# The SAST & The Furious

## Introducing Security Scanners to your pipelines

Zack Lott

## Who am I?

### I’m Zack

- Engineering Manager at Chip *(FinTech Company)*
- Runs the Security working group within the business
- Software developer for 10 years
- Programming in PHP for 9 years
- Originally worked with TYPO3 back in 2016 on version 6 and 7
- Recently picked up programming in Go
- Welsh and not English

## What is Application Security?

## Application Security is:

- The process of developing, adding or testing Security features within your websites/web applications.
- Preventing users from being able to gaining unauthorised access and modifying or deleting data.

**Well that is a bit vague?**

https://owasp.org/www-project-top-ten/

# OWASP Top 10

- Broken Access Control
- Cryptographic Failures
- Injection
- Insecure Design
- Security Misconfigurations
- Vulnerable and Outdated Components
- Identification and Authentication Failures
- Security Logging and Monitoring Failures
- Server-Side Request Forgery

OWASP

## That’s quite a lot?

### How do people normally find these?

- Code reviews
- Previous experience

- Create their own tools
- Use of Security Tools

**But what tooling is available?**

## Static Application Security Testing (*SAST*)

## What are SAST Scanners?

### How do they benefit us

SAST scanners are tools that allow us to statically analyse your applications for vulnerabilities and misconfigurations.

*Think of PHPStan but focus purely on security concerns*

- Allows you to run tests during the development life cycle
- Doesn’t require any infrastructure due to it scanning the code
- Lean on the shoulders of the communities to identify issues

## Different types of scanners

aqua trivy

Acunetix

snyk

SYNOPSYS®

sonarqube

## What we will be using!

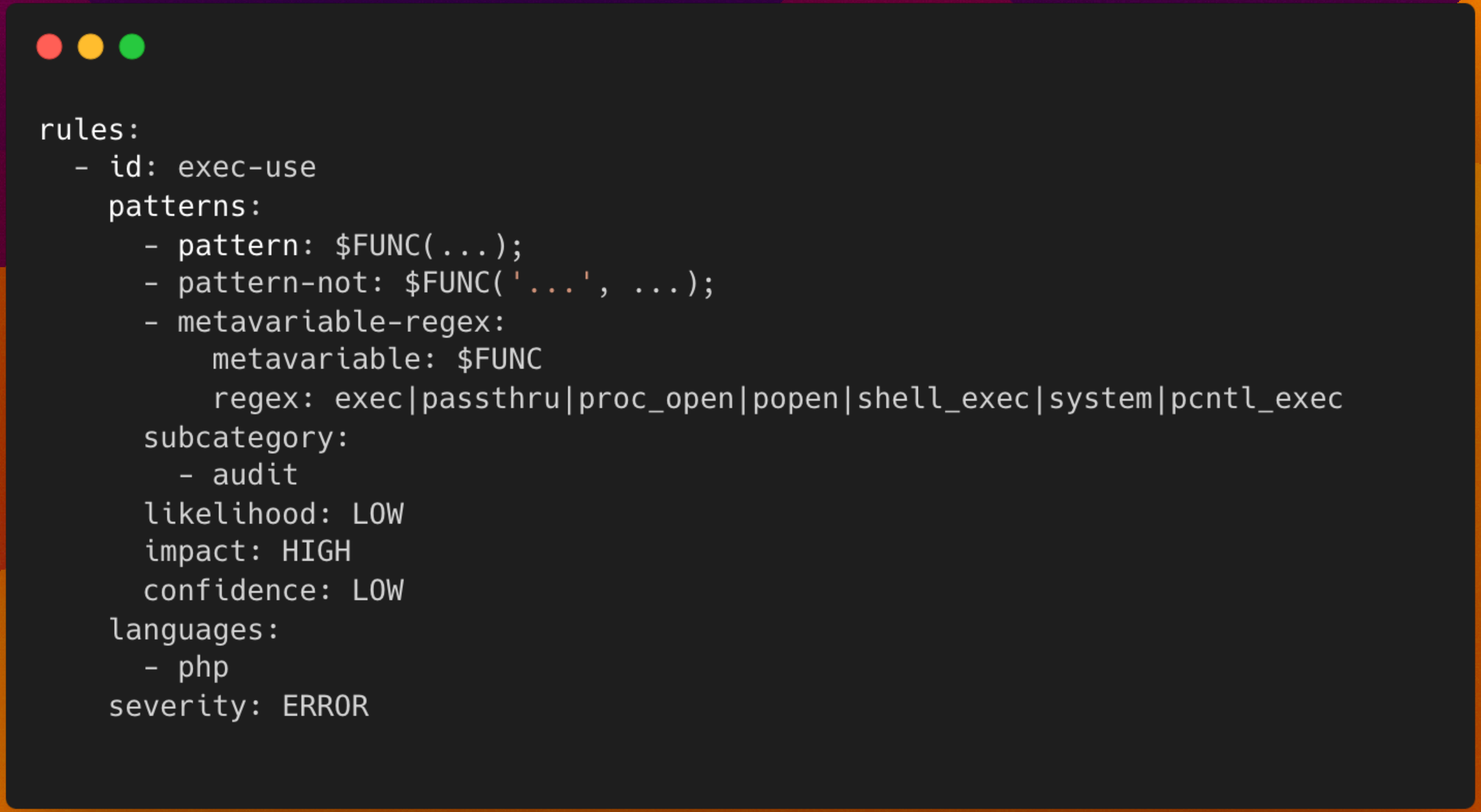
Semgrep

- Open source
- Supports PHP & JS natively
- Ability for custom rules
- Easy to install locally
- Easy to intergrate with your pipelines
- You can run it locally without a login

## So how is this beneficial for you with PHP

When it comes to PHP, Semgrep allows you to look for common issues such as checking for functions that might introduce security misconfigurations.

There are also rules that allow you to check for missing properties on forms or if someone has disabled key security features.



```yaml
rules:
  - id: exec-use
    patterns:
      - pattern: $FUNC(...);
      - pattern-not: $FUNC('...', ...);
      - metavariable-regex:
          metavariable: $FUNC
          regex: exec|passthru|proc_open|popen|shell_exec|system|pcntl_exec
      subcategory:
        - audit
      likelihood: LOW
      impact: HIGH
      confidence: LOW
    languages:
      - php
    severity: ERROR
```

## Different languages and rules

- Semgrep comes with a range of rules and allows you to make your own custom ones not just for PHP but also for JavaScript.
- Their rules are part of what they call the registry *(https://semgrep.dev/explore)*
- Semgrep also allow us to define these into rulesets and even offer their own quick start ones for checks
  - *395 JavaScript rules*
  - *88 PHP rules*
- https://semgrep.dev/p/owasp-top-ten

## How do I use Semgrep?

```
# Mac
brew install semgrep

# Linux & Window WSL
# install through pip
python3 -m pip install semgrep
```

Locally

```
SAST Scan:
  image: semgrep/semgrep:latest
  script: semgrep-agent
```

Pipelines

## Demo: Setting up Semgrep with your PHP application

## So what's next?

- So your code and your colleagues code is being checked for common vulnerabilities
- But what about those composer and NPM packages?
- Can servers and containers installs be insecure?

**But I didn't write the code for this, so why should I care?**

## Supply Chain Attacks

What is a supply chain attack?

A **supply chain attack** is a type of cyberattack that targets a trusted third-party vendor who offers services or software vital to the supply chain

**CrowdStrike**

**How does this affect me?**

Who here knows every package they have installed and what packages those packages depend on?

And do you know if they have any CVE’s?

*(Common Vulnerabilities and Exposures)*

## Scanning your dependencies and packages

**You can use scanners to check over your package files for issues in the following:**

- Composer packages
- NPM packages
- APT or APK installs
- OS updates

**While they aren't the silver bullet.
They will identify common CVE or out of date packages**

## This is where Trivy comes along

### Scanner that checks for CVE’s within your third parties and more!

- Scans your OS and software dependencies for CVEs
- It scans your package manager files such as:
    - composer.lock
    - package-lock.json
    - yarn.lock
- Also scans your OS package managers such as APK, YUM, APT and DPKG.

## So how can I use this?

Trivy can be used in multiple ways

- Installing it directly on your server or local machine and running the file system scanning
- Setting it up to scan against your repository using the repository scanning
- Scanning your docker images from your container registry using the image scanning

## Demo: Setting up Trivy with your PHP Application

## So now that we have it all set up, now what?

Reports

## What type of reports can we run?

- For users not using pipelines such as Github Actions or Gitlab CI
  - Both Semgrep and Trivy support JSON or SARIF for CVE reports
  - SBOMs are your friend for listing your packages used

- When it comes to Gitlab you can send your vulnerabilities to a dashboard if you or your organisation is on the ultimate package

## So now that we have all of this set up are we secure?

Sadly not

but congrats on talking the first steps to making your application more secure and getting involved in Application Security

Security is all about layers and this is a great way to prevent common vulnerabilities being apart of your application

Thank you all for putting up with me while you wait for the PHPUnit talk

*If you don't feel comfortable asking here, I will be about all event so just ask whenever you see me!*

## Any questions?

/zacklott

/zacklott

RoadSigns