DefectDojo

The Good, the Bad and the Ugly

OWASP Stammtisch Hamburg
Tilmann Haak
Manuel Schneider
2018-05-31

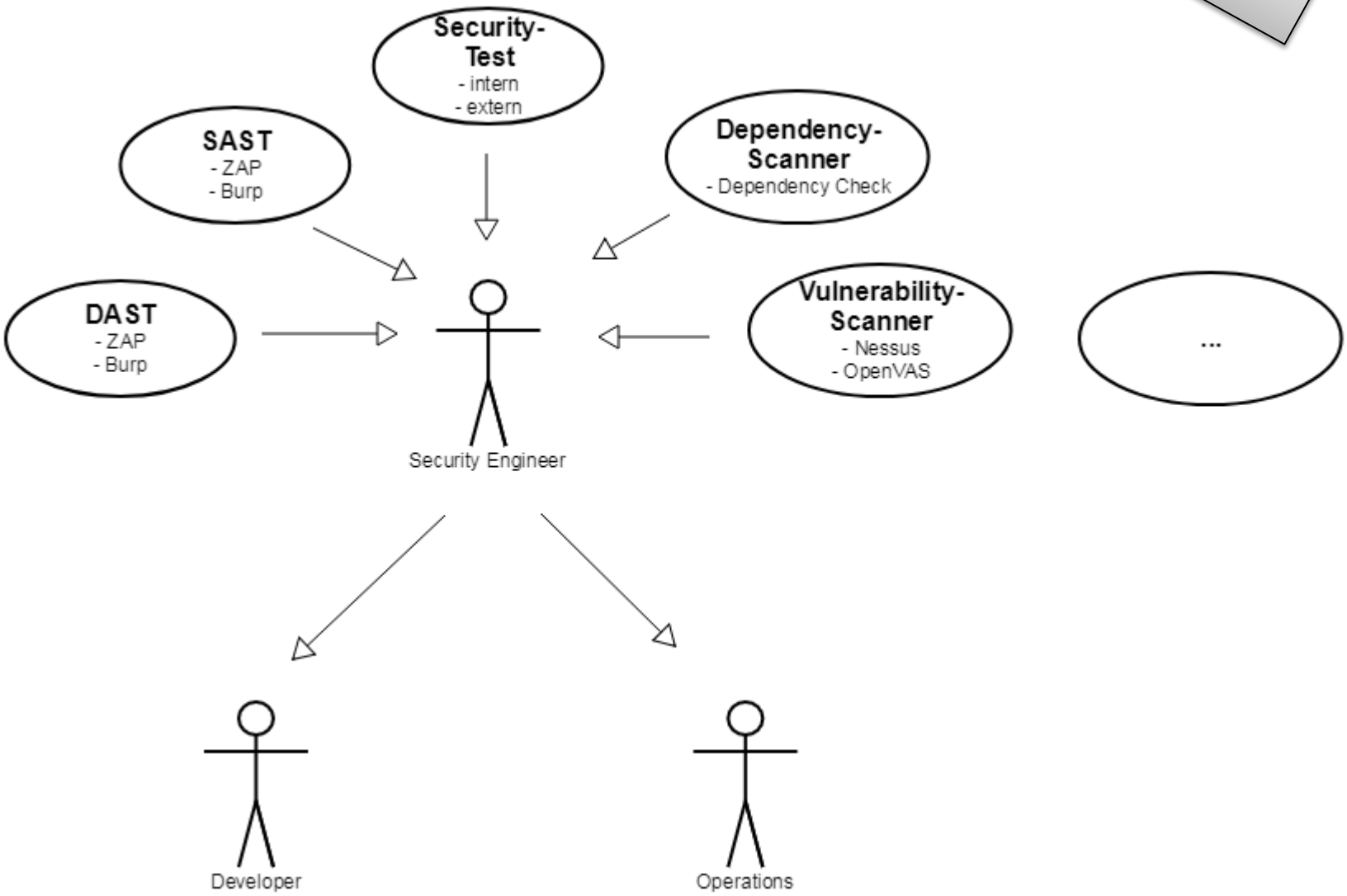
PREFACE

CIO: „What is the security posture of our applications?“

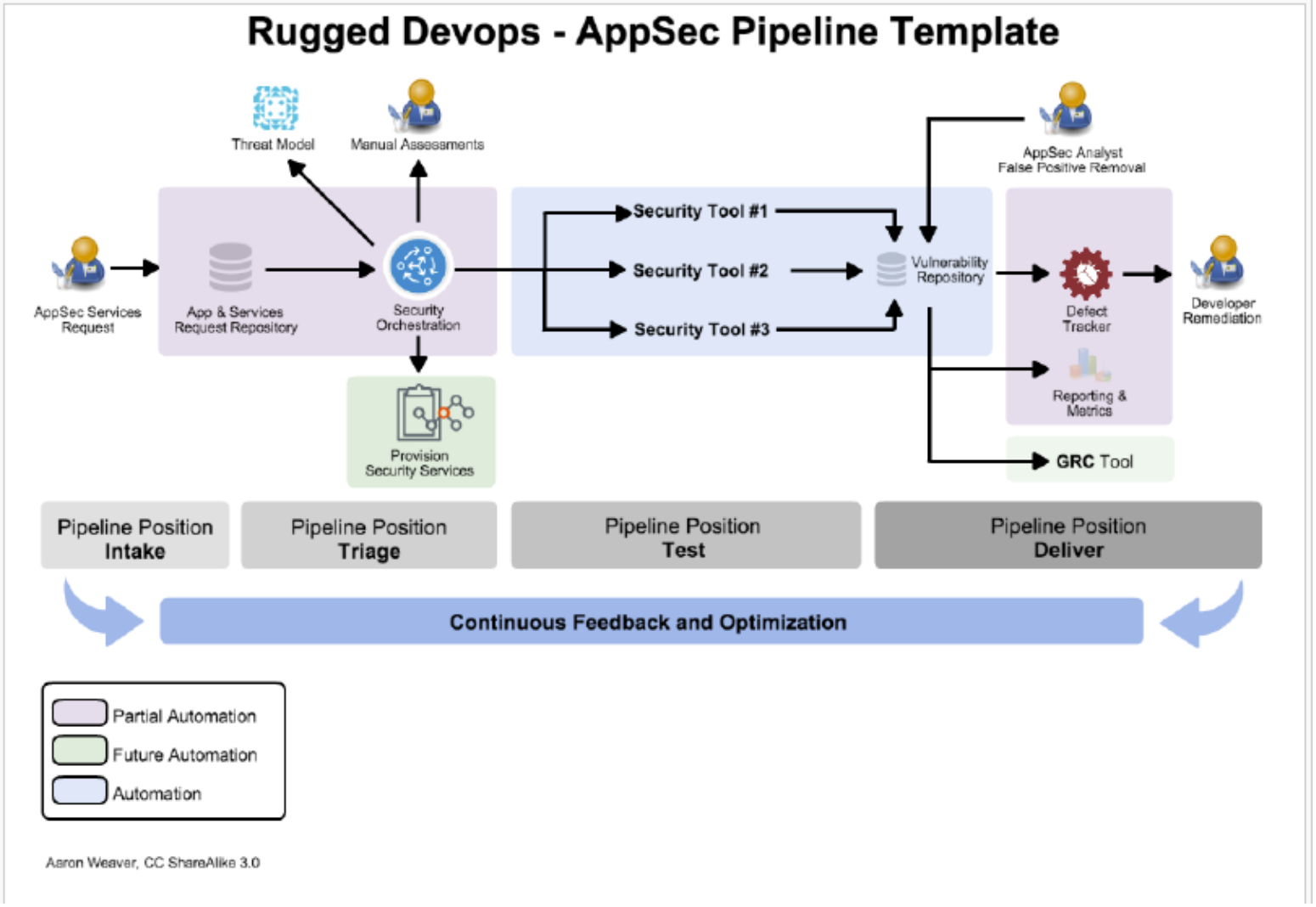
How do you handle and communicate vulnerabilities of (web-)applications?

A normal workday …

simplified



Application Security Pipeline



Application Vulnerability Corelation (AVC)

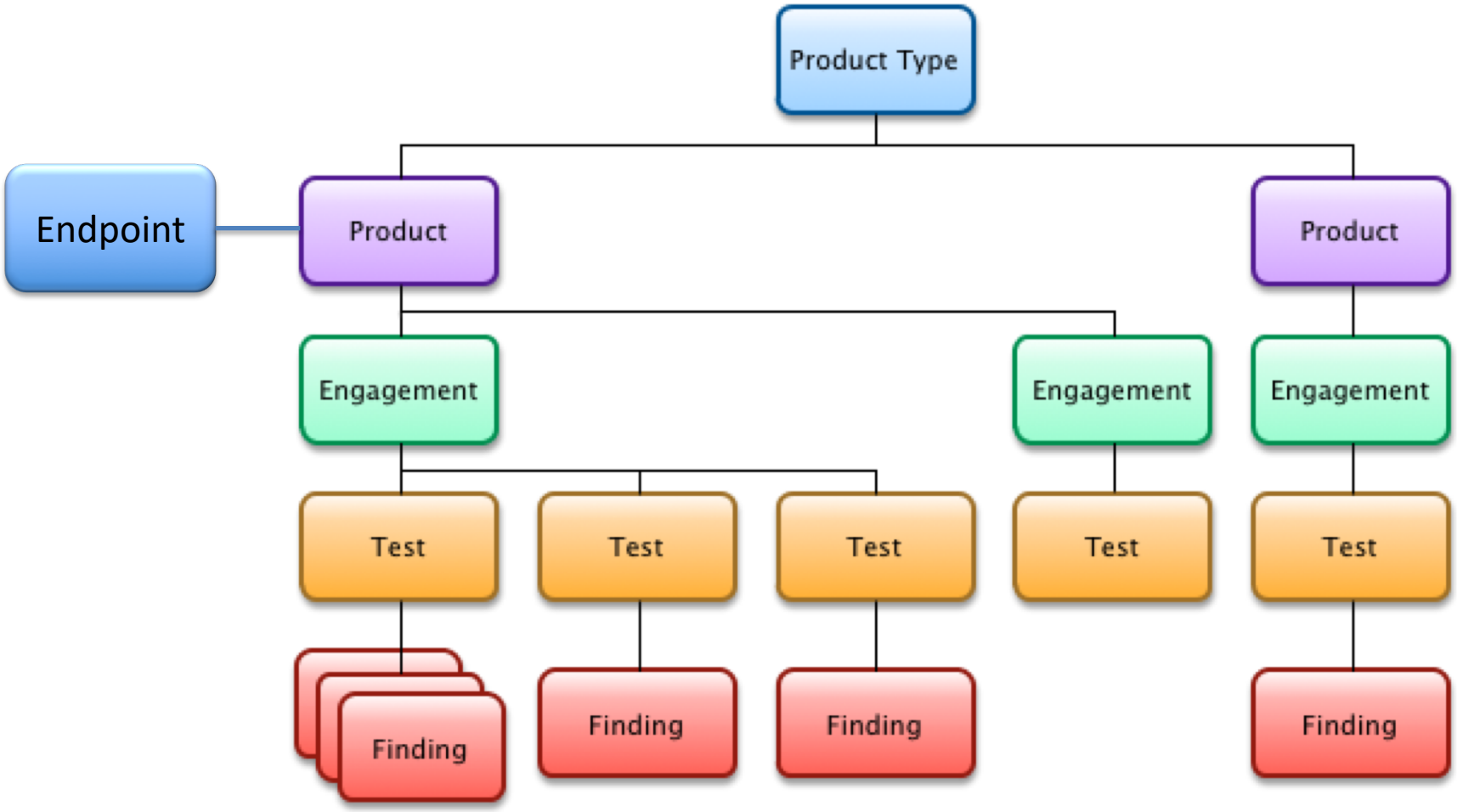
- “application security workflow and process management tools that aim to streamline SDLC application vulnerability remediation by incorporating findings from a variety of security-testing data sources into a centralized tool.”
- New tool category defined by Gartner
- Commercial tools
- Open source -> DefectDojo

DEFECTDOJO

What is the Promise of DefectDojo?

- Vulnerability Management Tool
- Security Program-/Test- Management Tool
- Importers for many scanners
- De-duplication
- REST API
- Free and Open Source (BSD 3-Clause)
- Uses Python Django, which makes it to integrate various plugins

DefectDojo Data Model



Docker

Seems to be easy!

Get it:

```
$ docker pull appsecpipeline/django-defectdojo
```

Run it:

```
$ docker run -it -p 8000:8000 \
  appsecpipeline/django-defectdojo
```

Web interface:

```
$ open http://localhost:8000/
```

Livedemo

- High Level Walkthrough DefectDojo
  - Typical workflows
  - Manual creation of a finding
  - Upload of report
  - De-duplication
  - Reporting
- Data needed/Products, etc.
  - DB-Export MySQL
    - manage.py / Django Data export/import
    - DB Tools

DEFECTDOJO @REAL LIVE

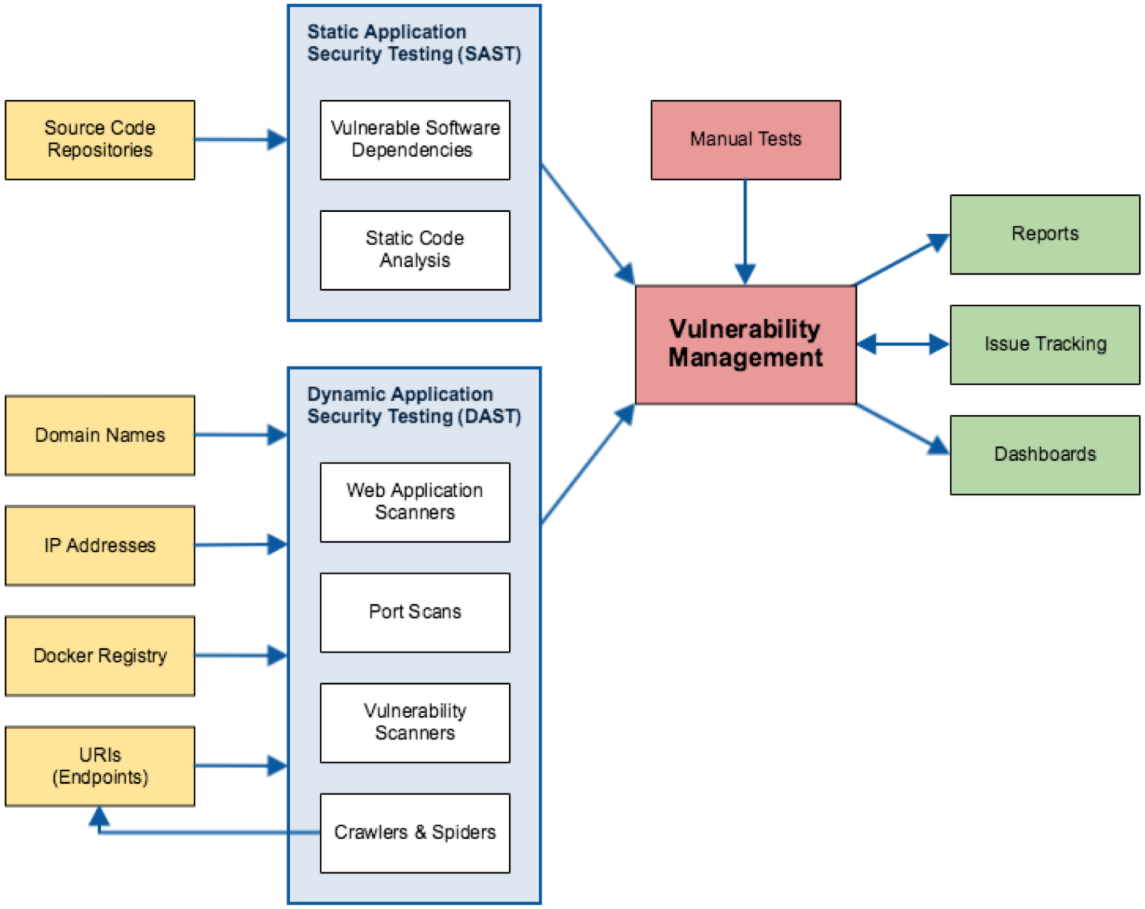
DefectDojo at Company A

- Existing inventory of platform applications
- Existing inventory of internal software
- Existing inventory of Micro-Sites
- OWASP Dependency Check for all known software projects
- Automated with Jenkins CI
  - Jenkins jobs (XML) generated with ERB (embedded ruby) templates
  - and uploaded via Jenkins API
- Central issue tracking with JIRA

Too much Software at Company A

- Many subsidiaries
- More than 100 own software applications
- Many engineering teams writing code
- 50+ Micro-Sites (esp. marketing)
  - Maintained by 17 external agencies
- 7+ mobile apps (Android, iOS, Windows)
- 2500+ hosts in two data centres (500+ physical, 2000+ VMs)
- A growing number of Docker containers (800+)

AppSec Pipeline at Company A

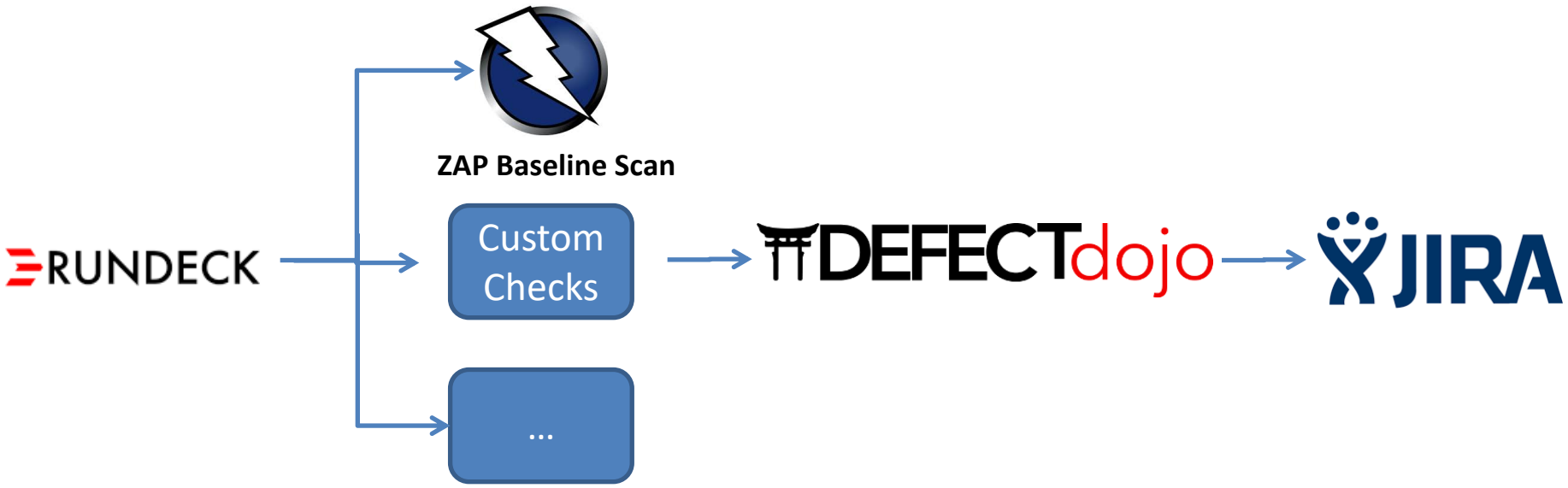


Company A's AppSec Pipeline

DefectDojo @Company B

- Motivation -> Security Assurance
  - Application Security Pipeline
- Large amount of Internet facing applications worldwide
- Baseline security scanning for Internet facing applications
- Push of vulnerabilities to JIRA
  - Distribution to devs/ ops via Jira
- Status planning/pilot
- Focus vulnerability documentation/ consolidation
  - Not test-management/ intake

AppSec Pipeline @Company B



OWASP Dependency Check for all projects @Company A

- Own software inventory
- Docker image with OWASP Dependency Check (and Ruby's bundler-audit)
- Generate Jenkins jobs for every software project to scan source code repository
- Push findings to DefectDojo
- De-duplicate + review with DefectDojo
- Push to JIRA (and get status changes via Webhook)

Dynamic Scanning @Company A
Scan all endpoints e.g. with Arachni

- Configure endpoints for all DefectDojo products based on our own software inventory
- Jenkins job pulls all endpoints from DefectDojo
- Scan all endpoints

And from here on, you know the drill:

- Push findings to DefectDojo
- De-duplicate + review with DefectDojo
- Push to JIRA (and get status changes via Webhook)

Dynamic Scanning @Company B
Scan all endpoints e.g. with ZAP

- Rundeck-Jobs for each application
  - Perform ZAP Baseline Scan
  - Upload to DefectDojo
  - Review results
  - Push to Jira
  - Distribute to dev/ops

Manual findings @Company B

How to handle findings from internal audits, external pen-tests

- Upload burp report to DefectDojo
- Enter findings for affected product in DefectDojo
  - Templates
- Push to JIRA vunerability project
- Clone and move to dev/ops teams

Manual findings @Company A

How to handle findings from internal audits, external pen-tests, and security researchers?

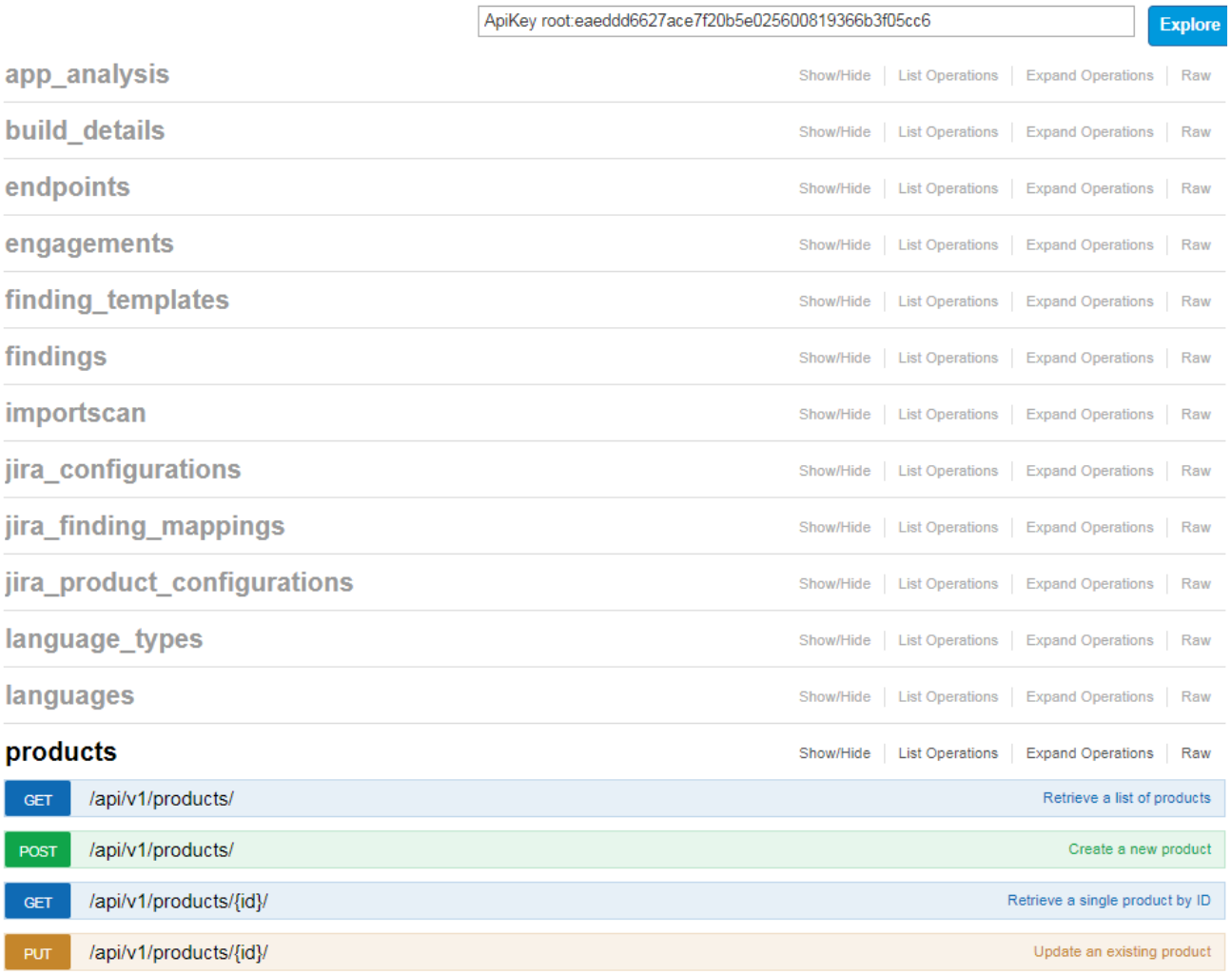
- Enter findings for affected product in DefectDojo
- Push to JIRA (and get status changes via Webhook)

Easy!

FEATURES

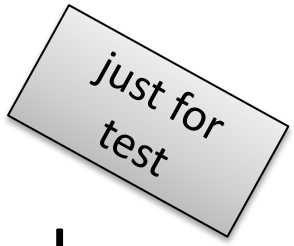
API examples

API V2 in Dev



https://github.com/aaronweaver/defectdojo_api - Python wrapper

Docker



Although the project claims to provide Docker images…

- Everything is cramped into a single container (bad!)
- My first try to split it up ended with approximately 1234 Docker images
- A high-availabilty Docker setup still requires some work

However: The docker images are a good starting point.

Supported Scanner

- Arachni Scanner
- AppSpider (Rapid7)
- Bandit
- Burp XML
- Contrast Scanner
- Checkmarx
- Dependency Check
- Generic Findings Import - CSV format
- Nessus (Tenable)
- Nexpose XML 2.0 (Rapid7)
- Nikto
- Nmap
- Node Security Platform
- OpenVAS CSV
- Qualys
- Retire.js
- SKF Scan
- Snyk
- SSL Labs
- Trufflehog
- Visual Code Grepper (VCG)
- Veracode
- Zed Attack Proxy

https://defectdojo.readthedocs.io/en/latest/integrations.html

WRAP UP

Lessons learned 1/2

- Don’t underestimate the total effort!
  - Although first steps are fairly easy (esp. with Docker), the full setup including processes takes time
- Tests are important, esp. JIRA integration is tricky
- Feels overengineered, basic features missing
- Data model seems to be too ambitious
- Core team is quite responsive (Github, Slack), but has an own view on how to use DefectDojo
- Documentation somewhat dated, it does not keep up with to current development speed

Lessons learned 2/2

- Needs a lot of glue code to integrate into existing infrastructure (inventory, issue tracking)
- API – missing methods e.g. add metadata, add tags, …
- API is complicated (eg. query by product id, which has to be searched first)
- Operational challenge updates, stability
- User experience odd at times – no cancel buttons
- JIRA Webhooks

Not figured out, yet ;)

- Usage of Tags vs. Product-Type vs. Metadata
- Leading system for URLs/Endpoints/Application
  - DefectDojo
  - Asset-Management System
  - Links between systems
- Combining AppSec and NetSec vulnerability data
  - AppSec – web-applications
    - Output DAST, SAST
  - NetSec – IP-addresses
    - Output Nessus, OpenVAS, Qualys, …
- Reviewing fix of vulnerabilities/ automation – manual review needed

Future

- Active project, with many new ideas
- A new API implementation based upon Django's Rest Framework (https://github.com/DefectDojo/django-DefectDojo/pull/566) -> merged
- Add Meta Data / Additional Information to API (https://github.com/DefectDojo/django-DefectDojo/issues/459)
- Add to the API (https://github.com/DefectDojo/django-DefectDojo/issues/457)
- Sponsoring possible for support of product and enhancements
- Enhacements as part of OWASP Security Summit planned

Thanks for your Attention!

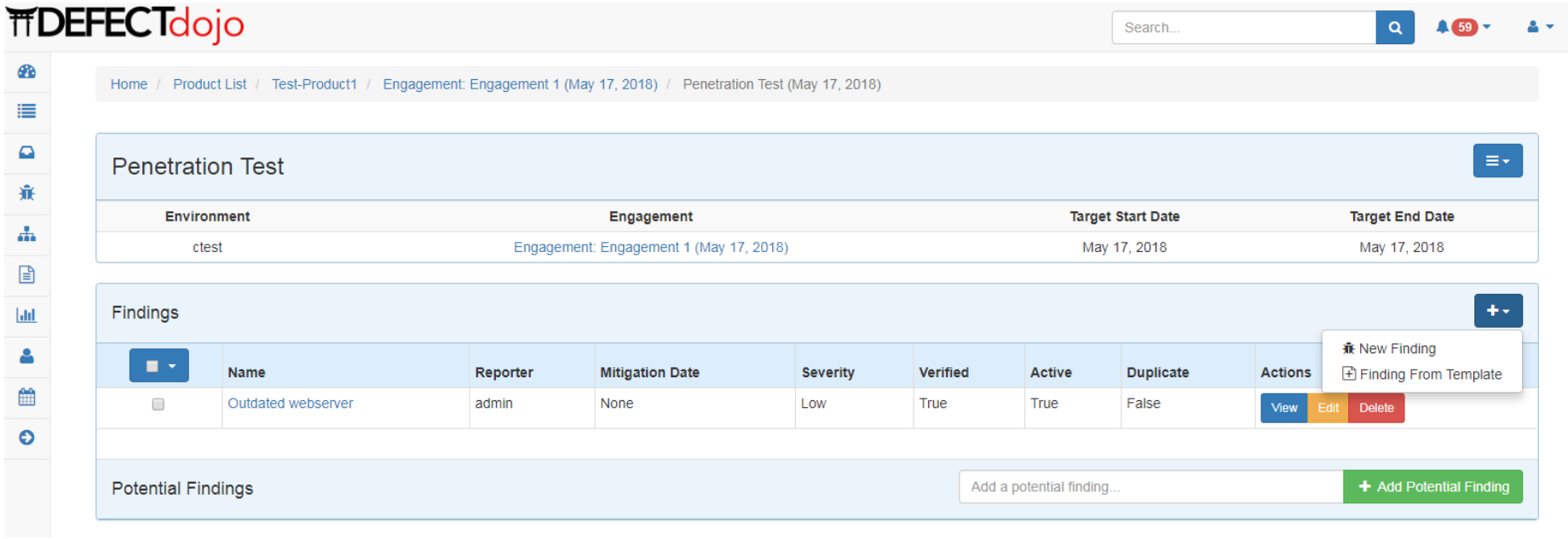
If there are any questions, comments, ideas – it’s your time now.

Links

- https://github.com/DefectDojo/django-DefectDojo
- https://www.denimgroup.com/resources/blog/2016/07/whats-in-a-name-why-gartner-picking-application-vulnerability-correlation-is-an-important-step-for-the-application-security-market/
- https://codedx.com/2017/11/08/gartner-identifies-the-next-step-in-software-vulnerability-management-application-vulnerability-correlation-avc/
- https://www.owasp.org/index.php/OWASP_AppSec_Pipeline#tab=Pipeline_Design_Patterns

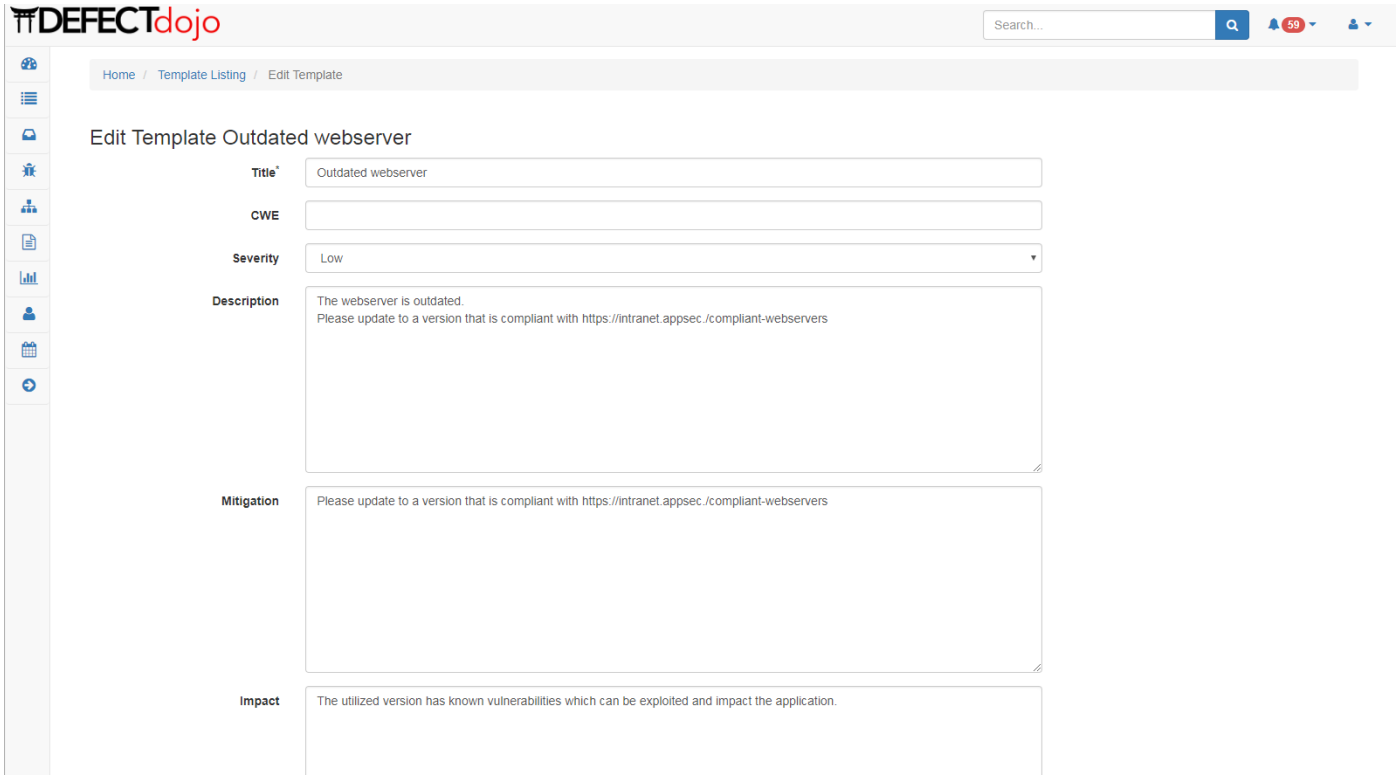
BACKUP

Manual creation of Finding



Templates

- Templates can be used for manual creation of vulnerabilties
- Links to policies, secure coding guideline, etc. can be utilized
- Standard texts for „standard“ vulnerabilities eg. XSS, Injection, …



Manual upload of Reports

- => Pain
- Demo -> Manuelles erzeugen von Engagement

- Scripting/ automation for the win