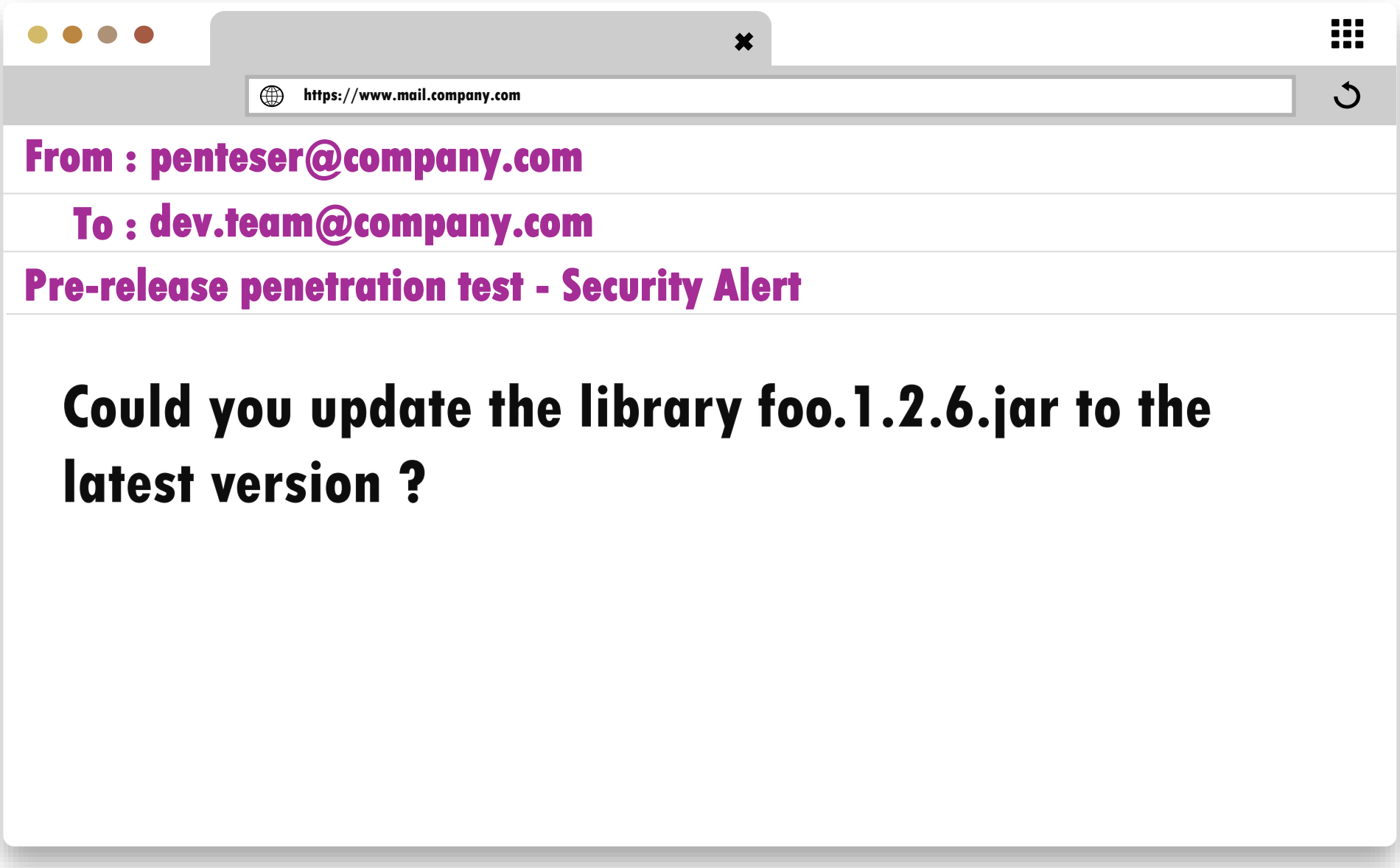
SECURING FAST & FURIOUS DEVOPS PIPELINES

BY : ABDESSAMAD TEMMAR

A PERSONAL STORY



A PERSONAL STORY

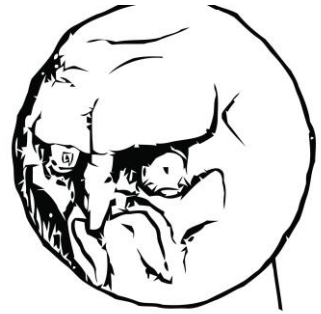
https://www.mail.company.com

From : dev.team@company.com

To : penteser@company.com

R : Pre-release penetration test - Security Alert

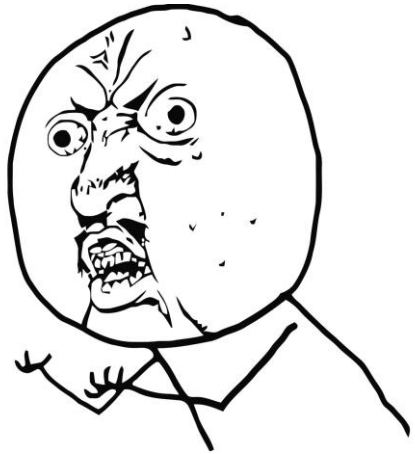
Sorry - that would take weeks, we don't have the resources to do that.



NO.

HOW THIS HAPPEN ?

- The dendepency is tightly related to other dependencies
- 100s of services to updates
- Some not touched in years



→ The problem : It's too late/hard to fix this issue !

WHY WE NEED AUTOMATED SECURITY TESTING ?

- Early feedback about security issues
- Manual security activities are bottlenecks for Software Dev.
- Business goals trump security needs
- Pentesters keeps finding basic issues

SO, WHAT AM I GOING TO TALK ABOUT TODAY ?

ABOUT ME

@T333333R

/in/abdessamad-temmar-6aa29a57

- Abdessamad TEMMAR
- Application Security Lead @ Amundi (from Siris Advisory)
- OSCP
- OWASP Contributor (OPC, MSTG & MASVS)
- Project : github.com/TmmmmmR/security_automation

ABOUT ME

"I AM A NICE SECURITY PROFESSIONAL, NOT MINDLESS VULNERABILITY SPEWING MACHINE. IF I AM TO CHANGE THIS IMAGE, I MUST FIRST CHANGE MYSELF.

DEVELOPERS ARE FRIENDS, NOT FOOLS."

- Bruce, Aaron and Matt

AUTOMATED SECURITY TESTING CHALLENGES

Speed

Integration

Ease of use

Accuracy

Communication

Portability

EXISTING TOOLS

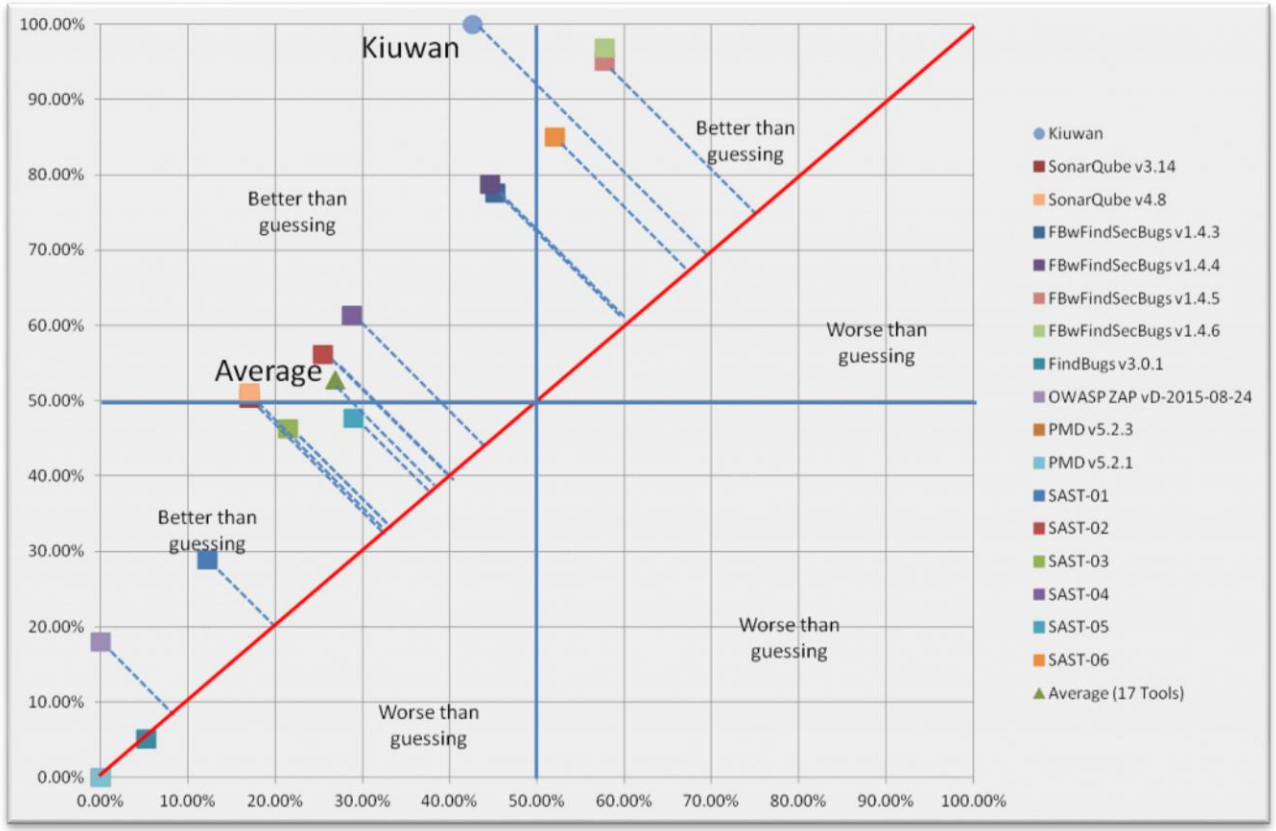
SCA (Dependencies Analysis)	• Parse dependencies definition file (pom.xml) to check for potential vulnerable components
DAST (Dynamic Scans)	• Black/Grey box testing • Simulates live attacker • Sends HTTP request and Analysis responses
SAST (Static Analysis)	• Whitebox testing • Intermediate code representation (Data Flow) • Checks all possible execution paths
IAST (Interactive Testing)	• Hybrid analysis : correlates SAST and DAST results • Use agents to monitor application runtime • Use SAST output to improve DAST coverage

EXISTING TOOLS

	SCA (Software Composition Analysis)	SAST (Static Application Security Testing)	DAST (Dynamic Application Security Testing)	IAST (Interactive Application Security Testing)
Speed	Minutes to Hours	Instant to Hours	Hours to Days	Instant to Hours
Integration	IDE, Build, Binary	IDE, Build, Binary	Build	Test Automation
Ease of Use	Fairly Easy	Varies. Can be Complex	Requires some security skills	Easy (once deployed)
Relevance	Hard to determine actual impact	Can be overwhelming	Focus on Front end (but some FPs)	Very relevant
Actionability	Not always straight forward	Right on. Points to Line Code	Difficult	Right on. Points to Line of Code

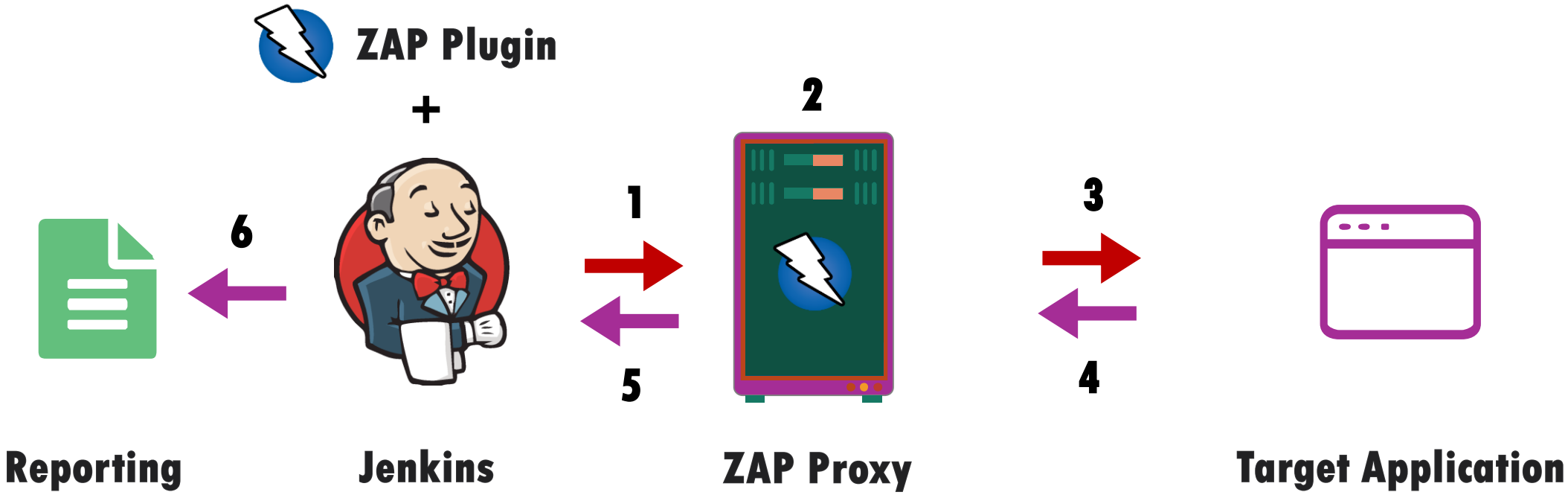
EXISTING TOOLS

Scanning tools : how to select the right one ?

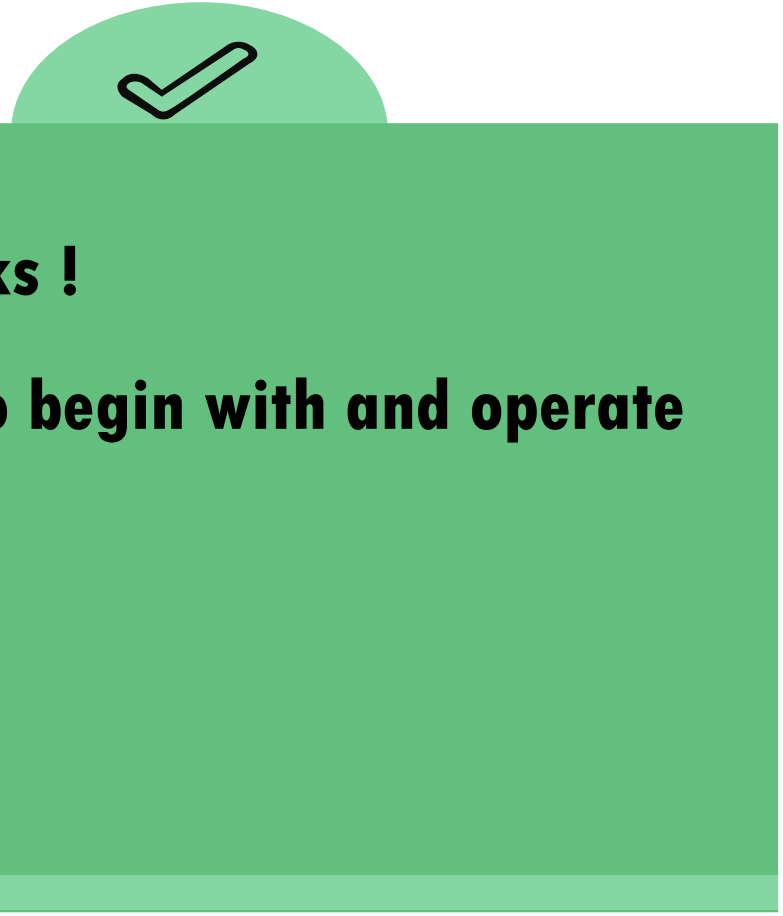


MY JOURNEY FOR SECURITY AUTOMATION AT HIGH SCALE

SOLUTION #1 : PLUGINS TO INVOKE REMOTE SECURITY SCANNERS



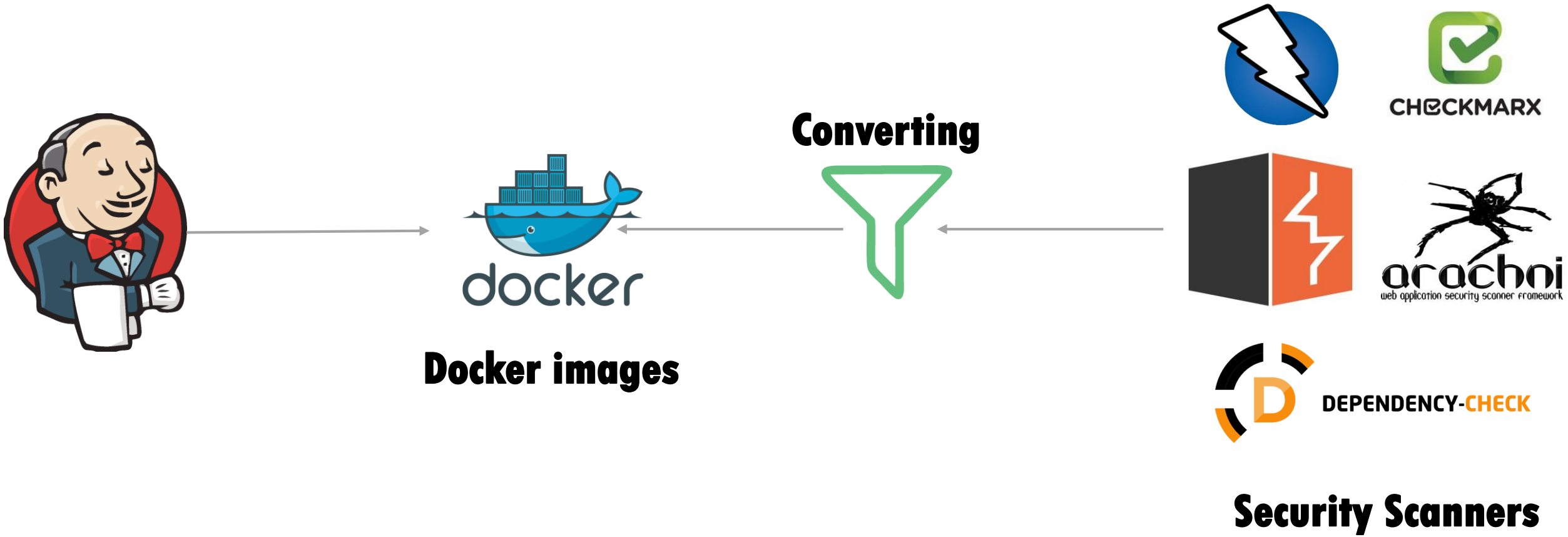
SOLUTION #1 : TAKE-AWAYS



- It works !
- Easy to begin with and operate

- Performance
- Lack of portability

SOLUTION #2 : DOCKERZING SECURITY SCANNERS



SOLUTION #2 : TAKE-AWAYS

- Portability : configure/build once , scan from everywhere
- Easy to run/inject inside a CI/CD pipeline

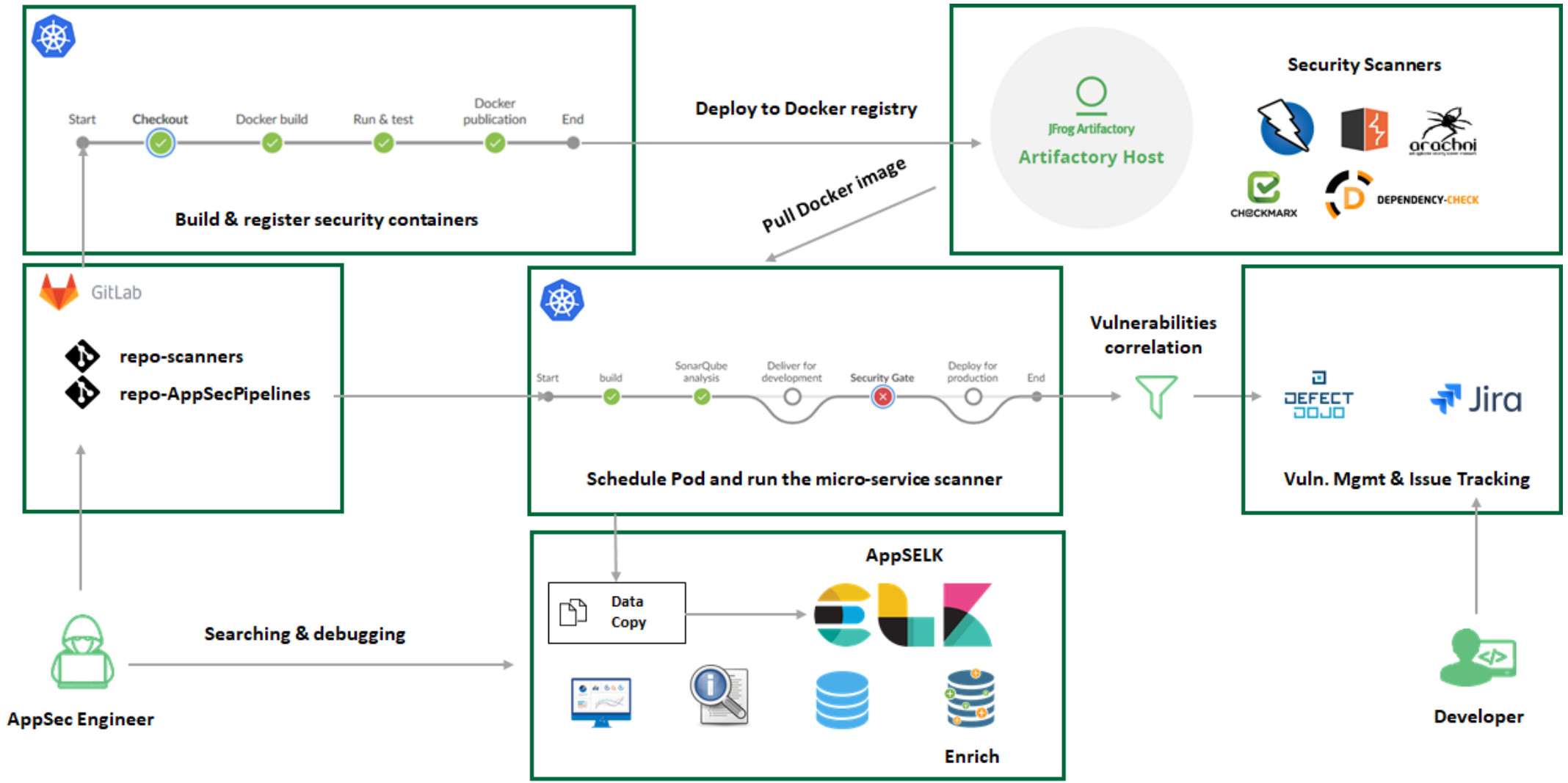
- Difficulties to manage the deployment & maintenance of your containers
- We didn't yet solve the performance issue !


WE NEED GUNS
LOTS OF GUNS

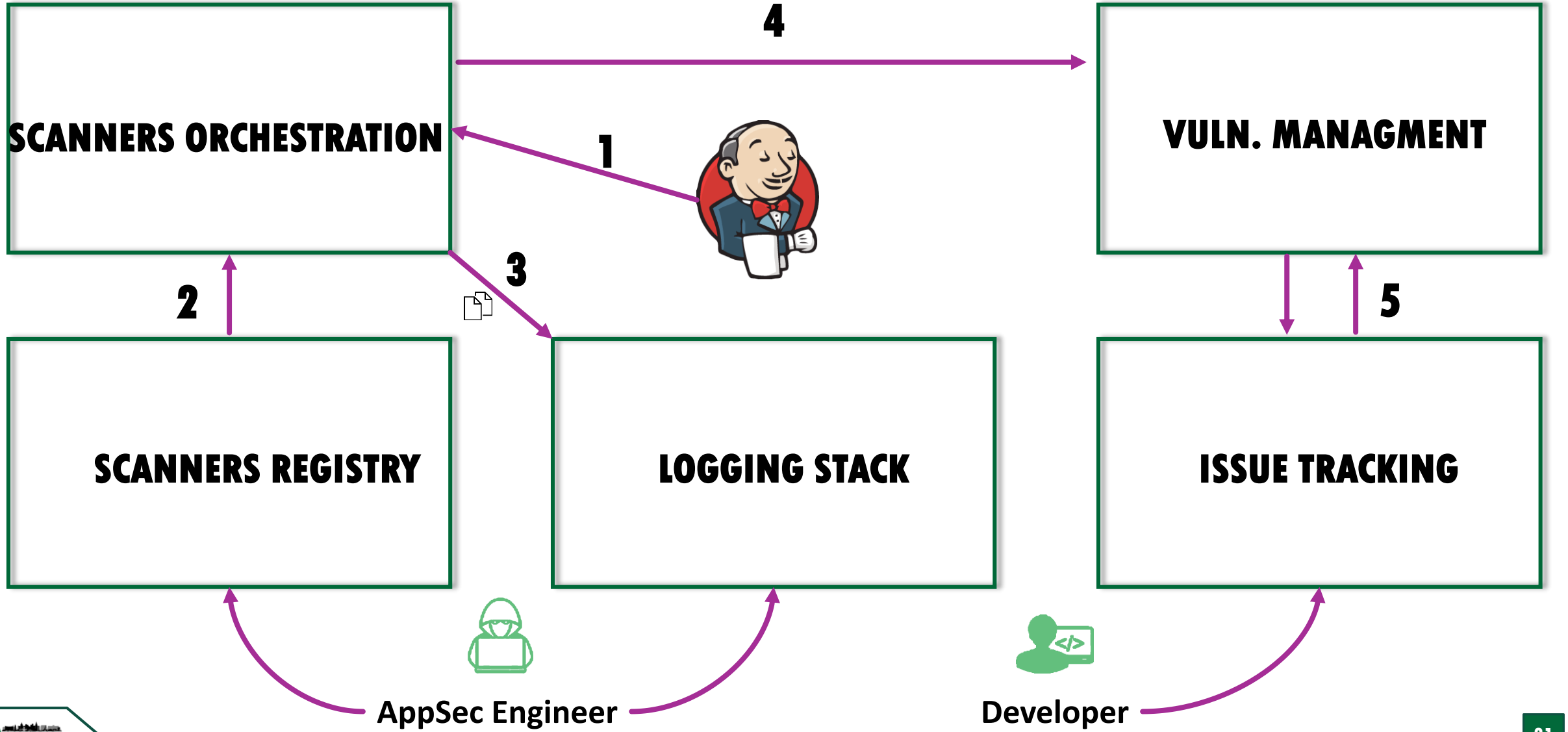
ROAD TO CONTAINERS (SCANNERS) ORCHESTRATION

01	Moving to container-based workload
02	Solving performance issue
03	New slave/fresh workspace for each scanning job
04	Micro-service architecture (Scanning As A Service)

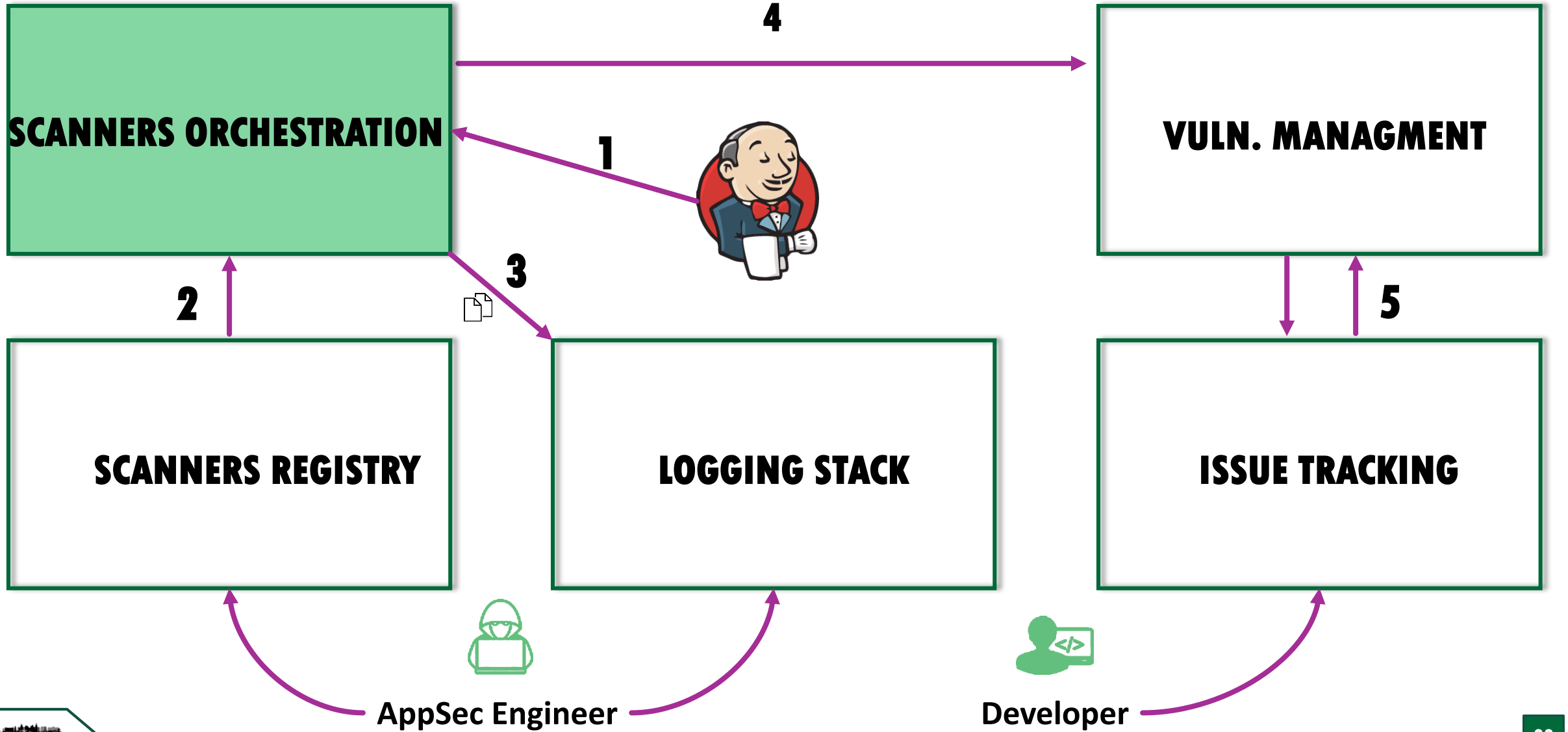
SOLUTION #3 : MICRO-SERVICE ARCHITECTURE



SOLUTION #3 : MICRO-SERVICE ARCHITECTURE



SOLUTION #3 : MICRO-SERVICE ARCHITECTURE



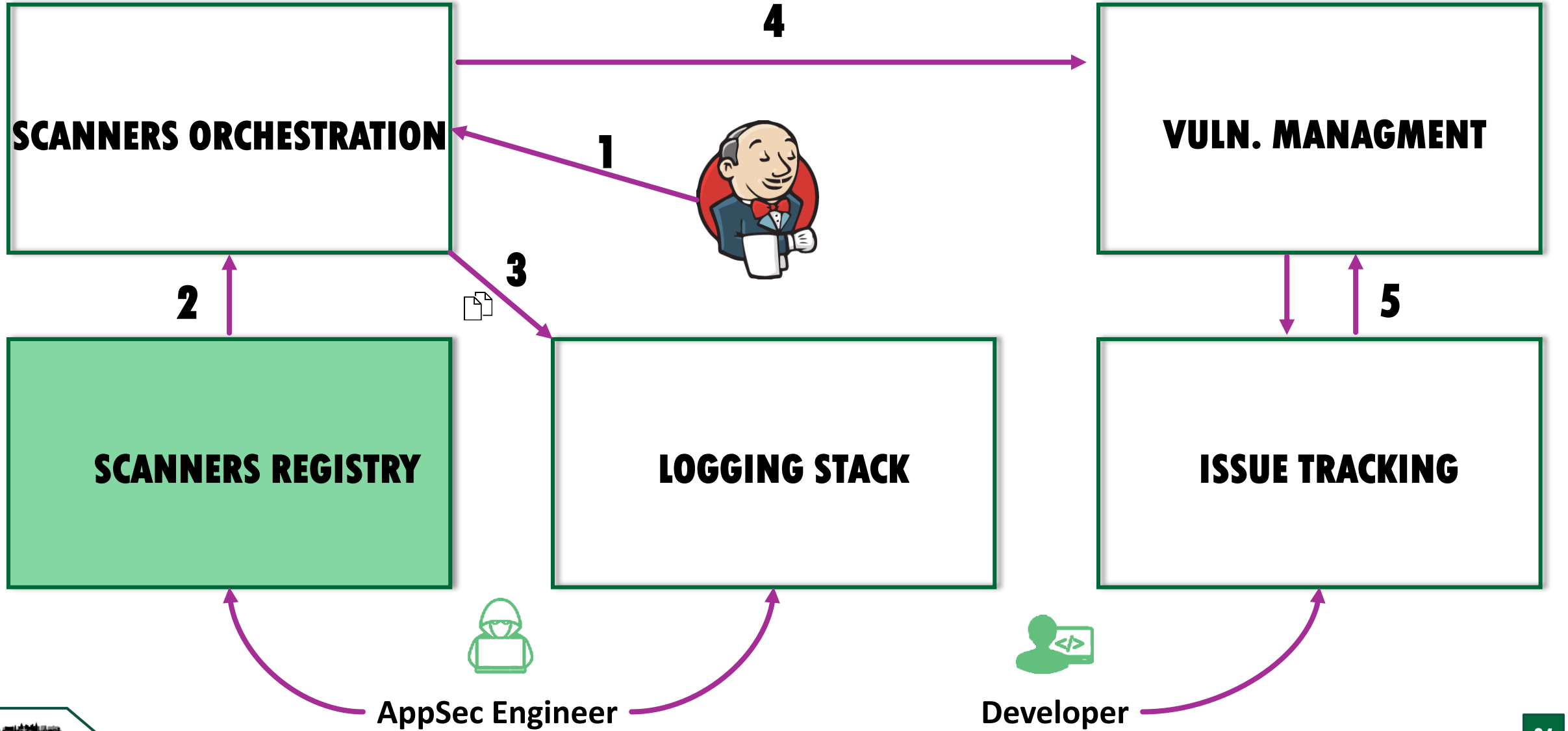
SCANNERS ORCHESTRATION : KUBERNETES

- Distribute the scanning workload over the Kubernetes cluster
- Scalable and ephemeral instances of scanners

SCANNERS ORCHESTRATION

Schedule a Pod and run micro-service scanner

SOLUTION #3 : MICRO-SERVICE ARCHITECTURE



SCANNERS REGISTRY

- Unify how we use/run security scanners
- Continuous improvement of the scanners tool set

SHARED LIBRARIES FOR SECURITY TOOLING

- **Explicit definition : Definition As Code**
- **Define common scan policy for security tooling**
- **Import multiple libraries/scanners in your Jenkinsfile**

SHARED LIBRARIES FOR SECURITY TOOLING

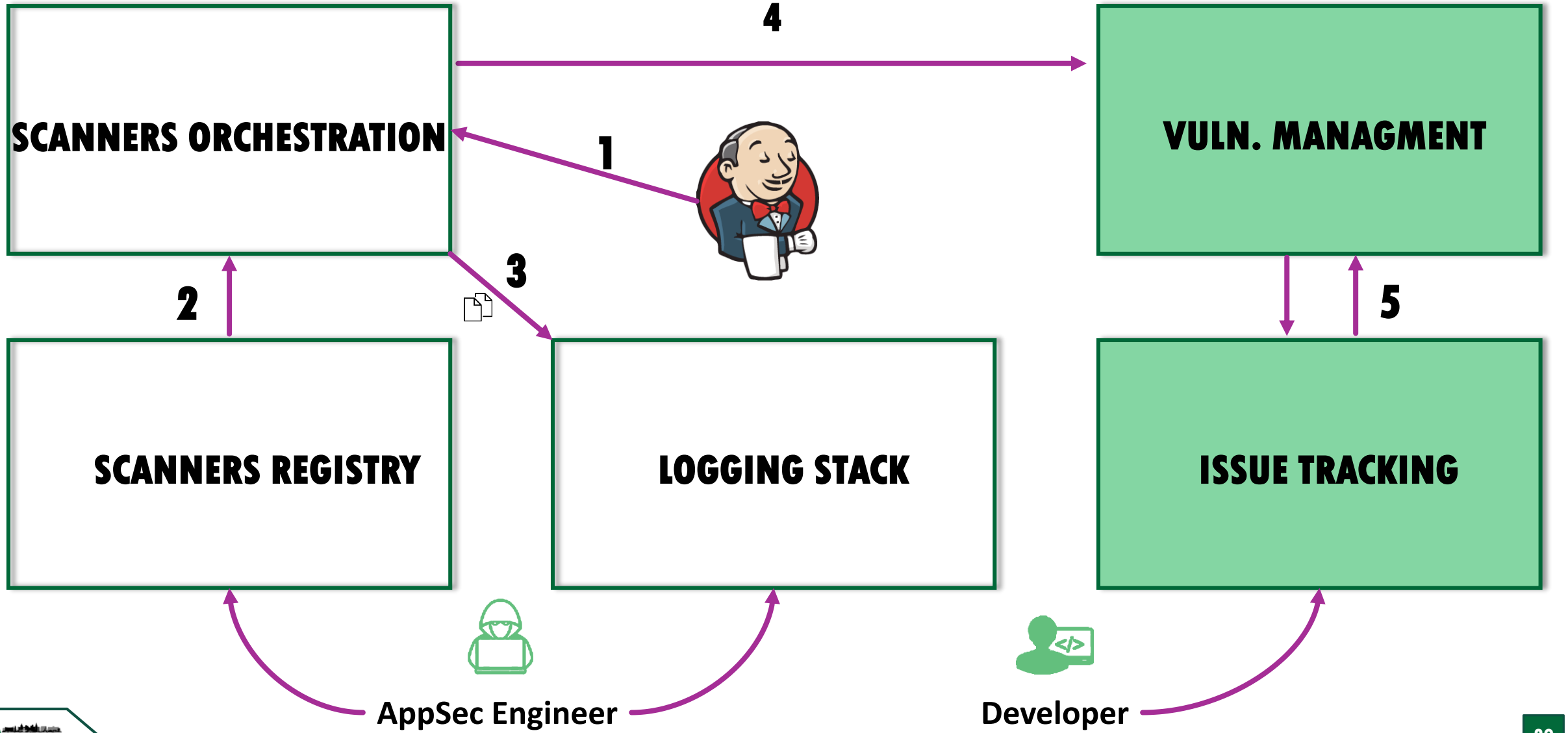
Jenkinsfile-checkmarx 109 Bytes

```
#!/usr/bin/env groovy
@Library(["CommonLibs", "SecLibs"]) _

Checkmarx{
    project_name          = "test"
}
```


YOU SEE...
ITS MAGIC

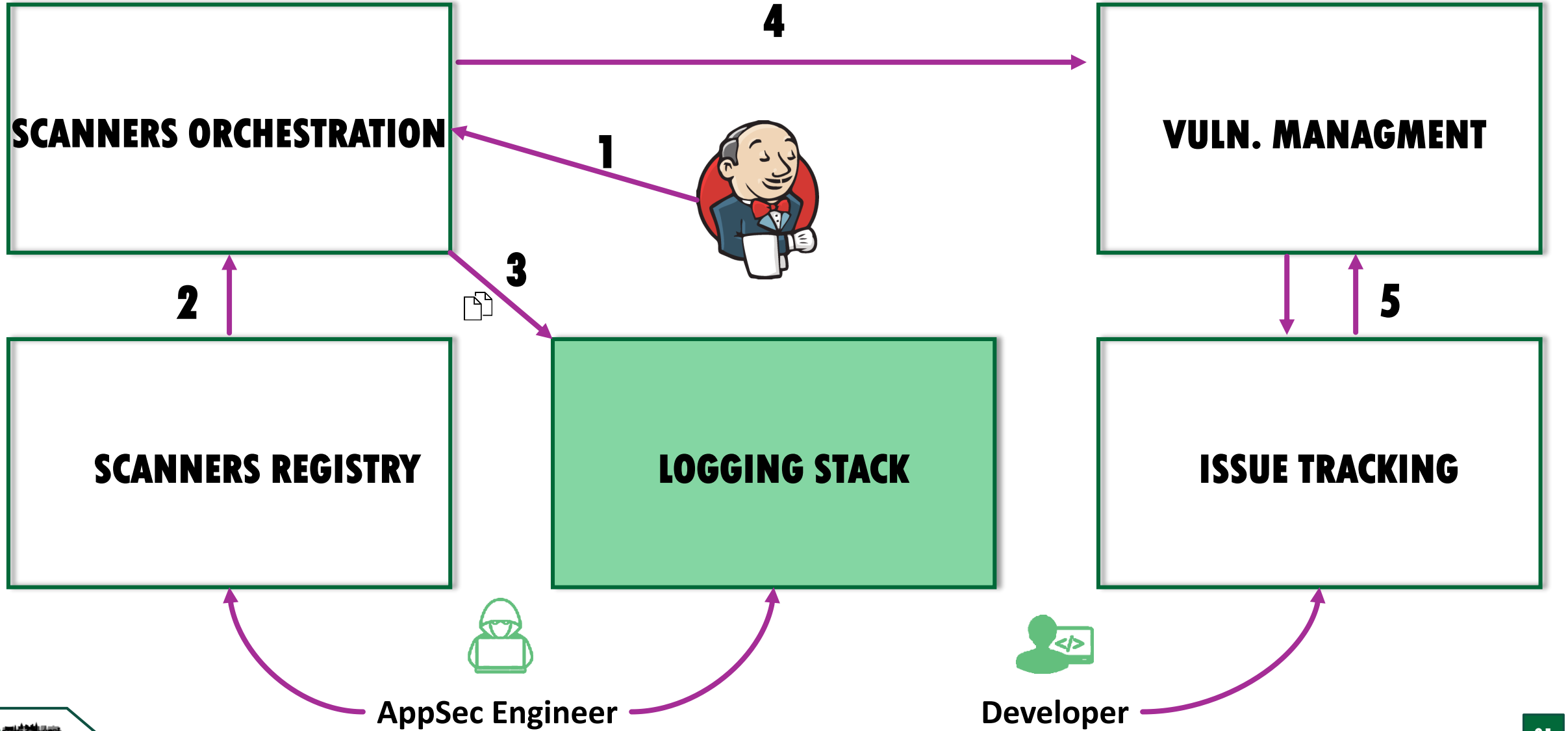
SOLUTION #3 : MICRO-SERVICE ARCHITECTURE



DEFECT-DOJO : MAKING VM LESS PAINFUL

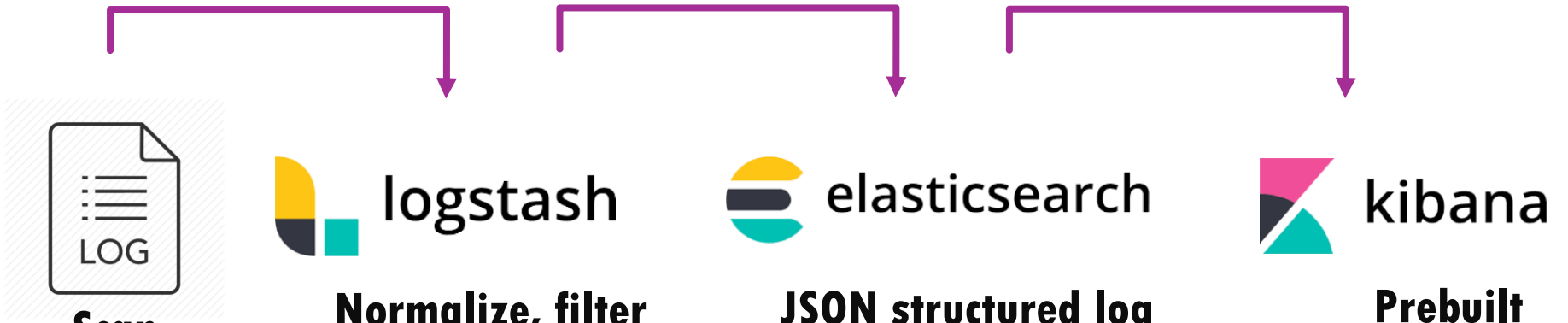
- **Vulnerability Management Tool**
- **Importers for many scanners**
- **De-duplication**
- **False Positive History**

SOLUTION #3 : MICRO-SERVICE ARCHITECTURE

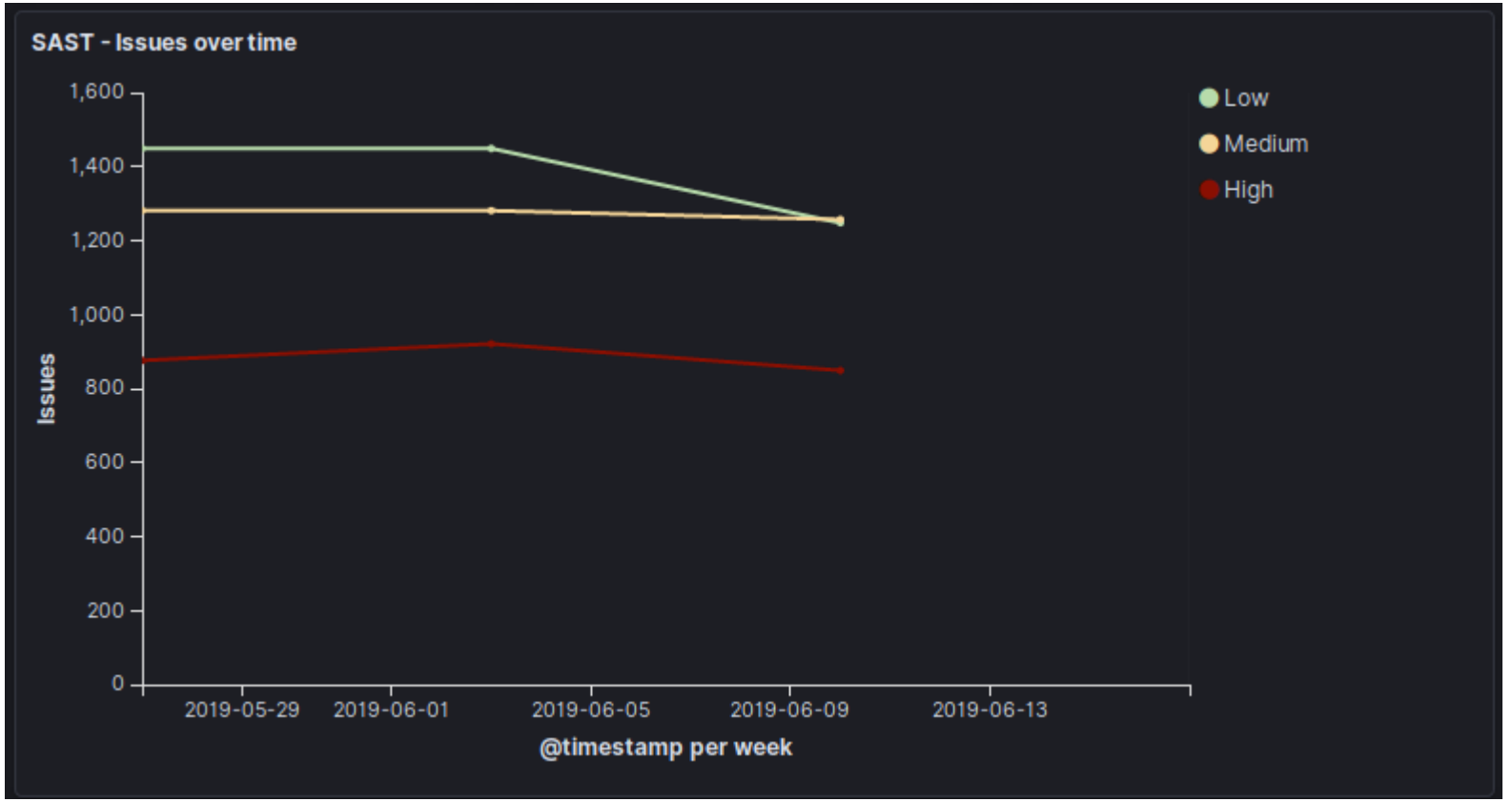


LOGGING STACK : ELK

- Create actionable data for dev team and metrics for managers
- Debugging the scanning activity (reports and logs)
- Customize risk scores



ELK STACK : ACTIONABLE VULNERABILITY SCANS



Tracking risk over time

SOLUTION #3 : TAKE-AWAYS

- Relying on Docker ecosystem
- Scalable and ephemeral instances of scanners
- Less tools to configure on the deployment servers

Nice problem to have :

- Monitoring / Metrics
- Resource Management

NEXT STEPS / QUESTIONS ?

- Explore more functionalities of ELK stack
- Deep fuzzing ! more detailed logs !
- Custom training based on the scanning results
- Establish a solid knowledge base for secure code snippets

THANKS!

Any questions?

You can find me at :

A.TEMMAR@SIRIS-AD.COM

@T333333R

/in/abdessamad-temmar-6aa29a57