Threat Modelling (Web)Apps Myths and Best Practices

Matthias Rohr

www.matthiasrohr.de
mail@matthiasrohr.de

OWASP

7.11.2012

The OWASP Foundation

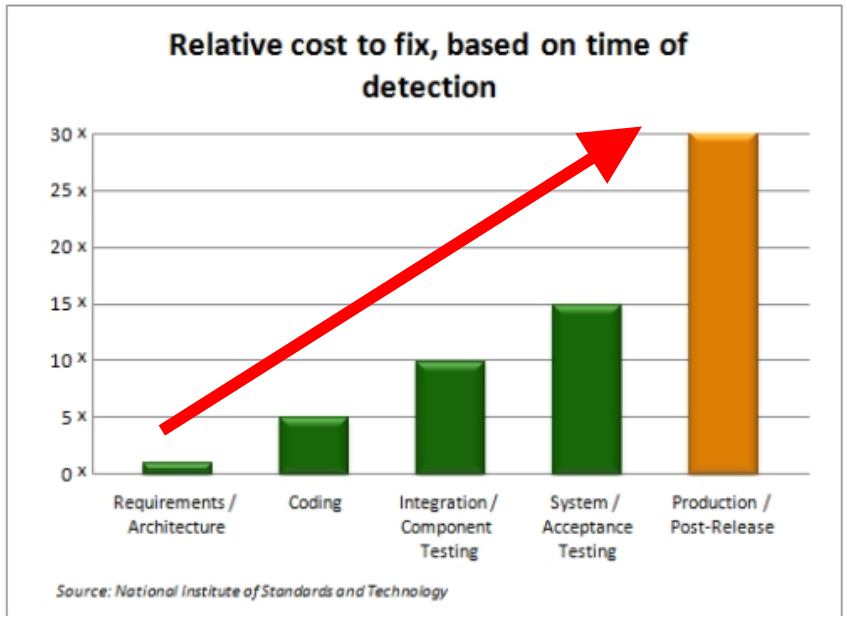
http://www.owasp.org

About me

- Matthias Rohr
- Dipl. Medieninf. (FH), CISSP, CSSLP, CCSK
- Focus: Application Security Management
- Contractor in London – from 2013 on back in Hamburg
- Active in OWASP since 2007:
  - OWASP ASVS/Java/Skavenger Project
  - Review of “BSI Baustein Webanwendungen”
  - WAF Best Practice Paper
  - OWASP Summits

Motivation I: Pushing Appsec Left in the SDLC

- **Costs** to fix a bug
- **Level of Security** (derived also from the costs)
- **Planability**: Sec tests may lead to "surprises"
- **Visibility** within SDLC**:**

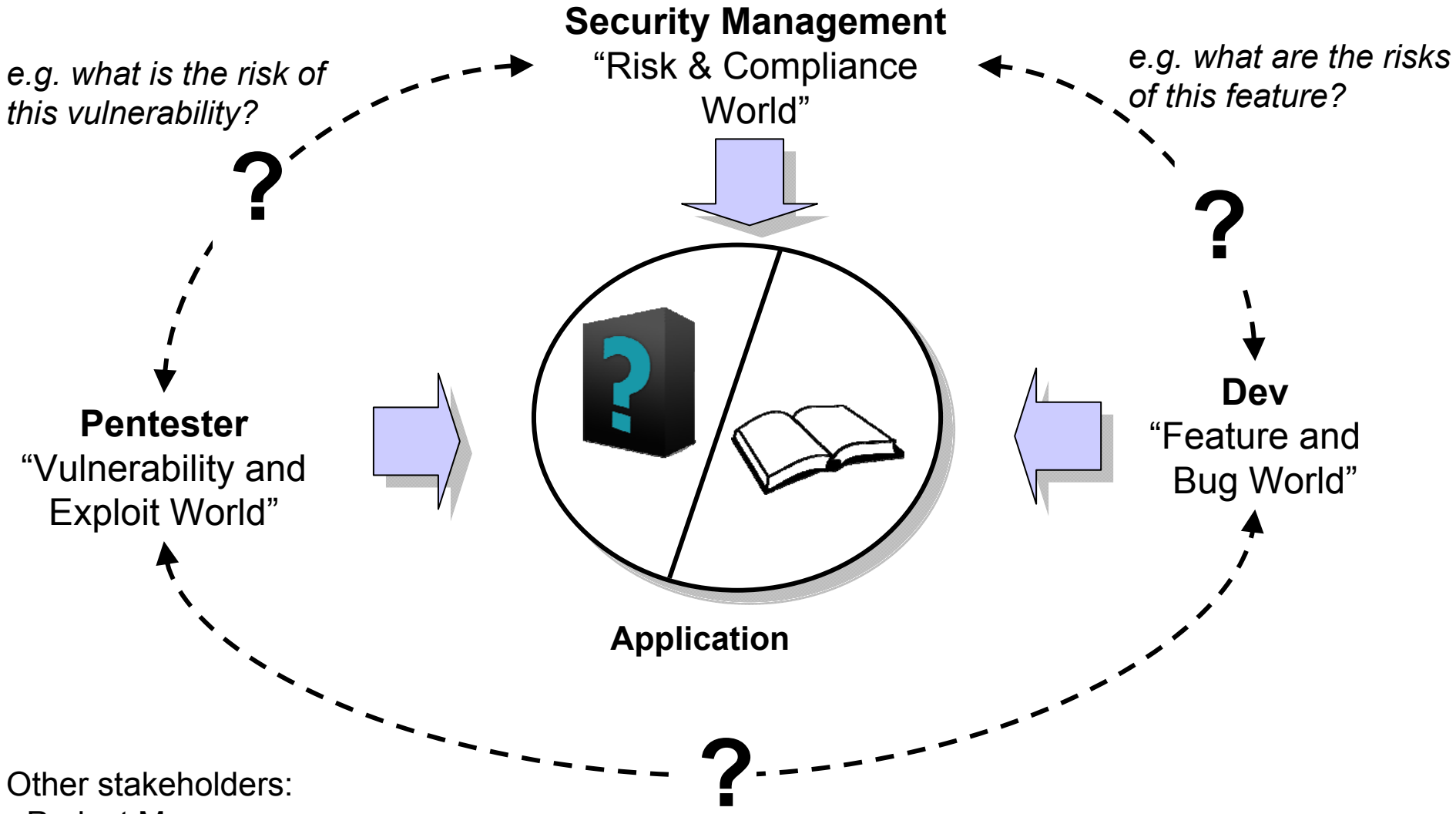


"**60%** of all weaknesses are visible in the application design"

(Principles of Software Engineering Management, T. Gilb)

OWASP

Motivation II: The Transformation Problem



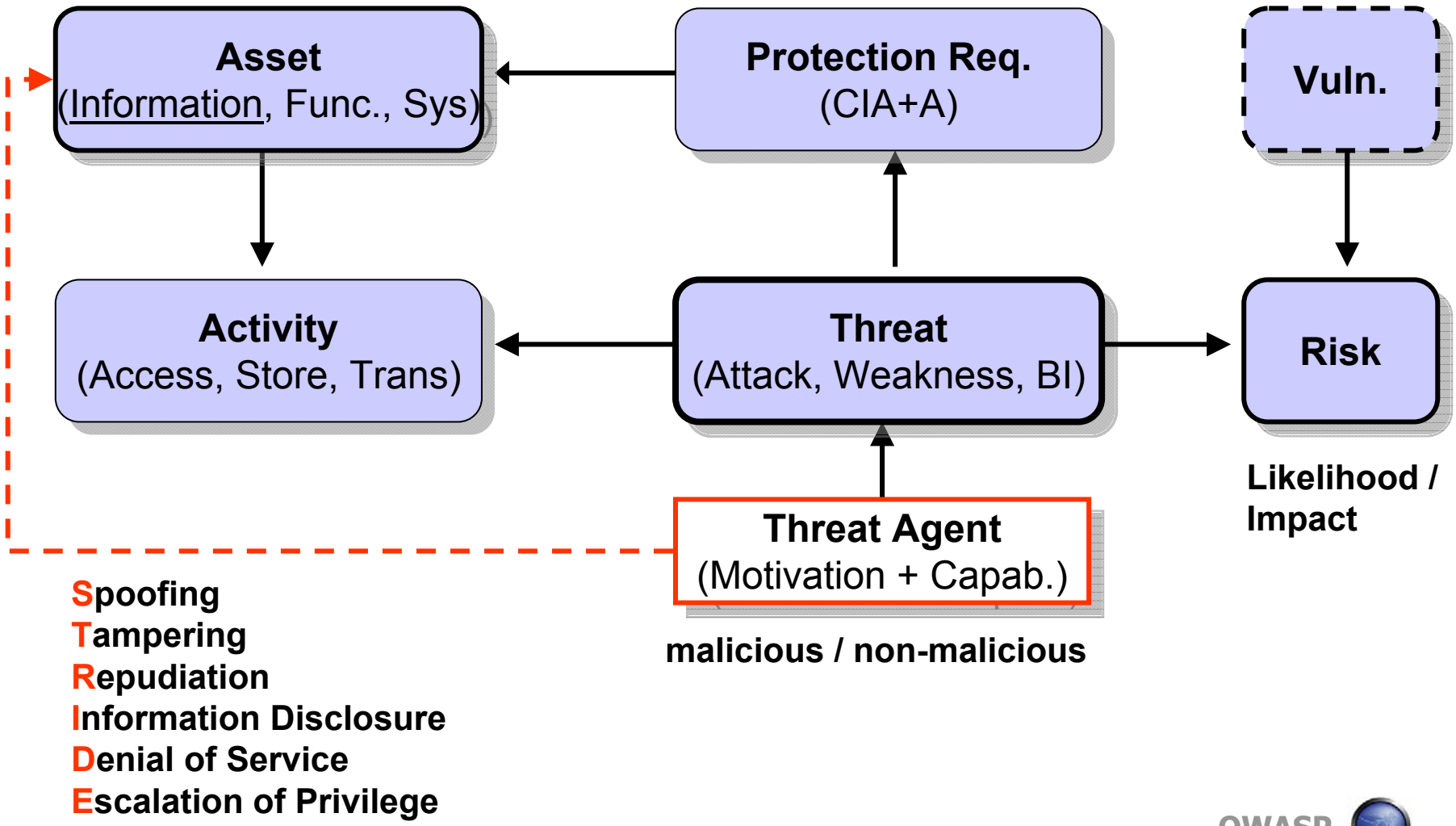
Other stakeholders:
- Project Managers
- QA
- Operation

OWASP

Threat Modelling - Goals

- **Primary**
  - Early identification, assessment and correction of potential security problems in an IT system (such as a Web application)
  - Link technical implementation to IT Risk Mgmt & ISMS
- **Secondary**
  - Improvement of planability & quality of later security tests (pentests, code reviews, etc.)
  - Documentation and discussion of the application security architecture

What is a Threat?



OWASP

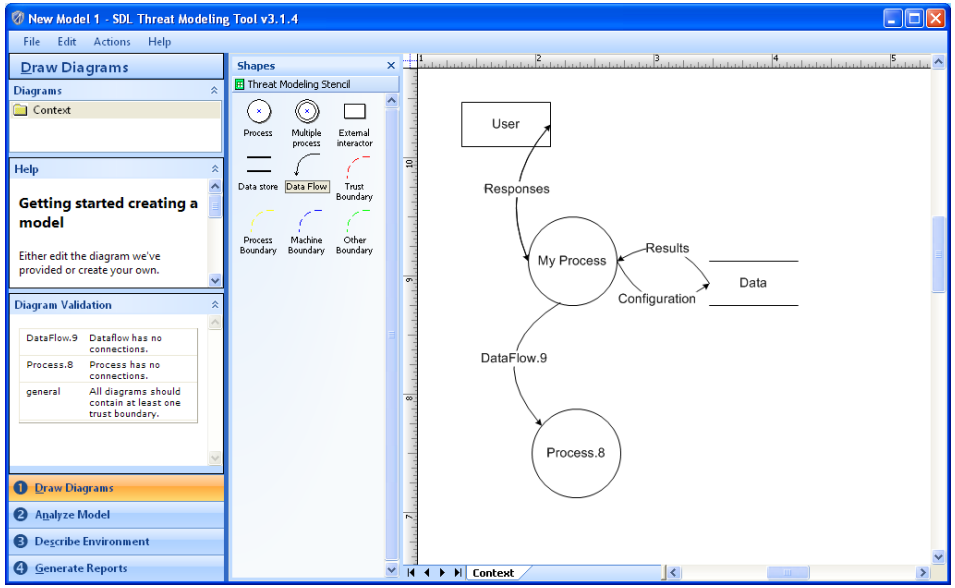
Existing Methodologies

- Microsoft I (2003, “DREAD”)
- Microsoft II (2009, “Bug Bars”)
- OWASP I + OWASP II
- PASTA
- T-MAPS
- PTA
- SANS
- Trike

Difficult to compare due to different concepts.

Tools

- Word, Excel, Visio or any Wiki, etc.
- Microsoft Threat Modelling Toolkit (TAM):
  free MS Visio Plugin, but limited (DfD* analysis only)



DfD = Data flow Diagram

OWASP

Myths

(or just misunderstandings……)

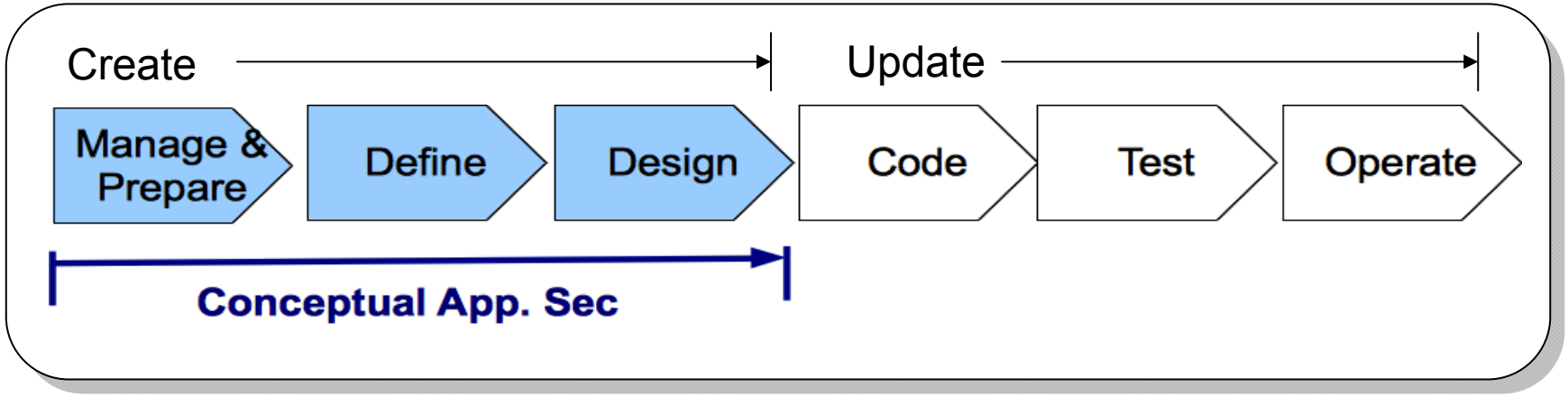
Myth 1: Threat Modelling is too Complicated

- Threat modelling is a **best effort** approach
  - Identifying only some threats is better than nothing at all
  - Objective is not 100% threat coverage
  - Learning and integration process: Start simple & informal
- Every stakeholder can conduct some sort of **threat assessment*** in principle (e.g. developers, project managers, …)

* A threat assessment is not necessarily a threat modelling!

Myth 2: Threat Modelling = Design Review

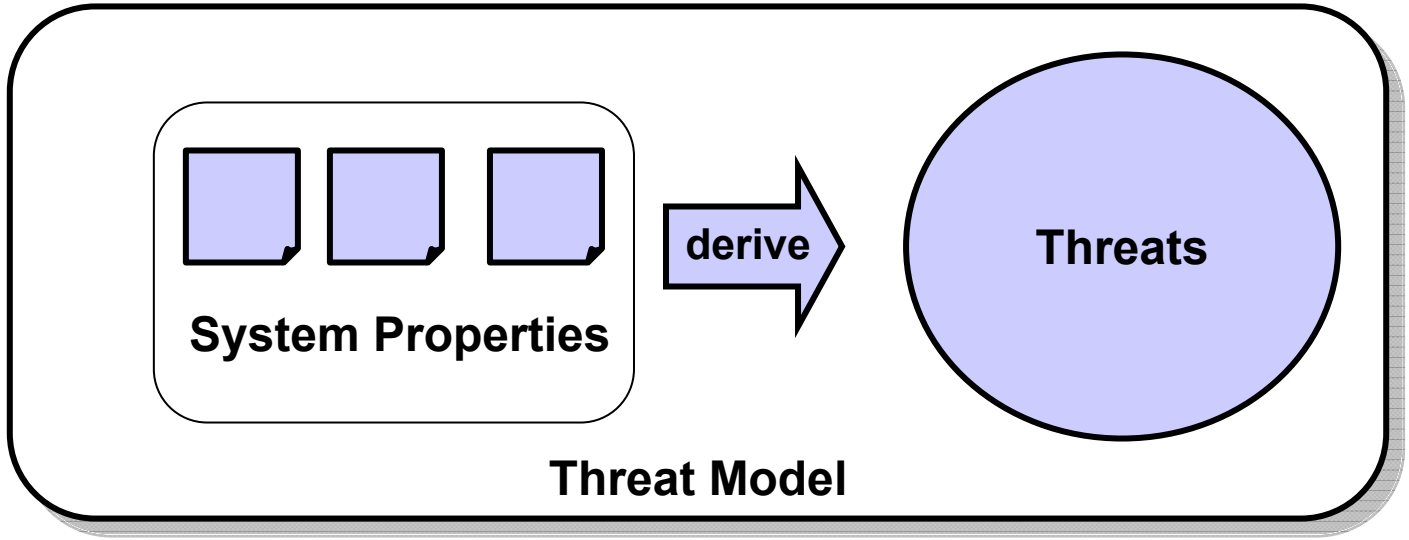
- Many threats are already visible in the specification!
- Hence: See TM as a **conceptual security analysis**!



- A threat model can be created in **iterations** (allows us to start very early and with a limited model)
- A threat model can be updated with details from implementation and operation phase.

Myth 3: TM Output = a List of Threats

- Lists are **static**, models can be **dynamic**
- Change of a system's property (e.g. a data flow) may effect its threats and therefore the threat model too.
- Lists as result of a generic "threat analysis" ok of course.



See also: http://www.curphey.com/2012/03/is-threat-Modelling-overrated

Myth 4: Decide for ONE Perspective

- **Attack-centric:** Focuses on attacks
  - May suit a pentester
  - Example: "XSS attack to steal cookies"
- **Software-/system-centric:** Focuses on weaknesses
  - May suit a developer or SW architect
  - Example: "Insufficient output validation controls"
- **Asset-/Risk-centric**: Focuses on business impact (BI)
  - May suit an infosec manager
  - Example: "Attacker may access customer data via …"

Multiple perspectives may lead to a lot **overlapping** threats, but will also increase **threat coverage!!!**

Myth 5: One Methodology suits them all

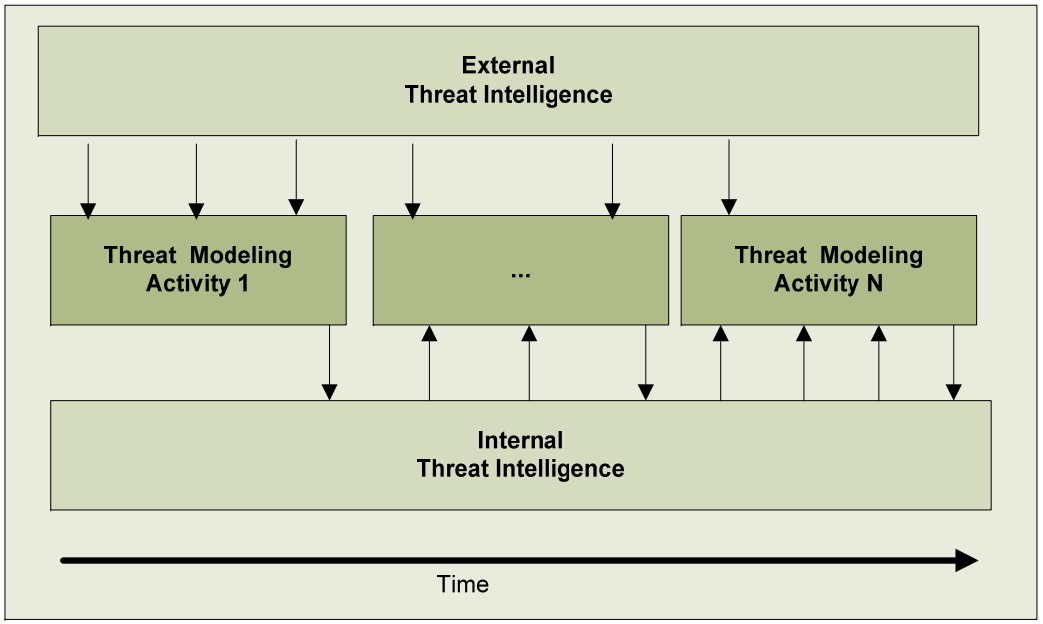
- For example Microsoft's TM:
  - Methodology is based on DfD analysis
  - Software-centric = focused on SW developers
- Instead, the approach should **be specific** to
  - The (development) organisation
  - Both SDLC and SDL
  - The qualification of the analyst
  - The protection requirements of the app
  - Existing resources
  - …
- Known as: **Tailoring**

Best Practices

(based on my personal experiences)

Threat Intelligence (TI)

- Main idea: Mapping of **expert know-how** and other intelligence to a threat modelling exercise
- Examples: Gen. threats, metrics, countermeasures, etc.
- Essential for integrating threat modelling into SDLC, improving quality & reducing resources



See also “Attack Models” practice in BSIMM study: http://bsimm.com/online/intelligence/am

OWASP

Step 0: Preparation

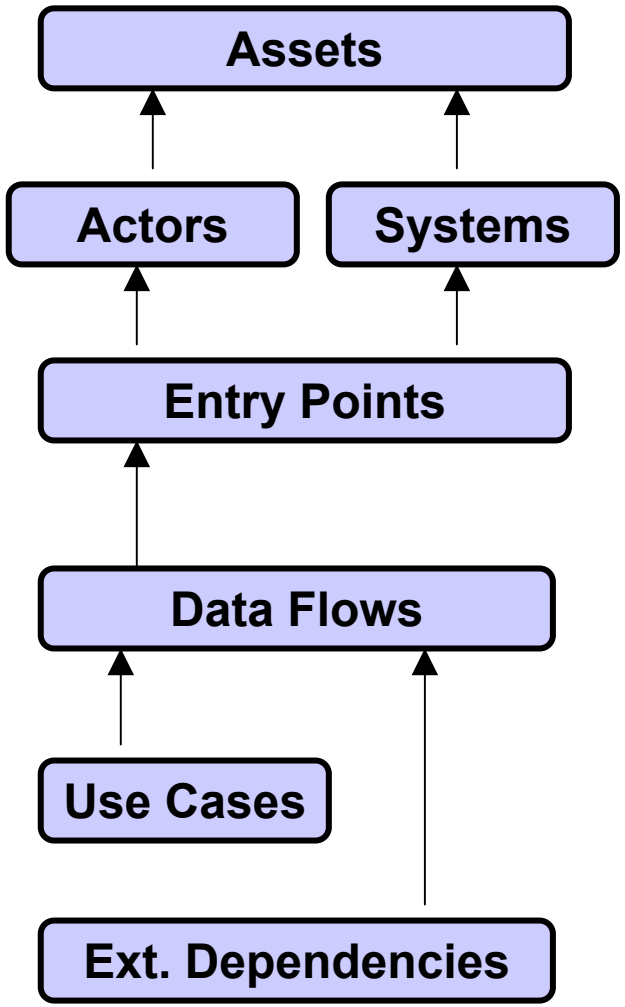
- Plan threat modelling exercise early in project mgmt:
  - Select suitable threat modelling methodology (**internal or external**)
  - Input requested from whom and when?
  - Output provided to whom and when?
  - Early kick-off (after this: update planning)
  - Estimate required SMEs*
- Consider exercise as a **quality gate**
- Use **RACI** to define responsibilities / estimate resources

*SME = Subject Matter Expert

Step 1: Assessment Definition

- Describe the application
  - Name, version, etc.
  - Business objectives
  - Sec requirements
  - Stakeholder
- Define **scope**
  - Target of Assessment (ToA)
  - Exclude platform, IDM, container, etc.
- Define **constrains**: Trust assumptions, etc.,
  - "Data from IDM or SAP FI system is trust worthy"
  - Irrelevant threat scenarios to be ignored

Step 2: Application Decomposition (AD)

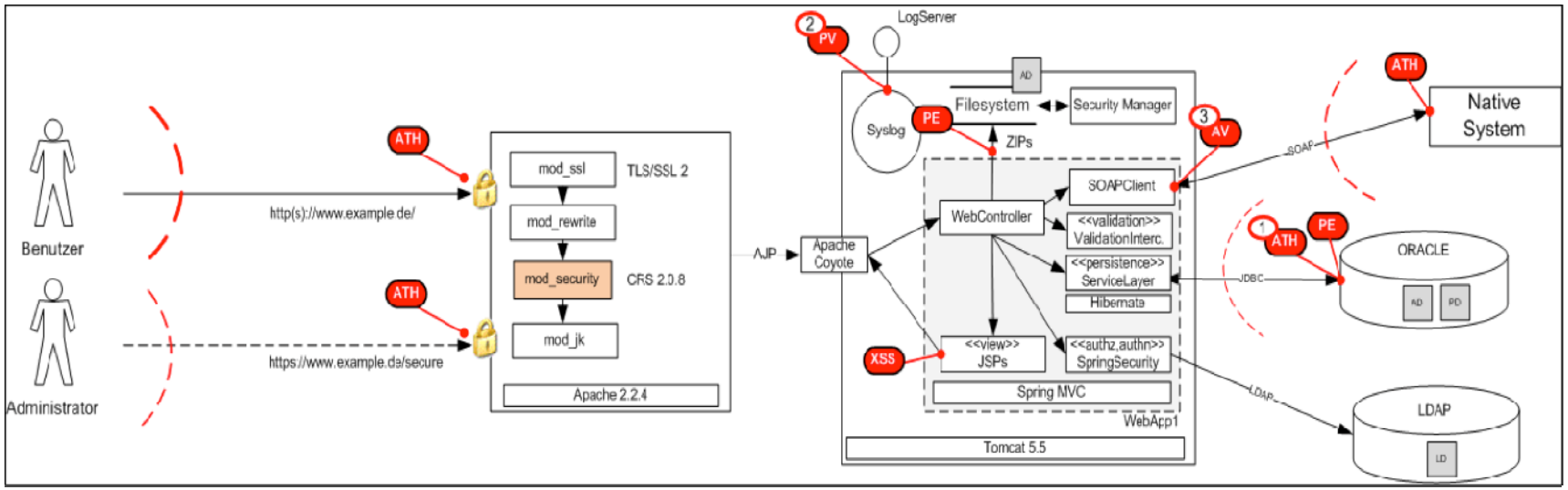


- **Identify** sub-systems, system boundaries and external dependencies.
- **Describe** assets, actors (including trust levels!), DfDs*, use cases*, entry points (channels)
- **Derive** (link) these information as shown left (e.g. using Word refs).
- This step may delivered as part of the development documentation.

* focus on DfDs and use cases that affect identified assets!

Step 2: AD: Application Overview

- Create a layer 7 view of the security architecture (no backup, cluster or other network devices).
- Don't bother with UML standards.
- Instead: use **hybrid diagrams**. Focus: **Visualisation**!

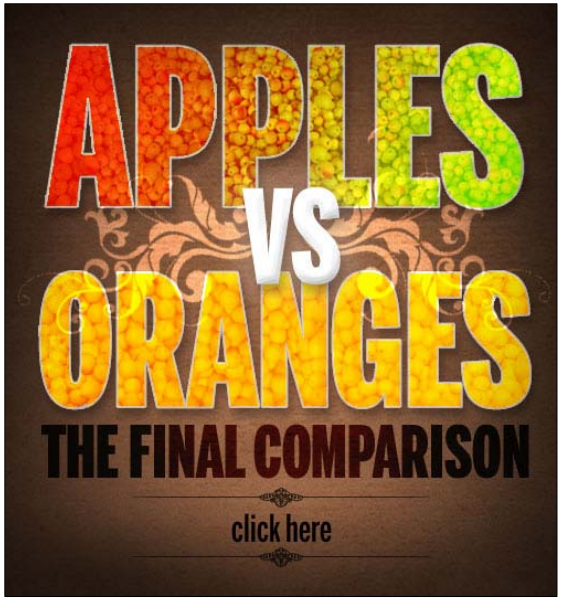


Dashed lines are **trust boundaries** (= architectural trust assumptions)

OWASP

Step 3: Clustering (optional)

- Applications can consists technically heterogeneous components leading to different **threat profiles**.
- Common example:
  - External Web interface for end-users
  - Internal admin GUI
- Clustering is used to identify such components and divide the **threat model** respectively.



Step 4: Threat Identification

- Objective: Maximization of coverage (don't be afraid of duplicates/overlapping threats!).
- Where/How may protection requirements of an assets be affected*:
  - **Primary**: Mainly confidentiality, integrity
  - **Secondary**: Authentication, loss of repudiation, etc.
  - **Indirect**: Design Principles (Least Priv., etc.)

* = potential damage to it

Step 4: Threat Identification – Building Blocks

- Questionnaires
- Attribute threat mapping
- Known vulnerability analysis
- Roles and permissions analysis
- Abuse & misuse case modelling
- Security control analysis
- Attack models / attack patterns
- Attack surface analysis
- Attack trees
- DFD analysis: STRIDE mapping, trust boundary analysis, …
- Input of pentests, other threat models, …

Step 4: Threat Identification - Tips

- Selection of activities depends on
  - **Protection requirements** (of the app)
  - Level of **maturity** (of the organisation)
  - **Qualification** (of the analyst)
  - **Resources** & time
- Tip: Do not focus on STRIDE*. Use own categories instead that helps you to derive threats from them:
  - e.g. "Threats regarding roles and permissions." (see example in appendix!)

* STRIDE = Spoofing identity, Tampering with data, Repudiation, Information disclosure DoS & Elevation of privilege. http://msdn.microsoft.com/en-us/library/ee823878.aspx

Step 4: Misuse & Abuse Cases

- **Misuse Case Modelling**
  - Based on use cases (of identified assets)
  - Analyze cases step-by-step:
    What could happened / should not happen that could cause damage to an asset?
- **Abuse Case Modelling**
  - Not based on use cases
  - What can a specific threat agent (e.g. admin, specific user such as a trader, hacker) do that could result in damage to an asset?

Step 4: Attribute Threat Mapping (ATM)

- Idea: Use threat intelligence to map application properties to generic (or known) threats (expert system).
- **Technical ATM (simple approach)**:

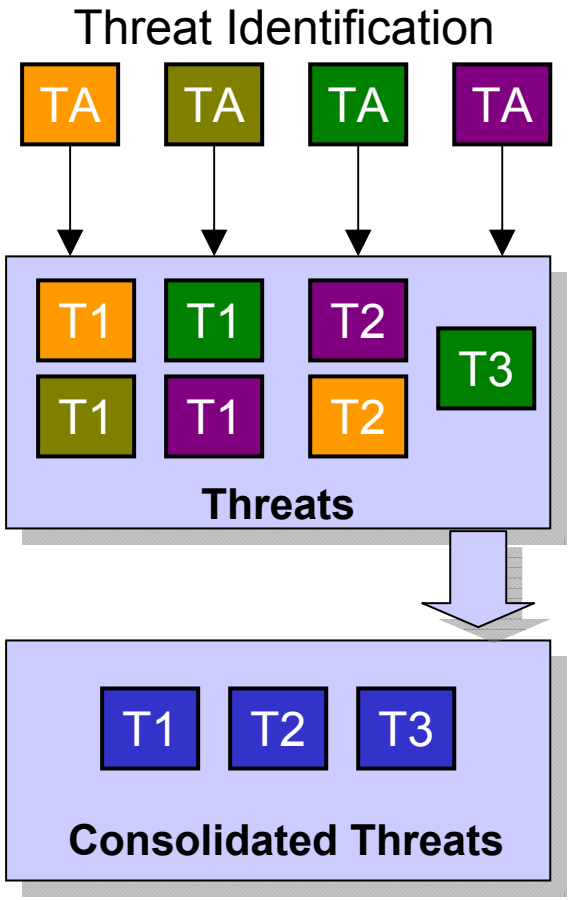
Attribute	Threats (Weaknesses, Attacks, BI)
Func.Register	▪ An attacker may enumerate users names ▪ Missing anti-automation
Func.Auth.Custom	▪ Insecure Session Identifier (CWE-330) ▪ Authentication Bypass (CAPEC-115) ▪ Insecure Password Storage (CWE-261) ▪ PW Eavesdropping (CAPEC-94)
Func.Auth.PWReset	▪ Weak Password Recovery (CWE-640)

Better approach: Map certain attributes using a logic (and, or, not) to specific threats.

- Create **threat profiles** for certain app types (e.g. collaboration, HR app, etc.)

Step 5: Threat Revision

- **Consolidation**
  Combine similar threats
- **Identify Mitigating Factors**
  Incl. controls, existing and planned



T = Threat
TA = Threat Ident. Activity

Step 5: Threat Revision

- **Consolidation**
  Combine similar threats
- **Identify Mitigating Factors**
  Incl. controls, existing and planned
- **Pre-Assessment (optional)**
  Check relevance / known issues

Step 6: Threat Rating

- **Threat Criticality Rating**
  - Option 1: DREAD: Criteria's are mapped indirectly to a numerical value using a metric (MS TM I)
    => Often very subjective!!
  - Option 2: CWSS: Similar to DREAD but more granularly and precise (= more work)
  - Option 3: Bug Bars: Criteria's that are mapped directly to low, medium, high, etc. (MS TM II)
  - …
- **Risk Assessment**
  - Threat Modelling → Risk Assessment

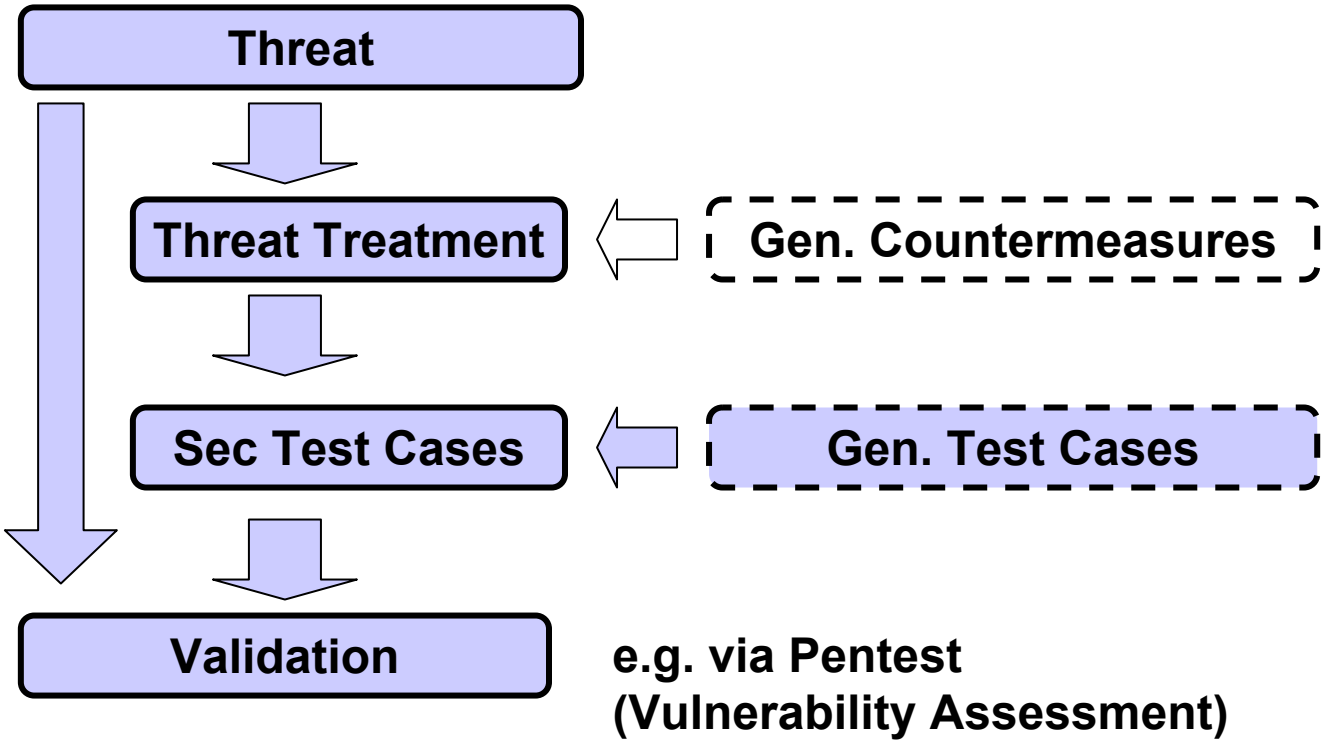
Bug Bars: http://msdn.microsoft.com/en-us/magazine/ee336031.aspx
CWSS: http://cwe.mitre.org/cwss/

Step 7: Threat Treatment (Countermeasures)

- Implemental
  - E.g. code changes
- Configurative
  - E.g. system hardening
- Architectural
  - E.g. installation of a PKI, IDM solution
- Other
  - Guidelines
  - Tests
  - …

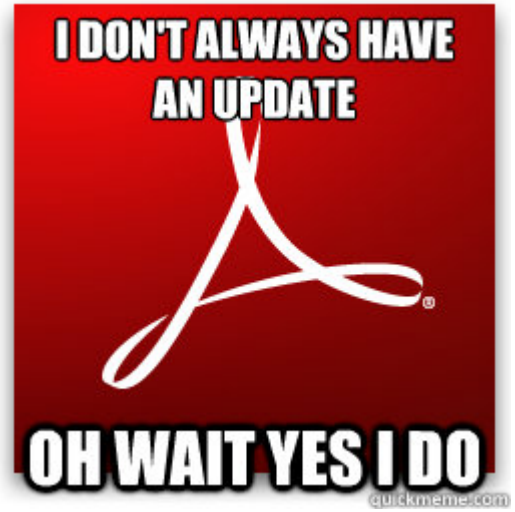
Threat 8: Threat Validation (Test Cases)

- Derive **test plan** & test cases from countermeasures
- Can easily include **generic test cases** (TI)
- Result: Threat-based security testing

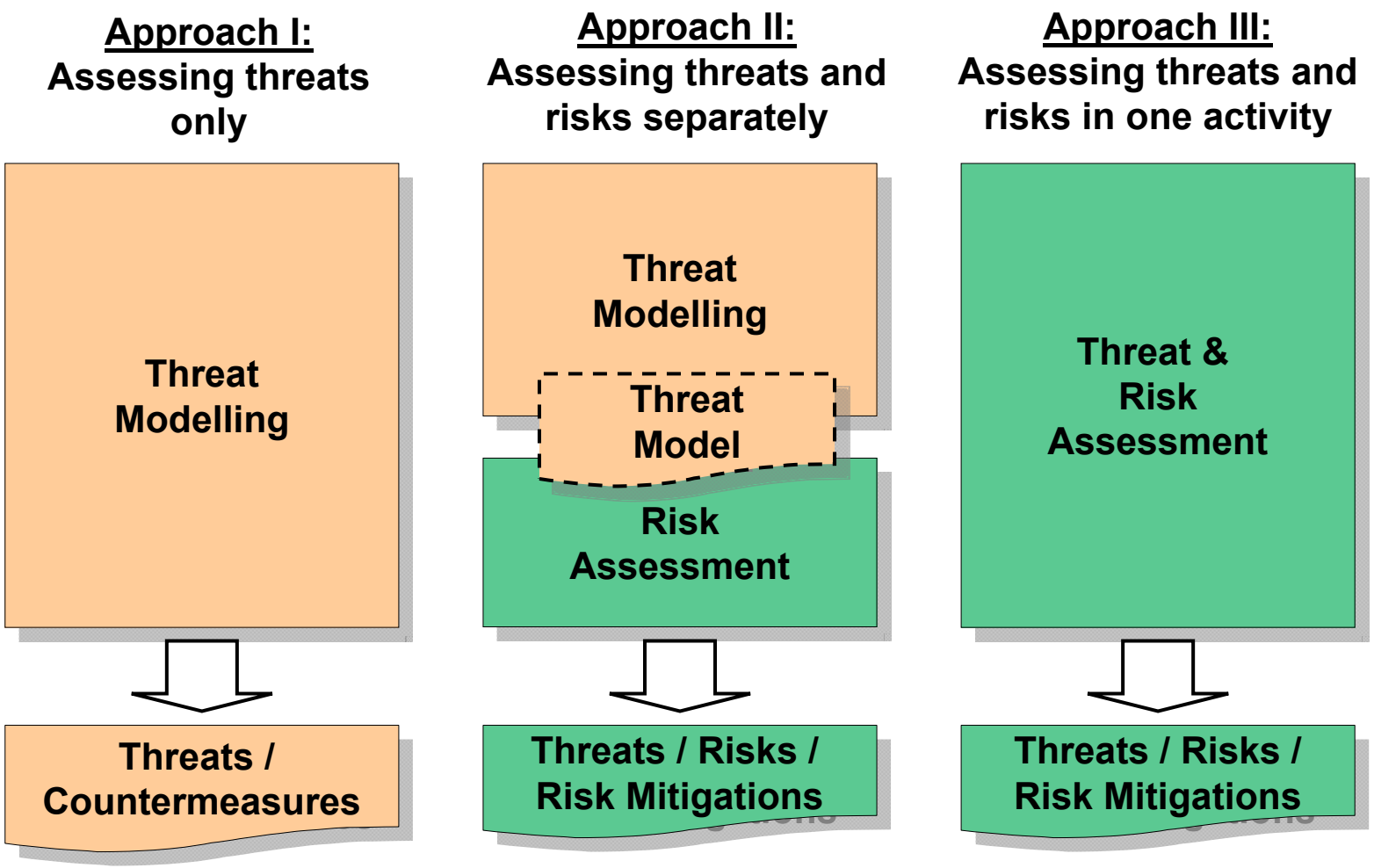


Step 9+10: Threat Retrospective & Update

- Update threat intelligence:
  - Known issues
  - Security test cases
  - Attribute threat mappings
  - Abuse cases
  - Metrics
  - …
- Continuous improvement of threat modelling exercises
- Update of the threat model after a specific time / changes



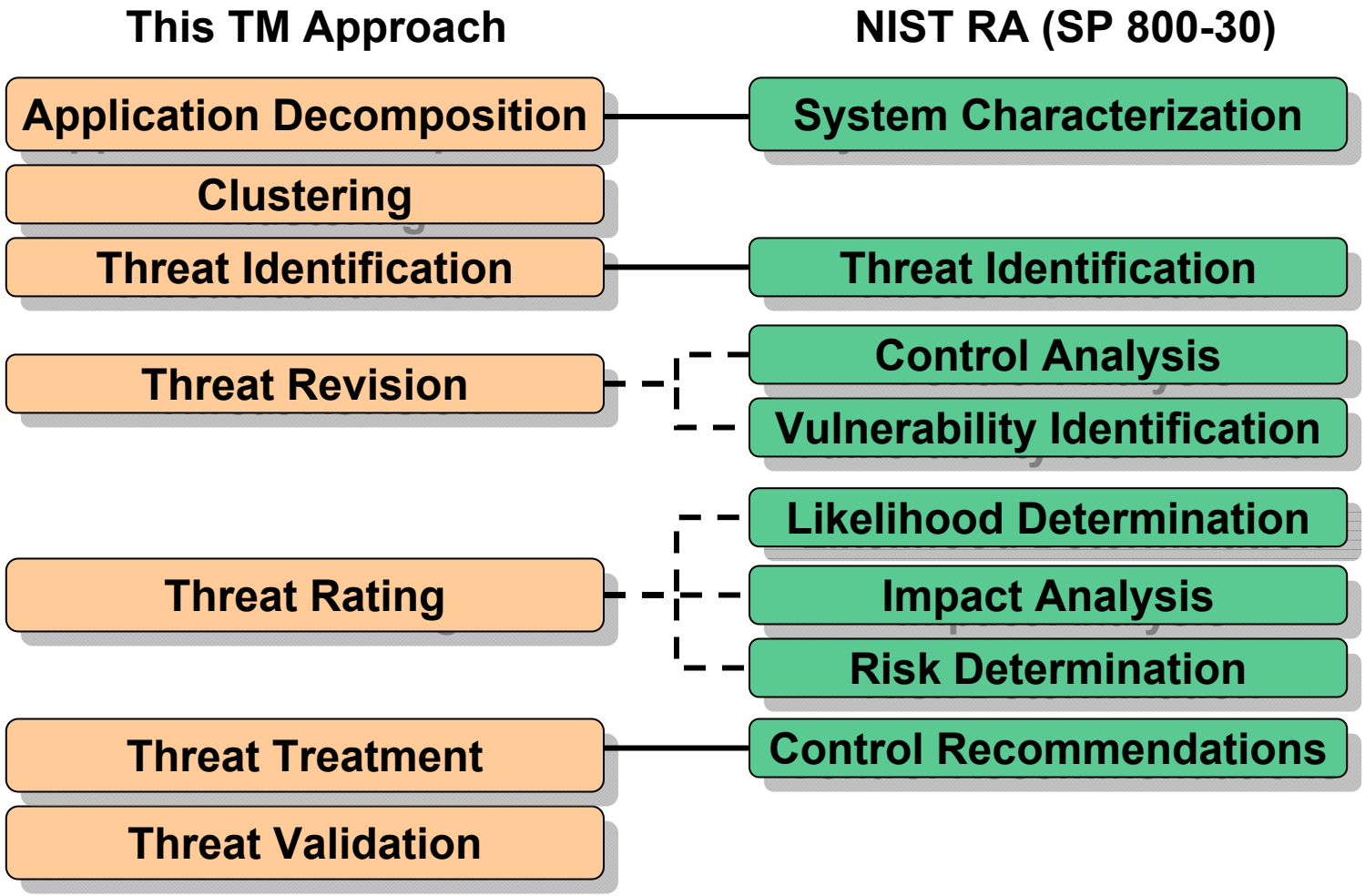
Threat Modelling & Risk Assessments



Try to implement approach II or III

OWASP

TM → RM: Example



Easy to combine both exercises. The WHERE is specific to an existing RM methodology!

OWASP

So Where to Start?

- Begin simple, informal and learn! (e.g. as a pilot)
- Collect threat intelligence wherever possible
  - **Lessons learned** after pentests, projects, etc.
- Integrate stakeholders: Dev team, TPMs, SME, pentester, etc.
- Build a **roadmap**:
  - Prioritize critical apps and platforms
  - Process maturity / SDLC integration
- **Get help**: E.g. let complicated threat models may be conducted by experienced consultants companies and learn from them!

Thank You! Any Questions???

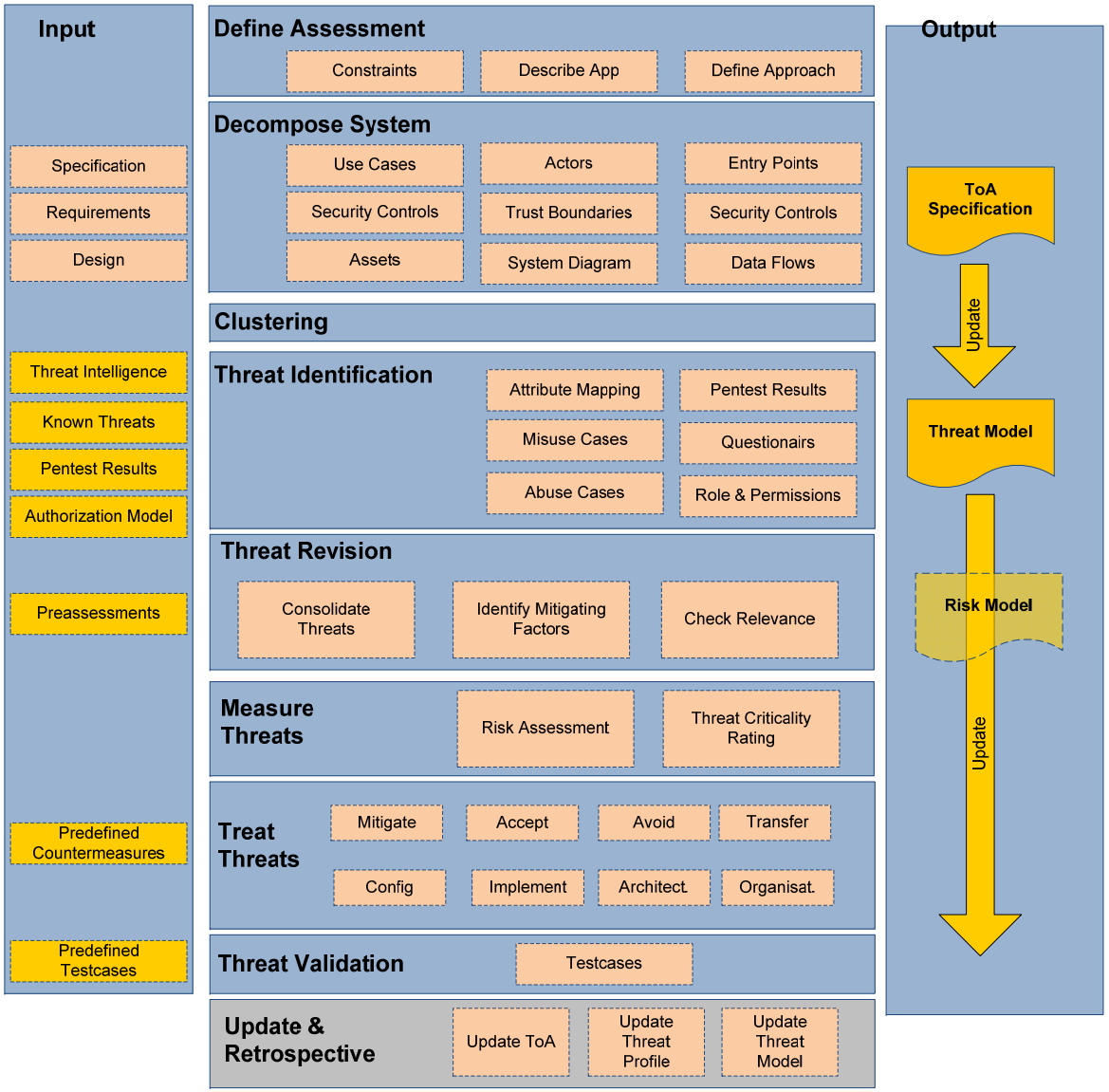
OWASP

APPENDIX

APPENDIX: Possible Threat Groups

- Insecure systems or missing hardening threats (HRD)
- Local threats (LOC)
- Threats by privileged users (PRV)
- Denial-of-Service threats (DOS)
- Threats to authentication & identities (ATN)
- Access control threats (ATZ)
- Threats regarding roles and permissions (RLP)
- Manipulation or disclosure of data in motion (DMM)
- Manipulation or disclosure of data at rest (DMR)
- Business-logic specific threats (BIL)
- Privacy threats (PRV)
- Accountability threats (ACC)

APPENDIX: Overview of Methodology



OWASP

APPENDIX: RACI Example

	Role			
Step	App Owner	Dev Team	Analyst	Sec Mgmt
Preparation	C	I	C	R/A
Assessment Definition	C	C	R	C/A
App Decomposition		C	R/A	
Threat Identification		C	R/A	
Threat Revision	C	C	R	I/A
Threat Rating	I	I	R	C/A
Define Action Plan	C	C	R	C/A

R – Responsible
A – Accountable
C - Consulted (in the loop)
I - Informed (in the picture)