Software Security

Group project:
application security verification
using OWASP ASVS

Assurance

Big challenge:

how can we provide assurance that an application is secure?

NB: it is *much* easier to

demonstrate that an application is *not* secure

than it is to

guarantee that it *is* secure

Assuring security of software

Before trying to get some assurance about the security of any piece of software, we should ask

1. What does it mean for this application to be secure?
2. What is the attacker model?
3. How important is it?
   - How big is the impact if it is insecure?
   - Does the application involve the crown jewels of the organisation?

   This involves risk assessment

You may decide that it's not worth the effort to provide some assurance.

Group Project

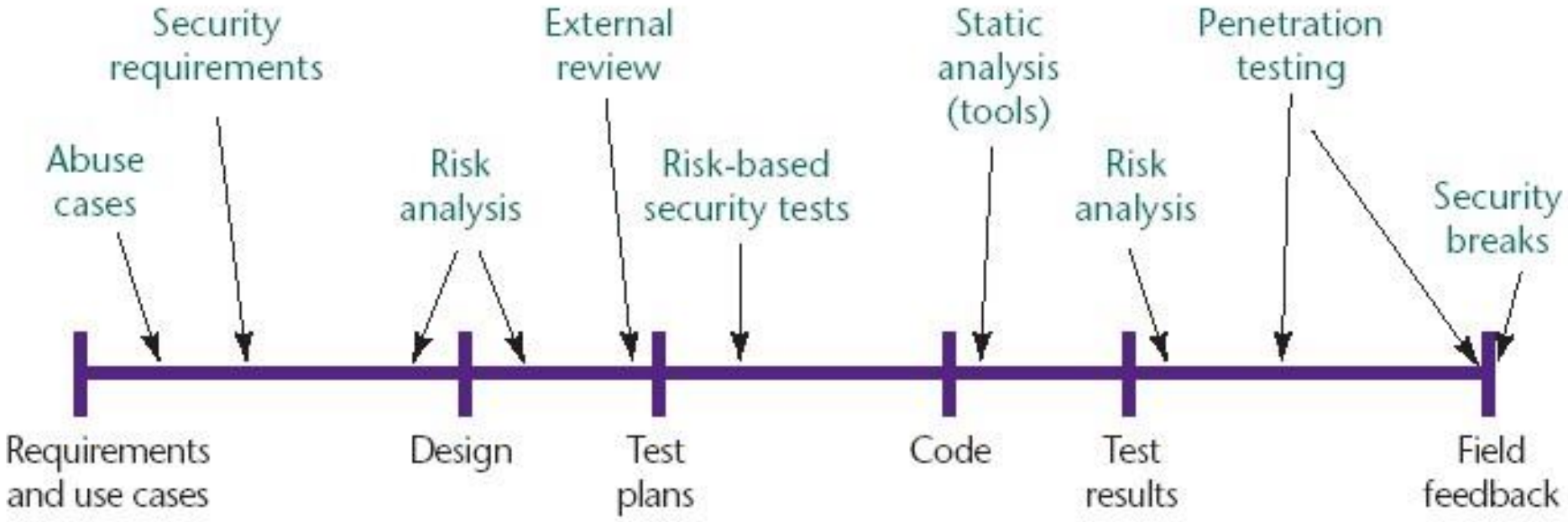
- We do a security code review of a web-application
- following the OWASP ASVS
  - Application Security Verification Standard (2016 edition, 3.0.1)
- trying out commercial source code scanners

Time for this +/- 30 hours,

ie one afternoon/morning per week for next 2 months.

We'll discuss and compare our findings in November (about the tools) and in December (overall)

(white box) code review vs (black box) pen test

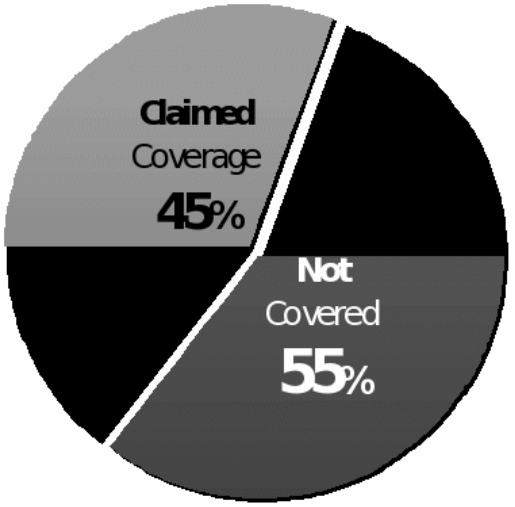


Goals

- Experiencing a software security review process
  - ie white-box code review, rather than a black-box pen test
- How useful are existing standards and approaches?
  - esp OWASP Application Security Verification Standard
- How good are modern static analysis tools?
  - Do they find many bugs? Many kinds of bugs? False positives? False negatives?
- How should security design and implementation decisions wrt security have been made and documented?

Gartner ranking of (static & dynamic) tool providers

Caveat: static tools cover at best 45%?



Research by MITRE showed

- All application security tool vendors' claims put together cover only 45% of the 695 known vulnerability types
- Very little overlap between tools, so to get 45% you need them all (assuming their claims are true)

It is hard to get objective evidence about quality of these tools

- despite efforts by eg NIST in Software Assurance Metrics and Tool Evaluation (SAMATE)

You have to experience using them to get an idea.

SECURITY INNOVATION · FORTIFY SOFTWARE · PALAMIDA · watchfire · Klocwork · Secure Software · AppSIC · CENZIC · CORE SECURITY TECHNOLOGIES · PARASOFT · Coverity · proServices · SofCheck · SPI DYNAMICS · VERACODE · OUNCE LABS

Non-goals

- For some, this is throwing you in at the deep end.
  I realise your experience varies a lot!
- Don't be tempted in copying results from other groups
  - Whether or not you find any security problems is not important, it's about forming an well-argued opinion about code reviews, the ASVS as guide for this, static analysis tools, etc.

OWASP ASVS (v3.0.1)

Application Security Verification Standard

aims to normalise the range in coverage & level of rigour in performing web application security verifcation

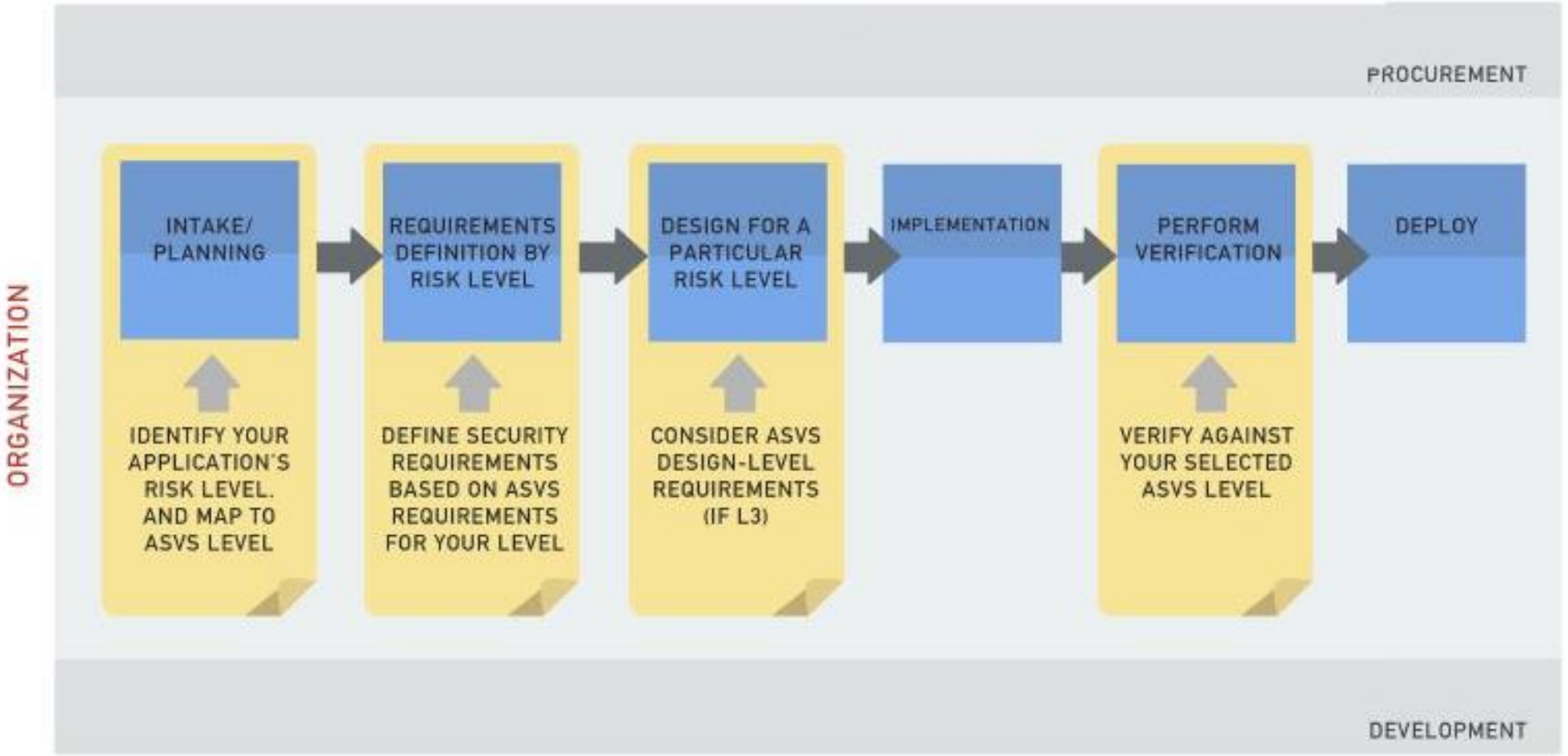
NB not "verification" in the mathematical or even testing sense

(New version V3.1 almost out; don't use it)

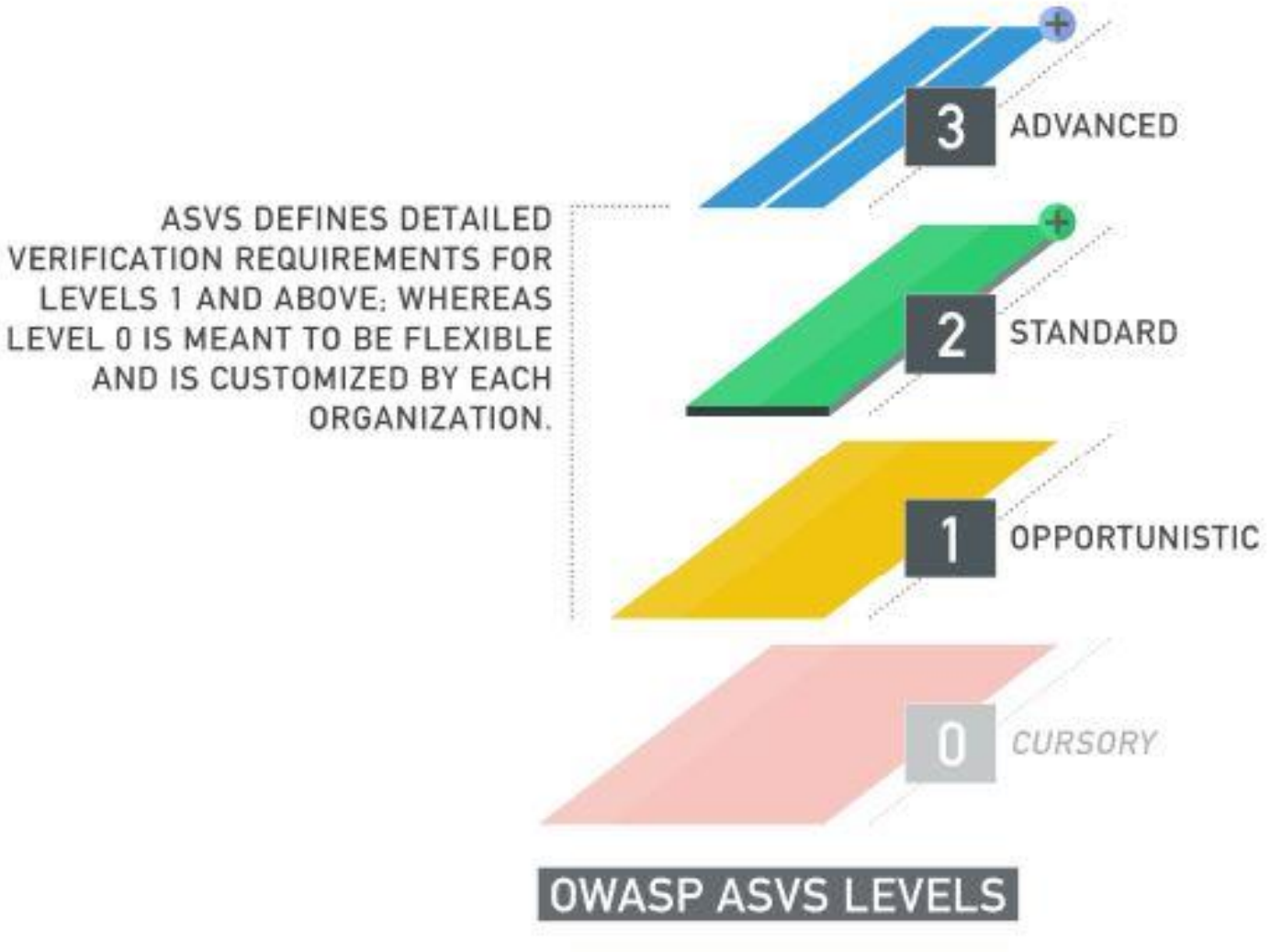
Builders

Defenders

OWASP ASVS Process



OWASP ASVS Levels



Categories of Verification Requirements

V1 Security Architecture

V2. Authentication

V3. Session Management

V4. Access Control

V5. Input Validation

V6. Output Encoding/Escaping

V7. Cryptography at rest

V8. Error Handling & logging

V9. Data Protection

V10. Communication Security

V11. HTTP Security

V13. Malicious Code Search

V15. Business Logic

V16 Files and Recourses

V17 Mobile

V18 Web Services

V19 Configuration

We will ignore V1, V10, V13, V15, V17-V19

ASVS Security Requirements

ASVS provides checklists of security requirements to check

- clustered in categories

where security requirements are stated in a 'positive' way , eg

- 'negative'
  there are no XSS attacks
- 'positive'
  all HTML output containing user-supplied input is properly escaped

Note: checking such positive instead of negative statements is very different.

Showing there ate XSS attacks is easier (if there is one) than arguing that there are not

Verification Techniques

- **Dynamic**
  - using running application
  - aka (penetration) testing

This can be

- manual application penetration testing
- automated application penetration testing

Static

using source code

aka code review

This can be

- manual
- automated using code analysis tools

Focus of the group project

Tool support for the ASVS

- A tool you could use to explore ASVS & find pointers to more info on the requirements

- https://www.owasp.org/index.php/OWASP_Security_Knowledge_Framework
- https://www.securityknowledgeframework.org/

Before you start

1. Form groups; possibly via Brightspace
   - Send me an email when you have a group
   - Fill in the questionnaire on the web page
2. Fix a weekly morning/afternoon to work on this
3. Keep a log what you are doing, and who does what

To start

1. Read the ASVS
2. Map the tool warnings to ASVS requirements

3. Look at the code
4. Install the code to get a feel for functionality?

5. Dig in deeper
   - Check tool warnings for false/true positives
   - Look into the other requirements
   - …

To complete

- Keep track of your findings in the .xls
  - Template will be provided in Brigtspace
- Produce findings in one PDF at the end

- We'll compare findings on the tool warnings November 16
- We'll compare overall finding December 21

  - So relevant .xls findings *before* those dates
  - Final report could be later, in January

Remember: we're skipping the most important steps

By jumping straight to look at the code using the ASVS

we skip the most important first steps of any security analysis:

1. **identifying security requirements and their importance**
   - **ie. threats and impacts, for a good risk assessment**
2. **defining attacker model**
   - **eg 'standard' online attacker, insiders, vandals, hacktivists, mafia, NSA, …**
   - **should also consider capabilities & motivation**