ZIGRIN SECURITY

Don't Leave Your Web Apps Vulnerable

Build a Fuzzing Framework with IAST

Agenda

ZIGRIN SECURITY

- Introduction
- Typical vulnerability scanners
- Fuzzing framework goals
- Approach & Architecture
- Our results
- Benefits & Challenges
- Final words

Introduction

- CEO and Cybersecurity Expert in Zigrin Security
- 12 years of cybersecurity experience
- Industries
  - SaaS
  - Military
  - Healthcare
  - Banking & Insurance
  - E-commerce
  - You can read about some of them here:

    www.zigrin.com/advisories

Company

ZIGRIN SECURITY

We are a team of **cybersecurity perfectionists** and **experts** who offer you specialized knowledge and years of experience in software and hardware security testing.

You can read about how we help our customers get more secure: www.zigrin.com/casestudy

Project background

ZIGRIN SECURITY

Research project inspired by

- Automatic Detection of Vulnerabilities in Web Applications using Fuzzing by Miguel Filipe Beatriz – https://fenix.tecnico.ulisboa.pt/downloadFile/563345090413029/ExtendedAbstract-MEICA-67039-MiguelBeatriz.pdf
- WPGarlic by Krzysztof Zając – https://github.com/kazet/wpgarlic

Commissioned by

LUXEMBOURG ARMED FORCES
CIS

Fuzzing framework goals

ZIGRIN SECURITY

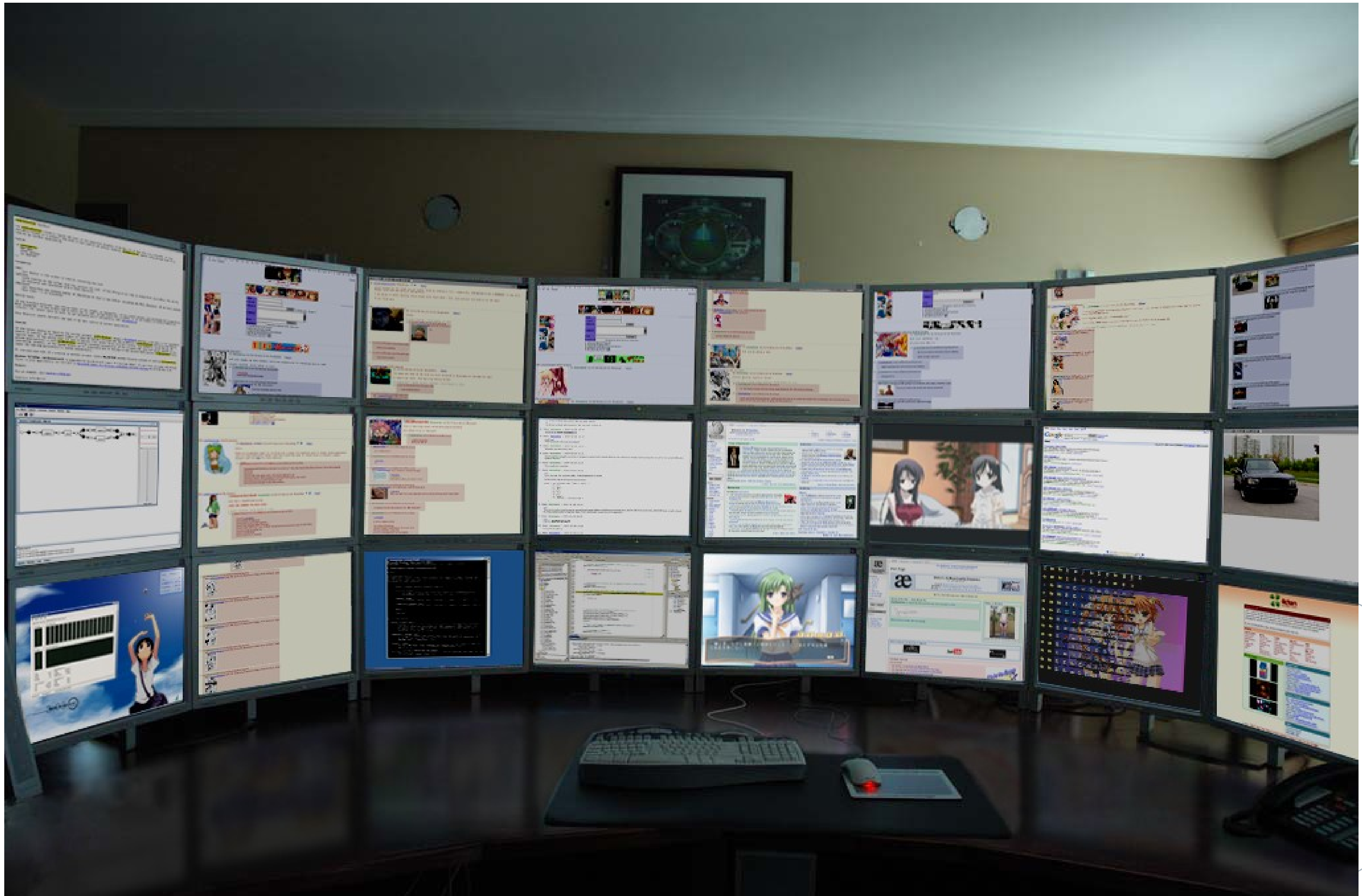
Approaches to discovering web vulnerabilities

- DAST – Dynamic Application Security Testing
- SAST – Static Application Security Testing
- **IAST – Interactive Application Security Testing**

Initial goals

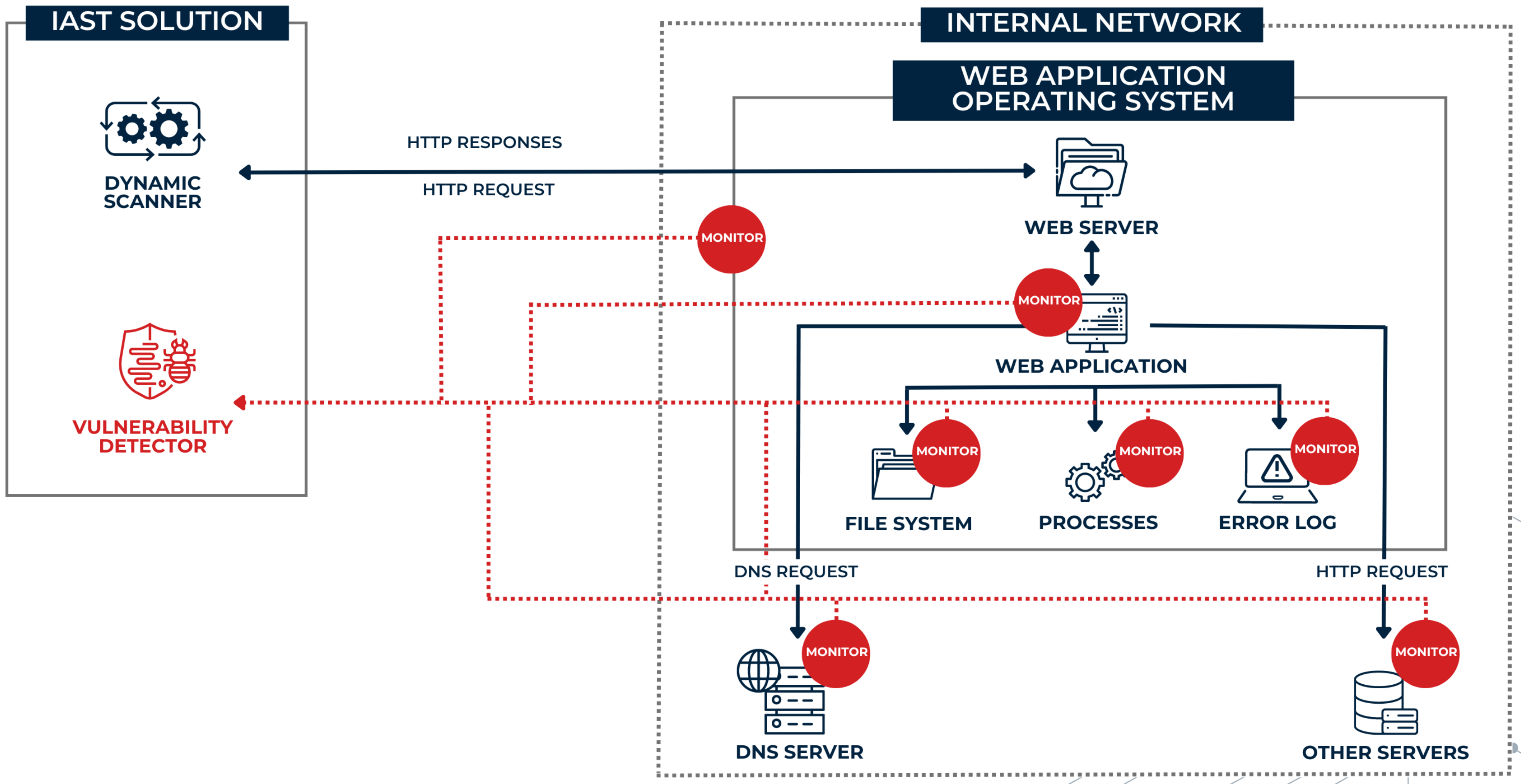
- Automated vulnerability discovery
- No pre-configuration required
- Results with minimal false-positive rate
- Minimal security knowledge required

Idea – Monitor everything



IAST Architecture

ZIGRIN SECURITY

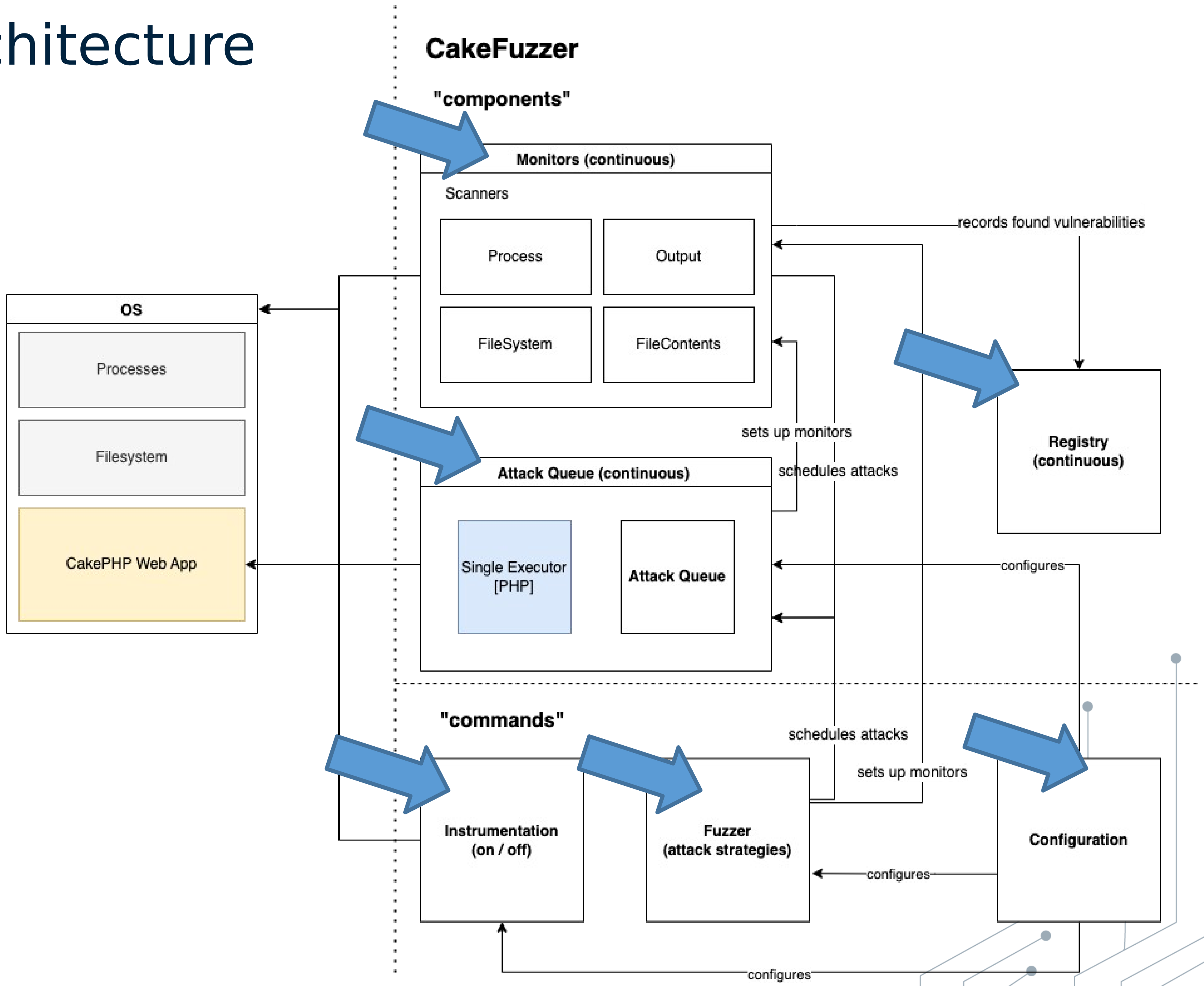


Approach - Focus

- CakePHP-based web applications
- CakePHP internals: routes, controllers, actions
- Faster because we don’t scan over HTTP

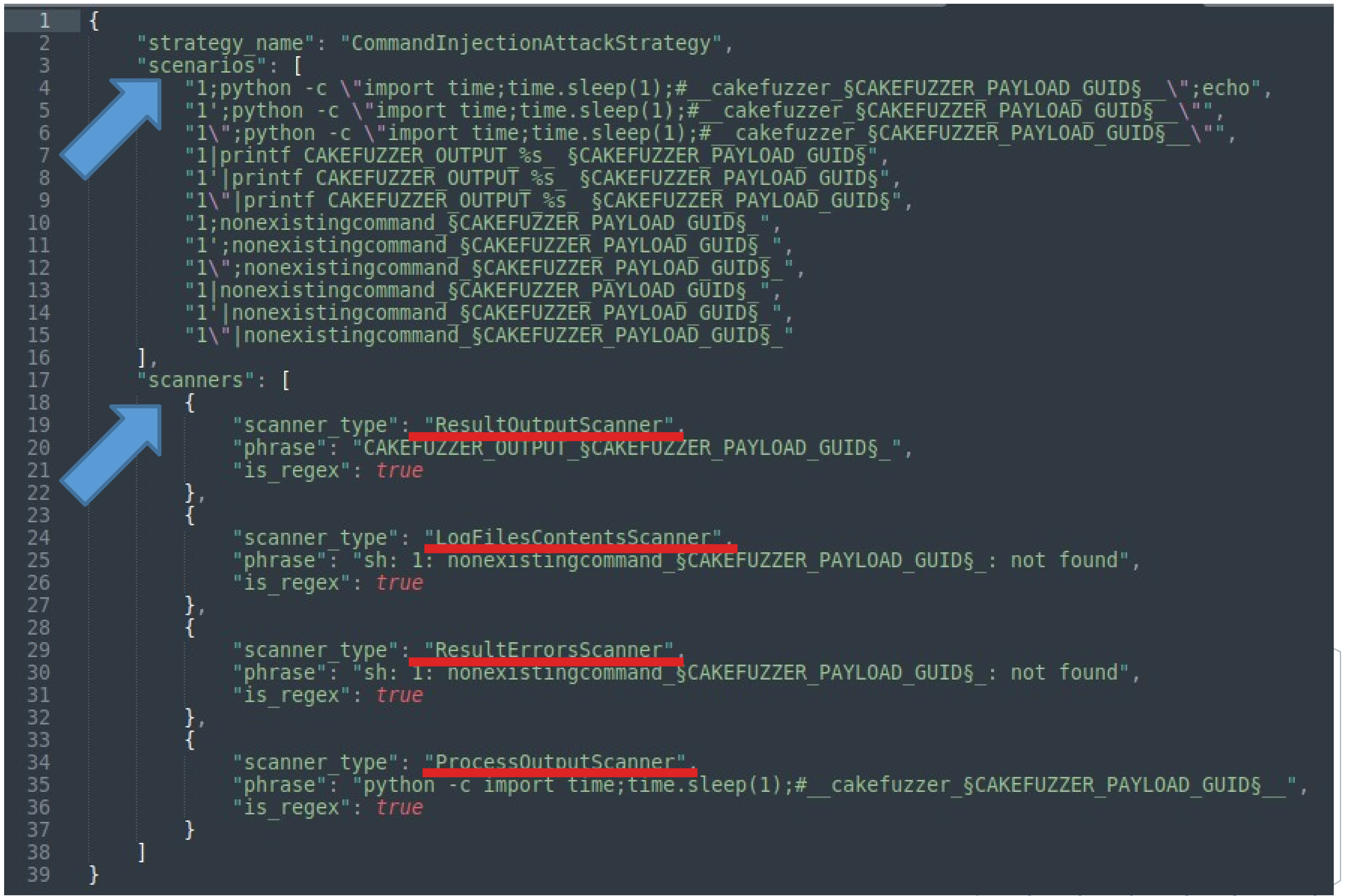
CakeFuzzer

Architecture



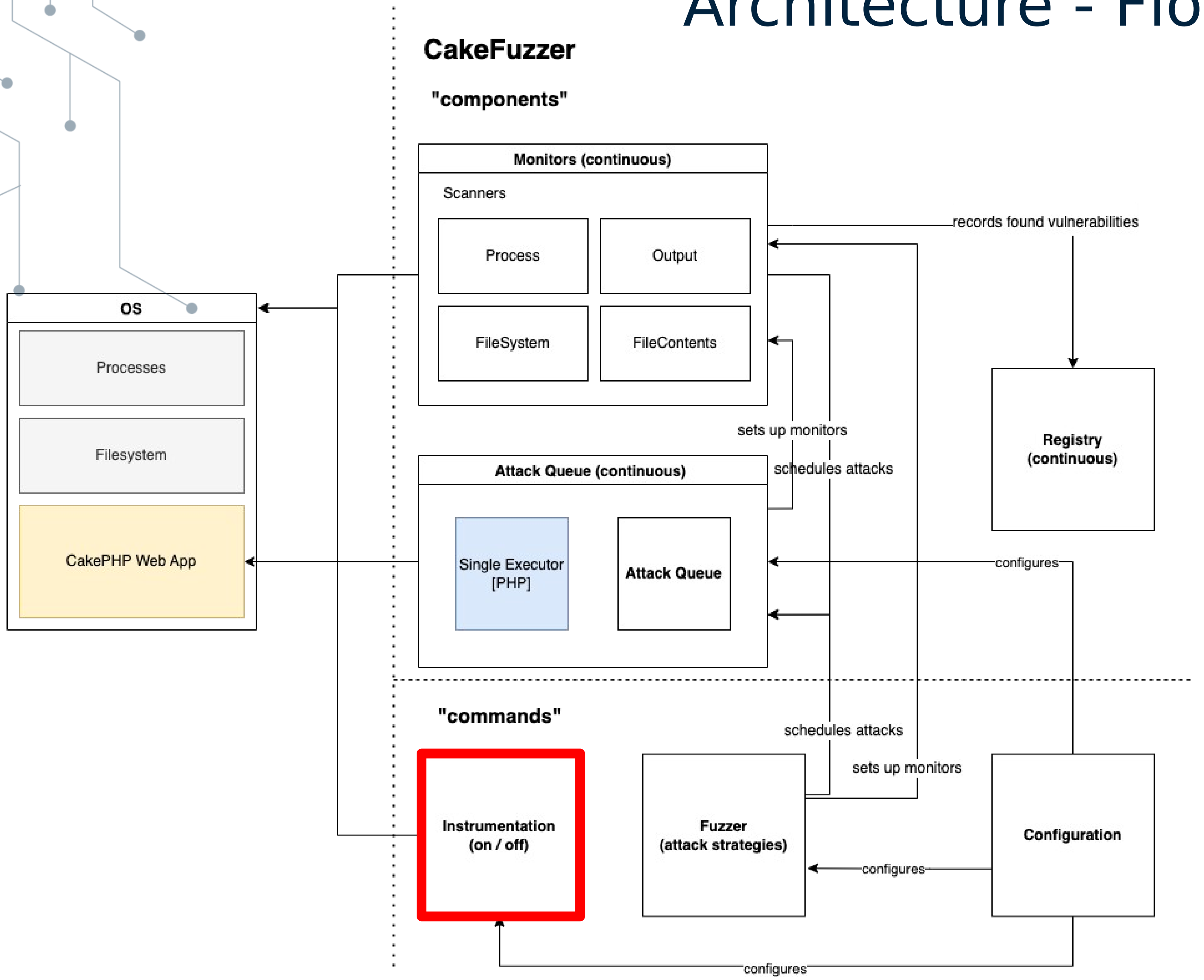
ZIGRIN SECURITY

Architecture - Strategy



```
{
    "strategy_name": "CommandInjectionAttackStrategy",
    "scenarios": [
        "1;python -c \"import time;time.sleep(1);#__cakefuzzer_§CAKEFUZZER_PAYLOAD_GUID§__\";echo",
        "1';python -c \"import time;time.sleep(1);#__cakefuzzer_§CAKEFUZZER_PAYLOAD_GUID§__\"",
        "1\";python -c \"import time;time.sleep(1);#__cakefuzzer_§CAKEFUZZER_PAYLOAD_GUID§__\"",
        "1|printf CAKEFUZZER_OUTPUT_%s _§CAKEFUZZER_PAYLOAD_GUID§",
        "1'|printf CAKEFUZZER_OUTPUT_%s _§CAKEFUZZER_PAYLOAD_GUID§",
        "1\"|printf CAKEFUZZER_OUTPUT_%s _§CAKEFUZZER_PAYLOAD_GUID§",
        "1;nonexistingcommand_§CAKEFUZZER_PAYLOAD_GUID§_",
        "1';nonexistingcommand_§CAKEFUZZER_PAYLOAD_GUID§_",
        "1\";nonexistingcommand_§CAKEFUZZER_PAYLOAD_GUID§_",
        "1|nonexistingcommand_§CAKEFUZZER_PAYLOAD_GUID§_",
        "1'|nonexistingcommand_§CAKEFUZZER_PAYLOAD_GUID§_",
        "1\"|nonexistingcommand_§CAKEFUZZER_PAYLOAD_GUID§_"
    ],
    "scanners": [
        {
            "scanner_type": "ResultOutputScanner",
            "phrase": "CAKEFUZZER_OUTPUT_§CAKEFUZZER_PAYLOAD_GUID§_",
            "is_regex": true
        },
        {
            "scanner_type": "LogFilesContentsScanner",
            "phrase": "sh: 1: nonexistingcommand_§CAKEFUZZER_PAYLOAD_GUID§_: not found",
            "is_regex": true
        },
        {
            "scanner_type": "ResultErrorsScanner",
            "phrase": "sh: 1: nonexistingcommand_§CAKEFUZZER_PAYLOAD_GUID§_: not found",
            "is_regex": true
        },
        {
            "scanner_type": "ProcessOutputScanner",
            "phrase": "python -c import time;time.sleep(1);#__cakefuzzer_§CAKEFUZZER_PAYLOAD_GUID§__",
            "is_regex": true
        }
    ]
}
```

Architecture - Flow



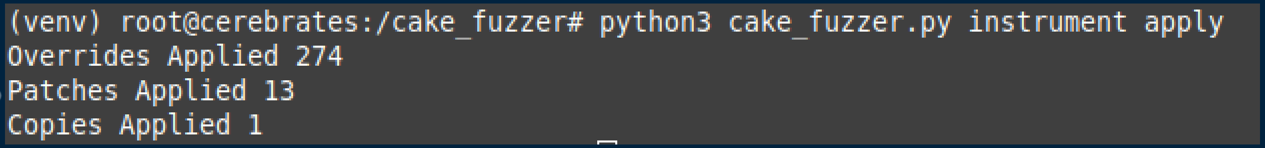
Instrumentation

- Preparing the application to launch attacks
- Disabling framework security checks
- Overwriting parts of the web app’s code

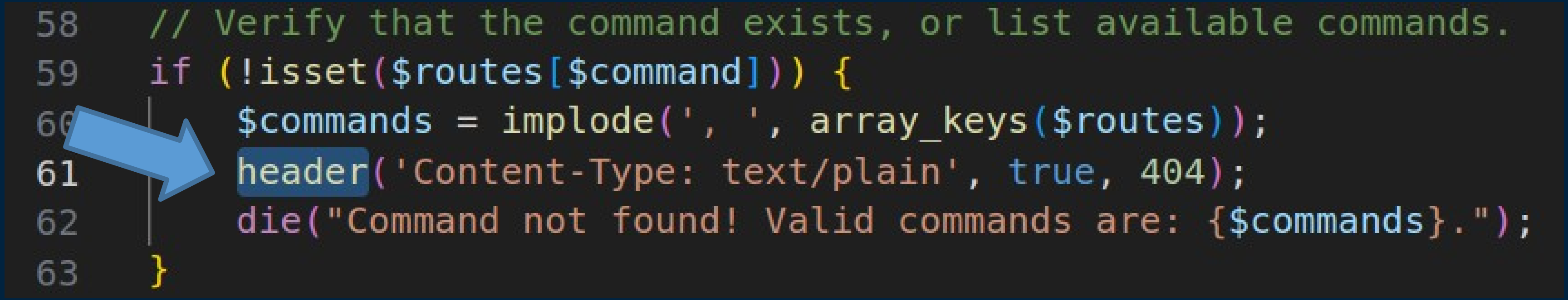
Architecture - Flow

ZIGRIN SECURITY

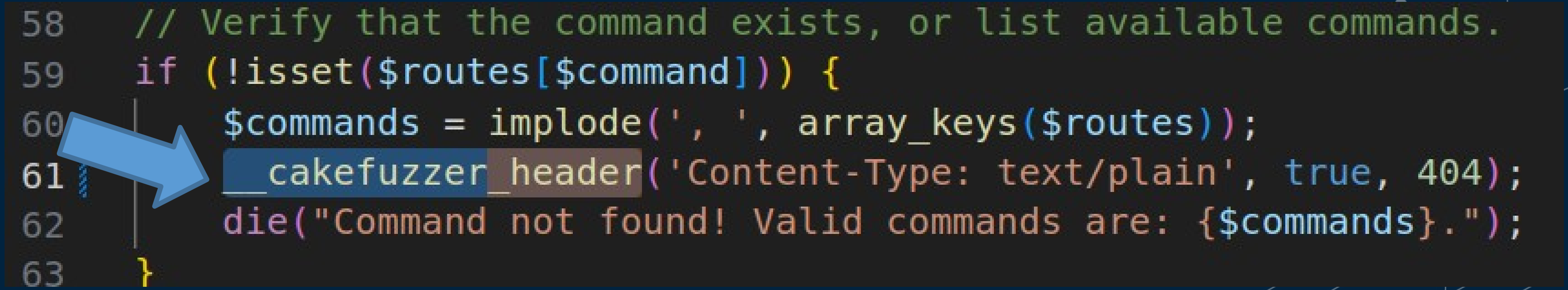
Instrumentation



```
(venv) root@cerebrates:/cake_fuzzer# python3 cake_fuzzer.py instrument apply
Overrides Applied 274
Patches Applied 13
Copies Applied 1
```

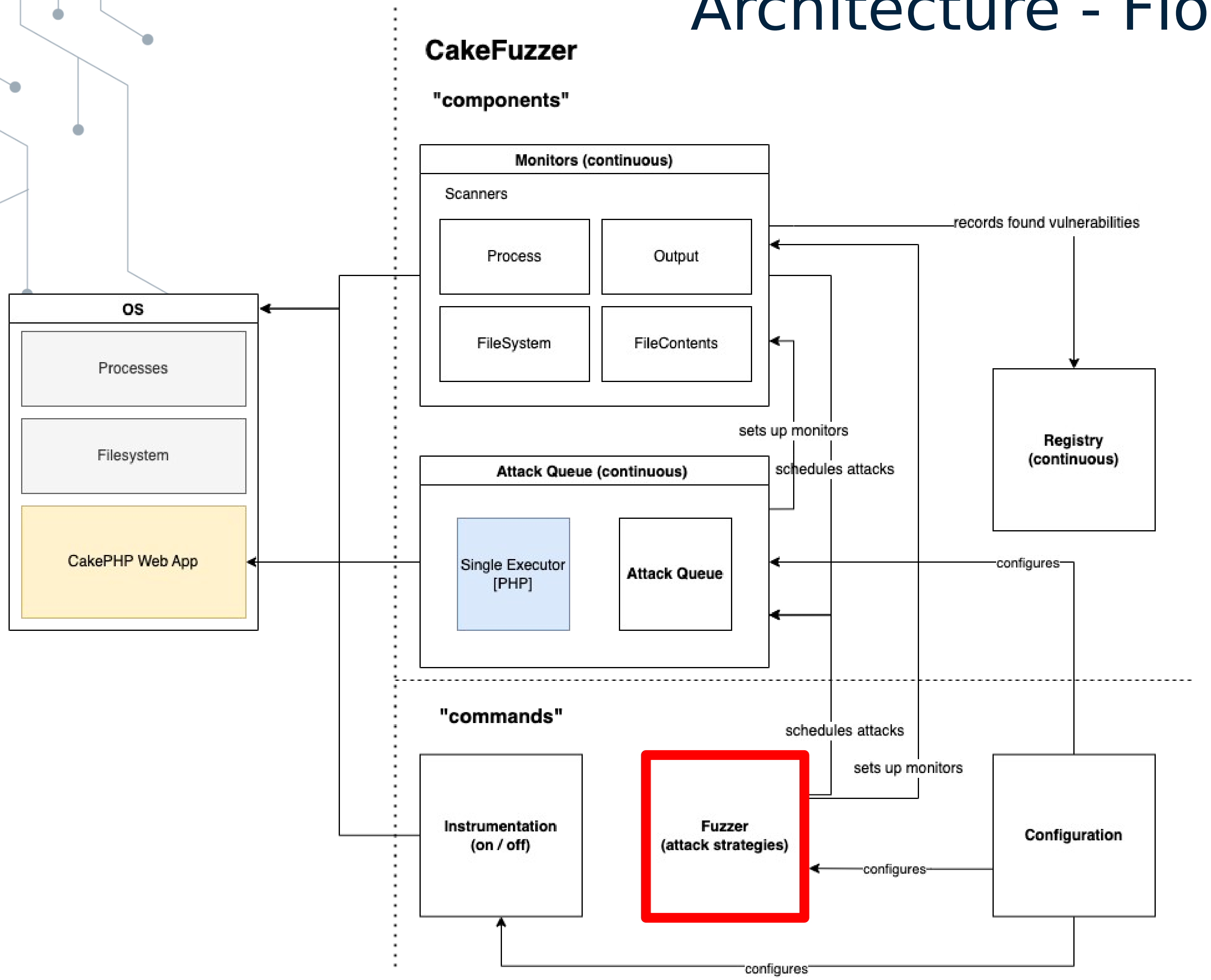


```
// Verify that the command exists, or list available commands.
if (!isset($routes[$command])) {
    $commands = implode(', ', array_keys($routes));
    header('Content-Type: text/plain', true, 404);
    die("Command not found! Valid commands are: {$commands}.");
}
```



```
// Verify that the command exists, or list available commands.
if (!isset($routes[$command])) {
    $commands = implode(', ', array_keys($routes));
    __cakefuzzer_header('Content-Type: text/plain', true, 404);
    die("Command not found! Valid commands are: {$commands}.");
}
```

Architecture - Flow

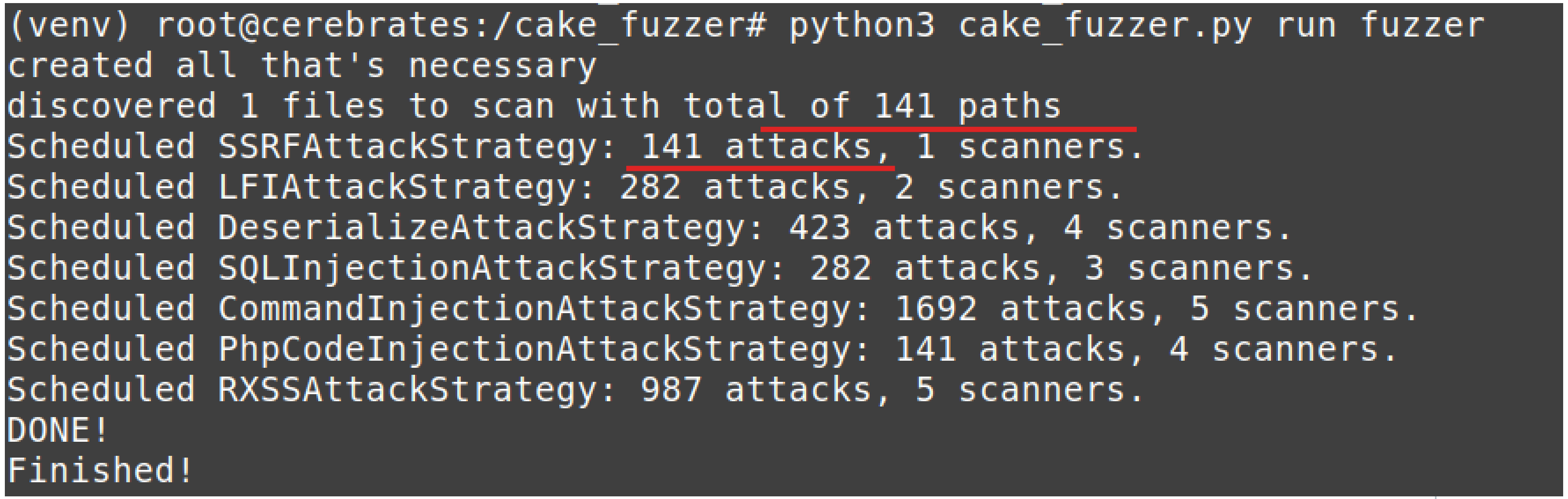


ZIGRIN SECURITY

Fuzzer – Attack scheduling

- Extracting info about the app
- Scheduling attacks
- Setting up scanners

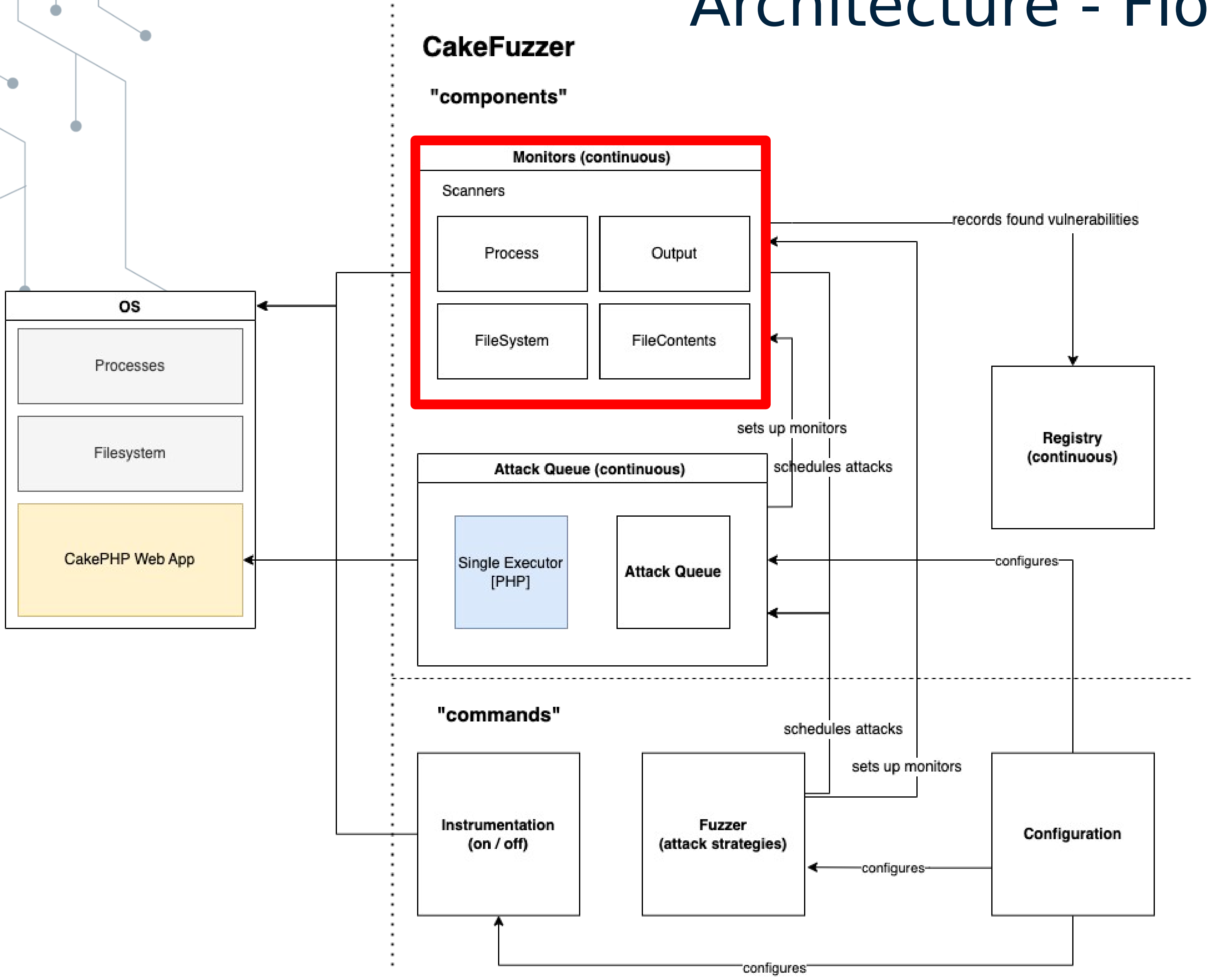
Architecture - Flow



```
(venv) root@cerebrates:/cake_fuzzer# python3 cake_fuzzer.py run fuzzer
created all that's necessary
discovered 1 files to scan with total of 141 paths
Scheduled SSRFAttackStrategy: 141 attacks, 1 scanners.
Scheduled LFIAttackStrategy: 282 attacks, 2 scanners.
Scheduled DeserializeAttackStrategy: 423 attacks, 4 scanners.
Scheduled SQLInjectionAttackStrategy: 282 attacks, 3 scanners.
Scheduled CommandInjectionAttackStrategy: 1692 attacks, 5 scanners.
Scheduled PhpCodeInjectionAttackStrategy: 141 attacks, 4 scanners.
Scheduled RXSSAttackStrategy: 987 attacks, 5 scanners.
DONE!
Finished!
```

Architecture - Flow

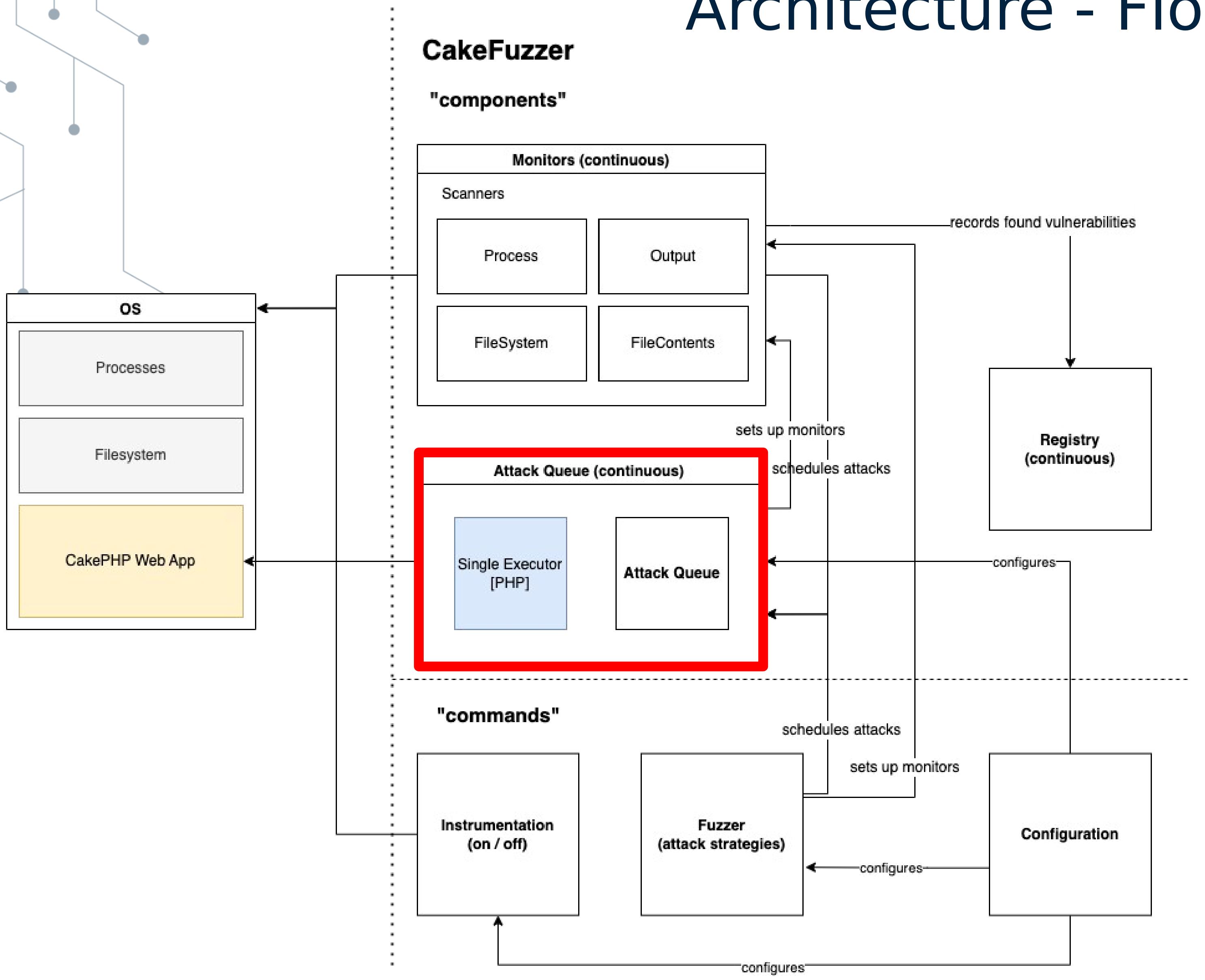
ZIGRIN SECURITY



Monitors

- Application response
- Standard error
- File system
- Error logs
- Operating system processes
- DNS connections

Architecture - Flow

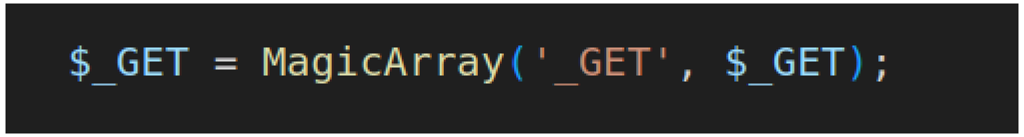


Attacking

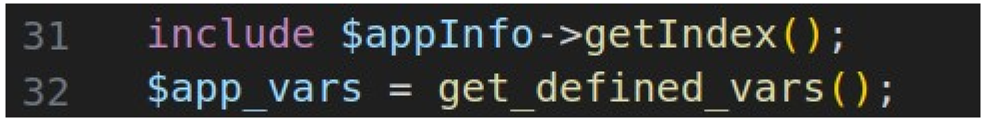
- Attacking every detected path
- Using all defined strategies
- Monitoring abnormal behaviors

Architecture - Flow

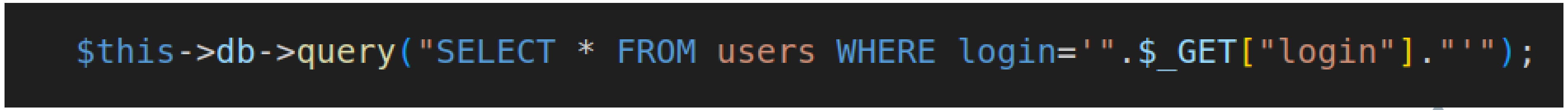
Attacking



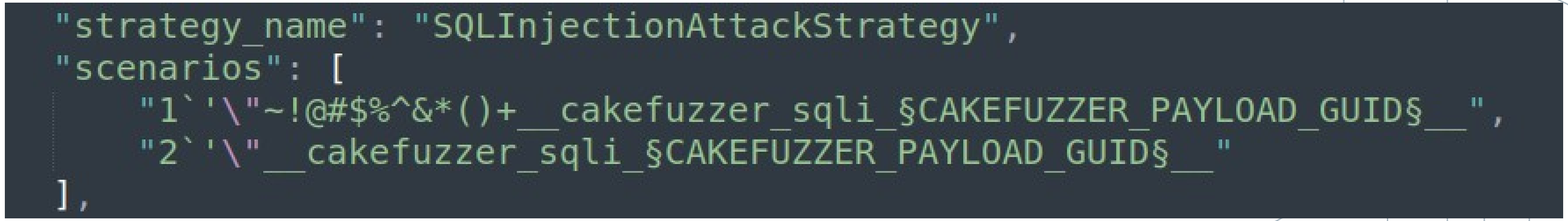
```php
$_GET = MagicArray('_GET', $_GET);
```



```php
include $appInfo->getIndex();
$app_vars = get_defined_vars();
```

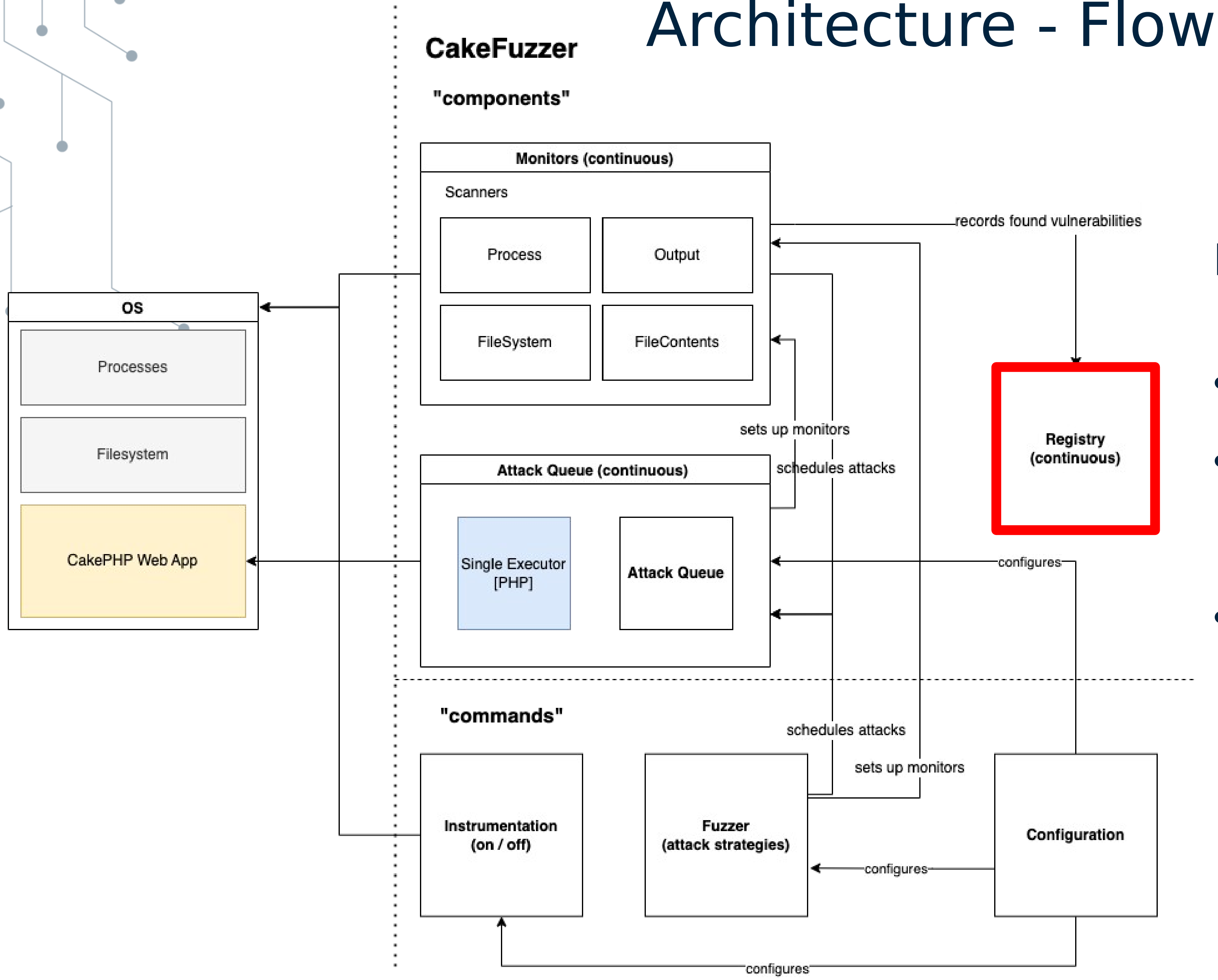


```php
$this->db->query("SELECT * FROM users WHERE login='".$_GET["login"]."'");
```



```json
"strategy_name": "SQLInjectionAttackStrategy",
"scenarios": [
    "1`'\"~!@#$%^&*()+__cakefuzzer_sqli_§CAKEFUZZER_PAYLOAD_GUID§__",
    "2`'\"__cakefuzzer_sqli_§CAKEFUZZER_PAYLOAD_GUID§__"
],
```

Architecture - Flow

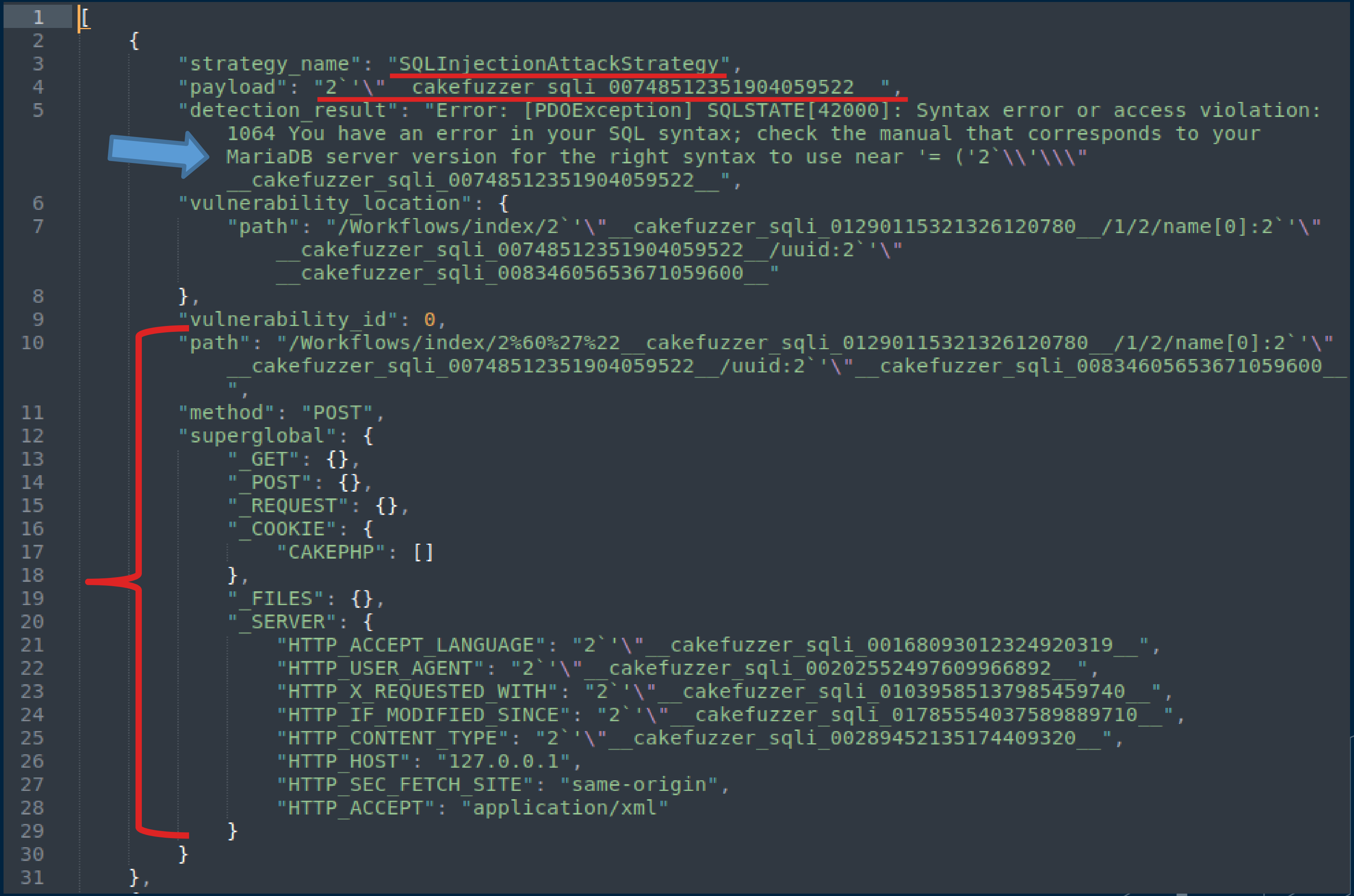


Extracting results

- Results saved in the registry
- Extracting detected vulnerabilities
- Extracting application responses

Results

ZIGRIN SECURITY



```
[
    {
        "strategy_name": "SQLInjectionAttackStrategy",
        "payload": "2`'\"__cakefuzzer_sqli_0074851235190405922__",
        "detection_result": "Error: [PDOException] SQLSTATE[42000]: Syntax error or access violation:
            1064 You have an error in your SQL syntax; check the manual that corresponds to your
            MariaDB server version for the right syntax to use near '= ('2`\\'\\\"
            __cakefuzzer_sqli_0074851235190405922__",
        "vulnerability_location": {
            "path": "/Workflows/index/2`'\"__cakefuzzer_sqli_0129011532132612078__/1/2/name[0]:2`'\"
                __cakefuzzer_sqli_0074851235190405922__/uuid:2`'\"
                __cakefuzzer_sqli_0083460565367105960__"
        },
        "vulnerability_id": 0,
        "path": "/Workflows/index/2%60%27%22__cakefuzzer_sqli_0129011532132612078__/1/2/name[0]:2`'\"
            __cakefuzzer_sqli_0074851235190405922__/uuid:2`'\"__cakefuzzer_sqli_0083460565367105960__
            ",
        "method": "POST",
        "superglobal": {
            "_GET": {},
            "_POST": {},
            "_REQUEST": {},
            "_COOKIE": {
                "CAKEPHP": []
            },
            "_FILES": {},
            "_SERVER": {
                "HTTP_ACCEPT_LANGUAGE": "2`'\"__cakefuzzer_sqli_0016809301232492031__",
                "HTTP_USER_AGENT": "2`'\"__cakefuzzer_sqli_0020255249760996689__",
                "HTTP_X_REQUESTED_WITH": "2`'\"__cakefuzzer_sqli_0103958513798545974__",
                "HTTP_IF_MODIFIED_SINCE": "2`'\"__cakefuzzer_sqli_0178555403758988971__",
                "HTTP_CONTENT_TYPE": "2`'\"__cakefuzzer_sqli_0028945213517440932__",
                "HTTP_HOST": "127.0.0.1",
                "HTTP_SEC_FETCH_SITE": "same-origin",
                "HTTP_ACCEPT": "application/xml"
            }
        }
    },
```

Results

ZIGRIN SECURITY

Results

ZIGRIN SECURITY

DETECTED KNOWN VULNERABILITIES

REFLECTED XSS:

1 CVE-2022-29533
6.1 MEDIUM

2 CVE-2021-3184
6.1 MEDIUM

3 CVE-2020-8893
6.1 MEDIUM

4 CVE-2019-10254
6.1 MEDIUM

DETECTED 0-DAY VULNERABILITIES

1 CVE-2023-28884 6.1 MEDIUM
XSS IN URL PARAM

2 CVE-2023-28883 9.8 CRITICAL
BLIND SQL INJECTION

3 CVE-2023-24070 3.0 LOW
XSS IN REFERER

4 CVE-2022-47928 6.1 MEDIUM
XSS IN UPLOADFILE

5 CVE-2022-48328 8.8 HIGH
SQLI IN CRUD

Results

ZIGRIN SECURITY

- XSS in uploadFile
- SQL injection in CRUD
- XSS in Referer header
- Blind SQL Injection
- Dom-Based XSS
- Blind SQL injection in order parameter
- Blind SQL injection in array input parameters
- Time-based SQL injection in /Logs/index
- Reflected Cross-Site Scripting in Galaxies

CVE-2023-48657 CVE-2022-48328

CVE-2023-48659 CVE-2023-48655

CVE-2023-48658 CVE-2023-24070

CVE-2023-28884 CVE-2023-48656

CVE-2023-28883 CVE-2022-47928

Benefits

ZIGRIN SECURITY

- Extendable fuzzing framework
- Bottom-up approach
- Discovering obscure parameters
- Manual tests possible
- Simple strategy definition

Challanges

- **Static patching**
- **Testing new features requires a lot of features**
- **Detecting duplicated vulnerabilities**
- **Discovering vulnerabilities that have multiple requirements to be triggered**

ZIGRIN SECURITY

Follow and Contribute

- Star the Cake Fuzzer project on Github! It’s open sourced!
- Follow Zigrin Security on LinkedIn

linkedin.com/company/zigrin-security

www.zigrin.com