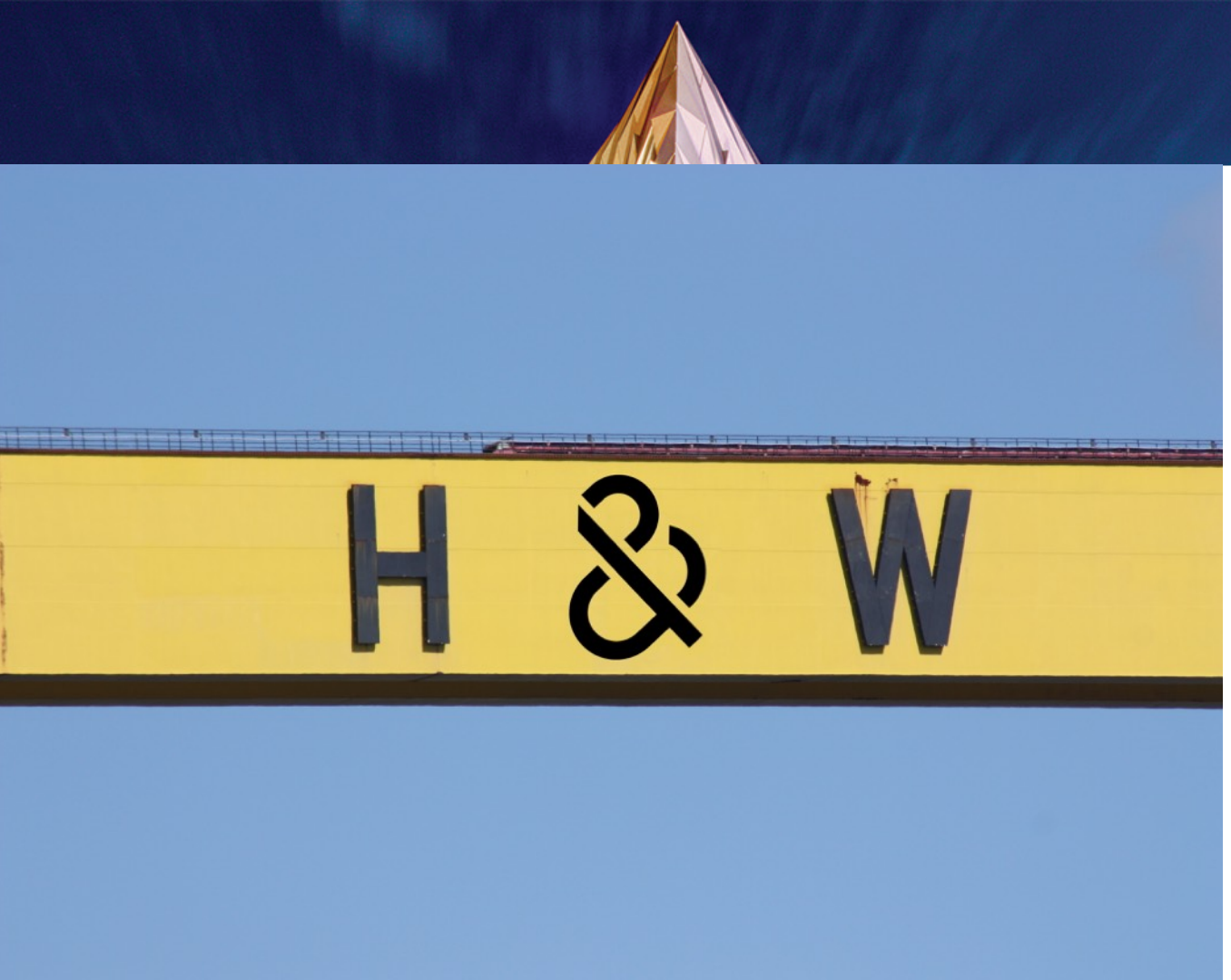

OWASP
AppSec EU
Belfast
8th to 12th
of May
2017
Waterfront
Conference
Center

The Rise of the Machines

Making AppSec Great Again

(I'm sorry, they made me do this)



OWASP AppSec EU Belfast

dun & bradstreet

- Rohini Sulatycki
- Kevin D'Arcy
- Nick Raite

Background

- Dun & Bradstreet is a 176 year old company
- One of the first companies to be publicly traded on the New York Stock Exchange
- Sells data on 240 million companies worldwide
- Lots of old systems
  - Cobol
  - Classic ASP etc.
- Lots of new systems
  - Java
  - AWS

- Outsourced development
- No consistent development methodology
- Insourcing of key systems, control and knowledge
- Security function in-sourced over last 4 years
- Application Security team founded in mid-2015


Faced with 100s
of applications


With a small team of 5
pentesters and 1 developer

Other Responsibilities:

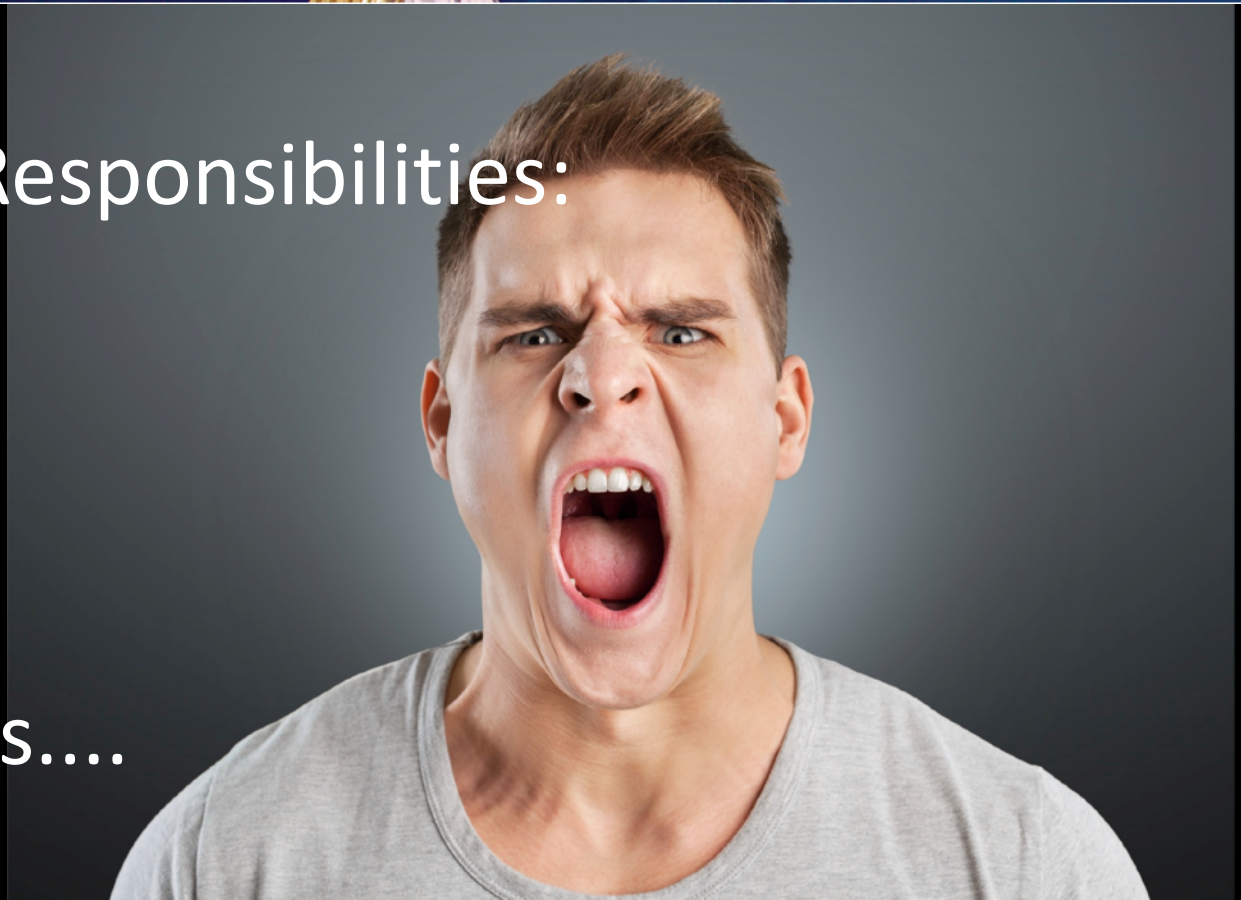
Audits

PCI

SOX

RFIs

Updates....



Anyone else in the same situation?

What if we could use "machines" to automate our dynamic and static scanning?

This is where we started building an automation framework

au·to·ma·tion
ȯ-tə-ˈmā-shən
Noun

1. the technique of making an apparatus, a process, or a system operate automatically

2. the state of being operated automatically

3. automatically controlled operation of an apparatus, process, or system by mechanical or electronic devices that take the place of human labor

- Continuous Scanning
- Build Integration
- Continuous Protection

- **Dashboards**
- **Metrics**
- **Vulnerability Patterns**

  SQLI?

  Weak Passwords?

  Default/No Passwords?

  Patching?

Integrate Toolset

Initially:

- The team started doing manual penetration testing
  - Burp
  - ZAP
  - SQLMap
  - nmap
- All scan data was held in reports (PDFs, XML etc.)

Dashboard:

- Collation of statistics into management dashboard
  - ELK - ElasticSearch, Logstash, Kibana
  - Custom coded
  - Splunk


JIRA

1. Use the same technology that developers use!
2. Create reporting dashboards in JIRA
3. Send all AppSec tickets to JIRA

Initial Steps

- Data centre vs Cloud vs Lab
    - Cost, speed of development, level of access
- Creation of Jenkins jobs to automate static scans
    - FoD plugin didn’t support proxy access
    - Preference for on-premise solution
- Automation of dynamic scans
    - Burp doesn’t lend well to automation
    - ZAP has issues with authenticated scans

Iteration 1

- Goal
    - Have all detailed scan results in one place
- Tools in scope
    - Jenkins, Threadfix, FoD, Jira
- Environment
    - Dublin office test lab

Iteration 1

- Results
    - Threadfix had issues with FPRs from FoD
    - CSVs considered as alternative
    - Cigital XML -> SSVL - > Threadfix
- Learnings
    - Lab was the wrong environment, too many access issues
    - Despite issues, Threadfix worked well
    - Too many tools in scope, concentrate on quick wins

Iteration 2

- Goal
    - Migrate to AWS environment
- Tools
    - Jenkins, Threadfix, Jira, HP SCA & SSC, Bag of Holding
- Environment
    - Development AWS account managed from Dublin

- Bag of Holding
  - Application Registry
  - Tech Stack
  - App Owners
  - Security Testing
  - Compliance
  - Data Classification

Overview | Engagements | Environments | People | Settings

Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goals are to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and aid teachers/students to teach/learn web application security in a class room environment.

Service Level Agreements (0)

There are no service level agreements.

Technologies (2)

MySQL	Data Store
PHP	Language

Regulations (0)

There are no regulations.

Custom Fields (0)

Name	Value

There are no custom field values.

Metadata

Business Criticality	Low
Platform	Web
Lifecycle	Sustain
Origin	Open Source
User Records	None
Revenue (USD)	None

OWASP ASVS

Not Assessed

Resources

Tags

There are no tags.

Iteration 2

- Results
  - AWS environment quick to set up and deploy to
  - Good access to external network
  - But AWS account used was not the correct one: no Jira access, also limited access to internal network
- Learnings
  - AWS was the right environment, but needed to be in hybrid cloud environment rather than development cloud
  - BoH good repository for app info and interactions
  - Don’t use spot instances :)
  - Document, document, document: you’ll regret it later when you try to remember how to installed/configured something

Iteration 3

- Goal
  - Migrate to correct AWS environment
- Tools
  - Jenkins, Jira, HP SCA & SSC, Bag of Holding, VulnReport, Slack
- Environment
  - Hybrid cloud AWS account managed from US

- Vulnreport
  - Open Sourced By SalesForce
  - Ruby on Rails
  - PostgresSQL
  - Saved us a LOT of time in report creation

Security Assessment for Test 4

The below report outlines vulnerability and informational findings that were discovered via a mixture of automated scanning and manual security penetration testing (application/network/data). It is possible that there are other vulnerabilities that were not discovered during testing and are therefore not included on this report. Not all instances of each vulnerability type were enumerated. These issues have been identified, rated based on a risk model, and require remediation within a timeframe corresponding with their risk. This report is CONFIDENTIAL and may only be distributed to application teams who have a need-to-know in order to fix identified issues.

The attack vector and evidence of vulnerability associated with each item is listed with a description of the vulnerability. Suggested remediation steps and additional educational resources for each class of vulnerability are also provided.

Please note that this is not a comprehensive list of vulnerabilities in your application. Similiar and additional vulnerabilites outside of this report may exist. Please look through your entire codebase for additional security issues.

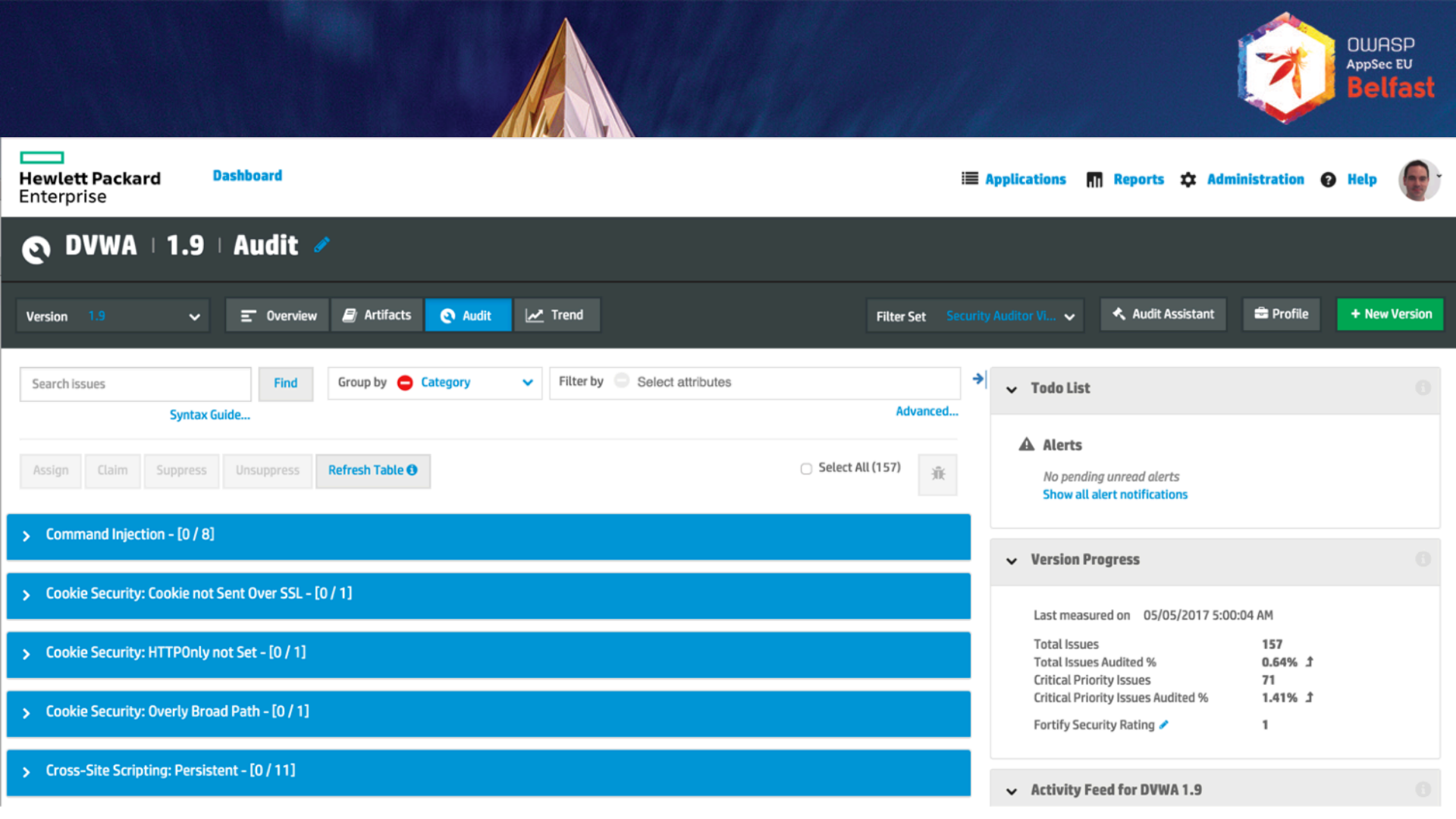
SLAs

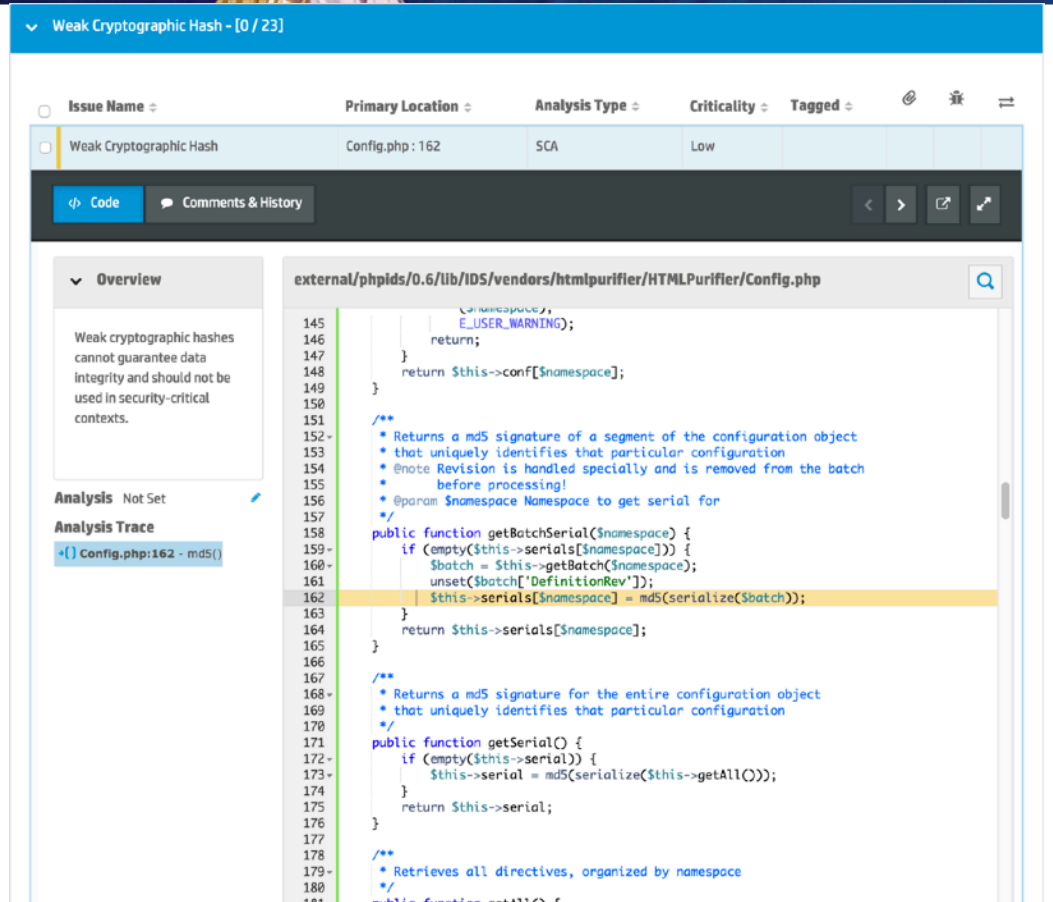
SEVERITY AND REMEDIATION SCHEDULE			
CRITICAL	HIGH	MEDIUM	LOW
30 DAY	60 DAYS	90 DAYS	1 Year or Next Major Version/Release

Contents

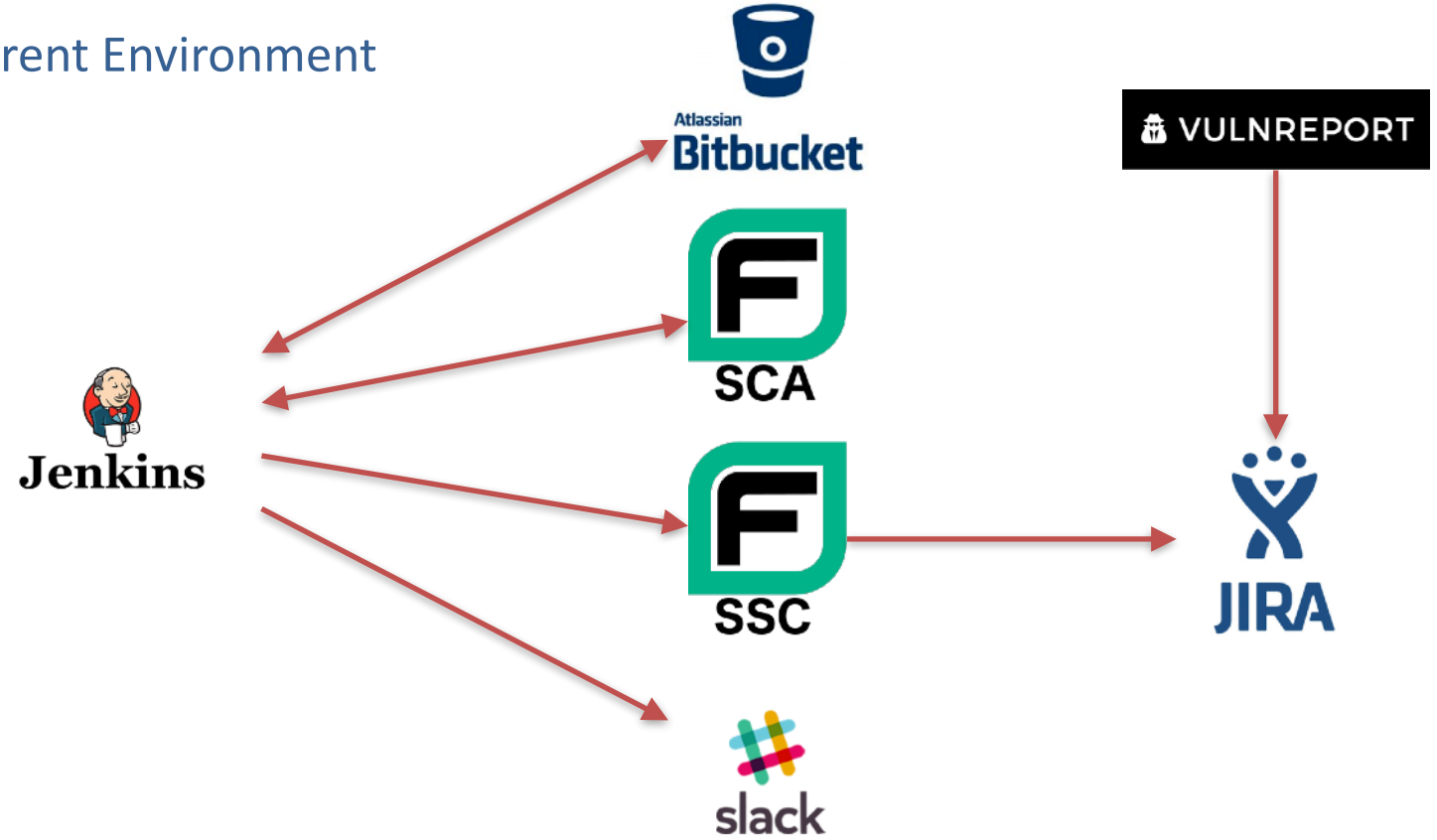
Iteration 3

- Results
  - Finally in correct AWS environment
  - Static scanning environment ramped up quickly (approx. 50 apps in 4 months)
  - Limited access to internal network (SSH, HTTP/S)
  - Stabilised and secure toolset (eat our own dog food)
- Learnings
  - Having admin access to EC2 instances is crucial to velocity
  - Integration with SSO is useful, but not important
  - Snapshot instances BEFORE you make changes
  - Give development teams access to as much info as possible
  - Grow the environment incrementally rather than doing too much too soon


OWASP
AppSec EU
Belfast
Hewlett Packard
Enterprise
Dashboard
Applications
Reports
Administration
Help
DVWA | 1.9 | Audit
Version 1.9
Overview
Artifacts
Audit
Trend
Filter Set Security Auditor Vi...
Audit Assistant
Profile
+ New Version
Search issues
Find
Syntax Guide...
Group by Category
Filter by Select attributes
Advanced...
Assign
Claim
Suppress
Unsuppress
Refresh Table
Select All (157)
Command Injection - [0 / 8]
Cookie Security: Cookie not Sent Over SSL - [0 / 1]
Cookie Security: HTTPOnly not Set - [0 / 1]
Cookie Security: Overly Broad Path - [0 / 1]
Cross-Site Scripting: Persistent - [0 / 11]
Todo List
Alerts
No pending unread alerts
Show all alert notifications
Version Progress
Last measured on 05/05/2017 5:00:04 AM
Total Issues 157
Total Issues Audited % 0.64%
Critical Priority Issues 71
Critical Priority Issues Audited % 1.41%
Fortify Security Rating 1
Activity Feed for DVWA 1.9


Weak Cryptographic Hash - [0 / 23]
Issue Name
Primary Location
Analysis Type
Criticality
Tagged
Weak Cryptographic Hash
Config.php : 162
SCA
Low
Code
Comments & History
Overview
Weak cryptographic hashes cannot guarantee data integrity and should not be used in security-critical contexts.
Analysis Not Set
Analysis Trace
Config.php:162 - md5()
external/phpids/0.6/lib/IDS/vendors/htmlpurifier/HTMLPurifier/Config.php
E_USER_WARNING);
return;
}
return $this->conf[$namespace];
}
/**
* Returns a md5 signature of a segment of the configuration object
* that uniquely identifies that particular configuration
* @note Revision is handled specially and is removed from the batch
* before processing!
* @param $namespace Namespace to get serial for
*/
public function getBatchSerial($namespace) {
if (empty($this->serials[$namespace])) {
$batch = $this->getBatch($namespace);
unset($batch['DefinitionRev']);
$this->serials[$namespace] = md5(serialize($batch));
}
return $this->serials[$namespace];
}
/**
* Returns a md5 signature for the entire configuration object
* that uniquely identifies that particular configuration
*/
public function getSerial() {
if (empty($this->serial)) {
$this->serial = md5(serialize($this->getAll()));
}
return $this->serial;
}
/**
* Retrieves all directives, organized by namespace
*/

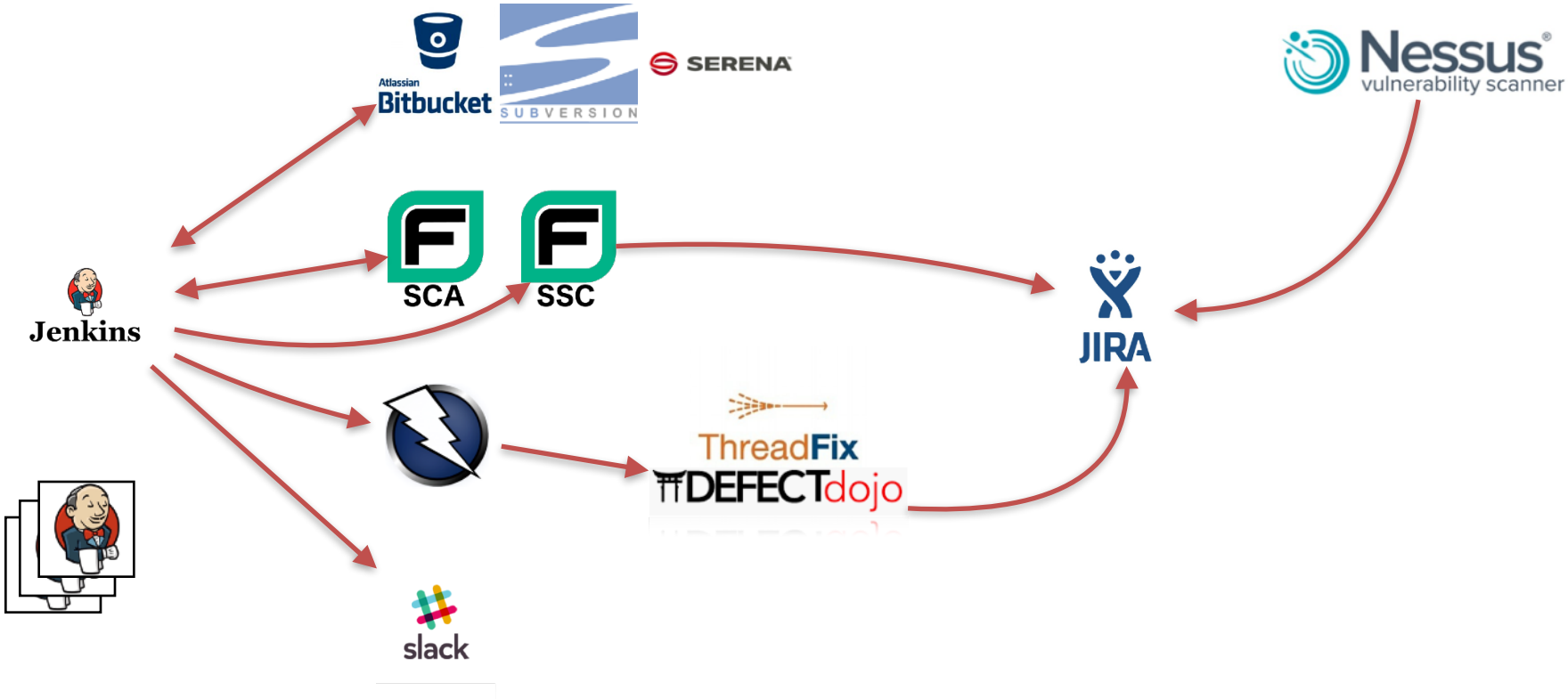
Current Environment



Iteration 4

- Include dynamic scanning automation
    - Start with unauthenticated scans
    - Resolve ZAP authentication issues
    - Choose repo tool
        - Threadfix or DefectDojo
- Gain internal network access
    - Jenkins slaves on internal network
- Build integration

Iteration 4


Atlassian Bitbucket
SUBVERSION
SERENA
Nessus vulnerability scanner
SCA
SSC
Jenkins
JIRA
ThreadFix
DEFECTdojo
slack

What did we do about the volume of issues being found?


GAMIFICATION
90%

We are trying to "gamify" :

- Learning
- Validations
- Installs
- Remediations

Lessons Learned

- Be agile - iterations are good!
- Choose your tools carefully but don’t be afraid to experiment
- Consider open-source
- Have a good relationship with Ops and Engineering
- Watch for icebergs ahead..they will sink you!

Demo

Questions?