Group Project
Application Security Verification using OWASP ASVS

Question

What would you do if you if someone asked you to check if some piece of software that they use (and possibly bought) is secure?

it is *much* easier to

demonstrate that an application is *not* secure

than it is to

guarantee that it *is* secure

Assurance

Big challenge:

how can we provide assurance that an application is secure?

Is the question above silly? NO, it is meaningless

We can hope to answer it n a meaningful way only if we know

- what the security requirements are
- what kind of attacks we should consider

Assuring Security of Software

Before trying to get some assurance about the security of any piece of software

1. What does it mean for this application to be secure?
2. How important is it? i.e., how big is the impact if it is insecure? Does the application involve the crown jewels of the organization?
3. What is the attacker model? i.e., what can attackers do to attack the system?

Thinking about abuse cases can be useful in all this.

You may decide after all this that it is not worth the effort to try and provide some assurance.

Web-application Verification Project

- A security code review of a web-application
  - in groups looking at different aspects
- following the OWASP standard
  - Application Security Verification Standard (ASVS) 3.0.1 (July 2016)
- trying out some source code analysis tools
- documenting the efforts,
  - incl. log, chronologically keeping track of who does what when

Time for this +/- 20 hours, ie half a day per week for next 4 weeks

Goals

- learning about web-app security
  - i.e., white-box code review, rather than a black-box pen test
- experiencing a software security review process
- how useful are existing standards (one specifically OWASP-ASVS) and approaches?
- how good are the tools we can use?
- how ***should*** security requirements, security design, and implementation decisions with respect to security have been made and documented?

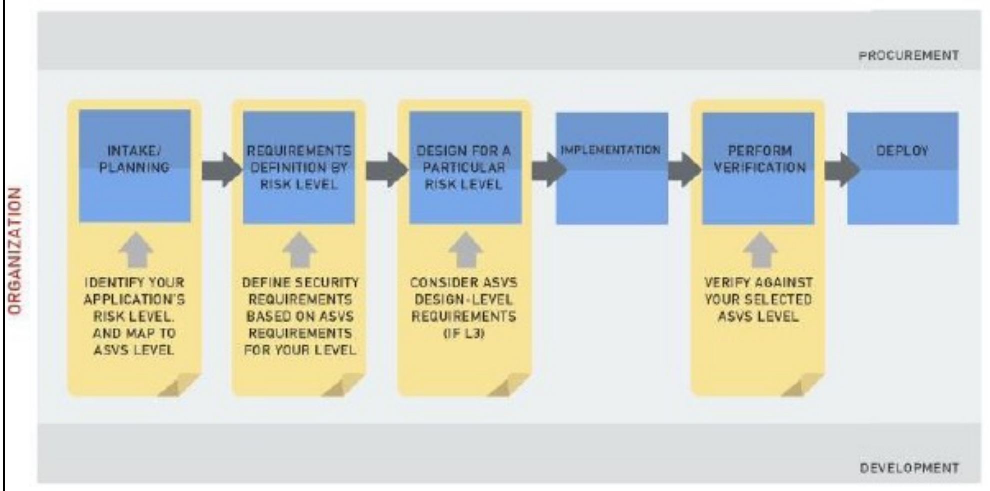
Non - Goals

- For some, this is comparable to "throwing you at the deep end". Not all of you have the same background / experience
- Do not be tempted to "copy" results from other groups
  - it is not important whether or not you find security problems
  - it **IS** important to form a conscious opinion about
    - code reviews
    - ASVS as a guide for doing so
    - static analysis tools
- My objective: getting some data and feedback on code review

Application Security Verification Standard (ASVS)

- OWASP standard on how to systematically evaluate the security of a web-application
  - *not "verification" in any formal sense*
- ASVS distinguishes
  - levels
    - the higher the level, the more rigorous the evaluation
  - techniques
    - dynamic vs static,
    - automated vs manual
  - sets of verification requirements
    - requirements to check

OWASP ASVS Process


PROCUREMENT
ORGANIZATION
INTAKE/ PLANNING
REQUIREMENTS DEFINITION BY RISK LEVEL
DESIGN FOR A PARTICULAR RISK LEVEL
IMPLEMENTATION
PERFORM VERIFICATION
DEPLOY
IDENTIFY YOUR APPLICATION'S RISK LEVEL. AND MAP TO ASVS LEVEL
DEFINE SECURITY REQUIREMENTS BASED ON ASVS REQUIREMENTS FOR YOUR LEVEL
CONSIDER ASVS DESIGN-LEVEL REQUIREMENTS (IF L3)
VERIFY AGAINST YOUR SELECTED ASVS LEVEL
DEVELOPMENT

ASVS Levels

- Level 1 – intended for all software
- Level 2 – for applications that contain sensitive data which requires protection
- Level 3 – for the most critical applications – that perform high value transactions, contains sensitive (e.g., medical) data or any application that requires the highest level of trust

Each level contains a list of security requirements, each (can be) mapped to security-specific features

ASVS Levels

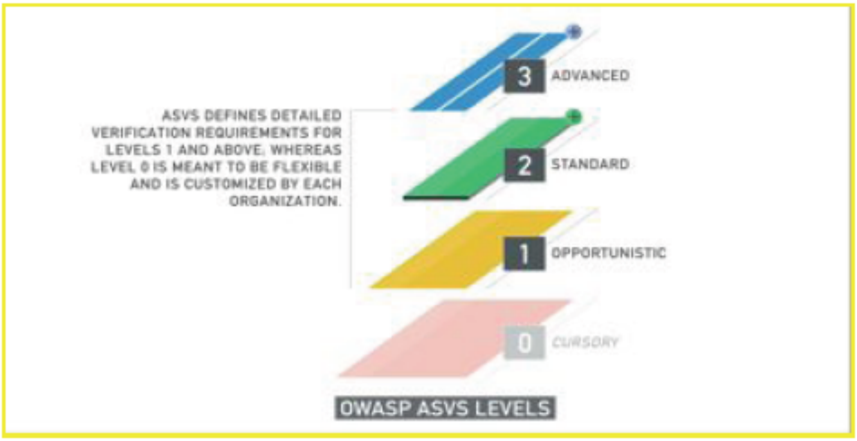

ASVS DEFINES DETAILED VERIFICATION REQUIREMENTS FOR LEVELS 1 AND ABOVE, WHEREAS LEVEL 0 IS MEANT TO BE FLEXIBLE AND IS CUSTOMIZED BY EACH ORGANIZATION.
3 ADVANCED
2 STANDARD
1 OPPORTUNISTIC
0 CURSORY
OWASP ASVS LEVELS

Figure 2 - OWASP ASVS Levels

11

ASVS Levels

- Level 1 – achieved if application resilient to easily discovered vulnerabilities (OWASP top 20 and similar) Simply automatically by tools or manually without source code
- Level 2 – achieved if application resilient to most risks to software, with attackers skilled and motivated focusing on specific targets within B2B-type applications
- Level 3 – for the most critical applications (military, health, safety, critical infrastructures). Achieved if it defends against advanced vulnerabilities after in-depth analysis of architecture, principles, coding, ...

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

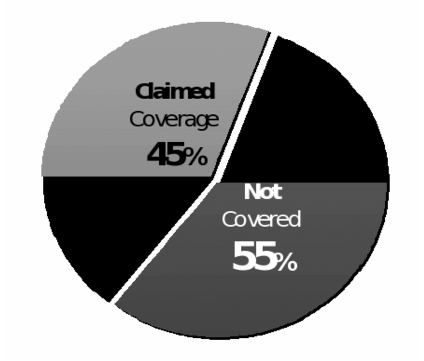
This can be

manual security code review

automated security code review

Group project focuses on this aspect

Caveat: tools cover at best 45% ? [study by MITRE]



- 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)

14

ASVS Security Requirements

ASVS provides checklists of security requirements to check

- clustered in several categories V1 ... V19

The security requirements are stated in a "positive" way:

- negative
  - e.g. there are no XSS holes
- positive
  - e.g. all HTML output that includes user-supplied input is properly escaped

Note: checking positive rather than negative statements is very different

Categories of Verification Requirements

V1. ~~Security Architecture~~

V2. Authentication

V3. Session Management

V4. Access Control

V5. Malicious Input Handling

V6. ~~Output Encoding / Escaping~~

V7. Cryptography at rest

V8. Error Handling & Logging

V9. Data Protection

V10. Communication Security

V11. HTTP Security *Config.*

V12. ~~Security Configuration~~

V13. Malicious Controls

V14. ~~Internal Security~~

V15. Business Logic

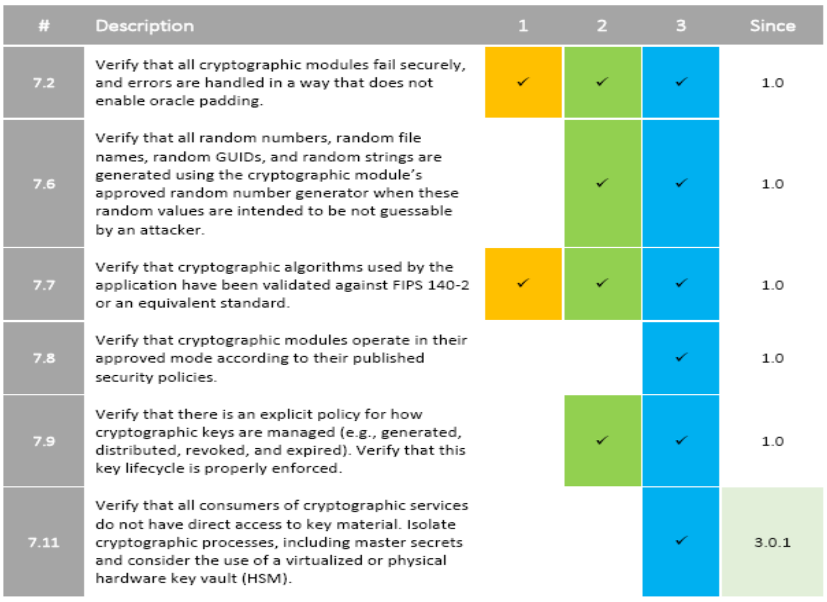
V16. Files and resources

V17. Mobile

V18. Web Services (new in 3.0)

V19. Configuration (new in 3.0)

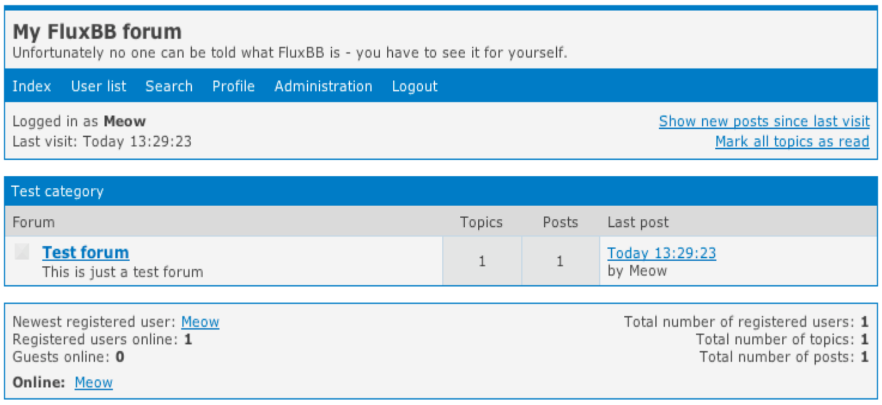
Cryptography at rest



#	Description	1	2	3	Since
7.2	Verify that all cryptographic modules fail securely, and errors are handled in a way that does not enable oracle padding.	✓	✓	✓	1.0
7.6	Verify that all random numbers, random file names, random GUIDs, and random strings are generated using the cryptographic module's approved random number generator when these random values are intended to be not guessable by an attacker.		✓	✓	1.0
7.7	Verify that cryptographic algorithms used by the application have been validated against FIPS 140-2 or an equivalent standard.	✓	✓	✓	1.0
7.8	Verify that cryptographic modules operate in their approved mode according to their published security policies.			✓	1.0
7.9	Verify that there is an explicit policy for how cryptographic keys are managed (e.g., generated, distributed, revoked, and expired). Verify that this key lifecycle is properly enforced.		✓	✓	1.0
7.11	Verify that all consumers of cryptographic services do not have direct access to key material. Isolate cryptographic processes, including master secrets and consider the use of a virtualized or physical hardware key vault (HSM).			✓	3.0.1

Web application case study: FluxBB

- Popular PHP web forum (bulletin board) implementation
- https://fluxbb.org



My FluxBB forum
Unfortunately no one can be told what FluxBB is - you have to see it for yourself.

Logged in as **Meow**
Last visit: Today 13:29:23

Show new posts since last visit
Mark all topics as read

Test category

Forum	Topics	Posts	Last post
Test forum This is just a test forum	1	1	Today 13:29:23 by Meow

Newest registered user: Meow
Registered users online: **1**
Guests online: **0**
Online: Meow

Total number of registered users: **1**
Total number of topics: **1**
Total number of posts: **1**

First project step: source code analyzers

- RIPS, RATS, …
  - basic old tools – glorified GREP/CTRL-F

You could also try any other tools you know or have experience with

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

http://blog.codacy.com/2014/05/06/php-static-analysis-tools/#gs.Ib3wkzI

First project phase

- Fix periodic meetings (once a week?)
- Keep a log of what you do
- Read the AVSV
- Run tool(s) over the source code.
- Look at the findings (interesting, useful, useful for you?)
- Start looking at the source code
- Document your findings and experiences.

  Deliverables: two/three pages (by Dec 9)
  - reporting on tool's findings about code, and how these can (or cannot) be used for your verification requirements
  - log

Problems you *will* run into

- Fundamentally
  - Tools produce way too many warnings
  - There is far too much code to look at
- Solutions
  - prioritize
  - sample (look at a few warnings/parts of the code)

  Use your intelligence here, and document choices & bases for choices

Skipping important steps

Be aware that we skip the important first steps of any security analysis, when we jump straight to verifying the security using AVSV:

- defining attacker model
  - standard online attacker
  - consider capabilities and motivations
- identifying security requirements and their importance
  - i.e. threats and impacts for a usable risk assessment

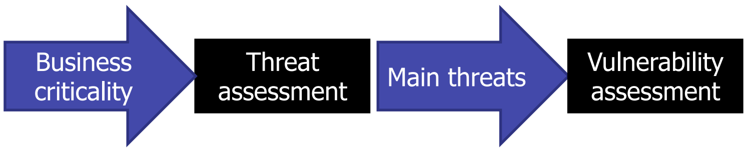
For a bulletin board, what could important considerations be here?

Is information in the board sensitive? Is integrity or confidentiality important?

Is membership of the bulletin board (privacy) sensitive?

Risk assessment

- Security evaluations should start with risk assessment
  - To prioritize efforts since means are limited
- Ideal picture:



Risk assessment for a web forum

- a.k.a. threat modeling
- What are the
  - assets
  - stakeholders
  - threats (to assets)
  - security requirements
  - attacks

  attack surfaces, potential vulnerabilities exploited by attacks, risks of attacks, possible countermeasures against attacks, ...

Classification of requirements & attacks

CIA

- Confidentiality
- Integrity
- Availability
- Non-repudiation

AAAA

- Authentication
- Access control/Authorization
- Auditing (requires logs)
- Action (requires auditing)

STRIDE

- Spoofing
- Tampering
- Repudiation
- Information Disclosure
- Denial-of-Service
- Elevation of Privilege

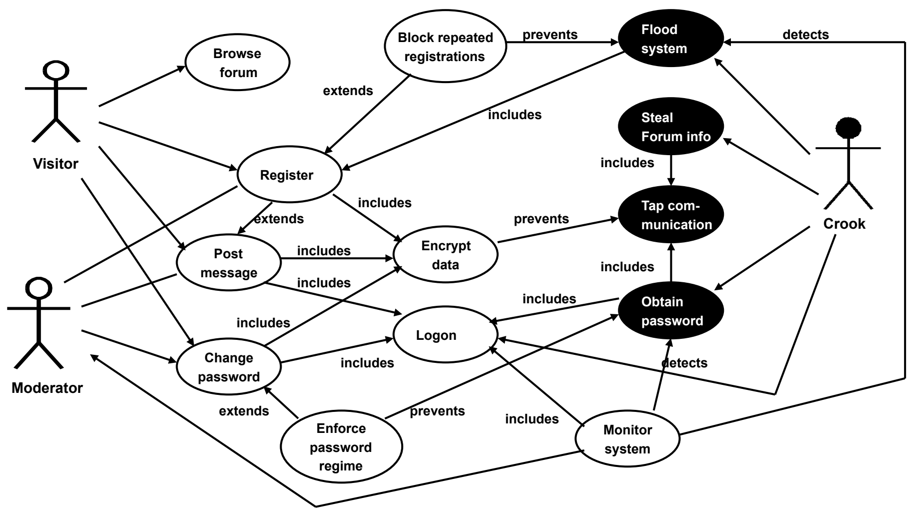
forum: assets

- forum content - availability, confidentiality, and integrity?
- user accounts
- passwords
- bandwidth (of the system hosting the forum)
- 'bandwidth' of users of the forum? – e.g., DoS attack by flooding forum
- CPU time (of the system hosting forum)
- disk space (for the database)
- the system hosting the forum
- the web browsers of the forum users (attacked via XXS via the forum?) – hence also: the systems of the forum

forum: stakeholders

- forum users
  - unregistered users, register users, moderator, forum owner,
- owner of the system hosting the forum
- possibly: others affected by the topics discussed on the forum
- attacker
  - which might be any of the categories above
  - hence for instance:
    - *elevation of privilege attacks (in case registered users are attackers)*
    - *logging (in case moderators are attacker)*

Modeling design, attacks and controls


Visitor
Moderator
Crook
Browse forum
Block repeated registrations
Flood system
Steal Forum info
Register
Tap com-munication
Post message
Encrypt data
Obtain password
Logon
Change password
Enforce password regime
Monitor system
prevents
detects
extends
includes