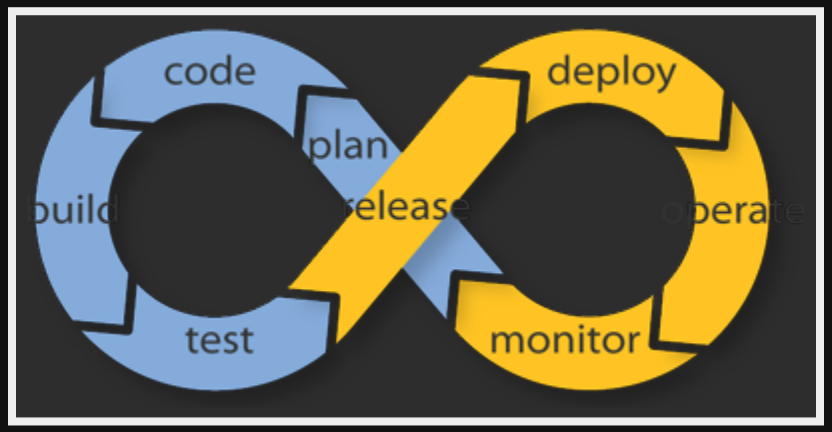
Security Automation in your Continuous Integration Pipeline



Jimmy Byrd

https://theangrybyrd.github.io/OWASPPipelineSlides

Jimmy Byrd?

- @jimmy_byrd
- Github
- Lead Developer at Binary Defense Systems

Continuous what now?

Continuous Integration

Continuous Integration is a software development practice where members of a team integrate their work frequently . . . Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible.

Martin Fowler

Why?

To prevent:

Merging code

Continuous Integration

1. Push
2. Build*
3. Test
4. Report

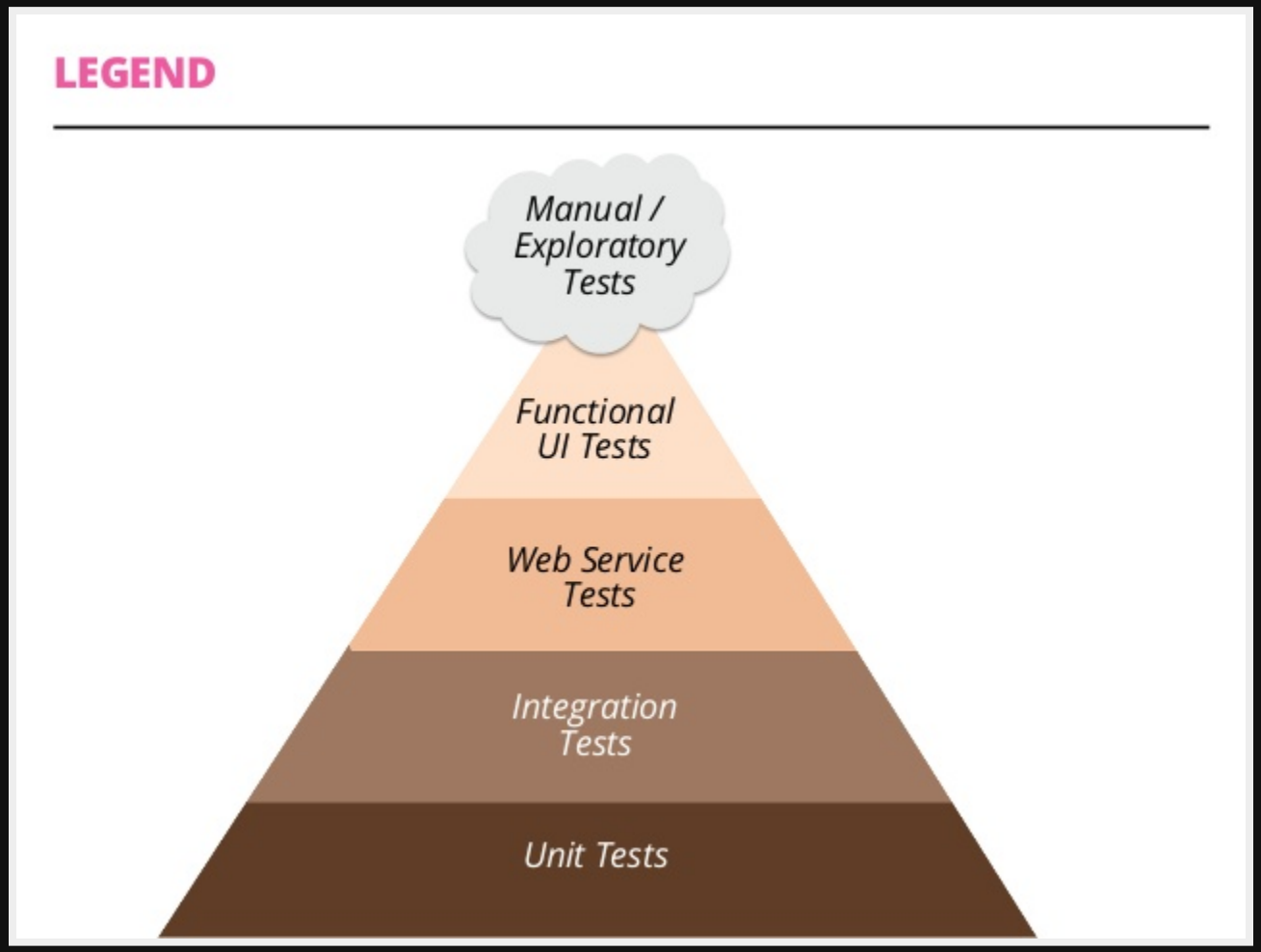
```
./build.sh

-----------------------------------------------------------------------
Build Report
-----------------------------------------------------------------------
Target           Duration
------           --------
RestoreNpm       00:00:32.4737389
PackWebAssets    00:00:03.7281990
Linter           00:00:03.3159012
Compile          00:00:22.2821302
RunTests         00:00:04.9936549
ScanCode         00:02:04.0223912
Status:          Ok
-----------------------------------------------------------------------


```

CI Tools

- Teamcity
- Jenkins
- Gitlab
- Travis
- AppVeyor

Software Testing

Automated testing hierarchy



Why aren't we writing security tests?

Writing your own crypto

Committing the production database password to source control

Storing plain text passwords

Sql Injection

Why don't we have both?

The Rugged Manifesto

I am rugged and, more importantly, my code is rugged.

I recognize that software has become a foundation of our modern world.

I recognize the awesome responsibility that comes with this foundational role.

I recognize that my code will be used in ways I cannot anticipate, in ways it was not designed, and for longer than it was ever intended.

I recognize that my code will be attacked by talented and persistent adversaries who threaten our physical, economic and national security.

I recognize these things – and I choose to be rugged.

I am rugged because I refuse to be a source of vulnerability or weakness.

I am rugged because I assure my code will support its mission.

I am rugged because my code can face these challenges and persist in spite of them.

I am rugged, not because it is easy, but because it is necessary and I am up for the challenge.

OWASP/Glue

Glue is a framework for running a series of tools. Generally, it is intended as a backbone for automating a security analysis pipeline of tools.

Github README

Recently renamed

```
1: s/pipeline/glue
```

Maintainers

- Matt Tesauro
- Aaron Weaver
- Matt Konda

Four simple concepts

- Mounters
- Tasks
- Filters
- Reporters

What's in the box? (MOUNTERS)

- Docker
- File System
- Git
- ISO
- URL

What's in the box? (TASKS)

- ClamAV
- Breakman (Ruby)
- Bundle-Audit (Ruby)
- Checkmarx (Code)
- DawnScanner (Ruby)
- File Integrity Monitoring
- FindSecurityBugs (Java)
- NodeSecurityProject (Javascript)
- OWASPDependencyCheck (Java and .NET)
- PMD Source Code Analyzer (Code)
- RetireJS (Javascript)
- Snyk (Javascript)
- Zap

What's in the box? (FILTERS)

- Jira
- Zap

What's in the box? (REPORTERS)

- CSV
- Jira
- Json
- Text

Getting started

Native

```
1: gem install owasp-glue
```

Docker

```
1: docker pull owasp/glue
2: docker run -i -t --entrypoint=/bin/bash owasp/glue
```

Help

```
glue --help
```

Hello World!

```
1: glue -t eslint,retirejs https://github.com/OWASP/NodeGoat.git
```

Hello World output

```
Finding: NodeGoat.git
Description: Package uglify-js-2.4.24 has known security issues
Timestamp: 2016-06-24 14:43:35 +0000
Source: {:scanner=>"RetireJS",
         :file=>"owasp-nodejs-goat->swig->uglify-js-2.4.24",
         :line=>nil,
         :code=>nil}
Severity: 0
Fingerprint:  041c4f08bd5a3decc502217f15b7787b654b800e092ffadb939bd99e4e
Detail:  https://nodesecurity.io/advisories/48


```

Tasks

Tools vs Labels

Tools vs Labels

Have to go code spelunking

```
1: cd ./lib/glue/tasks
```

Example from Brakeman.rb

```
1: def initialize(trigger, tracker)
2:     super(trigger, tracker)
3:     @name = "Brakeman"
4:     @description = "Source analysis for Ruby"
5:     @stage = :code
6:     @labels << "code" << "ruby" << "rails"
7: end
```

Important pieces

- Name (without spaces) = Tool
- Labels = Labels

All tools

```
av.rb:                "av"
brakeman.rb:          "brakeman"
bundle-audit.rb:      "bundleaudit"
checkmarx.rb:         "checkmarx"
dawnscanner.rb:       "dawnscanner"
eslint.rb:            "eslint"
fim.rb:               "fim"
findsecbugs.rb:       "findsecuritybugs"
nsp.rb:               "nodesecurityproject"
owasp-dep-check.rb:   "owaspdependencycheck"
pmd.rb:               "pmd"
retirejs.rb:          "retirejs"
scanjs.rb:            "scanjs"
sfl.rb:               "sfl"
zap.rb:               "zap"
```

Tools example

```
1: glue -t brakeman,eslint
```

This will run brakeman and eslint

All Labels

```
av.rb:                "filesystem"
brakeman.rb:          "code", "ruby", "rails"
bundle-audit.rb:      "code", "ruby"
checkmarx.rb:         "code"
dawnscanner.rb:       "code"
eslint.rb:            "code", "javascript"
fim.rb:               "filesystem"
findsecbugs.rb:       "code"
nsp.rb:               "code"
owasp-dep-check.rb: "code", "java", ".net"
pmd.rb:               "code"
retirejs.rb:          "code", "javascript"
scanjs.rb:            "code", "javascript"
sfl.rb:               "code"
zap.rb:               "live"
```

Labels example

```
1: glue -l ruby
```

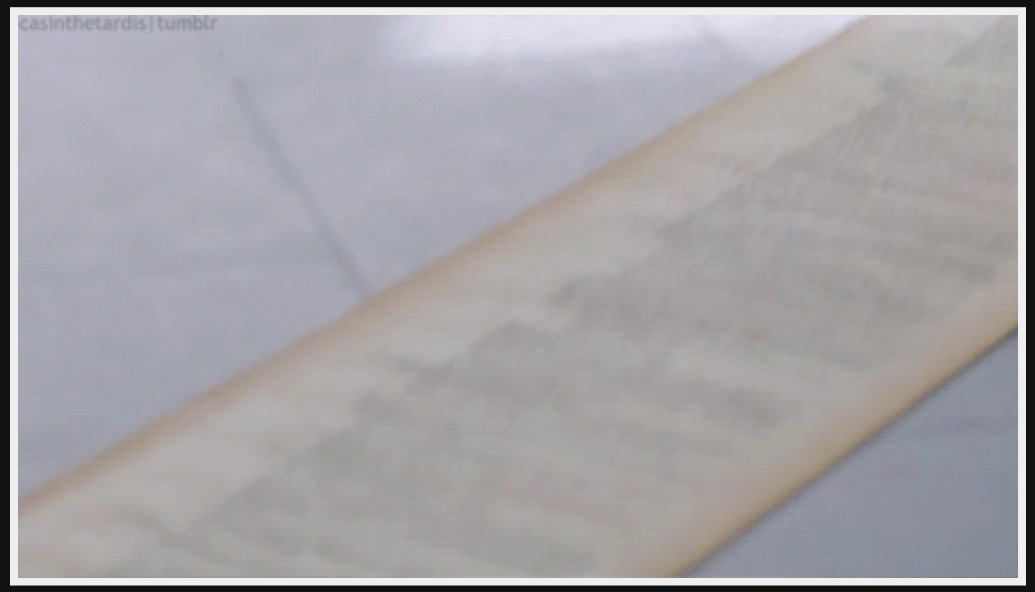
This will run brakeman and bundle-audit

Building your own task/filter/reporter is pretty easy

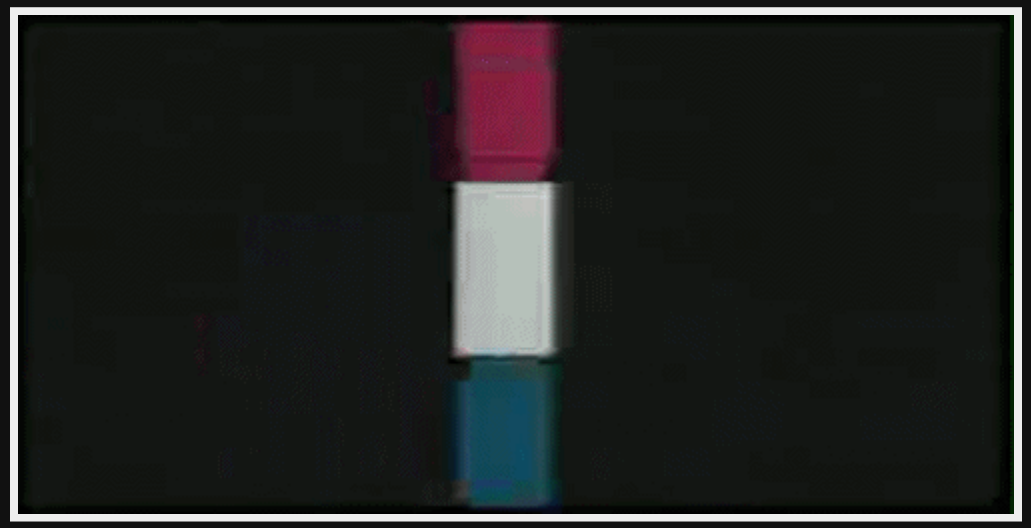
Challenge

Write a task for vigil

First time on your code



Knowing is half the battle



Resources

- OWASP Glue
- OWASP Glue Github
- Pipelines, DevOps and making things better - Matt Tesauro
- Design Approaches for Security Automation - Peleus Uhley