Evolutionary Fuzzing for Genetic Improvement

Toward Adaptive Software Defense

Jason Landsborough
Stephen Harding
Bryan Beabout

What Is The Problem We Are Trying To Solve?

Ecosystems lacking sufficient diversity lack adaptability and can easily falter and fail.

Problem: The low initial cost of same software everywhere has made our systems vulnerable and easy prey for widespread attack. *Attackers have a huge asymmetric advantage.*

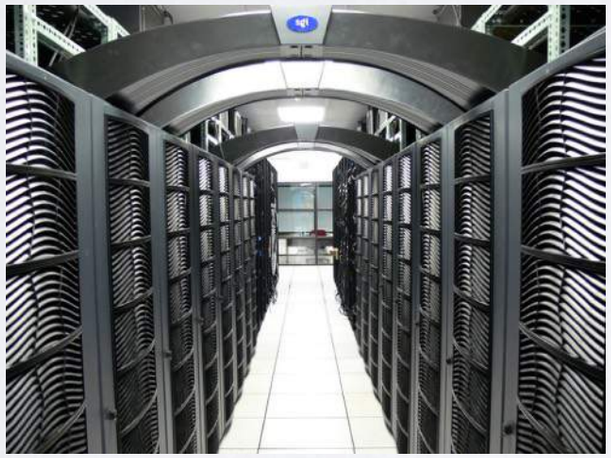


Photo taken by Christopher Browns, 31 January 2008. CC BY-SA 2.0

By Brett Jordan [CC BY 2.0 (http://creativecommons.org/licenses/by/2.0)], via Wikimedia Commons

Solution:

- Avoid unnecessary consistency.
- Implement facile and distributed moving target defense strategies.
- Use biology as inspiration.

Simulated software diversity

Genetic Improvement For Defense

We can use GI to create neutral variants

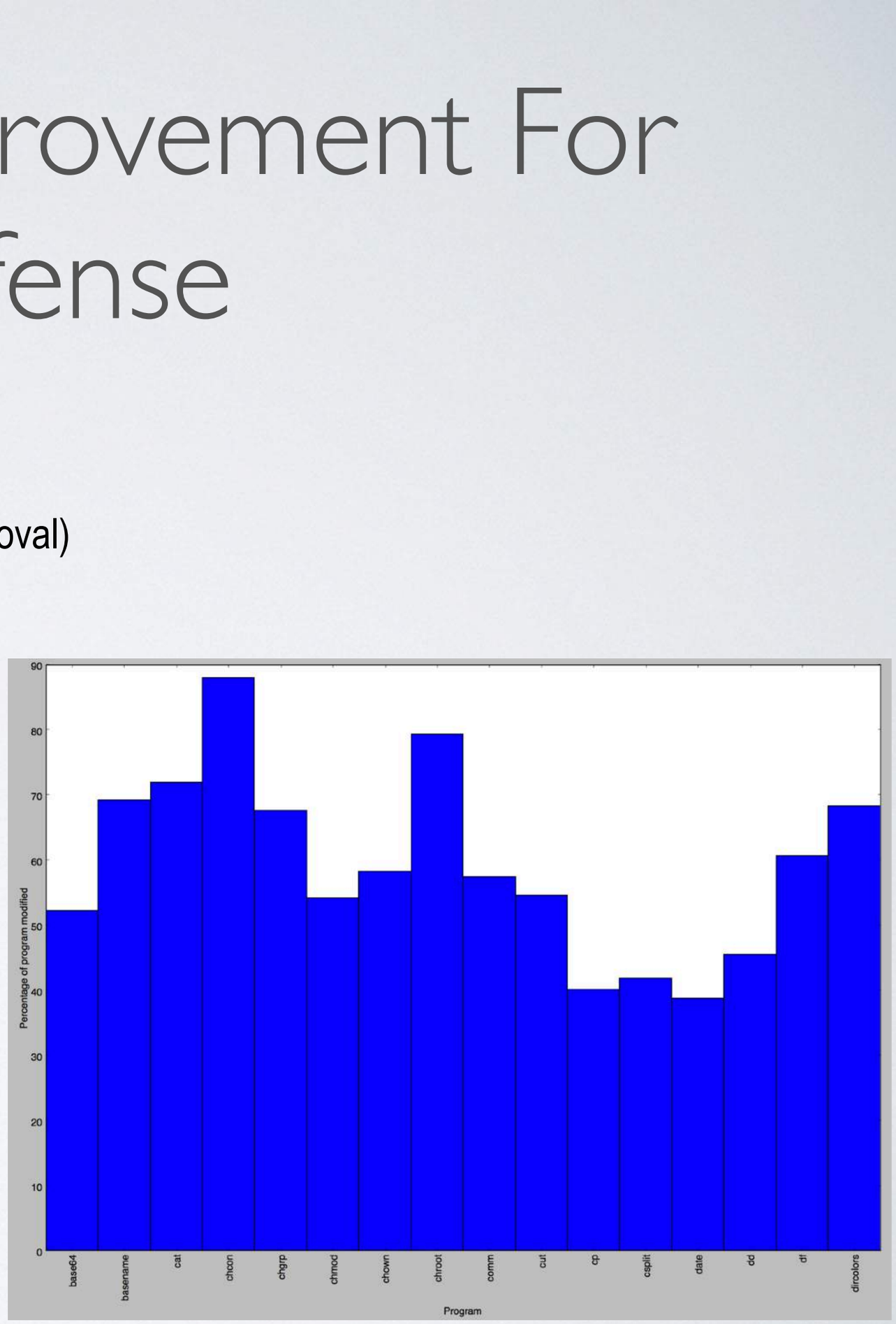
- Increase the likelihood that an exploit fails

Or further specialize:

- Mitigate a known vulnerability (repair or removal)

GI relies on test cases to ensure correctness.

Problem: What if tests suck?



Genetic Improvement For Defense

We can use GI to create neutral variants

- Increase the likelihood that an exploit fails

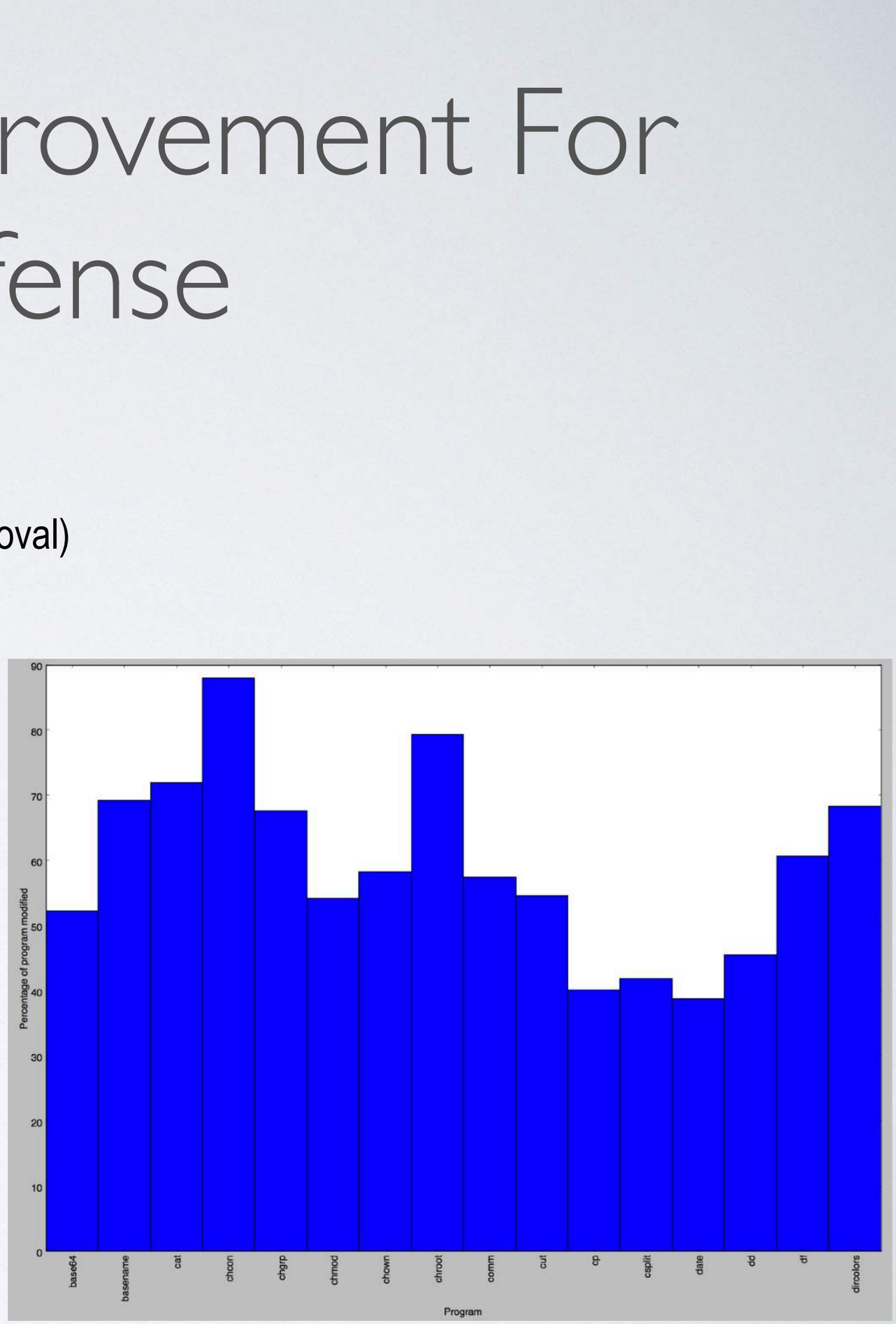
Or further specialize:

- Mitigate a known vulnerability (repair or removal)

GI relies on test cases to ensure correctness.

Problem: What if tests suck?

Solution: Give up



Genetic Improvement For Defense

We can use GI to create neutral variants

- Increase the likelihood that an exploit fails

Or further specialize:

- Mitigate a known vulnerability (repair or removal)

GI relies on test cases to ensure correctness.

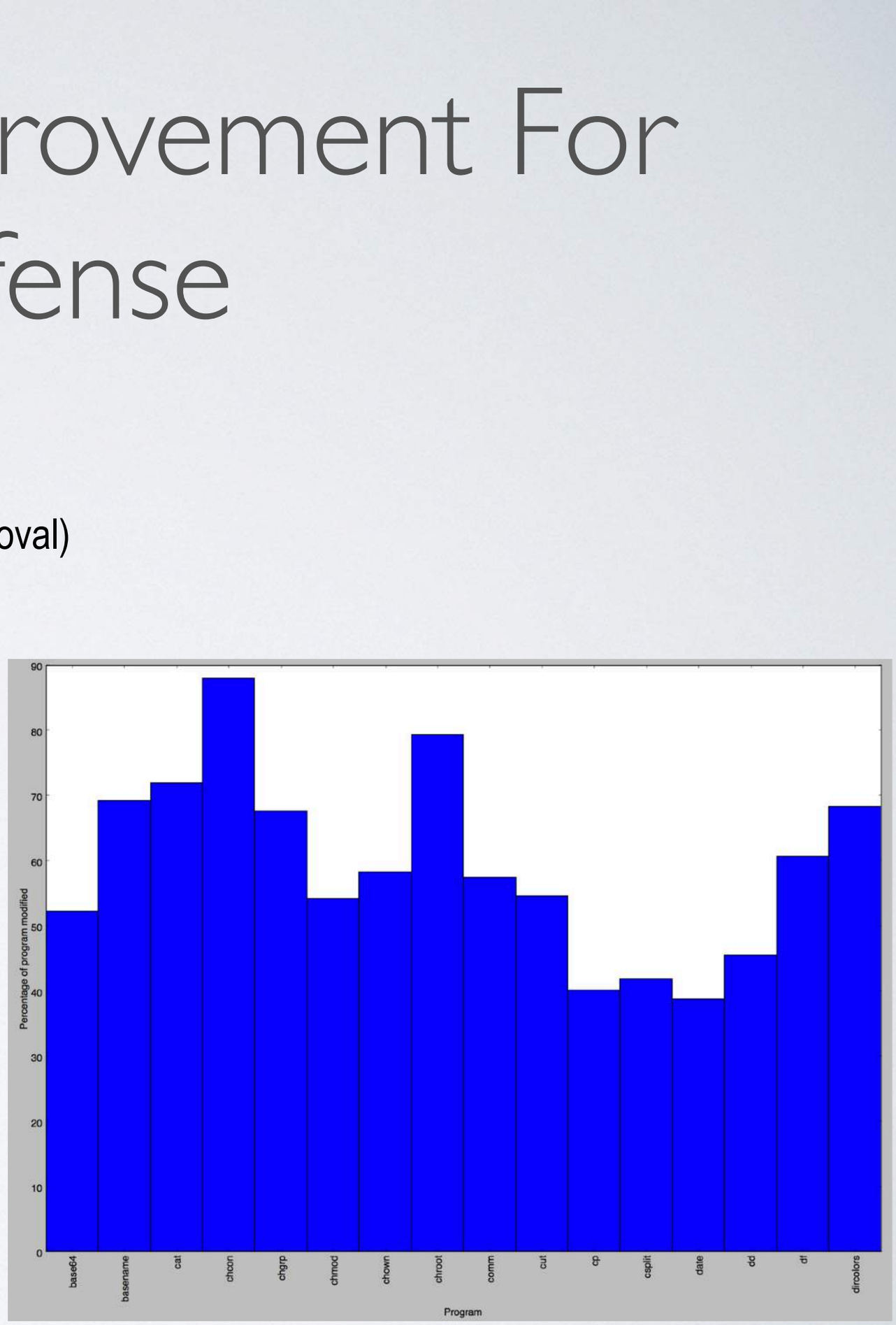
Problem: What if tests suck?

Solution: ~~Give up~~

***Solution:* Evolve better tests**

We need to go deeper

Photo taken by Manel Torralba, 16 November 2016. CC BY 2.0



Fuzzing

Used to stress test software, typically using random or semi-random test data, to uncover issues which may not be revealed in other testing.

Very popular for vulnerability discovery.

- Bug bounties
- Computing resources cost decreasing
- Some care about security

Types Of Fuzzers

Blackbox

- Traditional type of fuzzer
- Supply inputs and observe output (usually crashes)

Whitebox

- Use program analysis
- Microsoft Sage — found ~1/3 of bugs during development of Windows 7

Greybox

- Uses instrumentation
- Less effective than whitebox fuzzing, but more efficient

Dumb

- Unaware of input format or protocol

Smart

- Aware of input format or protocol

Evolutionary Fuzzers

A recent trend which uses evolutionary computation to guide fuzzing.

Earliest known work is by Dr. Jared DeMott (2006) [1]

- Evolutionary Fuzzing System (EFS), built on General Purpose Fuzzer
- Evolutionary greybox fuzzer

American Fuzzy Lop (AFL) (2014) [2]

- Most well known evolutionary greybox fuzzer

Vuzzer (2017) [3]

- Application aware evolutionary greybox fuzzer using dynamic taint analysis

American Fuzzy Lop

A mutation-based dumb evolutionary greybox fuzzer.

Favors tests with crashes, more branches hit, and/or new execution paths.

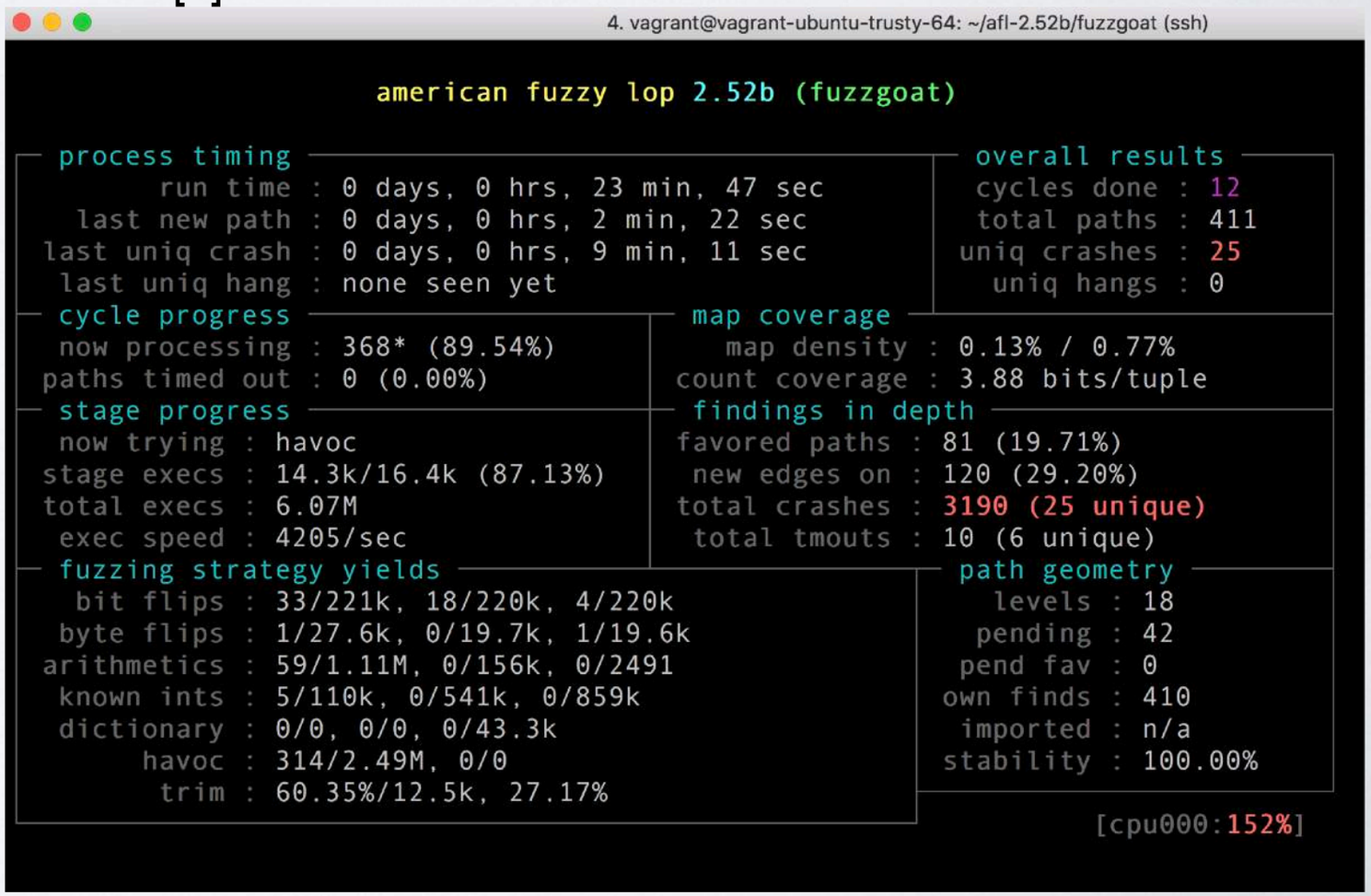
Photo taken by Lithonius, Public domain

Number of notable vulnerabilities found: 371* [2]

Supports fuzzing binaries using QEMU

Forks:

- Kernel system calls [4]
- Virtual Machines [5]



Interesting Fuzzing Examples

Evolutionary fuzzers have been able to learn protocols and file formats.

EFS can learn protocols, such as SMTP, using a handful of protocol related strings. [6]

AFL was used to learn the JPEG file format [7]

- Started with a text file containing: “hello”
- Discovered jpeg header byte
- Generated a grayscale 3x784 pixel image
- Generated images with other patterns:

https://lcamtuf.blogspot.com/2014/11/pulling-jpegs-out-of-thin-air.html

Fuzzgoat [8]

A vulnerable C program, based on a JSON parser

Numerous memory corruption bugs

Starting seed for AFL: {“”:””}

With test seed: 29.4% code coverage (kcov)

After ~43 hours of fuzzing with AFL

- 67 unique crashes found
- 0 hangs
- 944 test cases queued

With 1,011 tests: 90.5% code coverage (kcov)

Example Evolved Test Cases

```
{  =

{“q\Xc\q\Xc\\\\\k\¬L\"""""""\[\Xc\\z\q\'\q\Xc\\z\q\Xc\\z\[\Xc\\z\q\'\q\Xc]]],\q\Xc\\q\z\q\Xc\q\Xc\dz_q\\\\k\¬L\"\[\Xc\
\z\q\'\q\Xc\\z\q\Xc\\z\^M'\^M'k\qL\^W'\q]Xc\dz\qn\\\k\qL\"""""""\¬L\"""""""\[\Xc\\z\q\'\q\Xc\\z\q\Xc\\z\^M'\^M\"""""""\
[\Xc\\z\q\'\q\Xc\\z\q\Xc\\x\""""]]],\q\Xc\\q\z\Y\Xc\q\Xc\dz_q\\\\k\¬L\"""""""\[\Xc\\z\q\'\q\Xc\\z\q\Xc\
\z\^M'\^MGk\qL\^W"""""""\q]Xc\dz\qn\\\k\qL\"""""""]F],\]]],\q\Xc\\^@^P^P^@

{"\u0666!666<9b>6ÿÿöÿ\u0666666666ÿ@ÿÿ666O":{"":"'XX"',"":{"":{"":"'X"',"'^C":{"":"^LX"',"":"%X"',"":{"":
{"":"'X
]]],%X%X”}}}}}}}^@}}}

[{},{},[{},[{},{
},    ],{},[

{},   ],{}, ]     ,[{},[{},{},     ],[{},[{},{
},    ],{},[

{},   ],{}, ]     ,[{},[{},{},     ],{},[

{},   ],{}, ]     ,[{},[{},{},     ],[{},[{},{      …
```

Hunting And Fixing Bugs

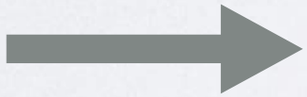
Fixing bugs in your sleep [9] application of dreaming device [10]

Online

By Pixabay. Public domain

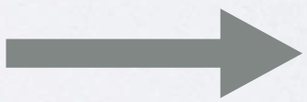
Offline

User generated bugs



Release

Validate



Hunting And Fixing Bugs

Fixing bugs in your sleep [9] application of dreaming device [10]

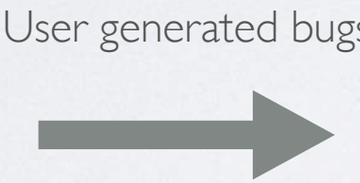
Online

Offline

Release

By Pixabay. Public domain

User generated bugs



Validate

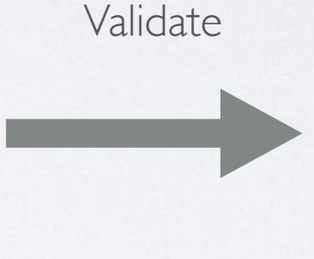
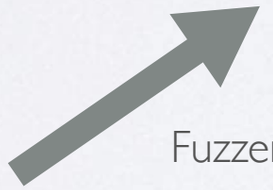


Photo taken by David Whelan, 24 May 2016. Public domain



Fuzzer generated
Bugs

Hunting And Fixing Bugs

Fuzzers can help generate tests, but may be a useful component in automated bug fixing.

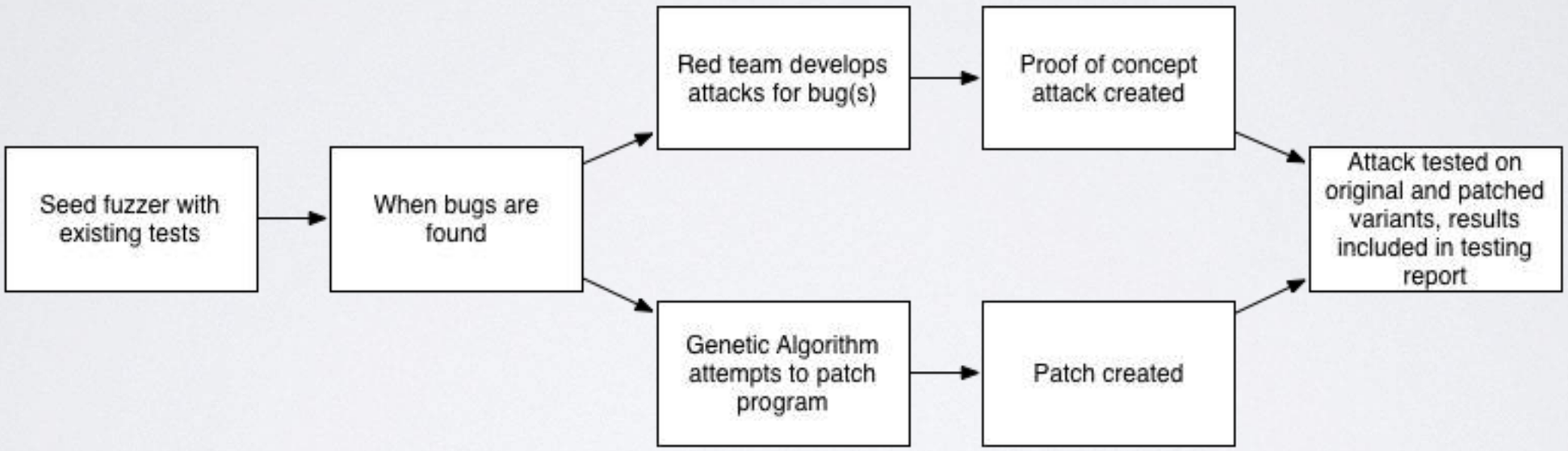
After ~43 hours of fuzzing fuzzgoat with AFL

- **67 unique crashes found**
- 0 hangs
- 944 test cases queued

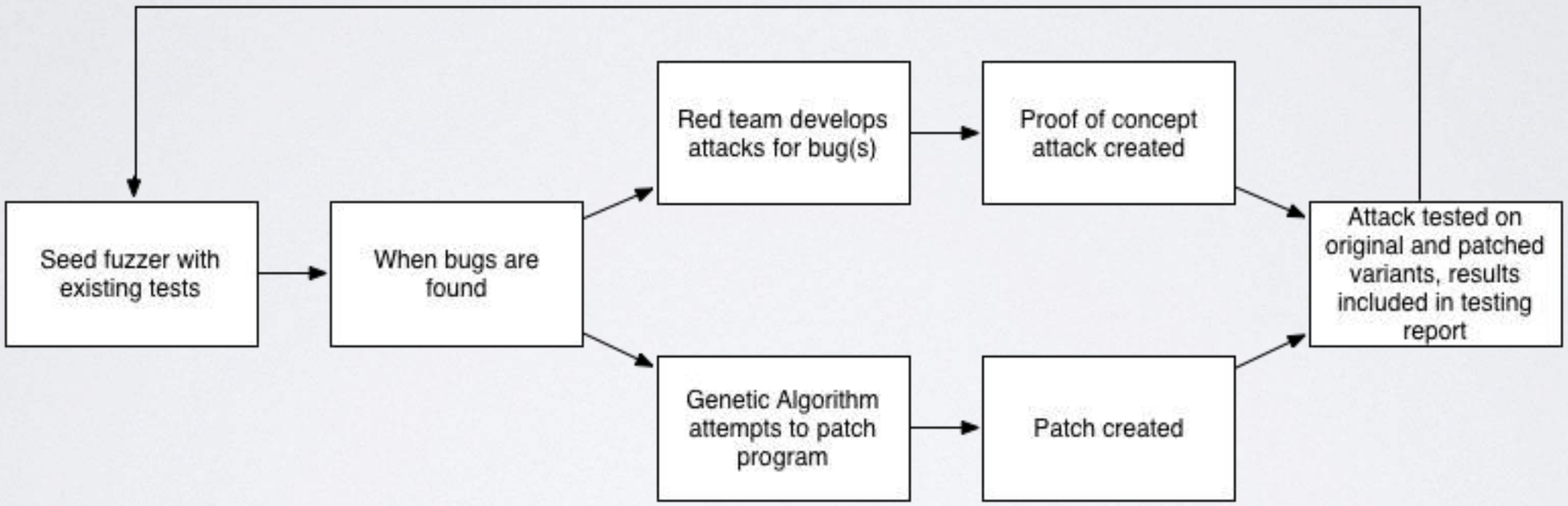
Treat crash test cases as bugs.

Incorporate return code (or lack of erroneous output) into fitness calculation.

Hunting And Fixing Bugs



Automated Hunting And Fixing Bugs



Genetic Improvement For Autonomic Computing

MAPE-K = Monitor Analyze Plan Execute over shared Knowledge

Enables self-* properties (self-configuration, self-healing, self-optimization, self-protection)

Autonomic system to tune and change GI parameters based on goals or priorities and system state.

By Peter-Lomas, Public domain

Photo taken by Kenueone, 5 May 2016. CC0

By Rizkyharis, 3 November 2016. CC BY-SY 4.0

By Sarah, 26 March 2010. CC BY-SA 2.0

Photo taken by Slashme, 5 January 2015. CC BY-SA 4.0

Questions?

GOTO by xkcd [CC BY-NC 2.5 https://creativecommons.org/licenses/by-nc/2.5/] via xkcd

References

- [1] DeMott, Jared. "The evolving art of fuzzing." *DEF CON* 14 (2006).
- [2] Michal Zalewski. American fuzzy lop, 2015. http://lcamtuf.coredump.cx/afl/
- [3] Sanjay Rawat, Vivek Jain, Ashish Kumar, Lucian Cojocar, Cristiano Giuffrida, and Herbert Bos. Vuzzer: Application-aware evolutionary fuzzing. 2017.
- [4] Vegard Nossum and Quentin Casasnovas. Filesystem fuzzing with american fuzzy lop. In Vault 2016, Raleigh, N.C., April 21 2016.
- [5] Jesse Hertz and helped by Tim Newsham. Project triforce: Run afl on everything!, 2016.
- [6] Jared DeMott, Richard Enbody, and William F Punch. Revolutionizing the field of grey-box attack surface testing with evolutionary fuzzing.
- [7] Michal Zalewski. Pulling jpegs out of thin air, November 07 2014. https://lcamtuf.blogspot.com/2014/11/pulling-jpegs-out-of-thin-air.html
- [8] Simon Kagstrom. fuzzgoat - a vulnerable c program for testing fuzzers, January 9 2018. https://github.com/fuzzstati0n/fuzzgoat
- [9] Saemundur O Haraldsson, John R Woodward, AlexanderEI Brownlee, and Kristin Siggeirsdottir. Fixing bugs in your sleep: how genetic improvement became an overnight success. In Proceedings of the Genetic and Evolutionary Computation Conference Companion, pages 1513–1520. ACM, 2017
- [10] Mark Harman, Yue Jia, William B Langdon, Justyna Petke, Iman Hemati Moghadam, Shin Yoo, and Fan Wu. Genetic improvement for adaptive soft- ware engineering (keynote). In Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, pages 1–4. ACM, 2014.
- [11] Jeffrey O Kephart and David M Chess. The vision of autonomic computing. Computer, 36(1):41–50, 2003