

Security DevOps

staying secure in agile projects

Christian Schneider

@cschneider4711



`whoami` www. } Christian-Schneider.net
mail@

- » Software Developer, Whitehat Hacker & Trainer
- » Freelancer since 1997
- » Focus on JavaEE & Web Security
- » Speaker at Conferences
- » @cschneider4711



Why Security DevOps?

- » Keep up with rollout pace in agile projects
- » Automate certain checks as best as possible within the build-chain
 - » Early feedback to Devs
- » Does **not** remove the pentest requirement!
 - » Aims to free pentesters' time to hunt more high-hanging bugs



Different levels of "Security DevOps" integration...

- » Security DevOps Maturity Model (SDOMM)
- » Can be seen as some automation tips within OpenSAMM's security practices
 - » Verification: Security Testing
 - » Verification: Code Reviews
- » Allows to define a RoadMap for projects implementing Security DevOps



... what levels will we cover?



implicit

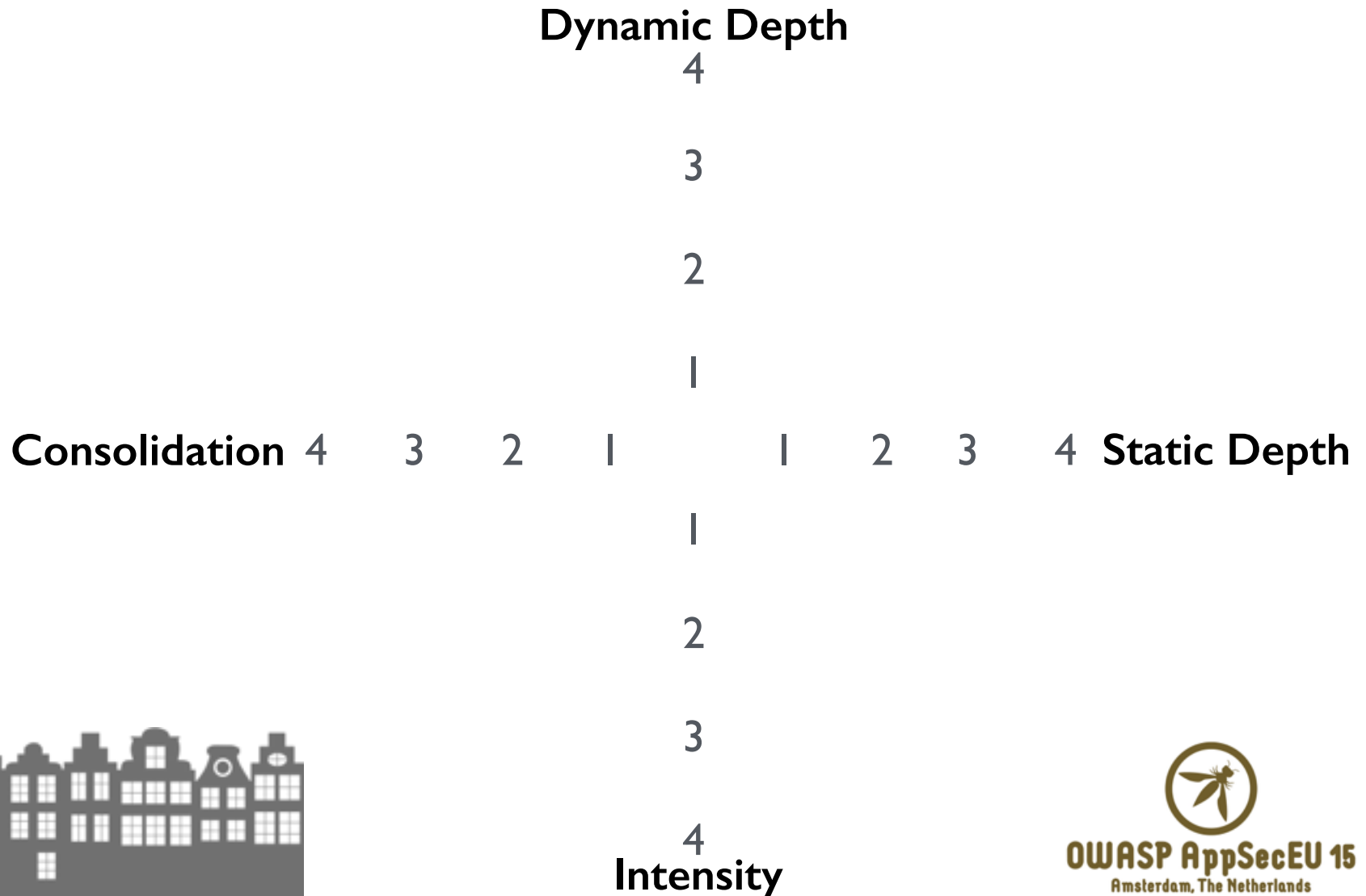
*four axes each with four
belts as incremental steps*

master

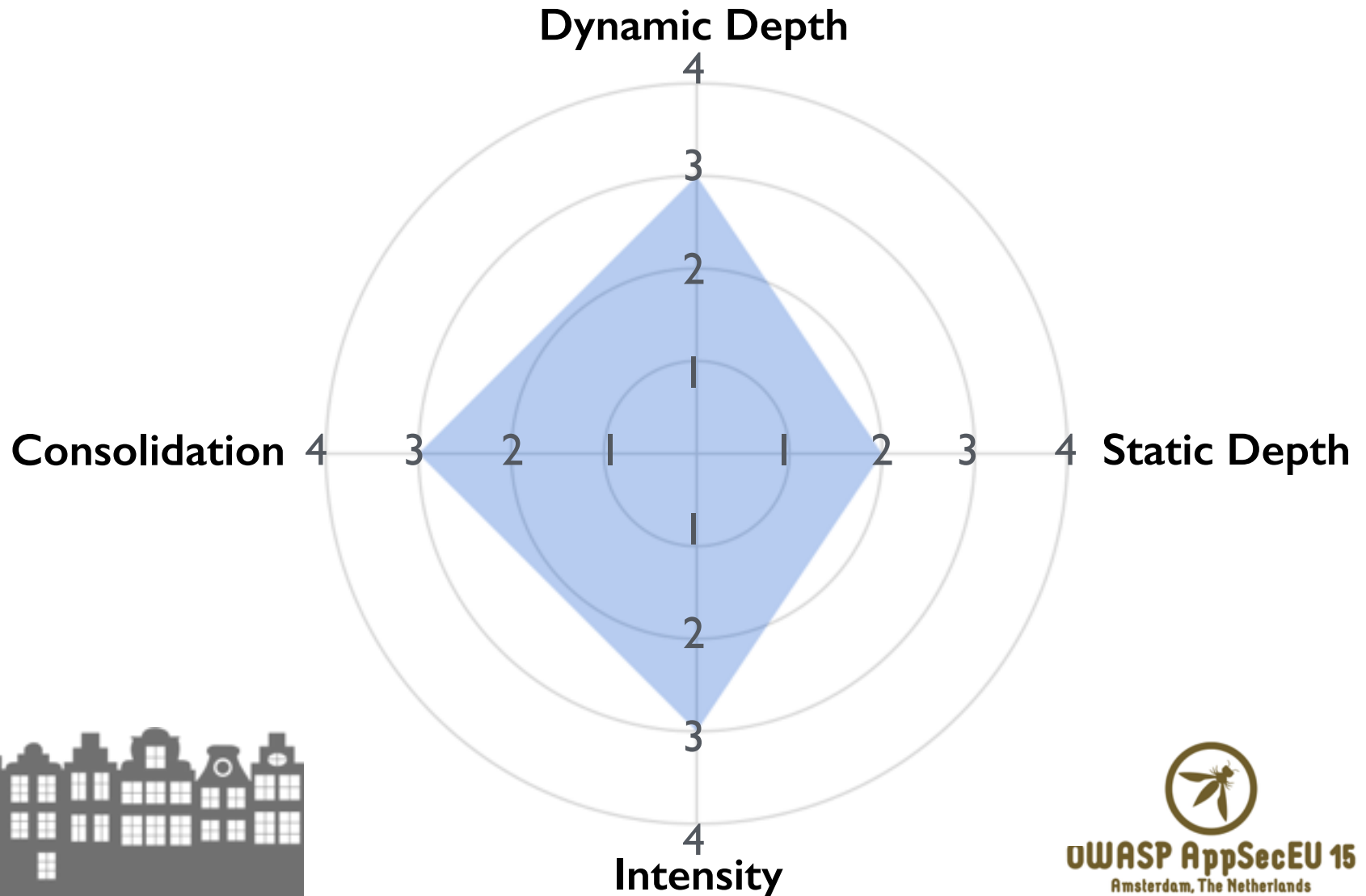


OWASP AppSecEU 15
Amsterdam, The Netherlands

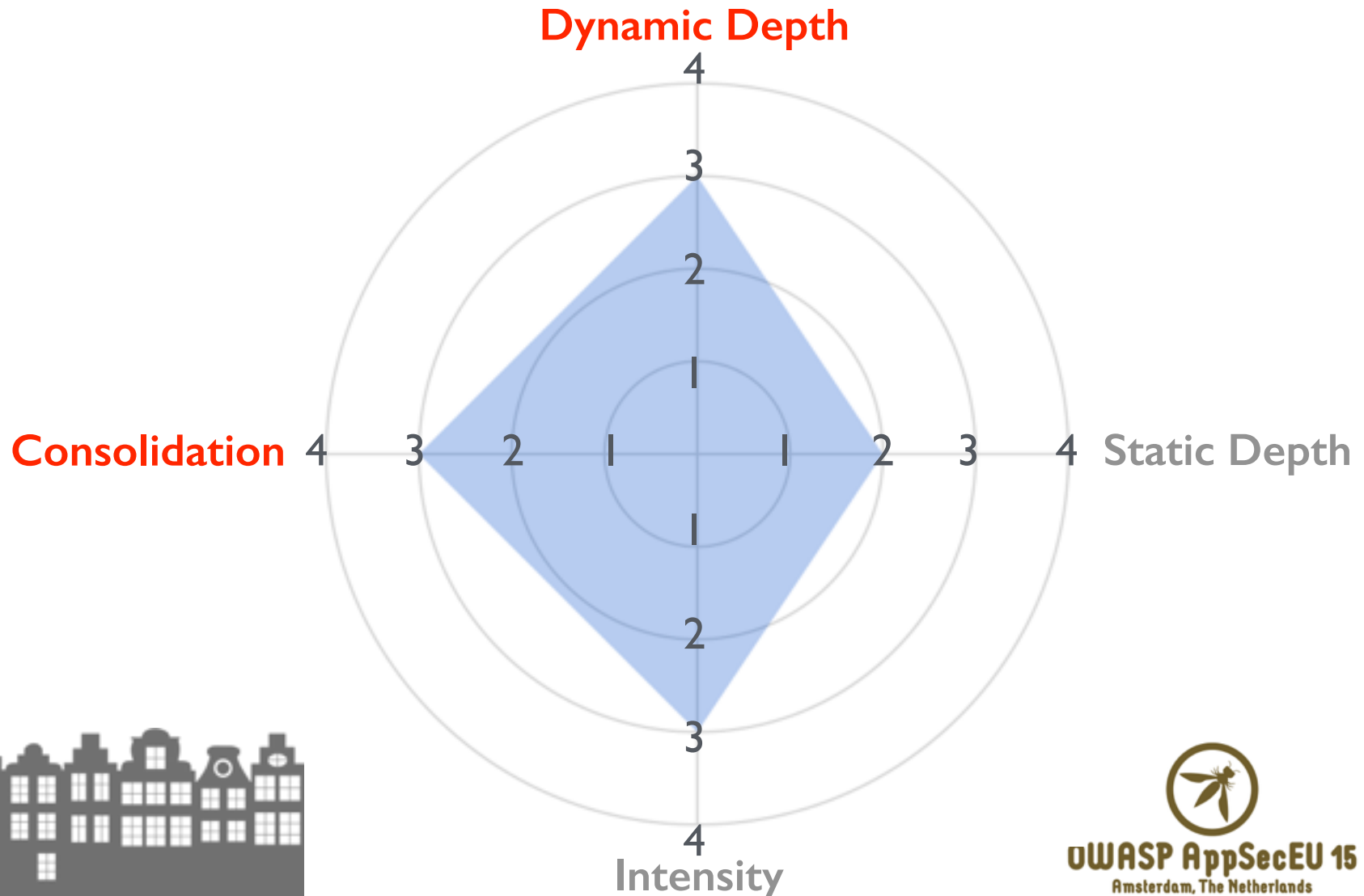
Four different axes



Four different axes



This talk covers **two** of them



Let's explore these axes ...

- » ... by showing how to implement this with OpenSource solutions used in the Security & Development domains.



Axis of "Dynamic Depth"



How deep are dynamic scans executed within a Security DevOps CI chain?

i.e. "where" are dynamic security tests applied?



OWASP AppSecEU 15
Amsterdam, The Netherlands

Axis "Dynamic Depth": Level I

Scanning of **public attack surface** (pre-auth):

- Spidering of UI layer
- No requirement to authenticate scanner with target
- Easy integration of scanner(s) in nightly build as post-step
- *"Throw tool at it (in CI-chain) and see what it generates..."*



OWASP AppSecEU 15
Amsterdam, The Netherlands

ZAP in SecDevOps?

"OWASP ZAP" features relevant for Security DevOps integration:

- Passive & active scanning
- Headless operation mode / daemon
- REST-API (with several language bindings as pre-built clients)
- Scriptable
- CLI



OWASP AppSecEU 15
Amsterdam, The Netherlands

ZAP + Jenkins = SecDevOps?

"OWASP ZAP" (spider & scanner) + Jenkins plugin "ZAPProxy"

- Allows us to "Spider & Scan" as step in build job via Jenkins plugin
- Point plugin config to URL of integration system to test
- Plugin saves HTML-report in project's job for inspection
- Best as separate Jenkins job to run during nightly build (duration)
 - Use different ZAP proxy ports for different builds to allow parallel execution of different project build jobs



Post Steps

☐ Run only if build succeeds ☐ Run only if build succeeds or is unstable

Should the post-build steps run only for successful builds, etc.

Execute ZAProxy

Admin configuration

Workspace used	C:\csr\@Security\@Tools\Jenkins\jobs\marathon\workspace
ZAProxy host	localhost (Configured in admin mode)
ZAProxy port	8500 (Configured in admin mode)

Startup

☒ Start ZAProxy in a pre-build step

Timeout for ZAProxy initialization

Enter a value in seconds

☒ ZAProxy is installed by Jenkins

Tool to use

☐ ZAProxy is already installed

Jenkins Plugin "ZAProxy": ZAP Startup

Setup

Load session

Target URL

☒ Spider URL

☒ Scan URL

ZAPProxy default directory

Choose policy to use

☒ Generate report

Choose format report

Filename for report

☒ Save session

Filename for session

Jenkins Plugin "ZAPProxy": ZAP Scan

Arachni in SecDevOps?

"Arachni Scanner" features relevant for Security DevOps integration:

- Passive & active scanning (Proxy / Spider)
- Uses internally a headless browser-cluster (for apps with lots of JS)
- Automation?
 - CLI + RPC API
- Web-UI (helpful when permanently running as server)



OWASP AppSecEU 15
Amsterdam, The Netherlands

Arachni + Jenkins = SecDevOps?

"Arachni Scanner" + Jenkins CLI step in build

- Start in build job as CLI step and point to URL of system under test
- Generate HTML report and place into workspace for inspection
- Better execute within nightly build job (due to duration)



OWASP AppSecEU 15
Amsterdam, The Netherlands

BDD-Security in SecDevOps?

BDD-based framework for functional and technical security tests:

- Technical security tests (i.e. check against XSS, SQL-Injection, XXE, etc.)
 - uses ZAP as scanning engine (among others)
- Functional security tests (i.e. check UserA can't access data from UserB)
- Tightly integrates with Selenium based app navigation workflows
- Uses JBehave for G/W/T stories & reporting
- Can run within CI (Jenkins, etc.) due to JBehave or as JUnit tests



```
Scenario: The application should not contain Cross Site Scripting vulnerabilities
Meta: @id scan_xss
Given a fresh scanner with all policies disabled
And the attack strength is set to High
And the Cross-Site-Scripting policy is enabled
When the scanner is run
And false positives described in: tables/false_positives.table are removed
Then no Medium or higher risk vulnerabilities should be present
```

BDD-Security Story: Scan for XSS

GauntIt in SecDevOps?

BDD-based framework for executing many security tools/scanners:

- Integrates scanners like Arachni, ZAP, sqlmap, etc.
 - Easy to integrate "your custom scanner tool" with GauntIt as well
- Allows to call different scan policies via BDD-stories (G/W/T)
- Integration with Jenkins (or other build servers) by either
 - Linking GauntIt's HTML report to build, or by
 - modifying how GauntIt calls Cucumber to produce JUnit output



Axis "Dynamic Depth": Level 2

Scanning of **authenticated** parts (= "post-auth") via UI layer

- Properly maintaining sessions
- Logout-detection & automatic re-login
- Different users / roles
- Spider & scan post-auth

Handling of hardening measures of application under test

- CSRF-Tokens, etc.



OWASP AppSecEU 15
Amsterdam, The Netherlands

Guide ZAP into Post-Auth in CI

Use ZAP manually (1x) to configure "Context":Auth, RegExps for Logged-In/Out Indicators, Users etc. + save as "ZAP Session-File" (could be in code repo)

- use that "Session-File" from code repo as starting point of scan (loaded as ZAP session during build job).

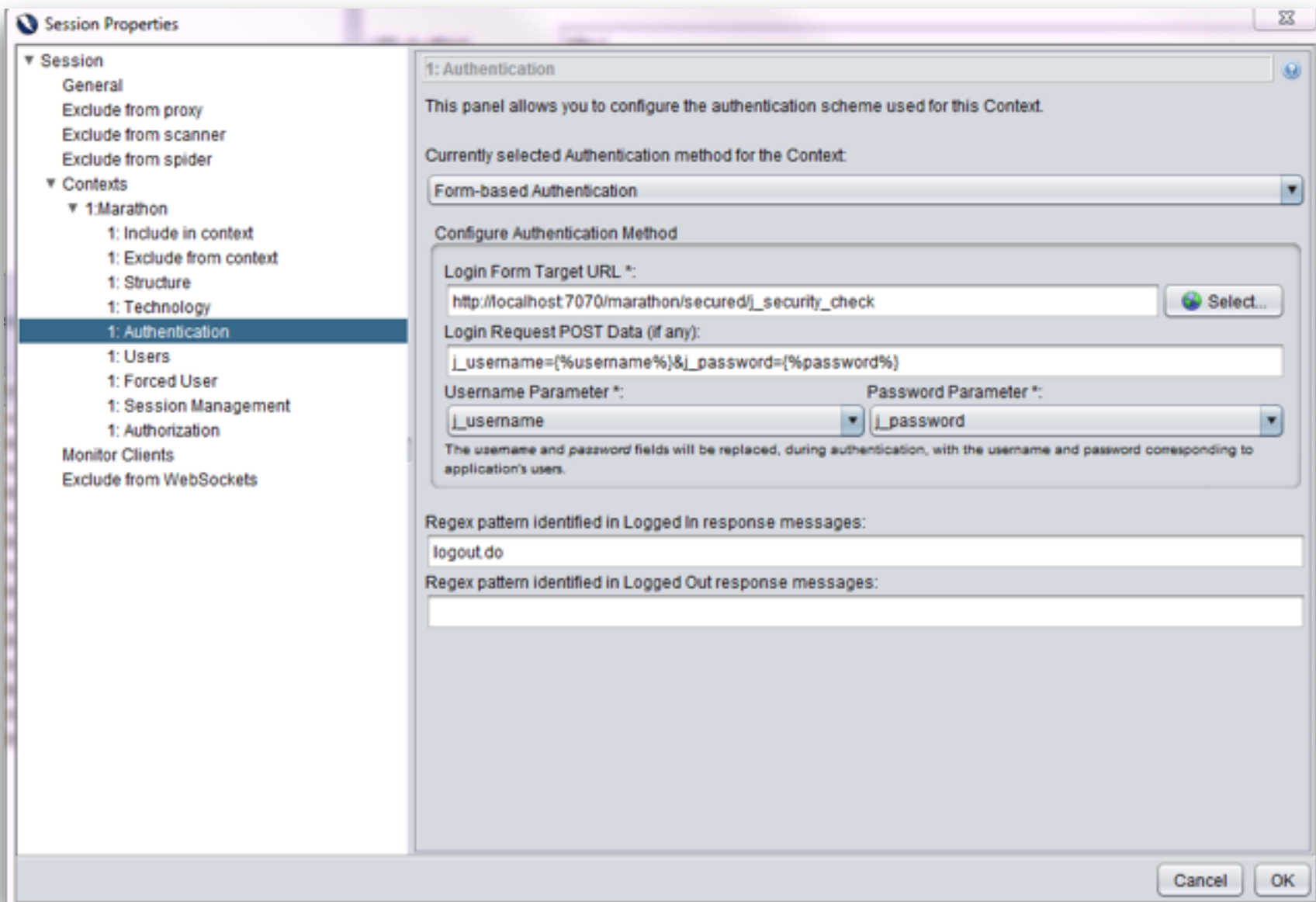
Note: Current version of ZAP has a bugfix pending for loading creds from session file

One can set these auth values and/or additional data via ZAP's REST-API during each build before scan starts (from Jenkins/Maven/...)

- use that to define current active session etc. during scan

Also Scripts in JavaScript or Zest can be registered in ZAP context to programmatically give authentication to ZAP





Login config example within ZAP

Setup

Load session	preseedsZAP\preseeding.session
Target URL	http://localhost:7070/marathon/
<input checked="" type="checkbox"/> Spider URL	
<input checked="" type="checkbox"/> Scan URL	
ZAProxy default directory	
Choose policy to use	
<input checked="" type="checkbox"/> Generate report	
Choose format report	<div>xml</div> <div>html</div>
Filename for report	reportsZAP/report

ZAProxy Jenkins Plugin: ZAP session use

Guide Arachni into Post-Auth

Give authentication infos to Arachni (Auth, Logged-In Indicators, Users)

- Use Arachni "autologin" plugin to specify via command line
 - Login URL, formfield names, credentials, logged-in indicator, excludes
- Alternatively write custom ruby script for "login_script" plugin
 - Individual custom login logic possible
 - Logged-In indicators (RegExp) to know when to re-login



OWASP AppSecEU 15
Amsterdam, The Netherlands

Login config example within Arachni (used in CI)

```
./arachni
--plugin=autologin:
    url=https://example.com/login.action,
    parameters='j_username=foo&j_password=bar',
    check='Logout'
--scope-exclude-pattern=logout.action
https://example.com/
```

Eventually also --session-check-url & --session-check-pattern

Or individual ruby script if more custom login logic required...



OWASP AppSecEU 15
Amsterdam, The Netherlands

Guide BDD-Security into Post-Auth

Use Selenium to navigate through the login process

- Based on excellent integration of BDD-Security with Selenium
- Separate app navigation code (Selenium) from Security testing code
- Use Selenium class (that handles login) within BDD stories
- Perform further spidering & active scanning (through ZAP) post-auth



OWASP AppSecEU 15
Amsterdam, The Netherlands

```
public class ShopApplicationScanHelper
    extends WebApplication implements ILogin {

    // ... integrates with BDD-Security via parent class & interface ...

}
```

```
public class ShopApplicationScanHelper
    extends WebApplication implements ILogin {
```

```
@Override
```

```
public void openLoginPage() {
```

```
}
```

```
@Override
```

```
public void login(Credentials credentials) {
```

```
}
```

```
@Override
```

```
public boolean isLoggedIn(String role) {
```

```
}
```



```
public class ShopApplicationScanHelper
    extends WebApplication implements ILogin {
```

```
@Override
```

```
public void openLoginPage() {
    driver.get(Config.getInstance().getBaseUrl() + "customer/login");
    verifyTextPresent("Login");
}
```

```
@Override
```

```
public void login(Credentials credentials) {
    UserPassCredentials creds = new UserPassCredentials(credentials);
    driver.findElement(By.id("username")).clear();
    driver.findElement(By.id("username")).sendKeys(creds.getUsername());
    driver.findElement(By.id("password")).clear();
    driver.findElement(By.id("password")).sendKeys(creds.getPassword());
    driver.findElement(By.name("_action_login")).click();
}
```

```
@Override
```

```
public boolean isLoggedIn(String role) {
    if (driver.getPageSource().contains("My Account")) {
        return true;
    } else {
        return false;
    }
}
```

Axis "Dynamic Depth": Level 3

Separate scanning of different application layers / backends

- Scan internal WebServices (e.g. SOAP / REST) = directly scan backends
- Detect and scan parameter positions within XML, JSON, ...
- Scan from "within" the different application's layers
 - IAST with distributed agents & instrumentation aims into that direction
- At least one simple step in that direction:
 - Use the proxy also between your backend service calls



Backend scans with ZAP

How to achieve this with ZAP?

- ZAP operates as proxy server: place it **between** backend calls
- ZAP can inject payloads in observed **XML** tags/attributes & **JSON** fields
- Capture service call traffic in integration test during CI while either
 - A. executing service tests that directly access the service endpoint, or
 - B. frontend UI tests execute service backend calls indirectly
- Automatically scan as new requests are seen: "**ATTACK Mode**"

Also keep an eye on an alpha-level SOAP-Scanner ZAP addon



OWASP AppSecEU 15
Amsterdam, The Netherlands

Backend scans with Arachni

How to achieve this with Arachni?

- Arachni can also operate as proxy: place it **between** backend calls
- Use passive proxy plugin to "train" Arachni of the XML / JSON requests
 - New addition in v1.1 to extract XML / JSON input vectors from it
- Use that collected input vector data to feed the active scan for the observed requests



OWASP AppSecEU 15
Amsterdam, The Netherlands

Axis "Dynamic Depth": Level 4

Targeted scanning of individual forms / wizards (UI) and service layers

- More individualised workflow coverage (not just simple spidering)
- Business-logic compliant usage patterns & inputs
 - *"fill shopping cart followed by checkout process"*
 - *"access backend WebServices in special order to test workflow", etc.*
- Custom coded security tests tailored to the application



ZAP with special workflows (1/3)

Many ways exist...

The simplest one could be:

Re-use existing UI tests (Selenium, ...)

- Proxy this traffic through ZAP in "ATTACK-Mode" (in security test phase of build)
- Optionally use ZAP Attack-Policies to specify/limit certain attack types



ZAP with special workflows (2/3)

A more customised handling of individual workflows can be achieved:

Re-use & enhance existing "UI test code" at the desired workflow steps with calls to ZAP's (REST)-API ordering attacks

- Basically it's like Unit-Test code that uses Selenium along with with ZAP-Calls at the proper positions in application workflow
- Type of "ordered attacks" can again be defined via policies
- Start ZAP as Daemon from Jenkins via plugin




```
public class ShopApplicationTest { // = regular JUnit unit test

    @Before
    public void setup() {

    }

    @Test
    public void testShippingAddressStep() {

    }

    @Test
    public void testBillingAddressStep() {

    }

}
```

```
public class ShopApplicationTest { // = regular JUnit unit test

@Before
public void setup() {
    // 1. start new proxy session in running ZAP (via REST-API call)
    // 2. create Selenium driver (proxying through running ZAP)
}

@Test
public void testShippingAddressStep() {

}

@Test
public void testBillingAddressStep() {

}

}
```

```
public class ShopApplicationTest { // = regular JUnit unit test

    @Before
    public void setup() {
        // 1. start new proxy session in running ZAP (via REST-API call)
        // 2. create Selenium driver (proxying through running ZAP)
    }

    @Test
    public void testShippingAddressStep() {
        // 1. use Selenium to fill shopping cart
        // 2. use Selenium to proceed to checkout
        // 3. use Selenium to provide reasonable shipping address data

    }

    @Test
    public void testBillingAddressStep() {

    }

}
```

```
public class ShopApplicationTest { // = regular JUnit unit test

    @Before
    public void setup() {
        // 1. start new proxy session in running ZAP (via REST-API call)
        // 2. create Selenium driver (proxying through running ZAP)
    }

    @Test
    public void testShippingAddressStep() {
        // 1. use Selenium to fill shopping cart
        // 2. use Selenium to proceed to checkout
        // 3. use Selenium to provide reasonable shipping address data
        // 4. set attack policy (types & strength) in running ZAP (API)
        /* 5. call ZAP (API) to actively scan the last seen URL
           (optionally define parameter excludes via API
            or ZAP "input vector scripts" if custom input format) */
    }

    @Test
    public void testBillingAddressStep() {

    }

}
```

```

public class ShopApplicationTest { // = regular JUnit unit test

    @Before
    public void setup() {
        // 1. start new proxy session in running ZAP (via REST-API call)
        // 2. create Selenium driver (proxying through running ZAP)
    }

    @Test
    public void testShippingAddressStep() {
        // 1. use Selenium to fill shopping cart
        // 2. use Selenium to proceed to checkout
        // 3. use Selenium to provide reasonable shipping address data
        // 4. set attack policy (types & strength) in running ZAP (API)
        /* 5. call ZAP (API) to actively scan the last seen URL
           (optionally define parameter excludes via API
            or ZAP "input vector scripts" if custom input format) */
    }

    @Test
    public void testBillingAddressStep() {
        // same idea as above ... just continue with the pattern
    }

}

```

See <https://github.com/continuumsecurity/zap-webdriver>
for a great working example of Selenium ZAP integration

ZAP with special workflows (3/3)

Alternatively "train" ZAP about the workflow by recording Zest scripts

- Keep an eye on **"Sequence Scanning"** alpha-level ZAP addon
 - Still alpha-level (as of May 2015), but interesting approach



OWASP AppSecEU 15
Amsterdam, The Netherlands

BDD with special workflows

Use Selenium to further drive BDD-Security initiated checks:

- Selenium-based test code navigates application workflows
- This code is integrated with BDD (via Java interfaces), so that:
 - BDD-Security stories can use that code to navigate and generate traffic
 - This generated traffic will be scanned by ZAP via BDD



If no Selenium test code exists?

Simply give developer teams access to ZAP to (at least) pre-seed the scanner:

- Developer teams use browser to navigate app workflows while proxying
 - Thereby seed the ZAP session(s) with navigation nodes/workflows
 - Save the ZAP session(s) and check-in into SCM (Git, SVN, ...)
- Point the Jenkins ZAP plugin to the saved ZAP session(s) as starting point
- Devs can add to this list of URLs for ZAP with each new UI

BTW: ZAP is also available as Docker image...



OWASP AppSecEU 15
Amsterdam, The Netherlands



Axis of "Static Depth"



How deep is static code analysis performed within a Security DevOps CI chain?

i.e. "where" are static security tests applied?



OWASP AppSecEU 15
Amsterdam, The Netherlands

Axis of "Intensity"



How intense are the majority of the executed attacks within a Security DevOps CI chain?

i.e. "what" is being checked for?



OWASP AppSecEU 15
Amsterdam, The Netherlands

Axis of "Consolidation"



How complete is the process of handling findings within a Security DevOps CI chain?

i.e. "how" are the results used?



OWASP AppSecEU 15
Amsterdam, The Netherlands

Axis "Consolidation": Level I

Generate human-readable (HTML) reports from tools and link them in Jenkins


- All relevant mentioned static and dynamic scanners generate HTML reports
- Collect and publish them in Jenkins build: via Jenkins "HTML Publisher Plugin"

Use simple criteria to "break the build" on heavy findings (ok, at least "unstable")

- Dependency-Check, BDD-Security (with the JBehave-stories), FindSecurityBugs (via Sonar when rated as blocker), Arachni (via Gauntlt execution with BDD-like stories), etc. all have capabilities to automatically flag the build
- For others: at least do a simple log parse from Jenkins "Log Parser Plugin" to flag the build as unstable and/or broken

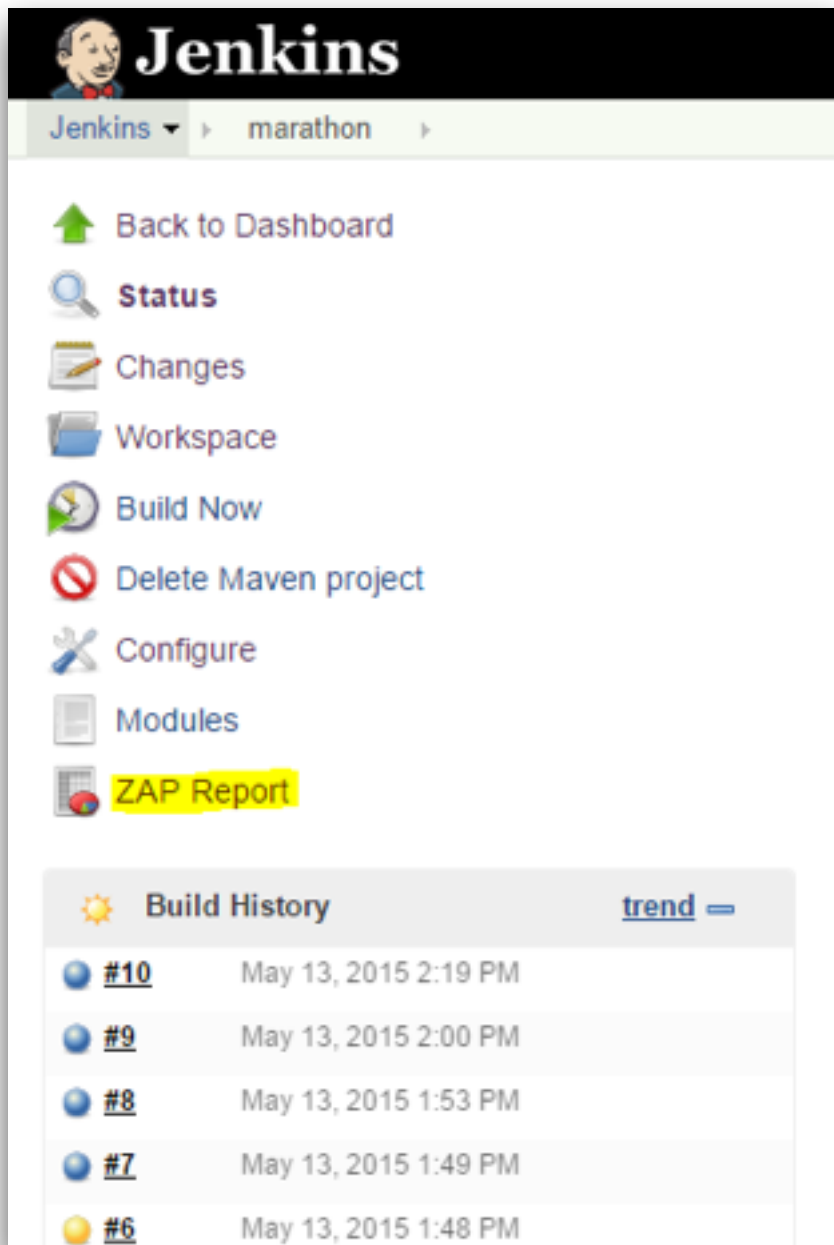


Post-build Actions

 Publish HTML reports

HTML directory to archive	Index page[s]	Report title	Keep past HTML reports
<input type="text" value="reportsZAP/"/>	<input type="text" value="report.html"/>	<input type="text" value="ZAP Report"/>	<input type="checkbox"/>

*Jenkins "HTML Publisher Plugin":
Configuration of HTML reports to link*



The image shows the Jenkins web interface. At the top is the Jenkins logo and name. Below it is a breadcrumb trail: "Jenkins" > "marathon". A sidebar on the left contains several links with icons: "Back to Dashboard" (green arrow), "Status" (magnifying glass), "Changes" (notepad), "Workspace" (folder), "Build Now" (play button), "Delete Maven project" (red circle with slash), "Configure" (wrench), "Modules" (document), and "ZAP Report" (document with red circle, which is highlighted in yellow). The main content area shows the "Build History" section with a "trend" link. It lists five builds: #10 (blue circle), #9 (blue circle), #8 (blue circle), #7 (blue circle), and #6 (yellow circle). Each build entry includes its number and the timestamp.

Build Number	Timestamp
#10	May 13, 2015 2:19 PM
#9	May 13, 2015 2:00 PM
#8	May 13, 2015 1:53 PM
#7	May 13, 2015 1:49 PM
#6	May 13, 2015 1:48 PM

*Jenkins "HTML
Publisher Plugin":
Result in build*

Axis "Consolidation": Level 2

Custom logic to make build unstable and/or broken depending on

- Type of vulnerability (CWE or WASC or ...)
- Confidence level (firm vs. tentative)
- Severity ranking (high risk)

Provide useful remediation info to developers

Respect suppression mechanisms to rule out false positives



OWASP AppSecEU 15
Amsterdam, The Netherlands

Flagging builds from reports

How (from within a CI job)?

- Most scanners also emit XML reports that can be parsed
 - Often a simple XPath count is just fine
- Alternatively fetch the results by accessing the scanner's API
- Be sure to only break build with (new?) findings of high severity and high confidence !!!
 - Less is more (when it comes to automation)...



Axis "Consolidation": Level 3

Consolidation goals:

- Consolidate & de-duplicate findings from different scanner reports (with better false positive handling)
- Push consolidated findings into established bug-tracker (known to devs)
- Delta analysis & trends over consolidated data sets



OWASP AppSecEU 15
Amsterdam, The Netherlands

ThreadFix as result consolidator

Use a local ThreadFix server, which imports native scanner outputs

- does the heavy lifting of consolidation & de-duplication
- pushes findings toward bug-tracker and IDE (via plugins)
- process can be customised using it's own REST-API
- ThreadFix imports findings of ZAP, Arachni, FindBugs, Brakeman, etc.



OWASP AppSecEU 15
Amsterdam, The Netherlands

Axis "Consolidation": Level 4

Measure the concrete code coverage of your security testing activities

- Find untested "white spots"
- Derive where static checks and code reviews should focus more to compensate

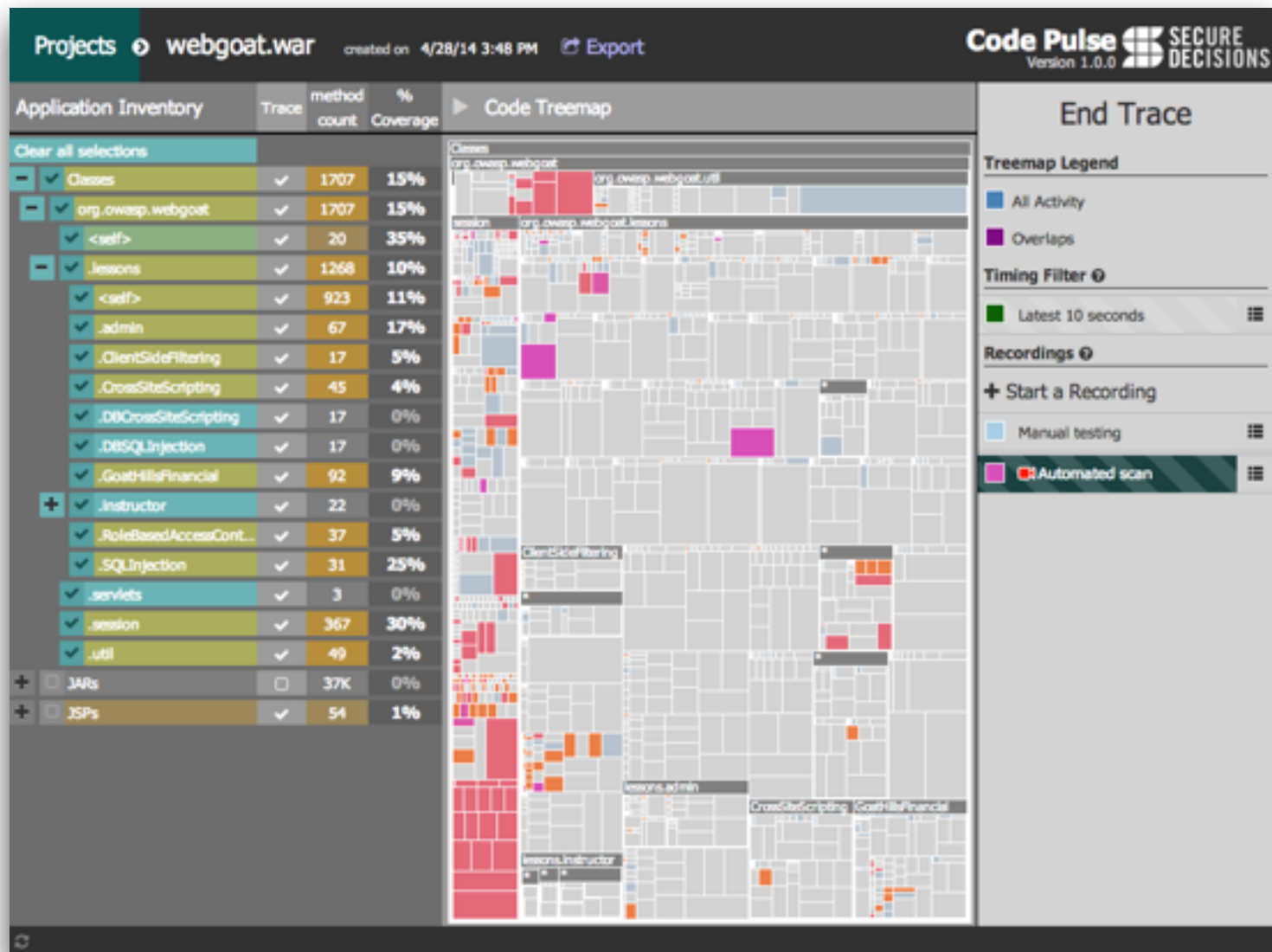


Code coverage analysis

Use "OWASP Code Pulse", which instruments your Java app via agent

- collects coverage data during dynamic security testing scans
- generates reports ("code treemaps") of coverage





Code Treemap of dynamic scan coverage



Thank you very much!

Links

OWASP ZAP

ZAP Selenium Demo

ZAP Jenkins Plugin

BDD-Security

Arachni

OWASP Dependency Check

OWASP Dependency Track

FindSecurityBugs

FindSecurityBugs-Cloud

retire.js

ScanJS

Jenkins Log Parser Plugin

ThreadFix

OWASP Code Pulse

Seccubus

vulnDb

fuzzdb

radamsa

https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

<https://github.com/continuumsecurity/zap-webdriver>

<https://wiki.jenkins-ci.org/display/JENKINS/ZAPProxy+Plugin>

<http://www.continuumsecurity.net/bdd-intro.html>

<http://www.arachni-scanner.com>

https://www.owasp.org/index.php/OWASP_Dependency_Check

https://www.owasp.org/index.php/OWASP_Dependency_Track_Project

<http://h3xstream.github.io/find-sec-bugs/>

<https://code.google.com/p/findbugs/wiki/FindBugsCloudTutorial>

<http://bekk.github.io/retire.js/>

<https://github.com/mozilla/scanjs>

<https://wiki.jenkins-ci.org/display/JENKINS/Log+Parser+Plugin>

<http://www.threadfix.org>

https://www.owasp.org/index.php/OWASP_Code_Pulse_Project

<https://www.seccubus.com>

<https://github.com/vulnDb/data>

<https://code.google.com/p/fuzzdb/>

<https://code.google.com/p/ouspg/wiki/Radamsa>

Interested in more web security stuff?

Visit my Blog: www.Christian-Schneider.net

@cschneider4711

