ForAllSecure

Welcom to the Mayhem Heroes Hackathon!

Hackathon Discord

Join us on Discord! You can message with questions, ideas, or anything you'd like:

Invite Link:

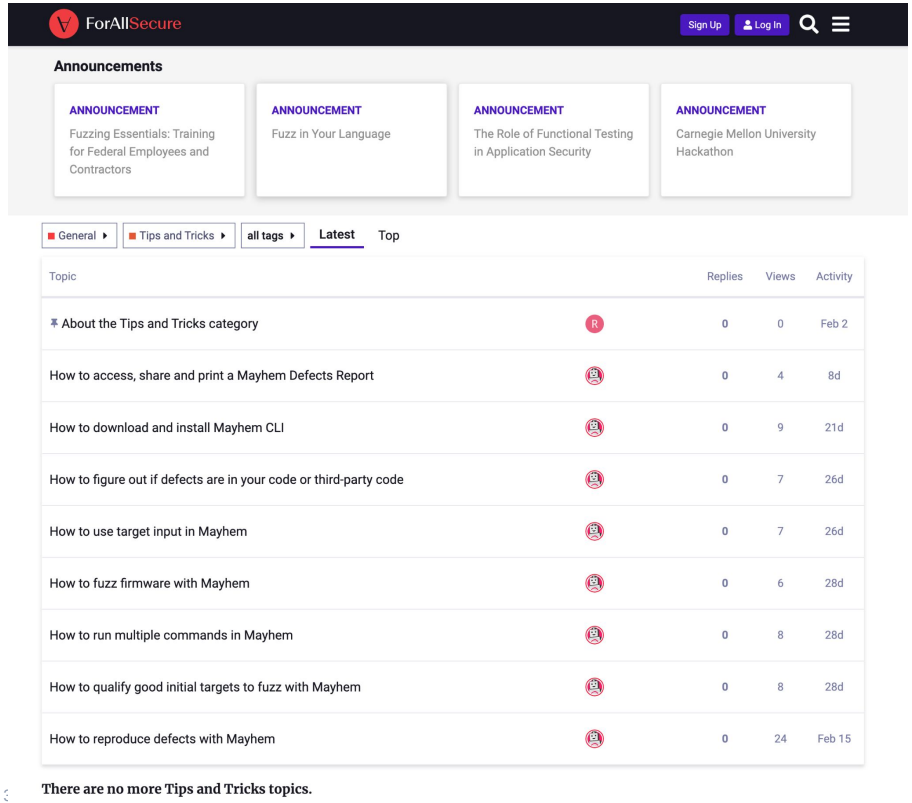
https://discord.gg/KhSGZnrEpp

Or, scan the QR Code with your phone:

ForAllSecure

Sign Up to the Community

- Navigate to **community.forallsecure.com**
- Click **Sign Up** at the top of the page
- Select **“With Github”**
- Login to Github if not already logged in
- **You are now ready to explore the Community**
- Navigate to **General > Tips and Tricks**
- **Learn Mayhem Hacking Best Practices!**



ForAllSecure

Agenda

- Mission
- Fuzzing Introduction ~45 mins
  - Mayhem Setup
  - Fuzzing Lighttpd
  - Fuzzing Overview and Fuzz Targets
- Getting Help ~10 mins
  - Community and Discord
- Mayhem CLI ~15 mins
- Packaging for Mayhem ~30 mins
  - Docker overview
  - Docker + Mayhem Exercise
- Build Systems ~30 mins
  - Common Patterns
  - CMake Exercise

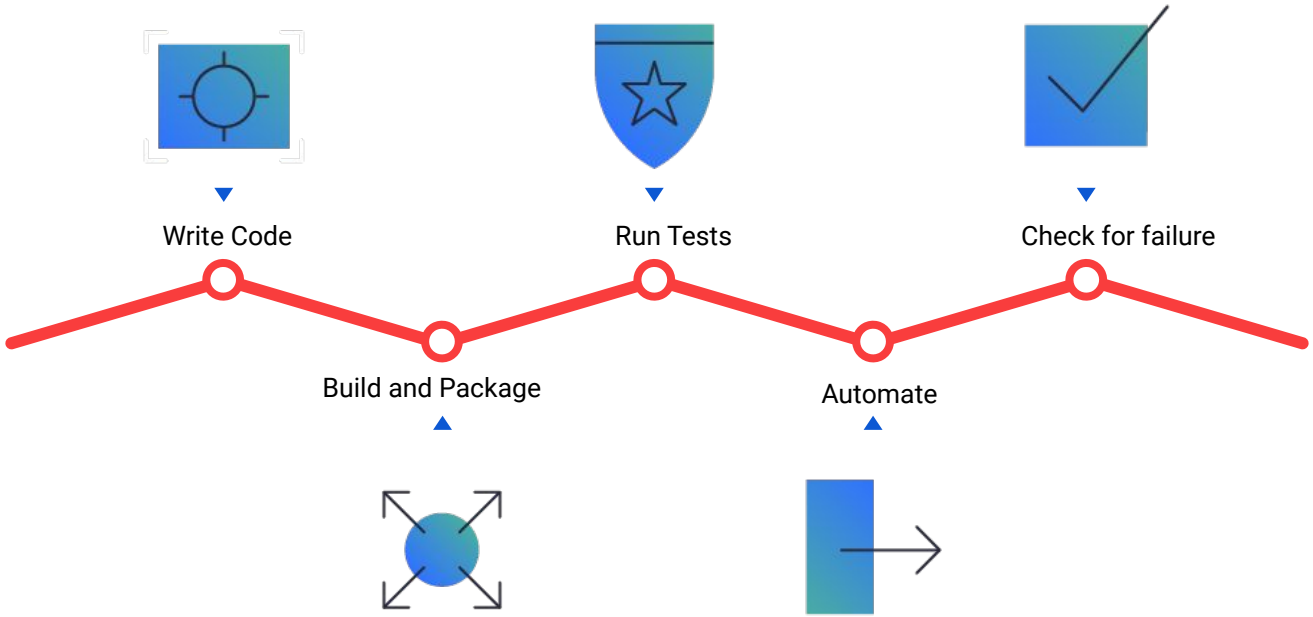
~ Break ~ 30 mins

- OSS Fuzzers ~30 mins
  - libFuzzer Example
  - libFuzzer Exercise
- GitHub Actions ~30 mins
- Case Study ~20 mins
- Conclusions ~10 mins
  - Training Survey
- Mayhem Heroes ~10 mins
- Survey and Conclusion ~10 mins

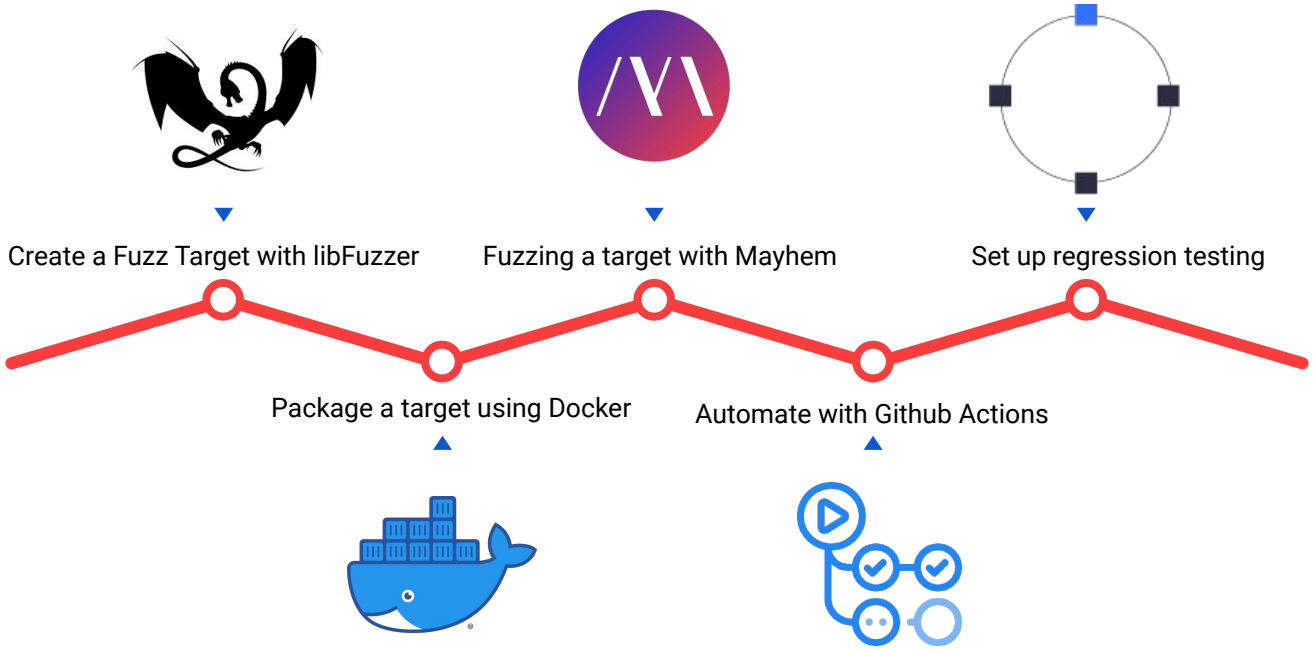
After this training, you will be able to

- Explain what fuzz testing is
- Fuzz an application with Mayhem
- Reproduce a found defect
- Package a fuzz target using Docker
- Build a fuzz target
- Add instrumentation to a target with libFuzzer
- Automate the whole process using Github Actions

Modern CI/CD Pipeline



Fuzzing CI/CD Pipeline



Mission

Fact

Growing cybersecurity talent shortage

570x

Developers than cybersecurity experts

Worse

We're shipping code faster than ever, as insecure as ever

87%

Of application security isn't automated today

Result

Offense has a permanent advantage today.

Status quo

Is not working.

GitHub, Microfocus, SANS

ForAllSecure

fluorescence

Richard Zhu

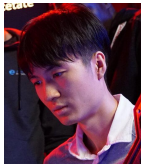
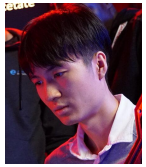
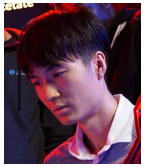
Image Credit: https://threatpost.com/pwn2own-tokyo-2019-amazon-echo-hackers/150033/

Vision

A world where computers automatically test and protect the world's software from exploitable bugs at speeds and scale faster than any attacker.

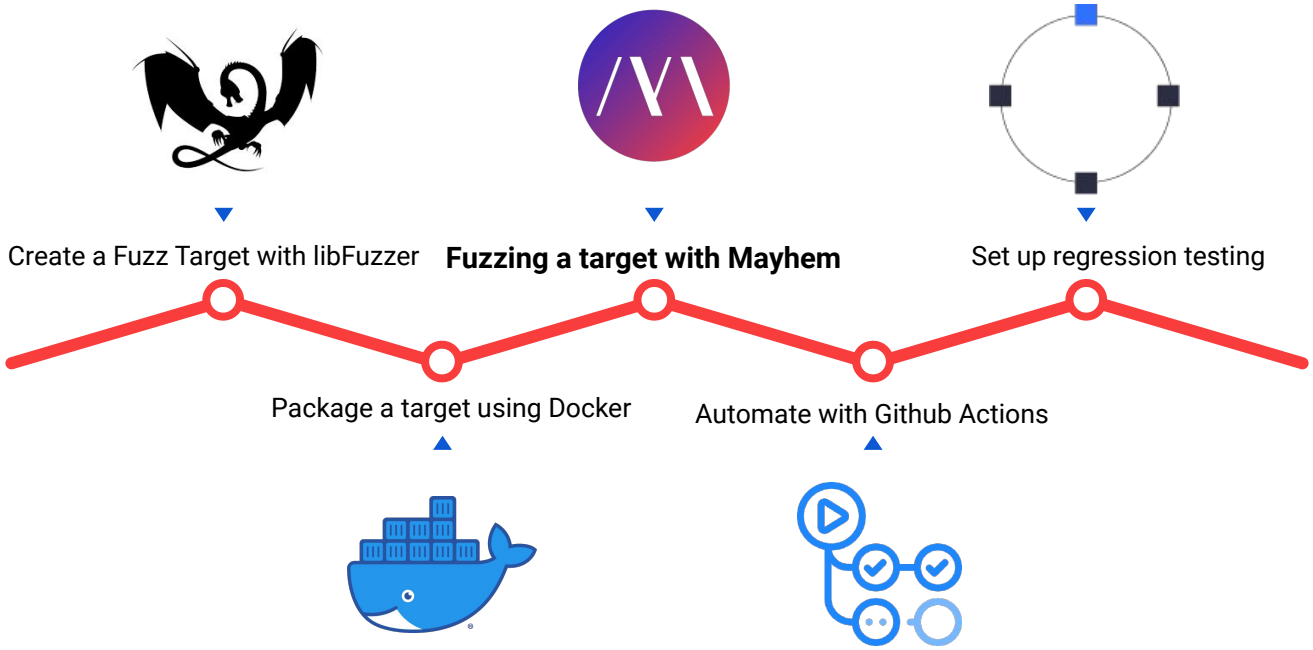
Making It Autonomous

Suppose you managed a team of 3 humans looking for zero days

1. **Work on the same app**
   - ➢ Software registry w/ docker
2. **Run the app the same way**
   - ➢ Mayhemfile
3. **Share work and reproduce results**
   - ➢ Results test corpus
4. **Focus on a particular part of app**
   - ➢ Harnessing
5. **Work intelligently**
   - ➢ Advanced algorithms

Fuzzing CI/CD Pipeline



Fuzzing / fəzing / v.

The process of automatically sampling and testing the program input space to elicit new program behaviors.

Fuzz testing is a technique appsec experts leverage.

Open Source software is everywhere. So let's fuzz it before the bad guys do.

Lighttpd (“Lighty”)

Lighttpd

lighttpd is an open-source web server optimized for speed-critical environments while remaining standards-compliant, secure and flexible. It was originally written by Jan Kneschke as a proof-of-concept of the c10k problem – how to handle 10,000 connections in parallel on one server,but has gained worldwide popularity. Its name is a portmanteau of "light" and "httpd". Wikipedia

Original author(s): Jan Kneschke

Written in: C

Available in: English

Type: Web server

License: 3-clause BSD

LX 4008 Switch

Siemens SINAUT
MD741-1 EGPRS-Router

Lighttpd (“Lighty”)

SHODAN lighttpd-1.4.15 Explore Pricing Enterprise Access

Exploits Maps

TOTAL RESULTS

557

TOP COUNTRIES

Germany	260
Korea, Republic of	98
Italy	56
Spain	41
United States	41

TOP SERVICES

HTTPS	382
HTTP	66
AndroMouse	29
HTTP (81)	26
Symantec Data Center Security	24

TOP ORGANIZATIONS

Deutsche Telekom AG	190
Korea Telecom	57
Telekom Deutschland GmbH	54
Telecom Italia Mobile	52
Telefonica de Espana	26

New Service: Keep track of what you have connected to the Internet. Check out Shodan Monitor

401 - Unauthorized

2.193.198.147
Telecom Italia Mobile
Added on 2021-03-13 19:42:12 GMT
Italy, Bari

SSL Certificate
Issued By:
|- Organization: Siemens AG
Issued To:
|- Organization: Siemens AG
Supported SSL Versions
SSLv2, SSLv3, TLSv1

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Digest realm="SINAUT MD741-1", nonce="6f7aff042563c0437ef134925ae27a22", qop="auth"
Content-Type: text/html
Content-Length: 351
Date: Wed, 24 Feb 2021 06:19:55 GMT
Server: lighttpd/1.4.15
```

401 - Unauthorized

37.84.234.242
Telekom Deutschland GmbH
Added on 2021-03-13 19:09:02 GMT
Germany, Bingen am Rhein

SSL Certificate
Issued By:
|- Organization: Siemens AG
Issued To:
|- Organization: Siemens AG
Supported SSL Versions
SSLv2, SSLv3, TLSv1

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Digest realm="SINAUT MD741", nonce="d19ae2ae87b3db41b79a0108be83e91c", qop="auth"
Content-Type: text/html
Content-Length: 351
Date: Sat, 13 Mar 2021 17:12:56 GMT
Server: lighttpd/1.4.15
```

401 - Unauthorized

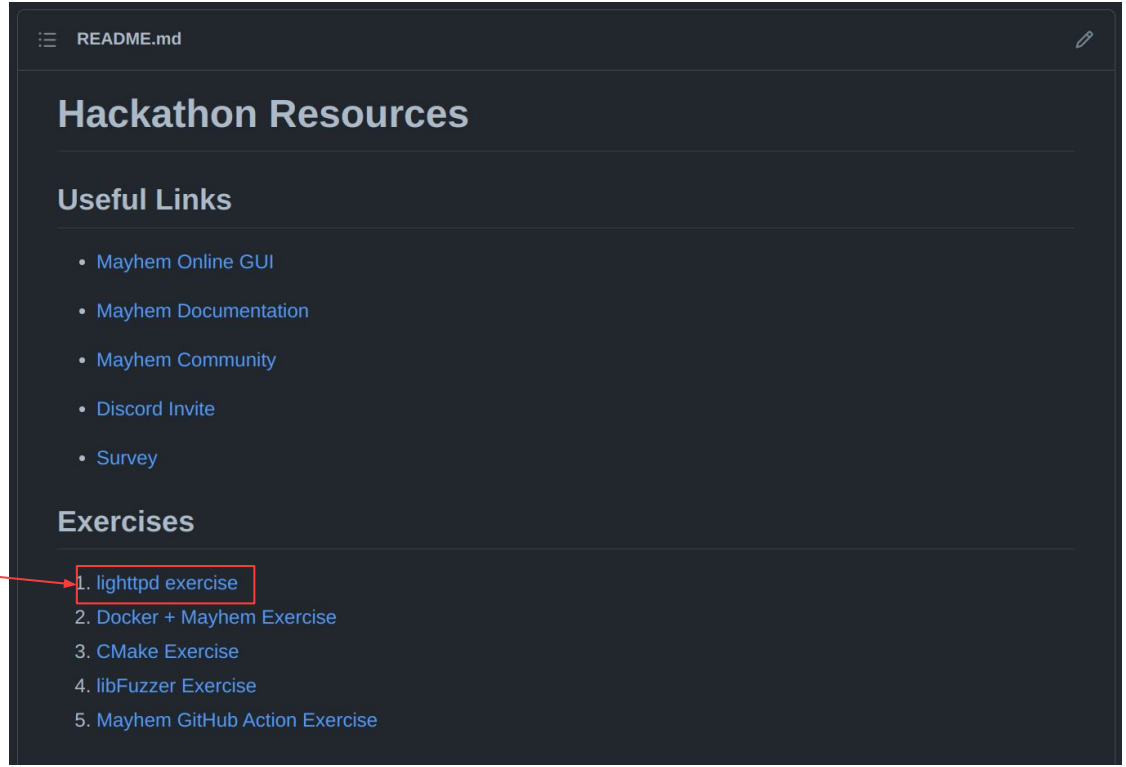
46.16.218.88
mdex46-16-218-88.publicip.mdex.biz
Wireless Logic mdex GmbH
Added on 2021-03-13 19:31:16 GMT
Germany, Norderstedt

SSL Certificate
Issued By:
|- Organization: Siemens AG
Issued To:
|- Organization: Siemens AG
Supported SSL Versions
SSLv2, SSLv3, TLSv1

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Digest realm="SCALANCE M-873-0", nonce="73eab94066df68e3d26f814d32c61890", qop="auth"
Content-Type: text/html
Content-Length: 351
Date: Sat, 13 Mar 2021 17:17:50 GMT
Server: lighttpd/1.4.15
```

https://github.com/mayhemheroes/hackathon-resources

Go here



Click here

Quiz 1

What do fuzz testing and electricity have in common?

A. They both are measured in Ohms

B. They both have lightning to thank for leading to their discovery

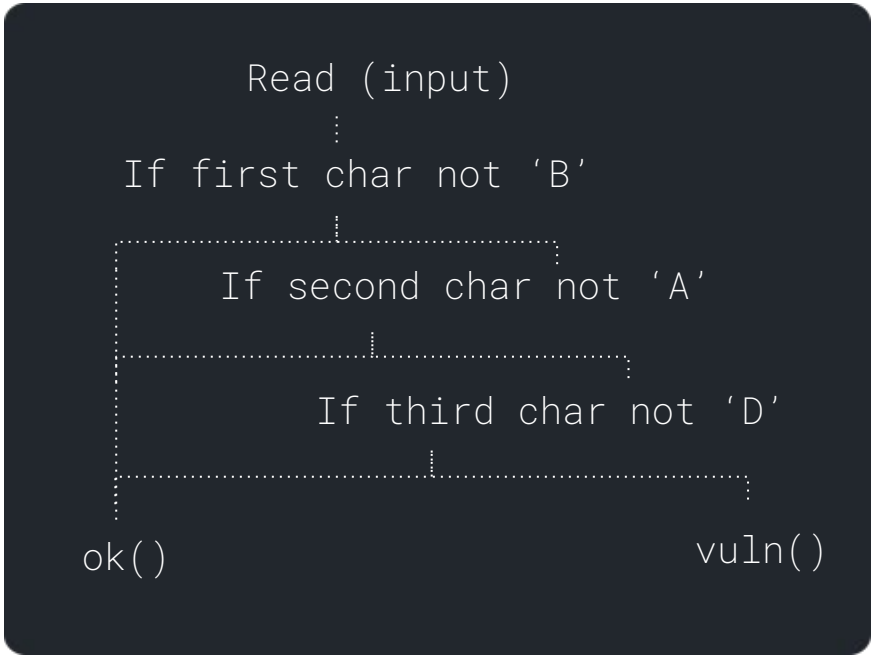
C. They were both unsuccessfully patented in the U.S

D. Nothing!

Introduction to Fuzz Testing

Covering the Bases

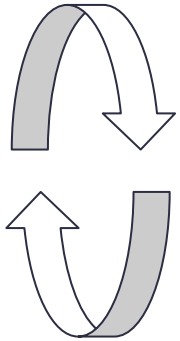
- “MOM” is ok()
- “DAD” is ok()
- “BOB” is ok()
- “BAR” is ok()
- “BAD” is vuln()!



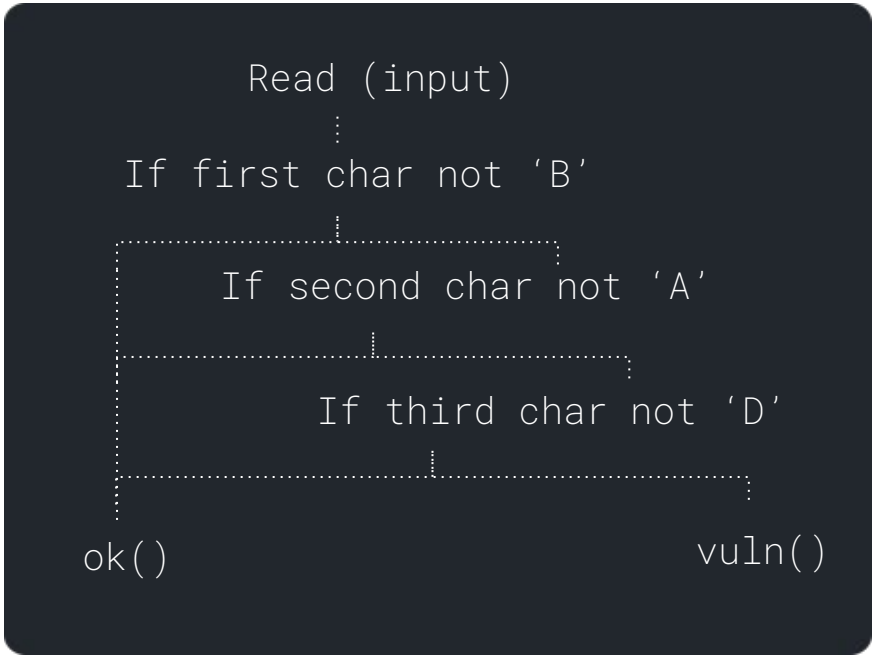
Simple Fuzzing Dataflow

Fuzzing Engine

- “MOM” is ok()
- “DAD” is ok()
- “BOB” is ok()
- “BAR” is ok()
- “BAD” is vuln()!

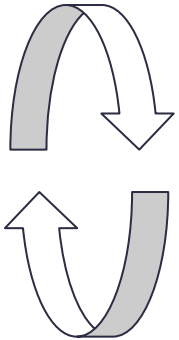


Fuzz Target



Simple Fuzzing Dataflow

Fuzzing Engine



Fuzz Target

How do we get here?

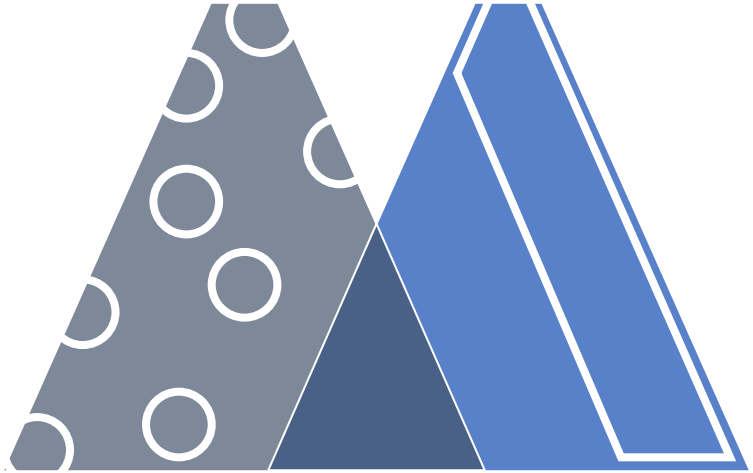
Mayhem Advanced Fuzz Testing

Guided Fuzzing

Uses sample inputs and target feedback to generate new test cases on-the-fly:

UNIQUE INPUTS: ↓

EXECUTION SPEED: ↑



Symbolic Execution

Models code as mathematical equations, and then solves them to come up with inputs:

UNIQUE INPUTS: ↑

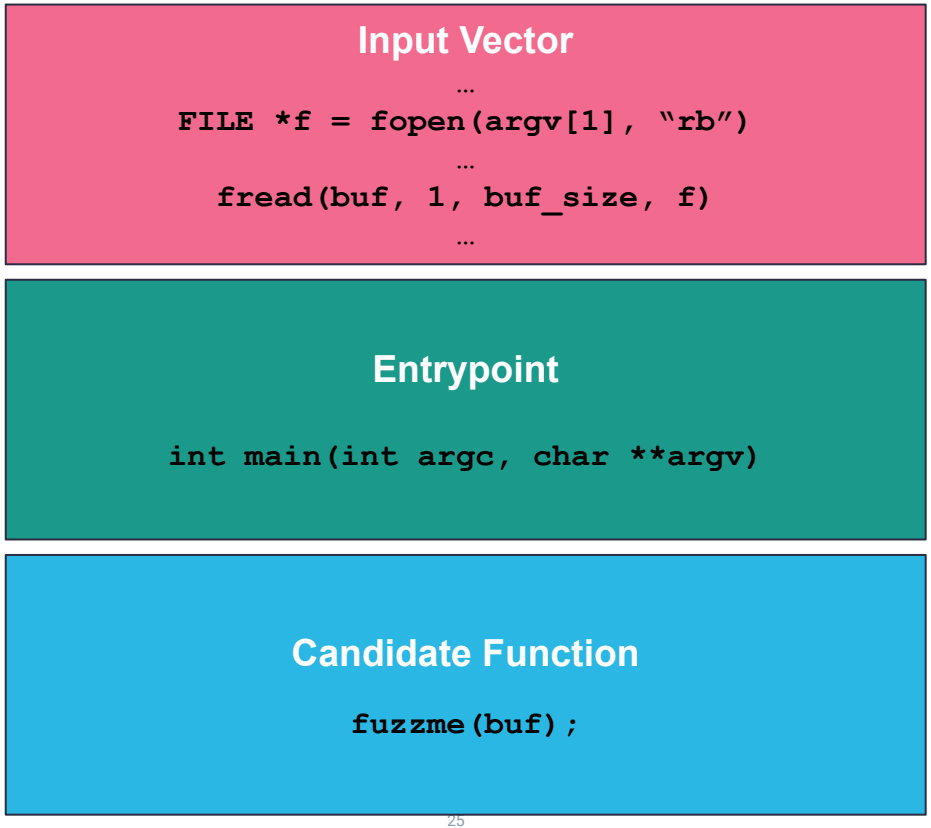
EXECUTION SPEED: ↓

Advanced Fuzzing

UNIQUE INPUTS: ↑

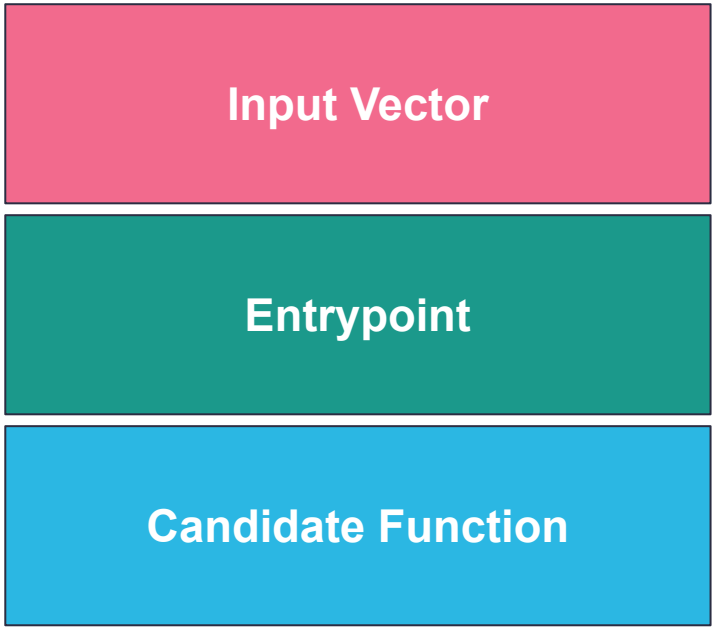
EXECUTION SPEED: ↑

Example Fuzz Target



Example Fuzz Target

```
#include <stdio.h>
#include <stdlib.h>

int fuzzme(char *buf)
{
  if(strlen(buf) >= 3)
    if(buf[0] == 'b')
      if(buf[1] == 'u')
        if(buf[2] == 'g') {
          printf("You've got it!");
          abort();
        }
  return 0;
}

int main(int argc, char **argv)
{
…
  if (NULL == ( f = fopen(argv[1], "rb") )) {
    fprintf(stderr, "could not open file for fuzzing\n");
    return 2;
  }

  if (!(sz = fread(buf, 1, sizeof(buf), f) )) {
    fprintf(stderr, "no data read from file\n");
    return 3;
  }

  fuzzme(buf);
…
}
```



Example Fuzz Target

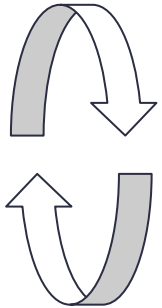
Input Vector

Entrypoint

Candidate Function

Fuzzing Engine

Mayhem for Code



./example_target @@

Fuzz Target

```
#include <stdio.h>
#include <stdlib.h>

int fuzzme(char *buf)
{
  if(strlen(buf) >= 3)
    if(buf[0] == 'b')
      if(buf[1] == 'u')
        if(buf[2] == 'g') {
          printf("You've got it!");
          abort();
        }
  return 0;
}

int main(int argc, char **argv)
{
…
  if (NULL == ( f = fopen(argv[1], "rb") )) {
    fprintf(stderr, "could not open file for fuzzing\n");
    return 2;
  }

  if (!(sz = fread(buf, 1, sizeof(buf), f) )) {
    fprintf(stderr, "no data read from file\n");
    return 3;
  }

  fuzzme(buf);
…
}
```

Quiz 2

Fuzz Testing has found *this* many defects in Linux alone

A. 110

B. 1,100

C. 11,000

D. 110,000

Mayhem CLI

Mayhem from the Command Line

```
$ mayhem --help
usage: mayhem [-h] [-y] [--verbosity
{info,debug}] [--version] CMD ...
Command line client for interacting with Mayhem

positional arguments:
    check               Check targets to see if
they are Mayhem-eligible.
    docker-registry     Gets the URI for
mayhem's docker registry.
    download            Download a target and
its test cases.
    init                Generate a Mayhemfile
    list                List projects and
targets you have run.
    login               Login to a Mayhem
server.
    logout              Logout from a Mayhem
server.
    package             Package the given target and
dependencies for Mayhem.
    run                 Run a target through Mayhem.
    show                Show Mayhem run(s).
    stop                Stop a Mayhem run.
    sync                Sync a package to its latest
state.
    validate            Validate a Mayhemfile.
    wait                Wait for a run to finish.

optional arguments:
  -h, --help            show this help message and
exit
  -y, --noninteractive  Accept the default options
for all prompts. Equivalent to setting the
`MAYHEM_NONINTERACTIVE` environment variable.
  --verbosity {info,debug}
                        Set mayhem verbosity level.
  --version             Get the version of the
Mayhem CLI.
```

Mayhem from the Command Line

```
$ mayhem --help
usage: mayhem [-h] [-y] [--verbosity
{info,debug}] [--version] CMD ...
Command line client for interacting with Mayhem

positional arguments:
    check                 Check targets to see if
they are Mayhem-eligible.
    docker-registry       Gets the URI for
mayhem's docker registry.
    download              Download a target and
its test cases.
    init                  Generate a Mayhemfile
    list                  List projects and
targets you have run.
    login                 Login to a Mayhem
server.
    logout                Logout from a Mayhem
server.
    package               Package the given target and
dependencies for Mayhem.
    run                   Run a target through Mayhem.
    show                  Show Mayhem run(s).
    stop                  Stop a Mayhem run.
    sync                  Sync a package to its latest
state.
    validate              Validate a Mayhemfile.
    wait                  Wait for a run to finish.

optional arguments:
  -h, --help              show this help message and
exit
  -y, --noninteractive    Accept the default options
for all prompts. Equivalent to setting the
`MAYHEM_NONINTERACTIVE` environment variable.
  --verbosity {info,debug}
                          Set mayhem verbosity level.
  --version               Get the version of the
Mayhem CLI.
```

Mayhemfiles

```
# Namespaced project name that the target belongs to
project: myRepo

# Target name (should be unique within the project)
target: myTarget

# Base image to run the binary in.
image: ghcr.io/xansec/myTarget:latest

# List of commands used to test the target
cmds:

    # Command used to start the target, "@@" is the input file
    # (when "@@" is omitted Mayhem defaults to stdin inputs)
    # NOTE: There is normally only one command per target. Mayhem does not support multiple
    # commands in this section
    - cmd: /myTarget
```

Lab 1b: Run with the Mayhem CLI

From lighttpd exercise, go to this section

Lab 1b: Run with the Mayhem CLI

Overview

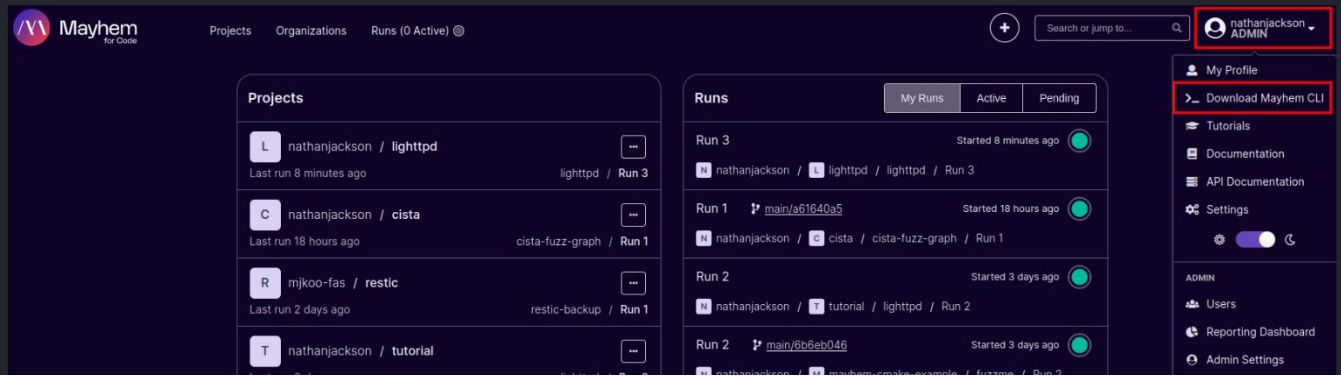
This guide will show you how to start a run using the Mayhem CLI.

Time to complete: About 5 minutes

Step 1. Install the Mayhem CLI

Before using the Mayhem CLI, you need to install it. Fortunately, Mayhem provides instructions for you.

- Click on "Download Mayhem CLI"



Quiz 3

Fuzz Testing can help you find the following types of bugs:

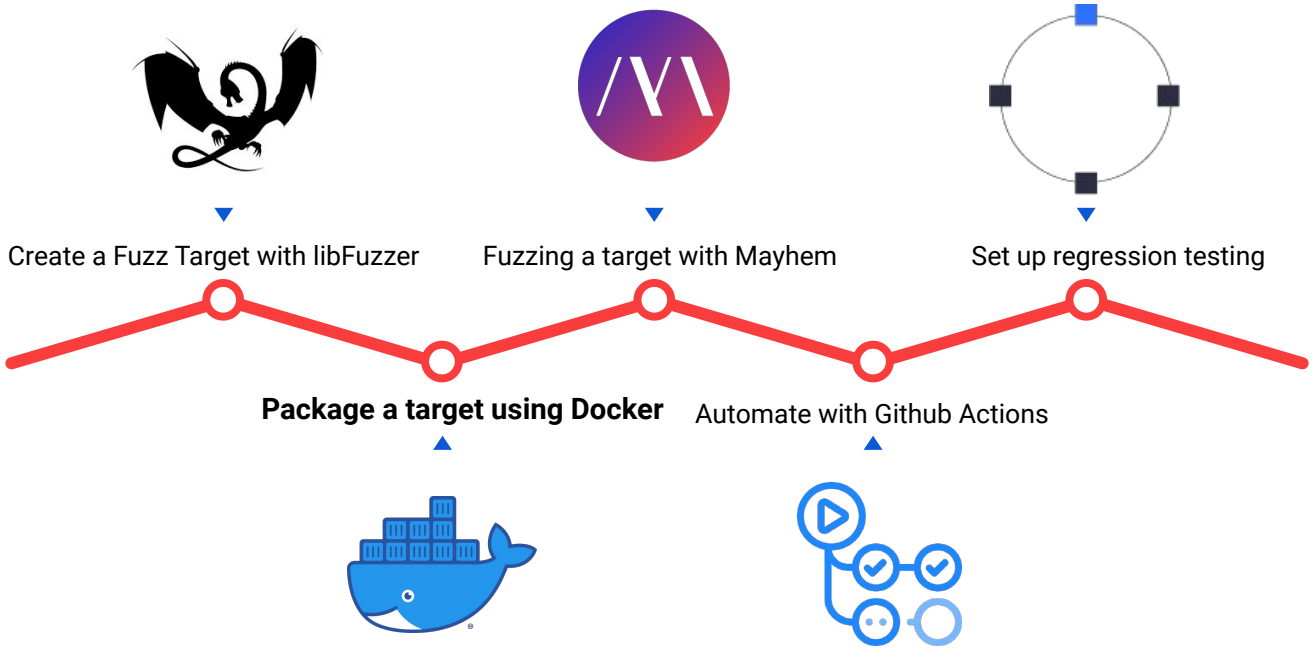
A. Memory Leaks

B. Program Crashes

C. Buffer Overflows

D. Uncaught Exceptions

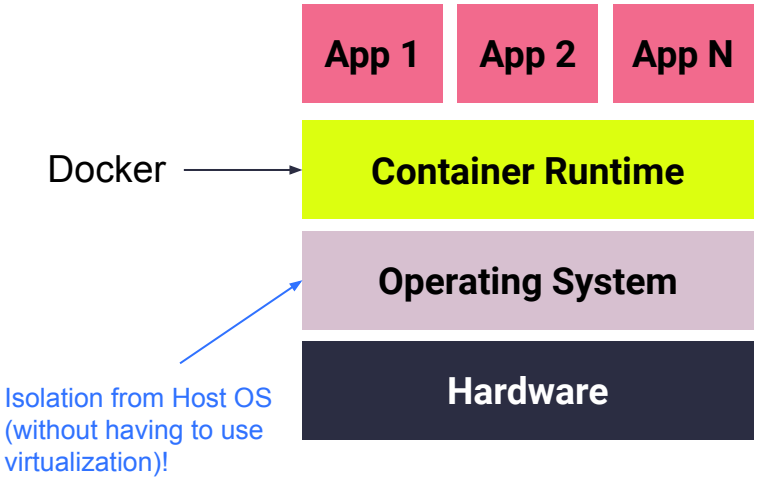
Fuzzing CI/CD Pipeline



Mayhem Target Requirements

- Mayhem performs Dynamic Analysis
  - Runs a target, monitor its behavior. Requires dependencies to be present at runtime.
- Mayhem is fuzzing, what happens when an input does something destructive?
  - Isolation Needed
- Targets need to run the same way over and over.
- Successful fuzzing depends on throughput (i.e., how fast can I run the target?)

What is Docker? Why do we need it?



- Mayhem accepts various types of packages, but the easiest to use is Docker images
- Docker is the industry standard for Containers.
  - Not just for Mayhem!
- Perfect for Mayhem's use case!
- We'll go over building, distributing, and running containers.

Docker Basics

Dockerfile - Configuration as Code that describes how to build a Docker Image

```
$ docker build -t <image-name> .
```

- This will build a Docker **image** based on the instructions in the Dockerfile

```
$ docker run <image-name>
```

- This will run a **container** based off of the specified image

```
$ docker push
```
or
```
$ docker pull
```

Either uploads a local image to a registry or downloads an image from a registry

A **Dockerfile** is like a cake recipe. A **container** is like a cake!

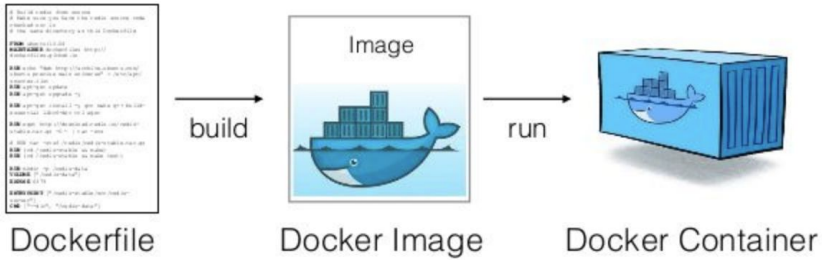
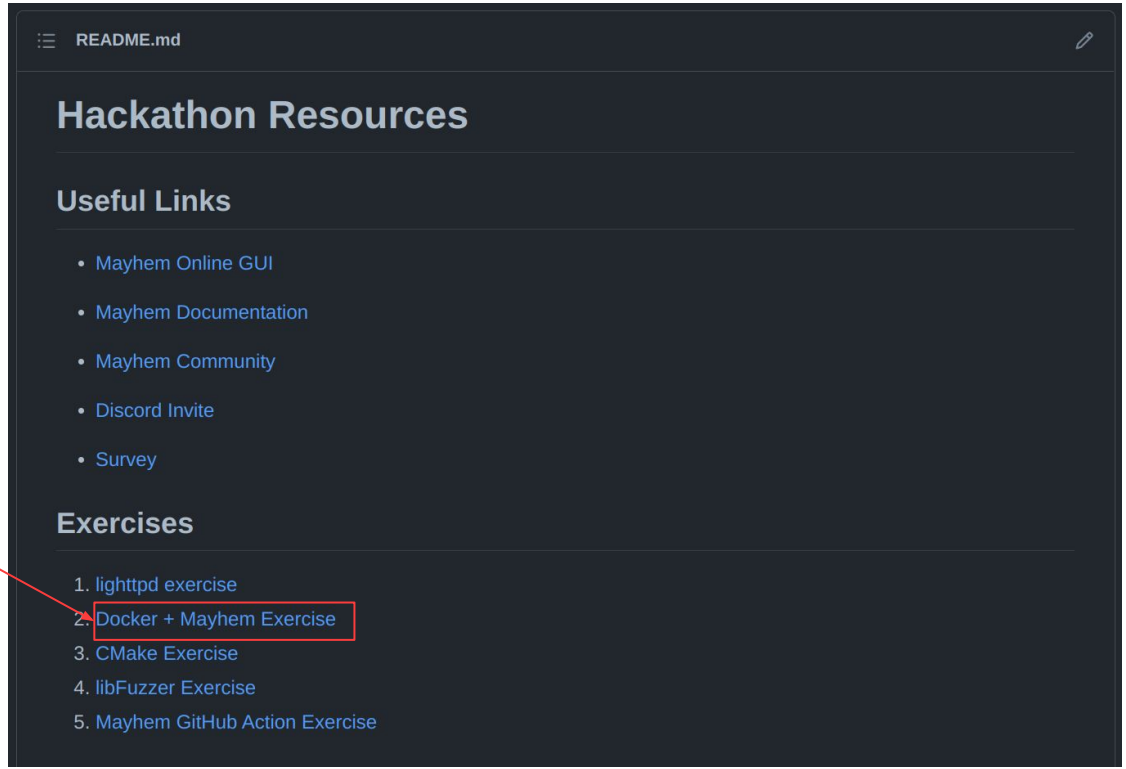


image source: https://medium.com/platformer-blog/practical-guide-on-writing-a-dockerfile-for-your-application-89376f88b3b5

Docker + Mayhem Exercise

https://github.com/mayhemheroes/hackathon-resources

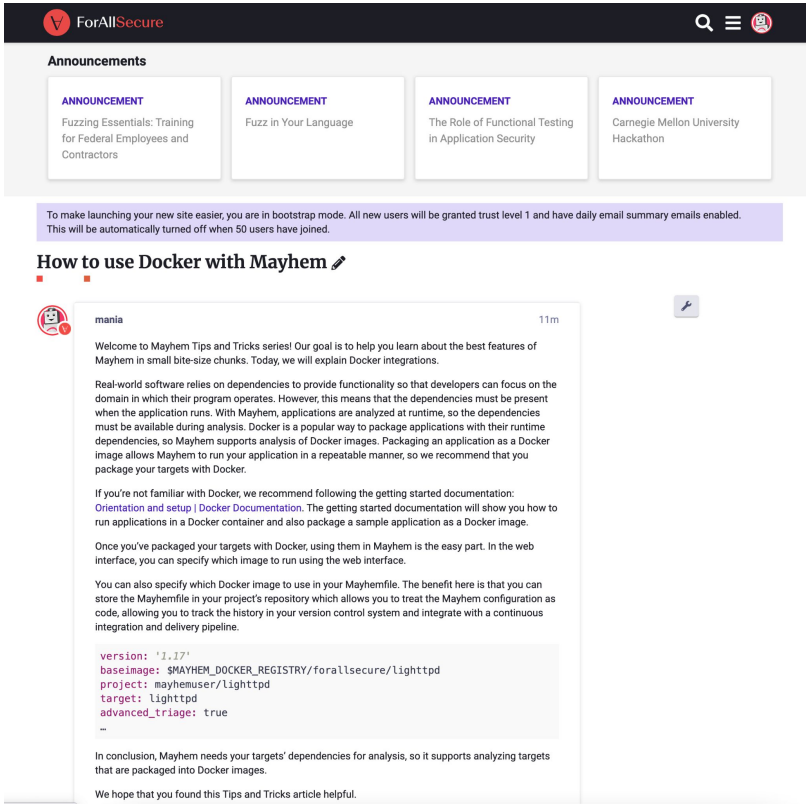
Go here



Click here

Docker Review

- Pulled Image from Docker Hub and Created a Container
- Built our own Docker image and pushed it into GitHub's Registry.
- Started a Mayhem Run that uses our new Docker image.
- See the community Tips and Tricks, under General.



Quiz 4

Fuzz testing led to the discovery of one of the most critical vulnerabilities in the TLS protocol in the past decade, called:

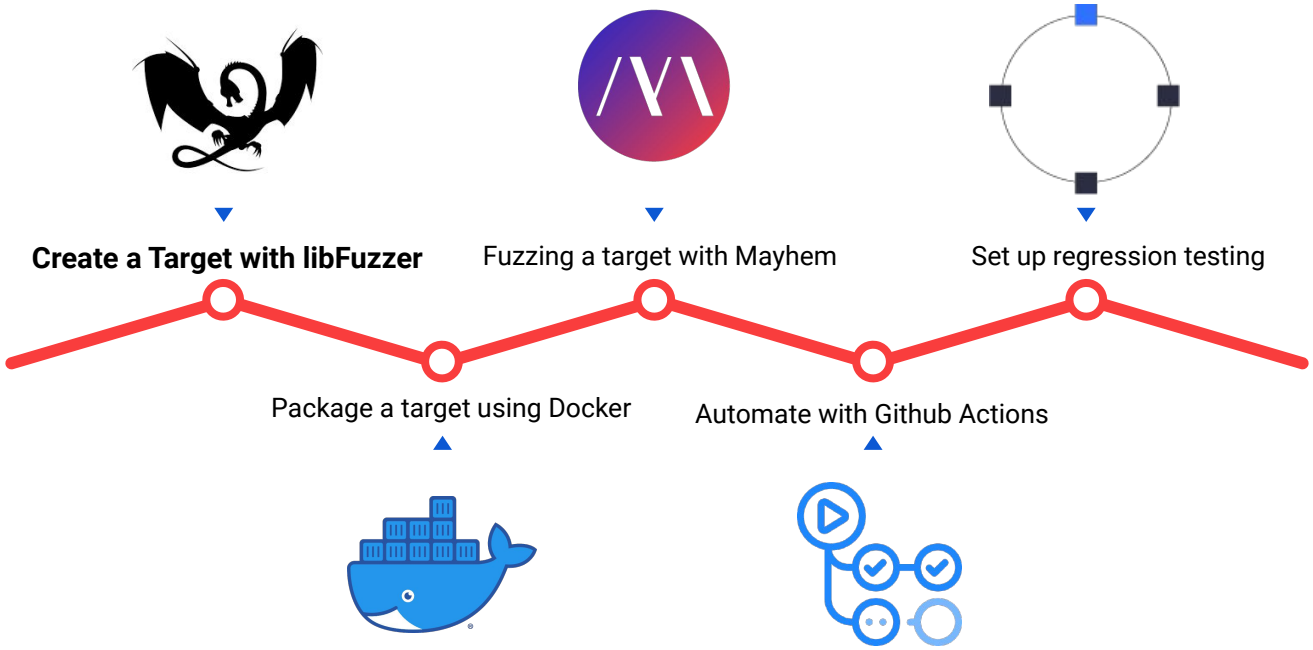
A. Hungerstrike

B. Fullthrottle

C. Heartbleed

D. Tidalwave

Fuzzing CI/CD Pipeline



Before we Begin: Build Systems

- OSS and Industry use build systems to enable automation and repeatability of compilation steps.
- Projects written in compiled languages need to be built before the code is run!
- Running the compiler manually or running IDE-specific project files don’t automate well

Common Patterns with Build Systems

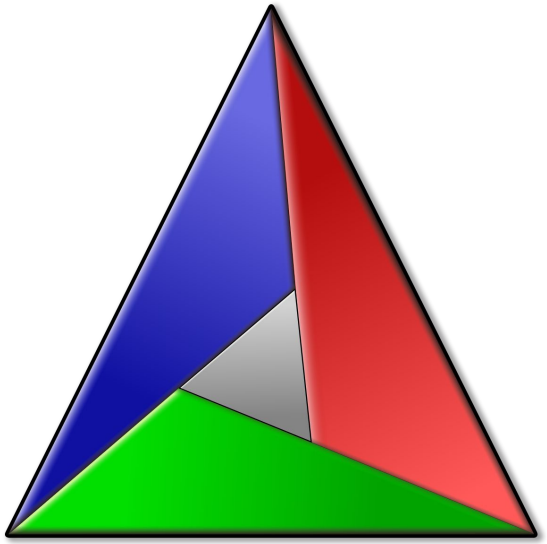
1. Configure Step - Generates configuration or other build system files.

2. Build Step - Use the configuration files generated in the previous step to build an executable

Identifying Build Systems

- C, C++
  - autotools - look for a “`configure`” script.
  - CMake - look for a “CMakeLists.txt” file.
- Rust - Uses Cargo
  - Look for “Cargo.toml”
- Go - Uses “go” CLI.

CMake Example Usage



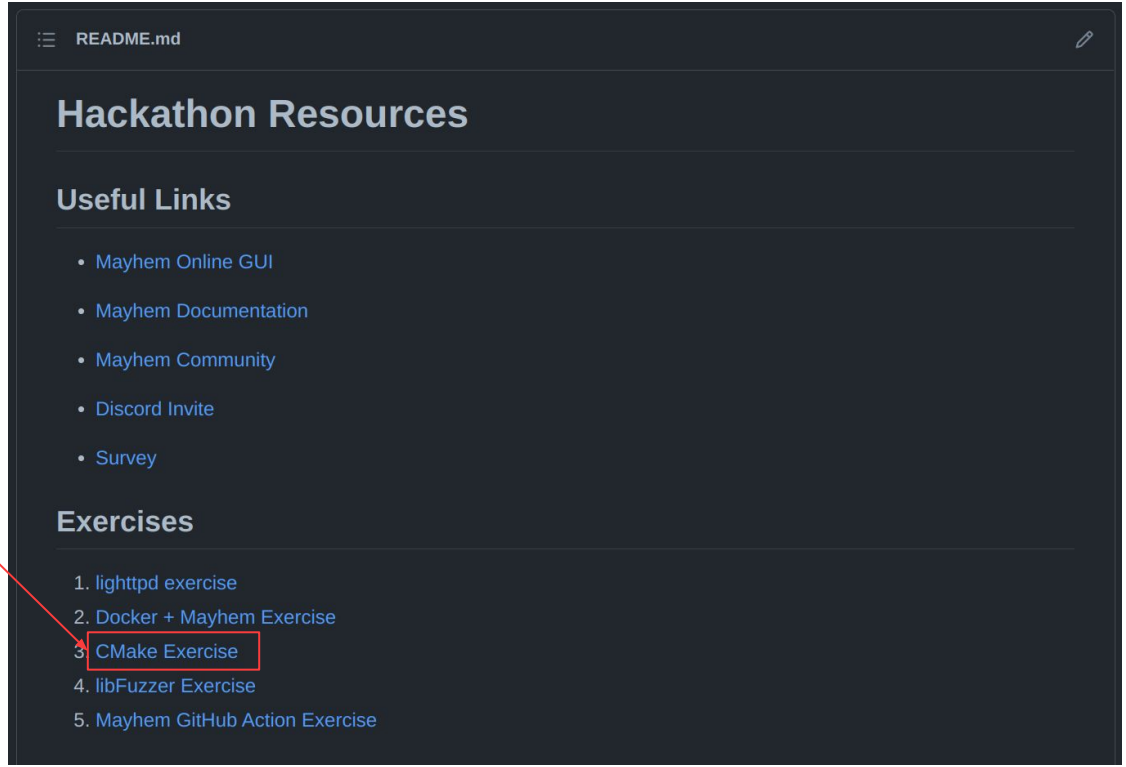
```
# Create directory for out-of-tree build.
$> mkdir build

# Change into build directory.
$> cd build/

# Generate Makefiles.
$> cmake ..

# Run make
$> make
```

CMake Exercise

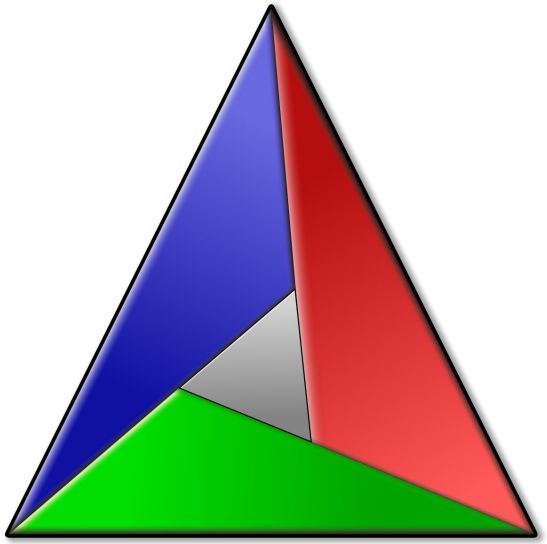
https://github.com/mayhemheroes/hackathon-resources

Go here

Click here



CMake Review



- Generated Build Files with `cmake`.
- Used `make` to perform build.
- Ran an executable generated by a CMake build system.
- Pattern:
  - Generate Build Files\Configure Step
  - Build Step (usually Make)

Quiz 5

Which of the following tools is NOT an open source fuzzing utility?

A. American Fuzzy Lop (AFL)

B. Wasp

C. libFuzzer

D. Honggfuzz

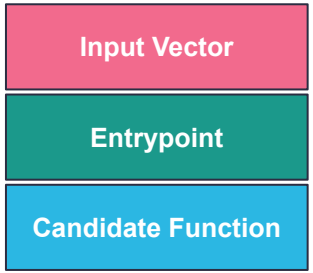
Mayhem + OSS Fuzzers

- Not only does Mayhem support binary-only analysis, but it also analyzes targets built with open source fuzzing technology.
  - AFL
    - Compile-time instrumentation with GCC/Clang (source required)
    - Works with programs that accept network of file input.
  - libFuzzer
    - Compile-time instrumentation with Clang (`clang -fsanitize=fuzzer`)
    - In-memory fuzzer (it runs really, really fast)
  - go fuzz, cargo fuzz
    - Open source fuzzers for Go and Rust - built into the language!
- Many of the repositories in the hackathon already have fuzzing targets. Bootstrap them in Mayhem!

libFuzzer

- Popular framework in Open Source and Industry for fuzzing.
- Part of the LLVM Project, requires clang\clang++.
- Requires the creation of a special function called: `LLVMFuzzerTestOneInput`
- Typically used with Address Sanitizer to find memory corruption bugs.
  - `clang -fsanitize=address,fuzzer`
- Vast majority of targets already have libFuzzer targets.

Example libFuzzer Target




```
#include <stdint.h>
#include <stdlib.h>

int fuzzme(char *buf)
{
  if(strlen(buf) >= 3)
    if(buf[0] == 'b')
      if(buf[1] == 'u')
        if(buf[2] == 'g') {
          printf("You've got it!");
          abort();
        }
  return 0;
}

int LLVMFuzzerTestOneInput( char* data, size_t len )
{
  fuzzme(data);
  return 0;
}
```

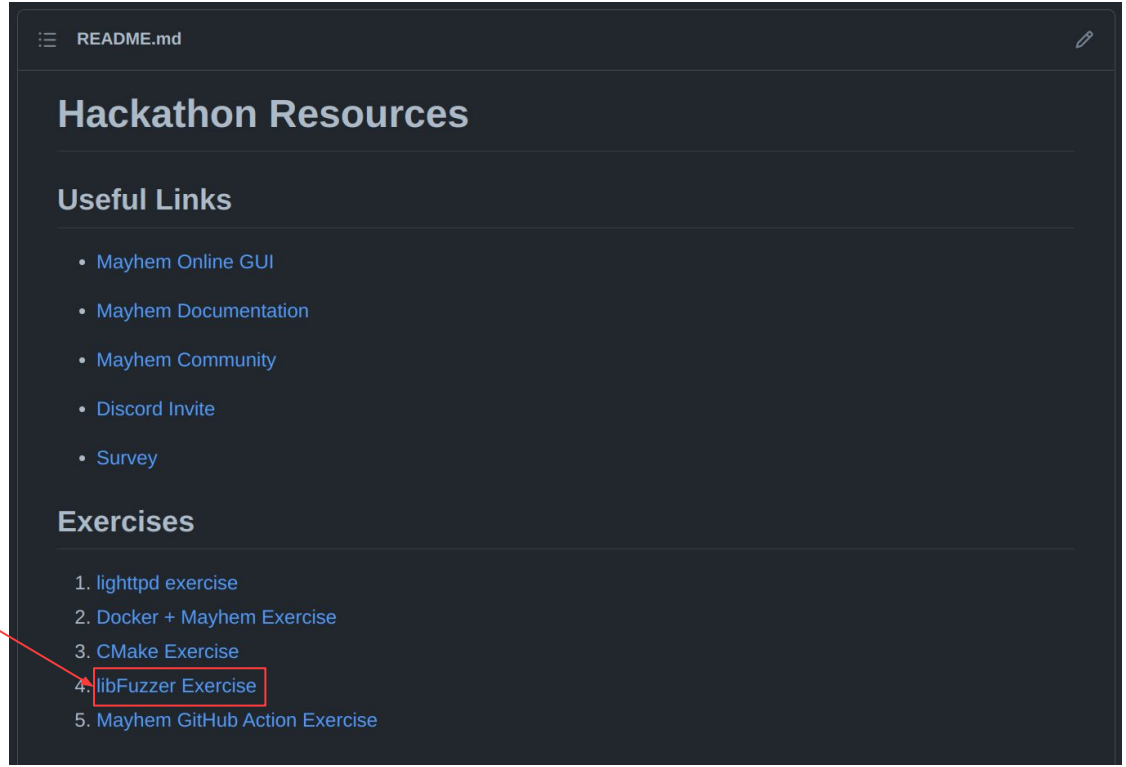
```
$> clang -fsanitize=fuzzer -c example_target.c

$> clang -fsanitize=fuzzer example_target.o -o example_target

$> ./example_target

INFO: Seed: 705835306
INFO: Loaded 1 modules   (1 inline 8-bit counters): 1 [0x4e8040, 0x4e8041),
INFO: Loaded 1 PC tables (1 PCs): 1 [0x4be930,0x4be940),
INFO: -max_len is not provided; libFuzzer will not generate inputs larger than
4096 bytes
INFO: A corpus is not provided, starting from an empty corpus
#2    INITED cov: 1 ft: 1 corp: 1/1b exec/s: 0 rss: 23Mb
#8388608    pulse  cov: 1 ft: 1 corp: 1/1b lim: 4096 exec/s: 4194304 rss: 23Mb
#16777216    pulse  cov: 1 ft: 1 corp: 1/1b lim: 4096 exec/s: 4194304 rss: 23Mb
```

libFuzzer Exercise

https://github.com/mayhemheroes/hackathon-resources

Go here

Click here



libFuzzer Review

- Created a libFuzzer target that tests a function “fuzzme”.
- Updated our CMake build system to handle a libFuzzer target.

Quiz 6

Who named fuzzing?

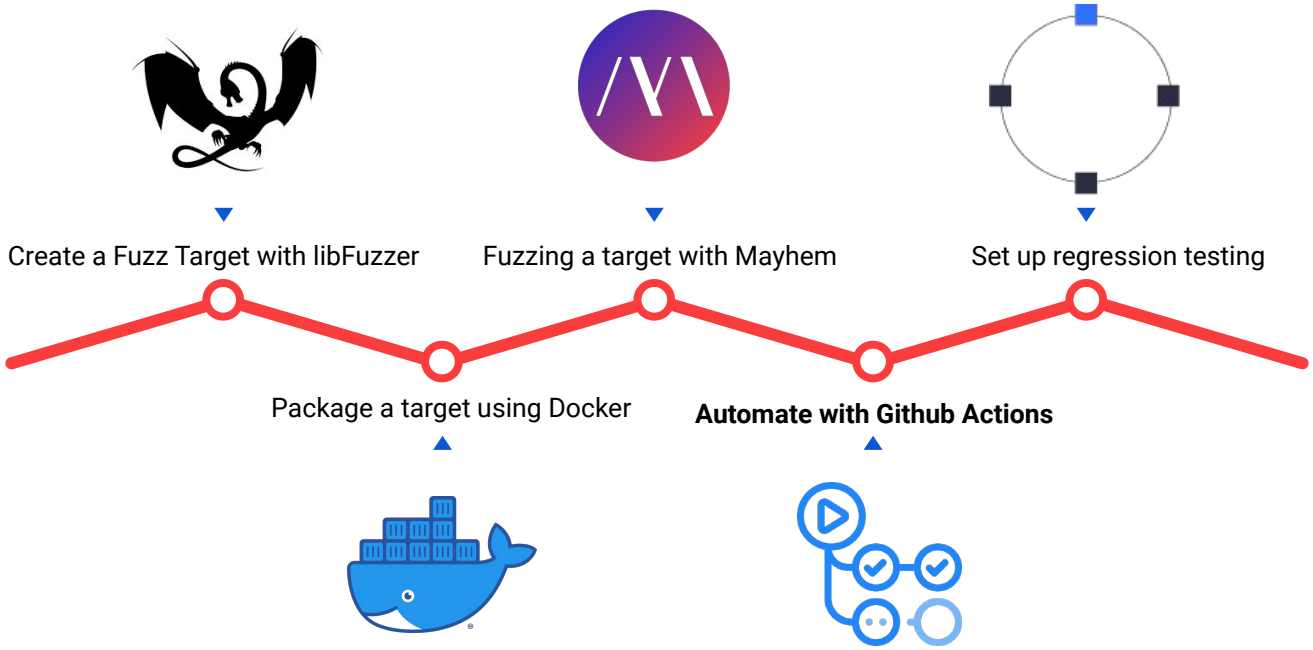
A. Barton Miller

B. David Brumley

C. Hanno Böck

D. Alexander Brewer

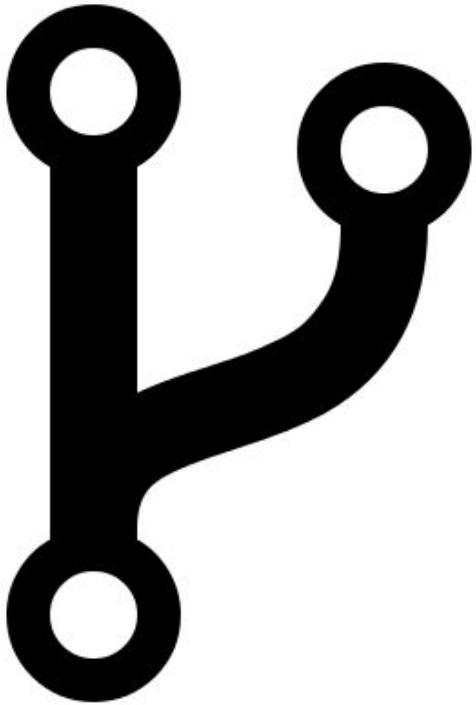
Modern CI/CD Pipeline



What is GitHub? Why does the Industry use it?

- Hosts Git repositories many open source projects.
- More than just a code repository.
  - Organizations
  - Issue Tracking
  - Continuous Integration (GitHub actions).
- Our objective: scale up fuzzing across popular projects hosted on GitHub!
  - Open Source projects are part of the supply chain for everyday products and services.

Contributing to OSS on GitHub



- OSS Developers store their work-in-progress code on “fork” repositories.
- Forks are copies of repos owned by someone other than the original developer or organization.
- Once a contributor has finished their changes, they submit a pull request to the original project.
  - Once approved, the changes are incorporated back into the project.

Continuous Integration\Delivery

- Open and closed source development projects usually have a test suite to ensure the software behaves as intended.
- CI\CD usually runs the test suites automatically whenever a developer pushes to a project.
- Helps to maintain software quality and takes some of the monotony out of certain developer workflows.

Continuous Integration\Delivery

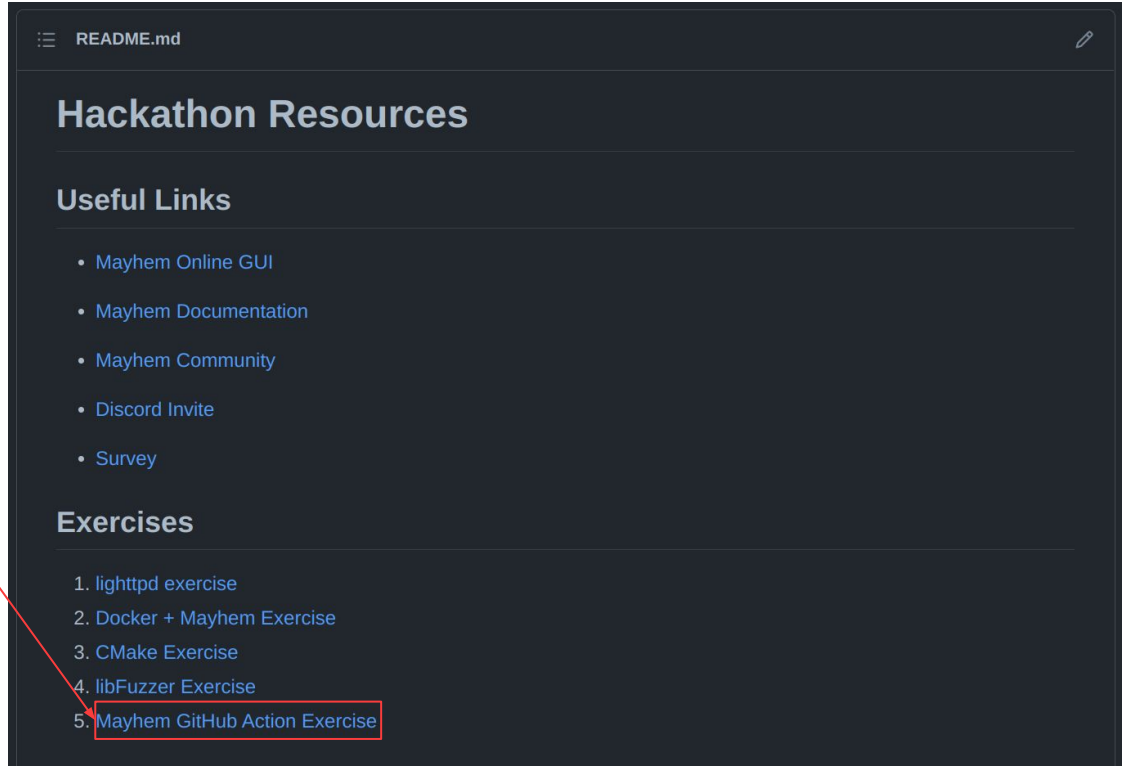
- GitHub Actions are workflows that run in response to project events (e.g., push to a repository, pull requests, etc).
- GitHub Actions can integrate with other tools, like Mayhem!
- Now, let's work through setting up CI/CD for our example CMake project.

GitHub Action Exercise

https://github.com/mayhemheroes/hackathon-resources

Go here

Click here



GitHub Actions Review

- Created a workflow that runs in response to push events.
- Automatically built a target
- Automatically started fuzzing run against target.

Quiz 7

Which of the following techniques are not used as part of Fuzz Testing

A. Binary Analysis

B. Reverse Engineering

C. Regression Testing

D. Virus Signature Scanning

Case Study: rant

Case Study: linked-list-allocator

Conclusions

Now What?

- We've worked through setting up Mayhem, CMake, OSS Fuzzers, and GitHub Actions.
- There are a lot of topics not covered here that might apply to specific targets. Google is your friend!
- If in doubt, ask in the community.

Training Survey

- Please take our training survey!
- < 5 minutes.
- https://dydbdnwi0qu.typeform.com/to/jZEKf0it
  - Also in the hackathon-resources repo under useful links (Survey 1)

Mayhem Heroes

Hackathon - Securing the World's Software

- **Goal:** Fuzz Open Source projects to help make the world's software more secure
- **Reward:** Up to $1k for each GitHub repository integrated
- **Timeline:** Submissions must be made in the next five weeks (before October 23, 2022) - paid 45 days after approval
- Legalese: Payment through 1099 via bank transfer for students eligible to work in USA

Hackathon - What Repos Qualify?

- Over **100 stars**
- **Active** (has a commit from at least 6 months ago or later)
- **Not already integrated** (in OSS-Fuzz or Mayhem)
- **Nothing inappropriate** - if you're not sure, just ask
- Legalese: ForAllSecure reserves the right to reject any submission at its sole discretion.

Hackathon - What do I get paid for?

- $100 - Integration (10 test cases)
- $200 - 100+ test cases generated
- $200 - 100+ test per second
- $200 - Function(s) harnessed
- $300 - Defect(s) discovered

Hackathon - How do I get started?

1. Select a target repository for integration.
2. Create a Fork of the Repository on GitHub
3. Get Mayhem to run with your target
4. Integrate Mayhem into your Fork
5. Submit this form (also in Hackathon resources): https://dydbdnwi0qu.typeform.com/to/YYJdU5wd
6. ForAllSecure will validate your target and eligibility. Assuming there are no issues with the target or your eligibility, you'll be asked to submit a pull request on GitHub.
    a. Once your changes have been approved in the pull request, you're eligible for payment!
    b. We may request additional changes. You can't really "fail", we want everyone to be successful and will point you in the right direction during the review process.

Hackathon - More Rules

- You can integrate - and get paid for - more than one Repo.
- If you and another student choose the same repo, the first student to submit a typeform gets paid (based on timestamp).
- You can work by yourself or in a group.
  - If working in a group, the group will split the payment.
  - When submitting as a group, group leader should inform ForAllSecure that this is a group submission.

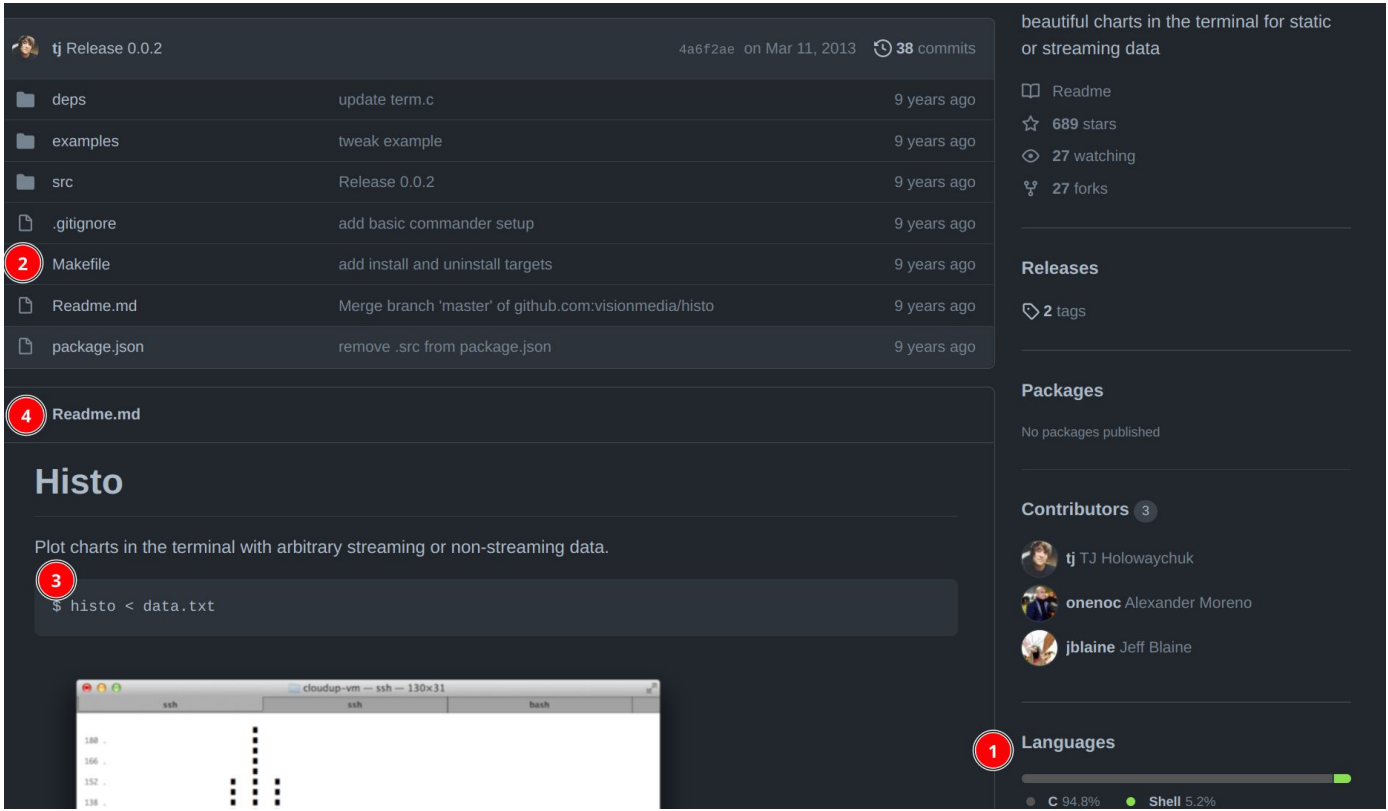
ForAllSecure

questions?

Bonus: Improving your Targets

What do I look for in a target?

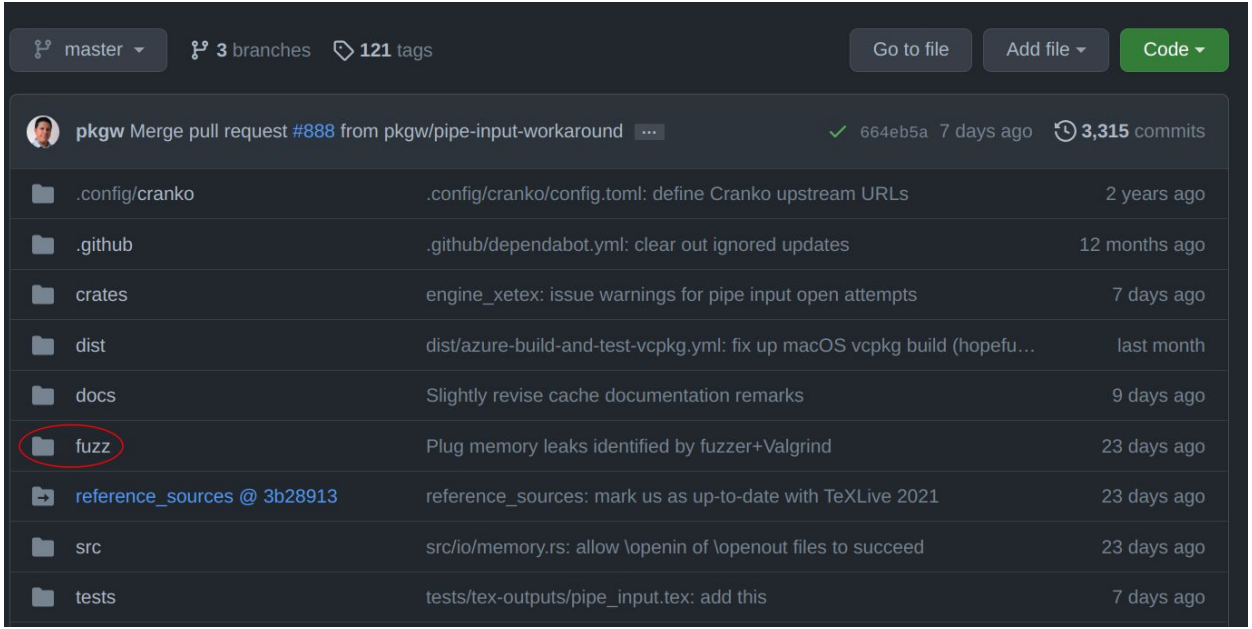
1. Language
2. Build System
3. Target Binary (if any)
4. README.md!



If it has a “fuzz/” folder…

Great! This should be easy. Integrate Mayhem with the existing fuzz target(s) underneath the fuzz/ folder

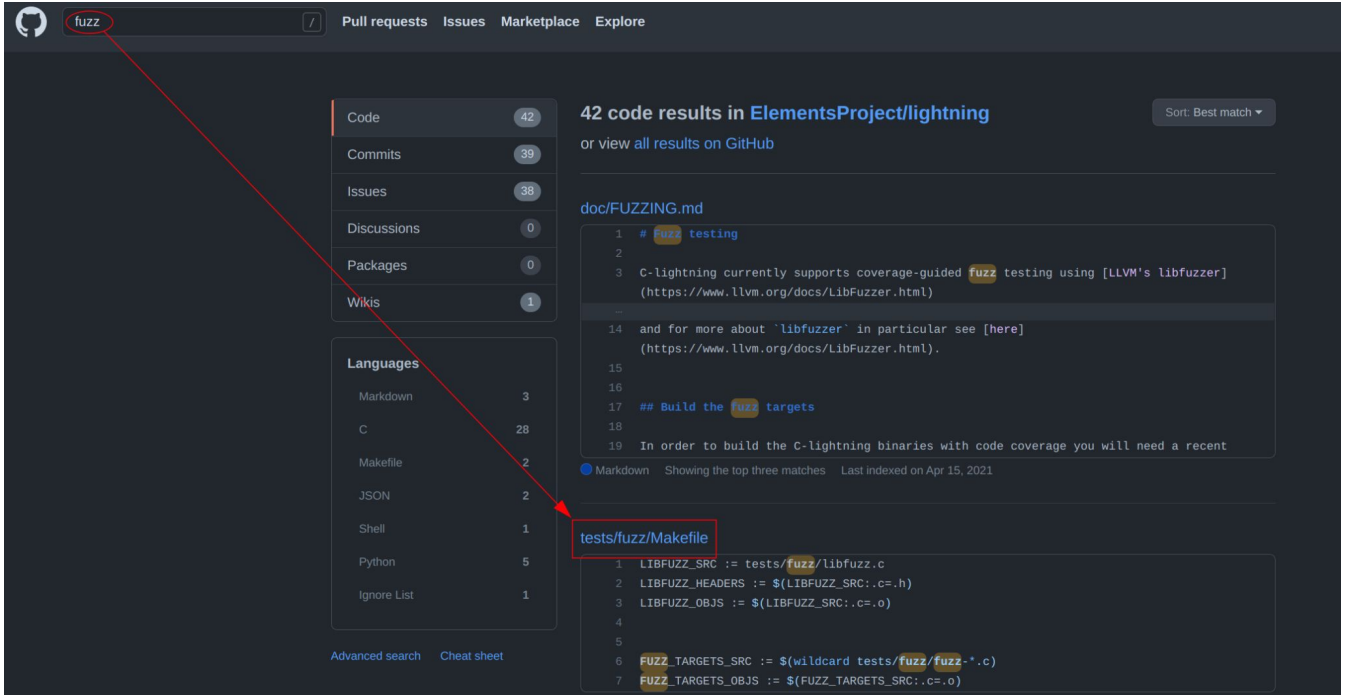
However, you won’t get credit for adding target coverage….



If it has a “fuzz/” folder…

Great! This should be easy. Integrate Mayhem with the existing fuzz target(s) underneath the fuzz/ folder

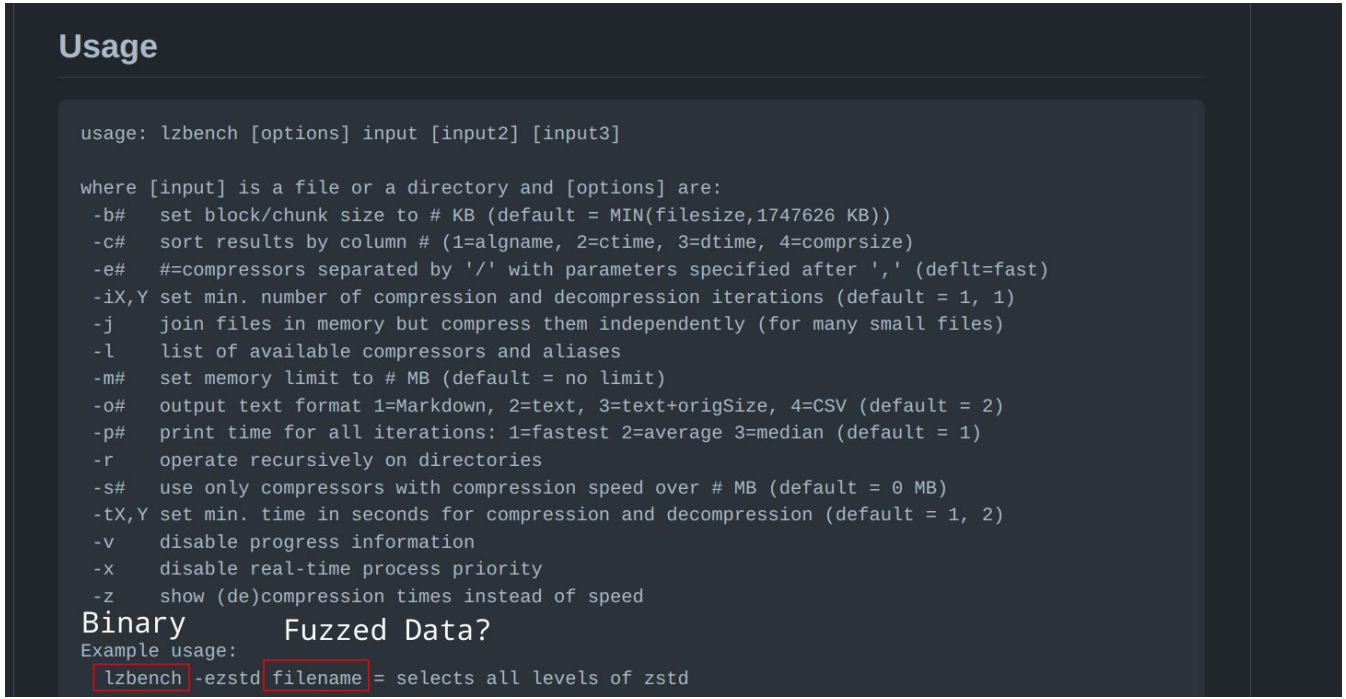
Hint: It might not be in the top level directory! Try searching for the word “fuzz”...



If it has a target binary that takes input

Via STDIN or a File

Also great! You should be able to send fuzzed data to the binary directly!



If it has a target binary that opens a port…

Via TCP/UDP

This is just like lighttpd! You should be able to send fuzzed data to the network port

Usage

Proxy

The recommended installation method is via Docker:

```
# This will expose an FF proxy on UDP port 1234 on the host and port 100 in the container
docker run --rm -it \
    -p 1234:100/udp \
    timetoogo/ff \
    --port 100 \
    -vvv
```

Or it can be installed locally from the source.

Arguments

Argument	Required	Description
`--port <port>`	Yes	The UDP port to listen for incoming requests
`--ip-address <ip>`	No	The IP address for which to accept incoming packets, defaulting to IPv4 wildcard address: *0.0.0.0*
`--ipv6-v6only`	No	When listening on IPv6 don't accept IPv4 connections
`--pre-shared-key <key>`	No	The pre-shared key used to decrypt incoming requests

If all else fails

You might need to develop the fuzz target yourself.

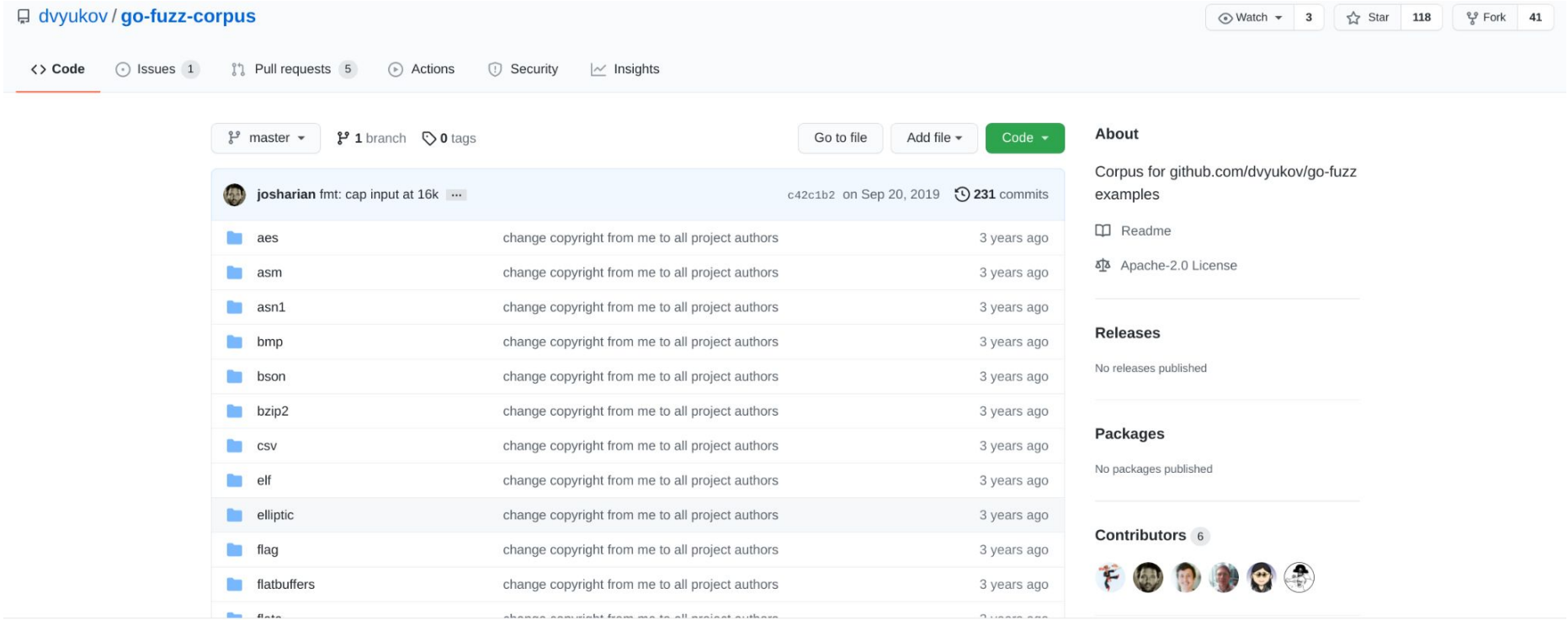
1. Look for a tests/ folder. You can usually leverage this code to create a fuzz target!
2. Read the README.md! It usually has all of the information you need to understand the repository
3. Ask for help!

Is my target being fuzzed well?

ForAllSecure

Adding a Seed Corpus

I can habeus corpus?



dvyukov / **go-fuzz-corpus**

Watch 3 | Star 118 | Fork 41

Code | Issues 1 | Pull requests 5 | Actions | Security | Insights

master | 1 branch | 0 tags | Go to file | Add file | Code

josharian fmt: cap input at 16k ··· c42c1b2 on Sep 20, 2019 **231** commits

aes	change copyright from me to all project authors	3 years ago
asm	change copyright from me to all project authors	3 years ago
asn1	change copyright from me to all project authors	3 years ago
bmp	change copyright from me to all project authors	3 years ago
bson	change copyright from me to all project authors	3 years ago
bzip2	change copyright from me to all project authors	3 years ago
csv	change copyright from me to all project authors	3 years ago
elf	change copyright from me to all project authors	3 years ago
elliptic	change copyright from me to all project authors	3 years ago
flag	change copyright from me to all project authors	3 years ago
flatbuffers	change copyright from me to all project authors	3 years ago

About

Corpus for github.com/dvyukov/go-fuzz examples

Readme

Apache-2.0 License

Releases

No releases published

Packages

No packages published

Contributors 6

Does it help?

0

Crashing Test Cases

756

Corpus Size

0

Crashing Test Cases

4,986

Corpus Size

Improve Target Speed

Bad

```
// Entry point for LibFuzzer.
extern "C" int LLVMFuzzerTestOneInput(const uint8_t* data,
size_t size) {

startServer();

 uLongf buffer_length = static_cast<uLongf>(sizeof(buffer));
 if (Z_OK != uncompress(buffer, &buffer_length, data,
                        static_cast<uLong>(size))) {
   std::cerr << "Error!" << endl;
 }
 return 0;
}
```

initialization on every run

unnecessary I/O

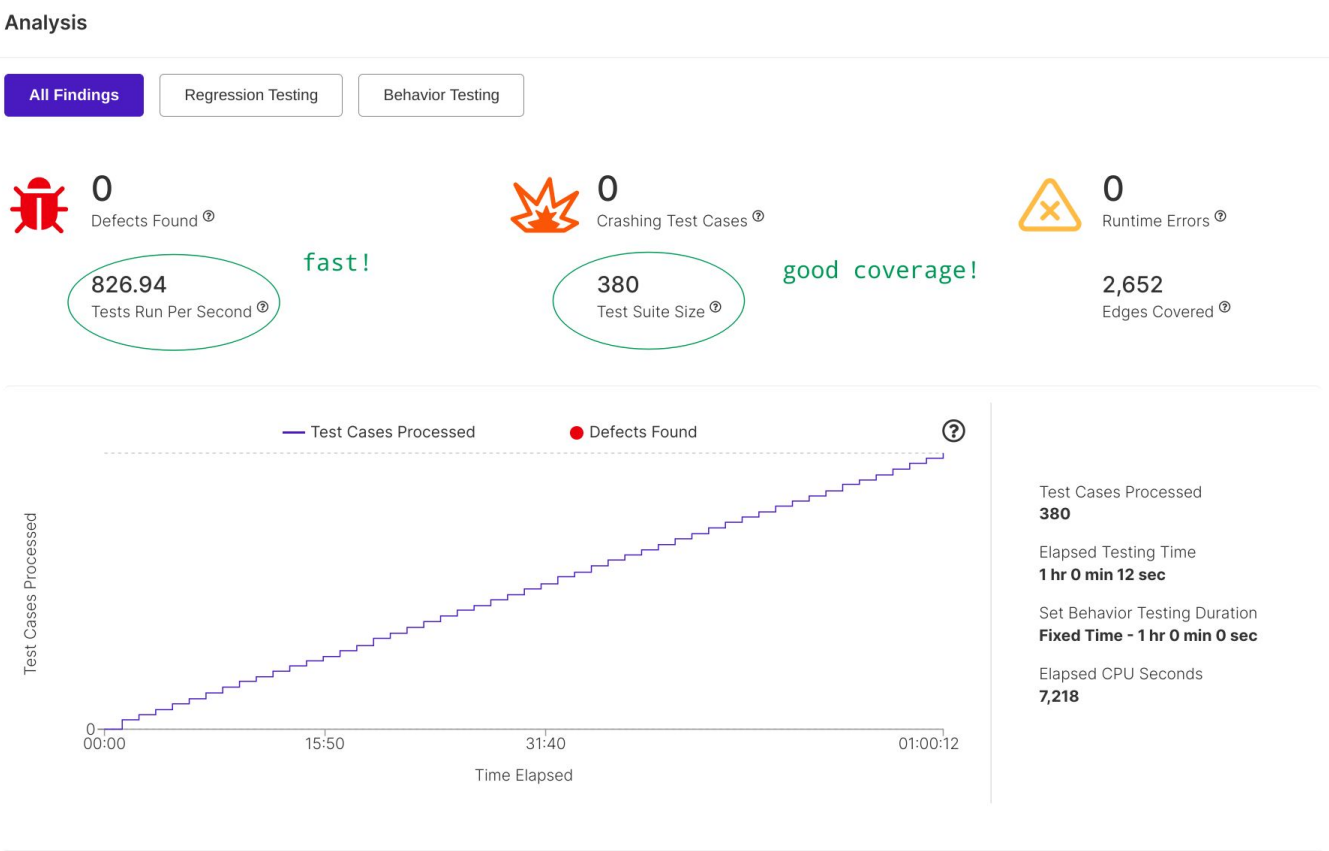
Good

```
// Entry point for LibFuzzer.
extern "C" int LLVMFuzzerTestOneInput(const uint8_t* data,
size_t size) {

static bool running = startServer();

if(running) {
 uLongf buffer_length = static_cast<uLongf>(sizeof(buffer));
 if (Z_OK != uncompress(buffer, &buffer_length, data,
                        static_cast<uLong>(size))) {
   return 0; //ignore
 }
 return 0;
 }
}
```

one-time init

no unnecessary I/O

That looks better!



ForAllSecure

ForAllSecure