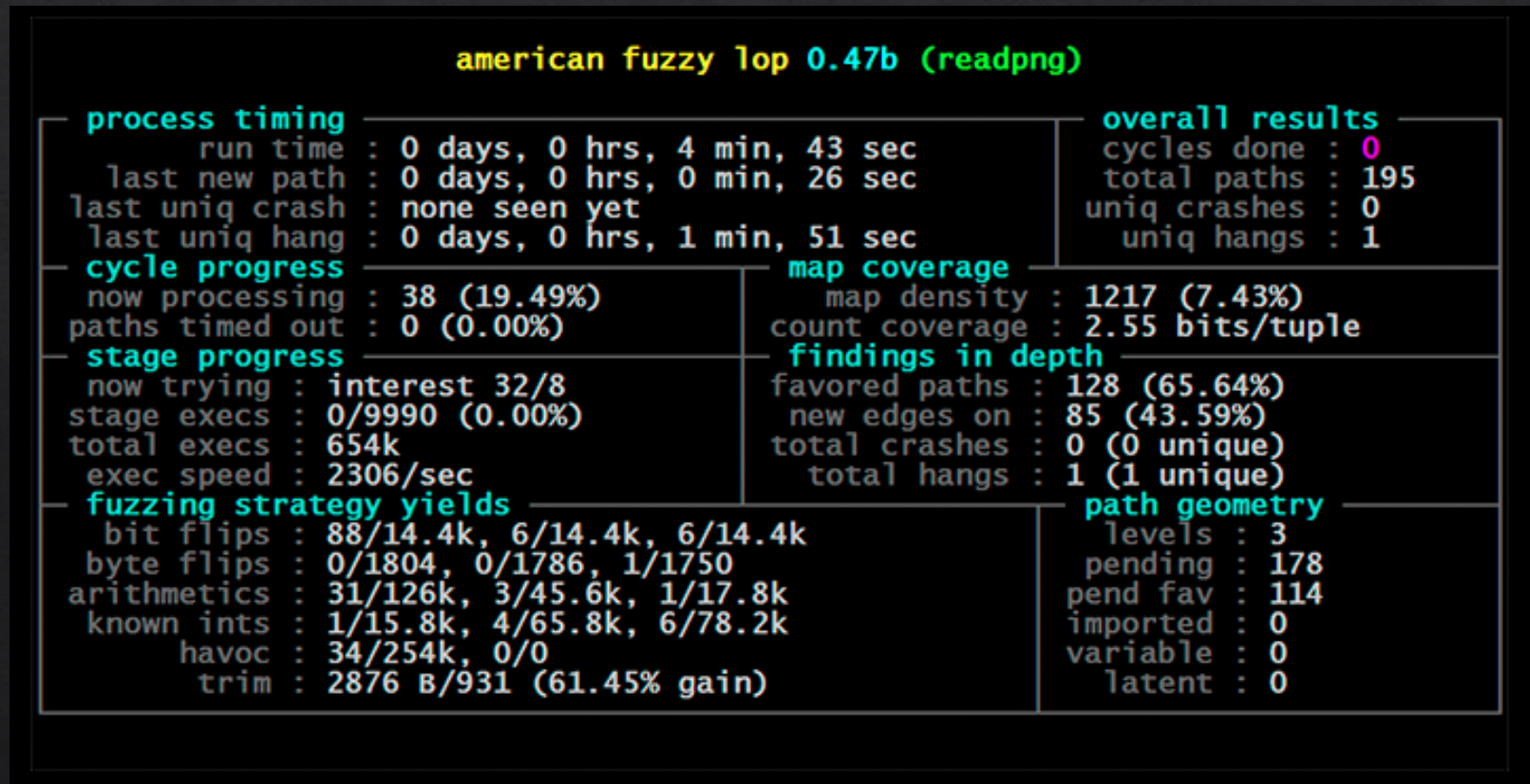
- Fuzzing & Fuzzing Techniques
- Current Tools

Intro – Outline

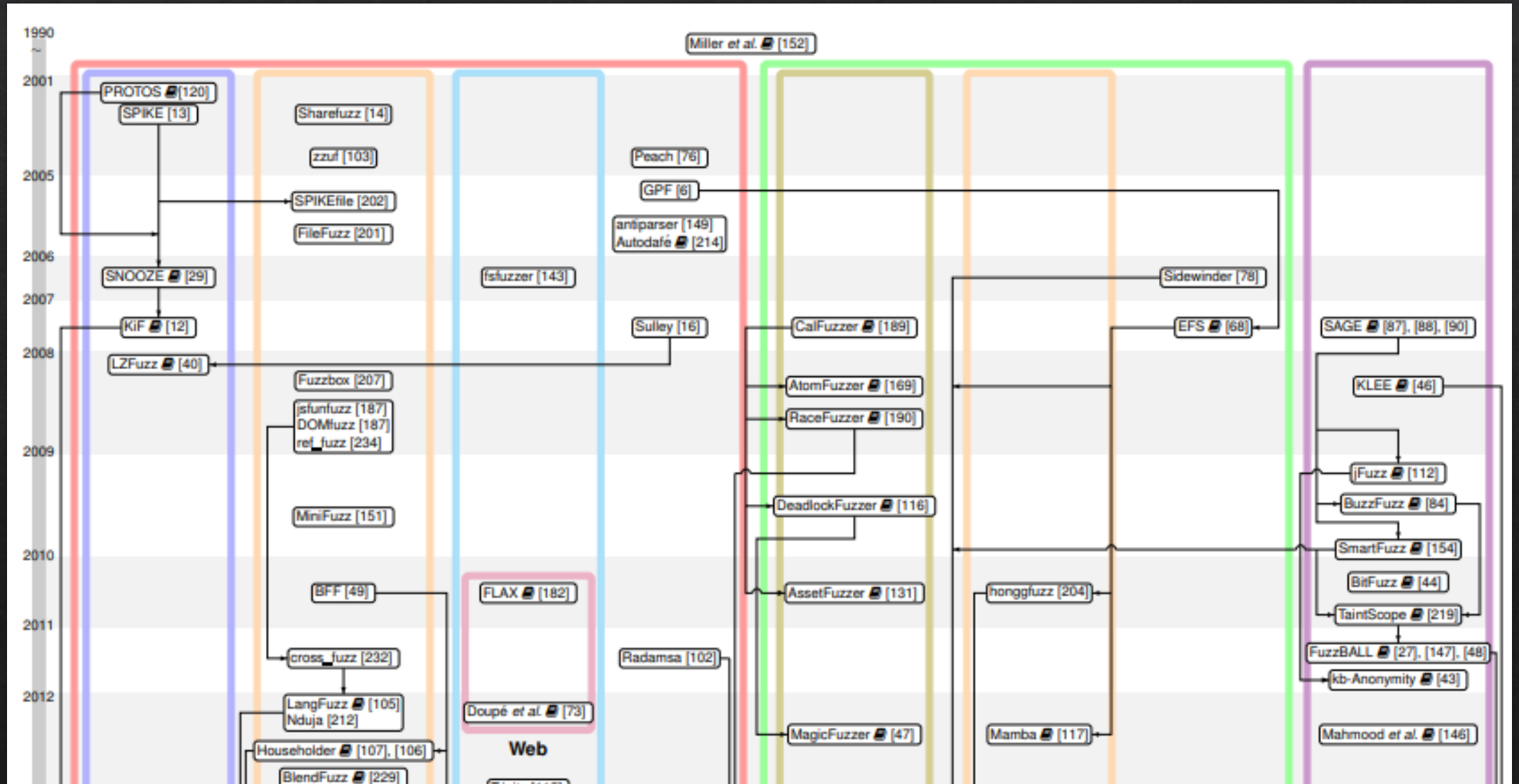
- Fuzzing & Fuzzing Techniques
- Current Tools

Current Tools

- Open Source, Guided Fuzzing
  - American Fuzzy Lop – AFL
  - LibFuzzer
  - Targets: File or API
  - Zillions of AFL flavors and others

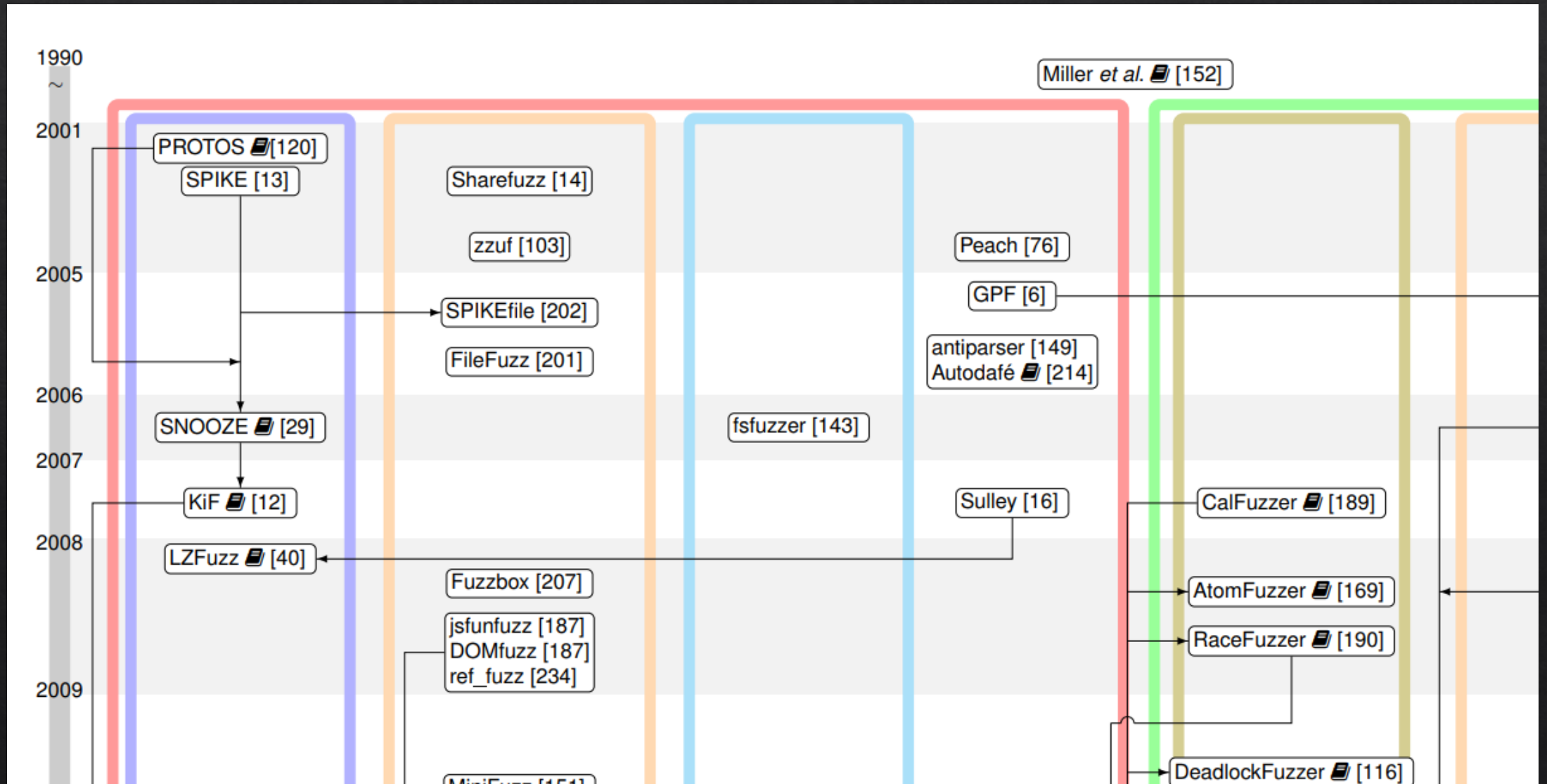


Current Tools



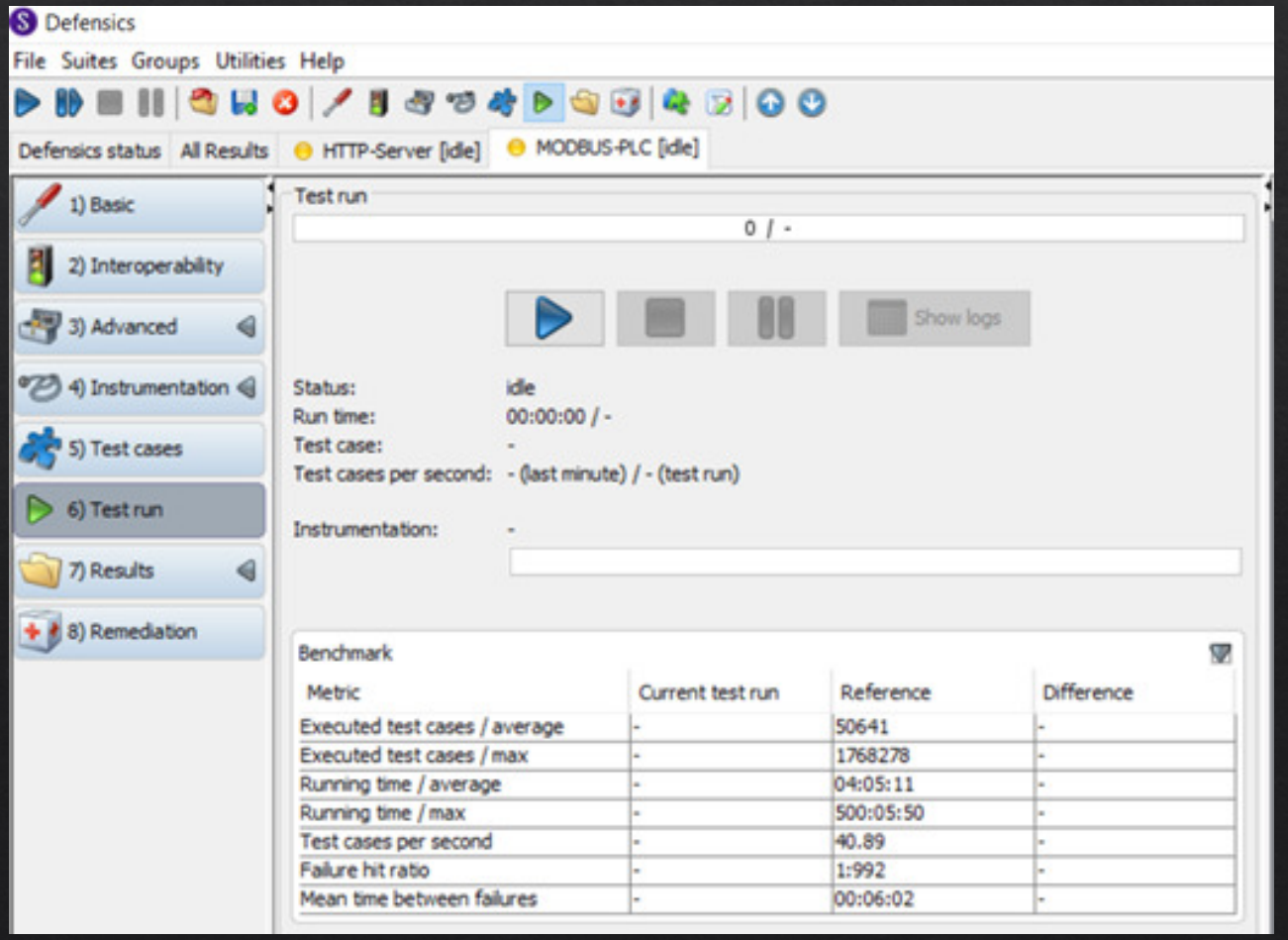
Current Tools

https://apps.dtic.mil/dtic/tr/fulltext/u2/a558209.pdf



Current Tools

- Closed Source, Generational Protocol Fuzzing
  - Defensics by Synopsys
  - Peach
  - Vendors sell specific protocol definitions



https://community.synopsys.com/s/question/0D53400004D2fALCAZ

Current Tools

- Open Source, Generational Protocol Fuzzing Frameworks
  - Peach (super old version)
  - Spike
  - Sulley
  - boofuzz

boofuzz

- Goal: Easy and quick data generation for network protocols
- Python
- Features
  - Mulitple transport layers (TCP, UDP, SSL, IP, Eth, Serial)
  - Shiny Web UI
  - Documentation
- Used by hobbyists & Fortune 100 companies

boofuzz: Fuzzing Techniques

- Random vs. Mutational vs. Generational
- Black box vs. White box
- Guided vs. Unguided
- Instrumentation Options
- Target type: Network Protocols

Outline

- Lecture: Fuzzing Intro
- Lecture: Basic Techniques with boofuzz
- Exercise 1 – Target Practice
- Exercise 2 – Target Practice 2
- Exercise 3 – On Your Own
- Lecture: Reverse Engineering & Advanced Topics

Outline

- Lecture: Fuzzing Intro
- Lecture: Basic Techniques with boofuzz
- Exercise 1 – Target Practice
- Exercise 2 – Target Practice 2
- Exercise 3 – On Your Own
- Lecture: Reverse Engineering & Advanced Topics

Basic Techniques

- Understand Your Protocol
- Define Some Messages
- Fuzz

Basic Techniques – Understand Your Protocol

- RFCs/Specifications
- Packet Captures
- Experimentation

boofuzz – Boilerplate

```
from boofuzz import *
# ...
session = Session(
    target=Target(
        connection=SocketConnection(target_host,
                                    target_port,
                                    proto='tcp'),
    )
)
```

boofuzz – Define Messages

```
s_initialize("user")
s_string("USER")
s_delim(" ")
s_string('ascii')
s_static("\r\n")
```

boofuzz – Define Messages

```
s_initialize("pass")
s_string("PASS")
s_delim(" ")
s_string('ascii')
s_static("\r\n")
```

boofuzz – Define Messages

```
s_initialize("stor")
s_string("STOR")
s_delim(" ")
s_string("AAAA")
s_static("\r\n")
```

boofuzz – Define Messages

```
s_initialize("retr")
s_string("RETR")
s_delim(" ")
s_string("AAAA")
s_static("\r\n")
```

boofuzz – Connect Messages

```
session.connect(s_get("user"))
session.connect(s_get("user"), s_get("pass"))
session.connect(s_get("pass"), s_get("stor"))
session.connect(s_get("pass"), s_get("retr"))
```

Outline

- Lecture: Fuzzing Intro
- Lecture: Basic Techniques with boofuzz
- Exercise 1 – Target Practice
- Exercise 2 – Target Practice 2
- Lecture: Reverse Engineering & Advanced Topics

Outline

- Lecture: Fuzzing Intro
- Lecture: Basic Techniques with boofuzz
- Exercise 1 – Target Practice
- Exercise 2 – Target Practice 2
- Lecture: Reverse Engineering & Advanced Topics

Exercise 1 – Target Practice

- Target: HTTP Server on Linux
- Follow your handout
- Go!

Outline

- Lecture: Fuzzing Intro
- Lecture: Basic Techniques with boofuzz
- Exercise 1 – Target Practice
- Exercise 2 – Target Practice 2
- Exercise 3 – On Your Own
- Lecture: Reverse Engineering & Advanced Topics

Outline

- Lecture: Fuzzing Intro
- Lecture: Basic Techniques with boofuzz
- Exercise 1 – Target Practice
- Exercise 2 – Target Practice 2
- Exercise 3 – On Your Own
- Lecture: Reverse Engineering & Advanced Topics

Exercise 2 – Target Practice 2

- Target: Modified HTTP Server
- Follow your handout
- Go!

Outline

- Lecture: Fuzzing Intro
- Lecture: Basic Techniques with boofuzz
- Exercise 1 – Target Practice
- Exercise 2 – Target Practice 2
- Exercise 3 – On Your Own
- Lecture: Reverse Engineering & Advanced Topics

Outline

- Lecture: Fuzzing Intro
- Lecture: Basic Techniques with boofuzz
- Exercise 1 – Target Practice
- Exercise 2 – Target Practice 2
- Exercise 3 – On Your Own
- Lecture: Reverse Engineering & Advanced Topics

Exercise 3 – On Your Own

- Target: smallftpd (FTP)
- Follow your handout
- Go!

Outline

- Lecture: Fuzzing Intro
- Lecture: Basic Techniques with boofuzz
- Exercise 1 – Target Practice
- Exercise 2 – Target Practice 2
- Exercise 3 – On Your Own
- Lecture: Reverse Engineering & Advanced Topics

Outline

- Lecture: Fuzzing Intro
- Lecture: Basic Techniques with boofuzz
- Exercise 1 – Target Practice
- Exercise 2 – Target Practice 2
- Exercise 3 – On Your Own
- Lecture: Reverse Engineering & Advanced Topics

Reverse Engineering – Outline

- Use Wireshark
- Analyze PCAPs
- Context
- ???
- Fuzzer!

Reverse Engineering – Analysis Tips

- Look at multiple samples
- Look for static values or sometimes-changing values
- Look for data structures
- Sequences of zeros may be
  - Filler bytes for fixed length fields
  - Unused fields
- Look at request vs reply format
- Look for hints
  - What else is happening in the PCAP?

Reverse Engineering – Analysis Example

- Two request+reply pairs
- Functionally similar
- Different hosts
- We will compare
  - Request A vs Request B
  - Request vs Reply
  - Reply A vs Reply B

Request A

```
0000    00 04 00 01 00 06 00 00 17 00 e4 88 00 00 08 00
0010    45 00 00 60 e6 c3 40 00 40 06 3d c5 0a 00 01 05
0020    0a 00 01 0b d6 df 08 01 94 68 ae 69 f4 2f b6 0e
0030    80 18 00 d2 16 62 00 00 01 01 08 0a 00 0f 48 a9
0040    1c 62 1b a3 80 00 00 28 d1 82 5e 7d 00 00 00 00
0050    00 00 00 02 00 01 86 a3 00 00 00 03 00 00 00 00
0060    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Request B

```
0000    00 04 00 01 00 06 00 00 17 00 2a ed 00 00 08 00
0010    45 00 00 60 13 c7 40 00 40 06 12 af 0a 00 00 11
0020    0a 00 00 12 03 1f 08 01 ff 30 1f 56 86 fd 15 cb
0030    80 18 00 d2 14 75 00 00 01 01 08 0a 07 f3 0c 90
0040    07 f1 8c fd 80 00 00 28 5c 1c ef 1b 00 00 00 00
0050    00 00 00 02 00 01 86 a3 00 00 00 04 00 00 00 00
0060    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Request A

```
0000    00 04 00 01 00 06 00 00 17 00 e4 88 00 00 08 00
0010    45 00 00 60 e6 c3 40 00 40 06 3d c5 0a 00 01 05
0020    0a 00 01 0b d6 df 08 01 94 68 ae 69 f4 2f b6 0e
0030    80 18 00 d2 16 62 00 00 01 01 08 0a 00 0f 48 a9
0040    1c 62 1b a3 80 00 00 28 d1 82 5e 7d 00 00 00 00
0050    00 00 00 02 00 01 86 a3 00 00 00 03 00 00 00 00
0060    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Reply A

```
0000    00 00 00 01 00 06 00 00 17 00 97 11 00 00 08 00
0010    45 00 00 50 ad ae 40 00 40 06 76 ea 0a 00 01 0b
0020    0a 00 01 05 08 01 d6 df f4 2f b6 0e 94 68 ae 95
0030    80 18 00 d2 62 c3 00 00 01 01 08 0a 1c 62 1b a3
0040    00 0f 48 a9 80 00 00 18 d1 82 5e 7d 00 00 00 01
0050    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Reply A

```
0000    00 00 00 01 00 06 00 00 17 00 97 11 00 00 08 00
0010    45 00 00 50 ad ae 40 00 40 06 76 ea 0a 00 01 0b
0020    0a 00 01 05 08 01 d6 df f4 2f b6 0e 94 68 ae 95
0030    80 18 00 d2 62 c3 00 00 01 01 08 0a 1c 62 1b a3
0040    00 0f 48 a9 80 00 00 18 d1 82 5e 7d 00 00 00 01
0050    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Reply B

```
0000    00 00 00 01 00 06 00 00 17 00 b1 07 00 00 08 00
0010    45 00 00 50 99 37 40 00 40 06 8d 4e 0a 00 00 12
0020    0a 00 00 11 08 01 03 1f 86 fd 15 cb ff 30 1f 82
0030    80 18 00 d2 26 46 00 00 01 01 08 0a 07 f1 8c fd
0040    07 f3 0c 90 80 00 00 18 5c 1c ef 1b 00 00 00 01
0050    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Reply B

```
0000    00 00 00 01 00 06 00 00 17 00 b1 07 00 00 08 00
0010    45 00 00 50 99 37 40 00 40 06 8d 4e 0a 00 00 12
0020    0a 00 00 11 08 01 03 1f 86 fd 15 cb ff 30 1f 82
0030    80 18 00 d2 26 46 00 00 01 01 08 0a 07 f1 8c fd
0040    07 f3 0c 90 80 00 00 18 5c 1c ef 1b 00 00 00 01
0050    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Advanced Topics – Outline

- Custom Transport Layers
- Future of boofuzz
- Conclusion

Advanced Topics – Custom Transport Layers

- Implement the `ITargetConnection` interface
- `SocketConnection` provides typical use case
- `SerialConnection` for serial ports

Advanced Topics – Custom Transport Layers

```
session = Session(
    target=Target(
        connection=SerialConnection(port=1,
                                    baudrate=9600),
    ),
    sleep_time=sleep_between_cases,
)
```

Advanced Topics – Future

- More robust data model – use Python construct package?
- More features
- Less bugs
  - Hot tip: If you think you found a bug in boofuzz… you probably did! :P
- Better built-in CLI
- Quality Protocol Definitions

Conclusion

- We learned how to…
  - Define protocols in boofuzz
  - Identify bugs using a network protocol fuzzer
  - Reverse engineer an unknown network protocol
- Remember:
  - Commercial tools are great if your employer can pay for them
  - Open source tools are best for custom protocols… and more fun! ☺
  - There is a wide world of fuzzing tools & research

Thank you and happy fuzzing!

- https://github.com/jtpereyda/boofuzz
- https://github.com/jtpereyda/boofuzz-ftp
- https://github.com/jtpereyda/boofuzz-http

- @boofuzz
- @jtpereyda
- @tim_clemans