# Whiteboard hacking – aka hands-on threat modeling

SECAPPDEV PRESENTATION 2019

## Sebastien Deleersnyder

- 5 years developer experience
- 15+ years information security experience
- Application security consultant Toreon
- Belgian OWASP chapter founder
- OWASP volunteer
- Co-founder BruCON

# Toreon: creating trust for a safer digital society

Security Governance & Privacy

Security Architecture

Ethical Hacking

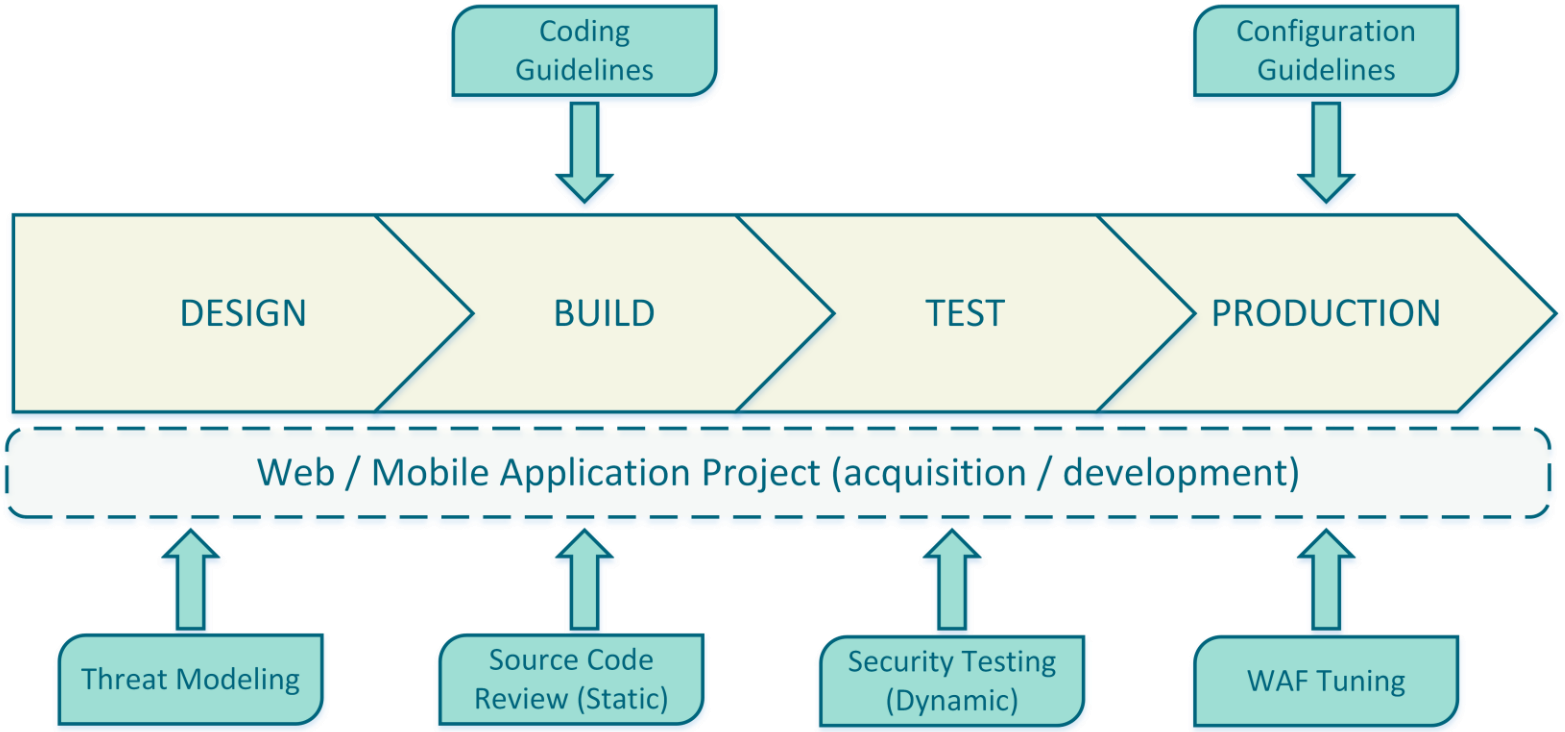
Application Security

Industrial Security & IOT

## Threat modeling introduction

1. Threat modeling in a secure development lifecycle
2. Threat modeling?
3. Threat modeling stages and examples
4. Integration in waterfall, agile and DevOps practices
5. Lessons learned
6. Threat modeling resources

## Secure development lifecycle


Coding Guidelines
Configuration Guidelines
DESIGN
BUILD
TEST
PRODUCTION
Web / Mobile Application Project (acquisition / development)
Threat Modeling
Source Code Review (Static)
Security Testing (Dynamic)
WAF Tuning

## Flaws versus bugs

### Security design flaws

- Errors in design, security requirements, architecture
- Need contextual knowledge
- No automation
- Costly to change in production

### Security coding bugs

- Coding errors
- Requires developers understanding secure coding
- Can be automated
- Patching less costly in production

## Threat modeling

- Threat modeling is the activity of identifying and managing application risks
- Also known as Architectural Risk Analysis

## Why perform threat modeling?

- Get team on same page with a shared vision on security
- Prevent security design flaws
- Identify & address greatest risks
- Prioritize development efforts based on risk weighting
- Increased risk awareness and understanding
- Cost justification and support for needed controls
- Document due diligence (GDPR…!)

## Different threat model methodologies (TMTOWTDI)

- STRIDE
- ATASM
- Pasta
- OCTAVE
- Trike
- VAST

## Threat modeling stages

**Step 1**

**Diagram**

**What are we building?**

**Step 2**

**Identify threats**

**What can go wrong?**

**Step 3**

**Mitigate**

**What are we doing to defend against threats?**

**Step 4**

**Validate**

**Validate steps 1-3**

**Report**

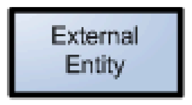
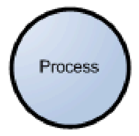
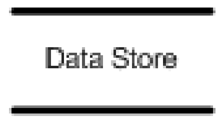
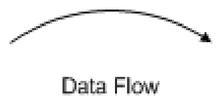
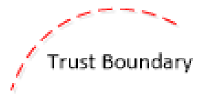
## Diagrams

- Define scope
- Good understanding context / objectives
- Understand how the software works
- Who interacts with the software?
- With Data Flow Diagrams, Sequence Diagrams, State diagrams ...
- Identify attack surfaces
- Foundation for threat analysis

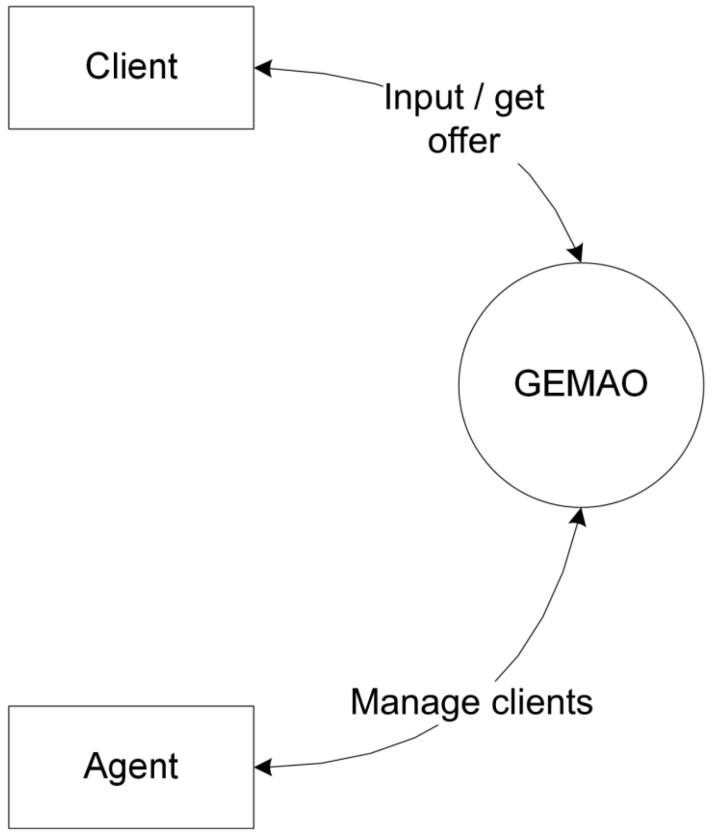
## Diagramming

- Preferably DFD or UML diagrams
  - Include processes, data stores, data flows
  - Include trust boundaries
  - Diagrams per scenario may be helpful
- Update diagrams as systems change
- Enumerate assumptions, dependencies
- Number or name everything

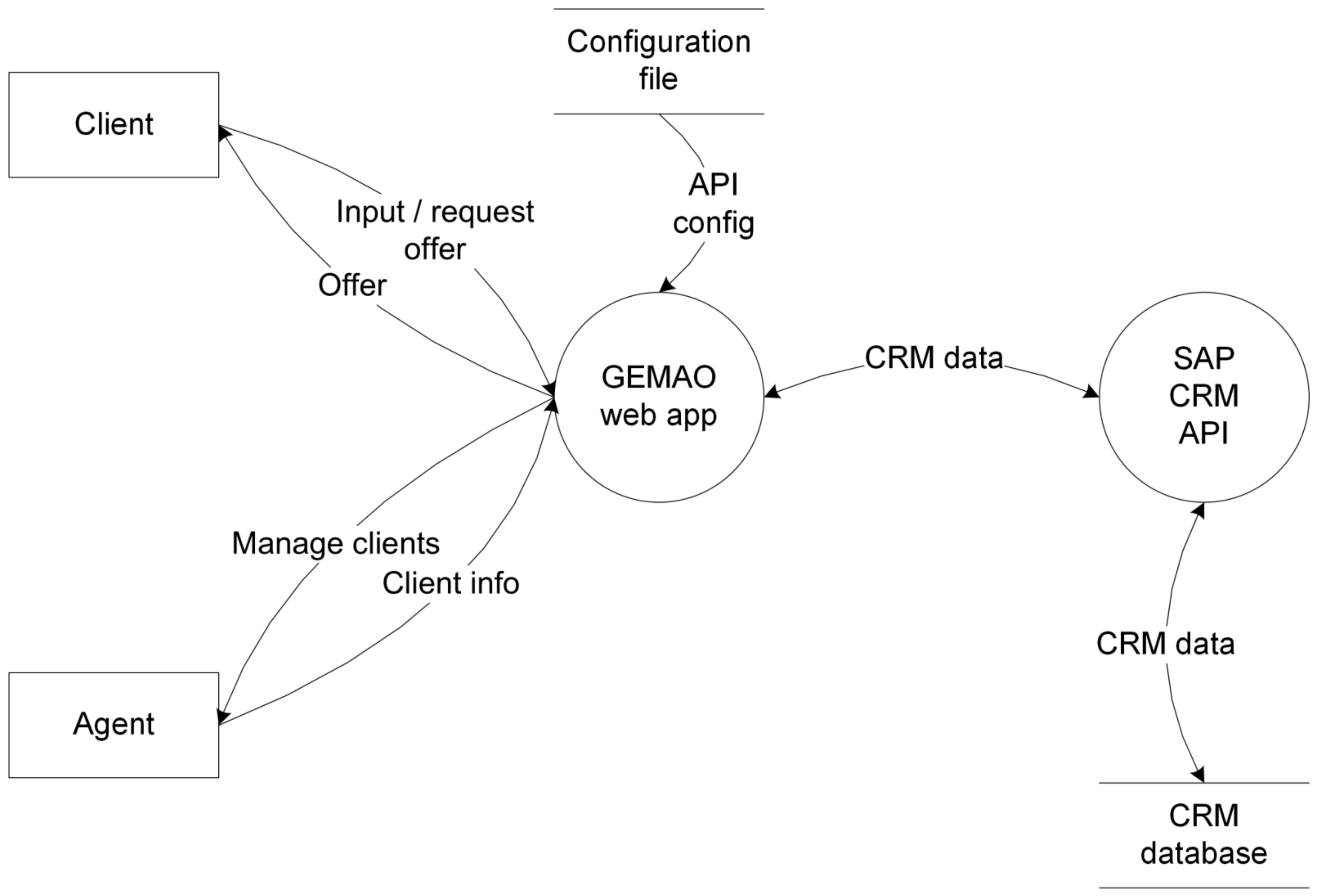
# DFD basics

| Symbol | | Description |
|---|---|---|
| External Entity |  | • Represents entities outside the application that interact with the application via an entry point |
| Process |  | • Represents tasks that handle data within the application; tasks may process data or perform actions based on the data |
| Data Store |  | • Represents locations where data is stored; data stores do not modify data, they only store it. |
| Data Flow |  | • Represents data movement within applications; the arrow tells the direction of data movement |
| Trust Boundary |  | • Represents the change of trust levels as data flows through the application |

## Context diagram GEMAO



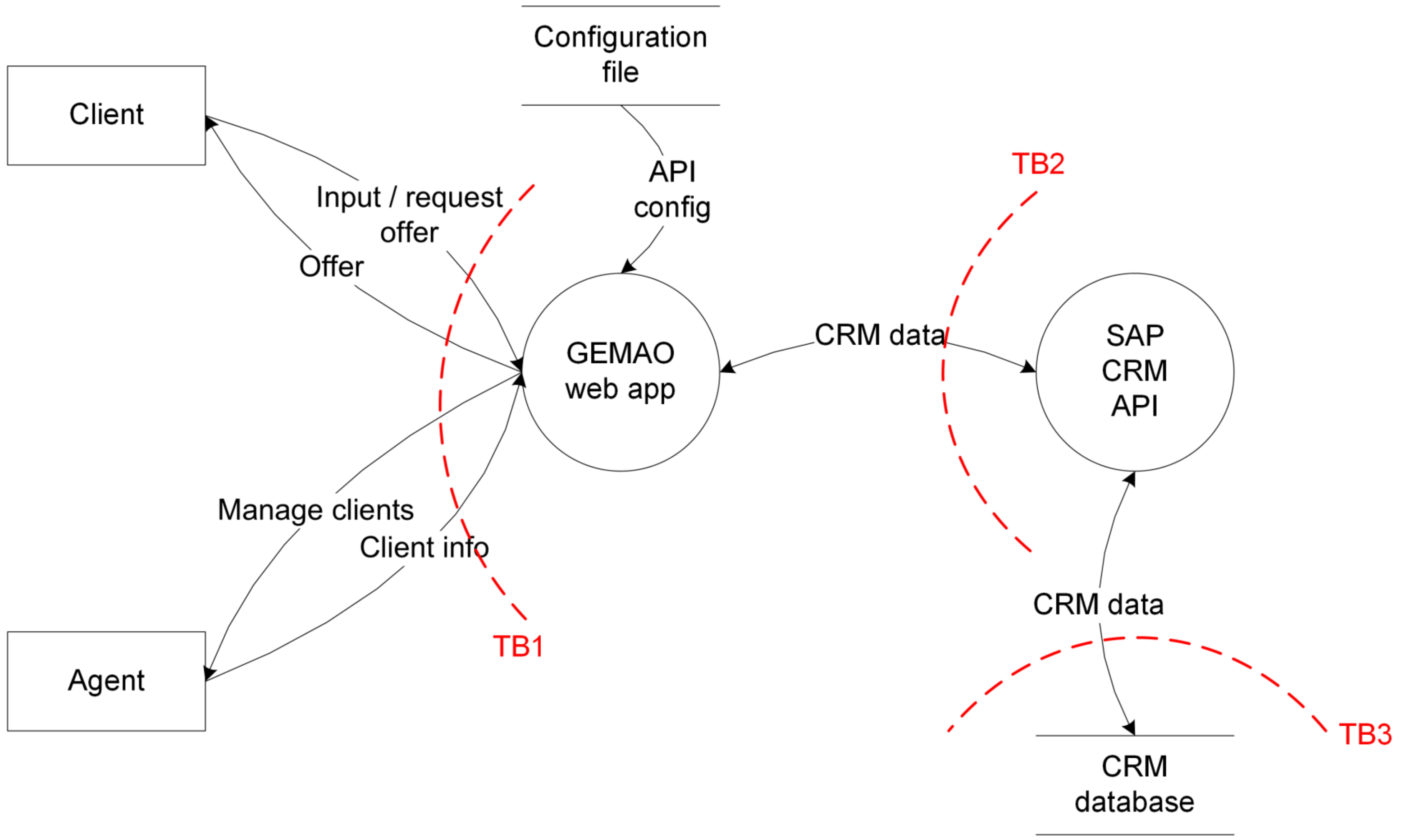
## DFD1 diagram GEMAO



## Diagrams – trust boundaries

- Trust boundaries intersect data flows
- Show where trust levels change
- Attack surface where an attacker can interject
- Examples: Machine boundaries, privilege boundaries, integrity boundaries
- Processes talking across a network always have a trust boundary

# Trust boundaries GEMAO



## Identify threats

- Based on diagrams
- STRIDE analysis
- Focus on identifying threats

|  |  |
|---|---|
| Spoofing | • Can an attacker gain access using a false identity? |
| Tampering | • Can an attacker modify data as it flows through the application? |
| Repudiation | • If an attacker denies doing something, can we prove he did it? |
| Information Disclosure | • Can an attacker gain access to private or potentially injurious data? |
| Denial of Service | • Can an attacker crash or reduce the availability of the system? |
| Elevation of Privilege | • Can an attacker assume the identity of a privileged user? |

## Apply STRIDE threats to each element

| | S | T | R | I | D | E |
|---|---|---|---|---|---|---|
| External Entity | ✓ | | ✓ | | | |
| Process | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Data Store | | ✓ | ? | ✓ | ✓ | |
| Data Flow | | ✓ | | ✓ | ✓ | |

# GEMAO threat table

| | Client / Agent | | > | | GEMAO | |
|---|---|---|---|---|---|---|
| **TB1** | **Mitigations** | **Vulnerabilities** | **Mitigations** | **Vulnerabilities** | **Mitigations** | **Vulnerabilities** |
| **S** | User / password authN | No 2FA for agent (V1) | | | TLS certificate | |
| **T** | | | TLS | | | No business validation input (V4) |
| **R** | | No audit trail (V2) | | | | No logging user actions (V5) |
| **I** | | | TLS | | | Clear text API credentials (V6) |
| **D** | | | | No fallback ISP (V3) | Load balanced web servers | |
| **E** | | | | | Access control | |

## Addressing threats

- Cover all threats
- Identify controls already in place
- Handle threats not (completely) covered

## Addressing each threat

Mitigation patterns

| | |
|---|---|
| Authentication | • Mitigating spoofing |
| Integrity | • Mitigating tampering |
| Non-repudiation | • Mitigating repudiation |
| Confidentiality | • Mitigating information disclosure |
| Availability | • Mitigating denial of service |
| Authorisation | • Mitigating elevation of privilege |

## Four ways to address threats

- Redesign to eliminate
- Apply standard mitigations
- Invent new mitigations (riskier)
- Accept vulnerability in design

## Risk-based threat management

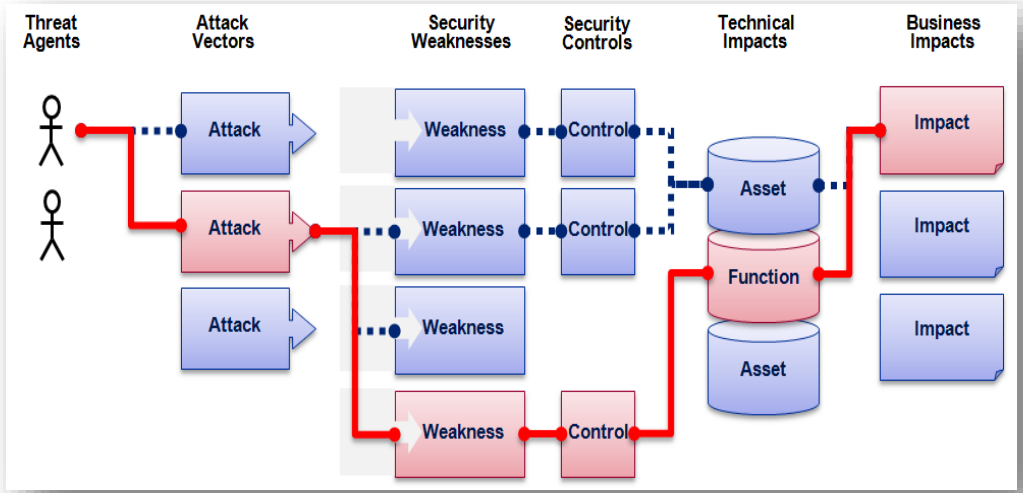
“The only truly secure system is one that is powered off, cast in a block of concrete, and sealed in a lead-lined room with armed guards - and even then I have my doubts. ”

Prof. Gene Spafford

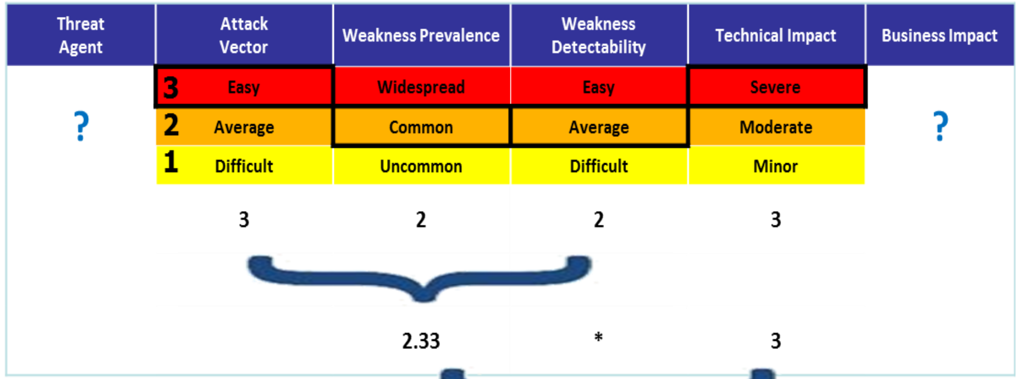
## Setting priorities

- Rank threats that are not mitigated or partially mitigated
- Be sure to cover the important threats
- Dual approach
  - Technical risk ranking, based on a quantified score
  - Business risk impact analysis

# OWASP risk rating



**Injection Example**



| Threat Agent | Attack Vector | Weakness Prevalence | Weakness Detectability | Technical Impact | Business Impact |
|---|---|---|---|---|---|
| | 3 Easy | Widespread | Easy | Severe | |
| ? | 2 Average | Common | Average | Moderate | ? |
| | 1 Difficult | Uncommon | Difficult | Minor | |
| | 3 | 2 | 2 | 3 | |

$$\underbrace{3 \quad 2 \quad 2}_{2.33} \quad 3$$

$$\underbrace{2.33 \quad * \quad 3}_{\text{7 weighted risk rating}}$$

## GEMAO example

| ID | Vulnerability | Vector | Prevalence | Detectability | Impact | Rating | Risk |
|---|---|---|---|---|---|---|---|
| V7 | No API endpoint restrictions | 3 | 2 | 2 | 3 | 7,0 | High |
| V4 | No business validation input | 2 | 2 | 2 | 3 | 6,0 | High |
| V1 | No 2FA for agent | 2 | 3 | 3 | 2 | 5,3 | Medium |
| V3 | No fall-back ISP | 2 | 2 | 1 | 3 | 5,0 | Medium |
| V6 | Clear text API credentials | 2 | 2 | 1 | 3 | 5,0 | Medium |
| V8 | Hardcoded DB credentials | 2 | 2 | 1 | 3 | 5,0 | Medium |
| V9 | Clear text DB connection | 2 | 2 | 1 | 3 | 5,0 | Medium |
| V11 | API access as DB admin | 2 | 2 | 1 | 3 | 5,0 | Medium |
| V2 | No audit trail | 1 | 2 | 2 | 1 | 1,7 | Low |
| V5 | No logging user actions | 1 | 2 | 2 | 1 | 1,7 | Low |
| V10 | No DB audit trail | 1 | 2 | 2 | 1 | 1,7 | Low |

**Low: <3, Medium: 4<=6, High: 7<=9**

## How to address threats - outline

1. Consider the enterprise context
2. Address threats in software with mitigation patterns
3. Add second and third order mitigations
4. Leverage proven security principles and tools

Specific mitigations for *your* threats

Generic advice that you should always keep in mind

## Document a threat model

- Most important step!
- Input for other security activities
- Basis for discussion
- Part of overall security documentation
- Describes (accepted) residual risk
- Input next iteration
- Update upon major changes in risk profile or software

## Communicate your threat model

To increase adoption

- Present the results to the audience, in person
- Discuss the countermeasures – cost vs. impact
- Complete the threat model with a proposed action list that you know is acceptable

**Architects**

- Should integrate the proposition to update the design

**Developers**

- Should benefit from the model transparently, through updated specification

**Security testing team**

- Now know precisely what to test!

**Software editor**

- If you are acquiring software, you can add the threat model to the software acceptance procedure

## Agile vs Waterfall

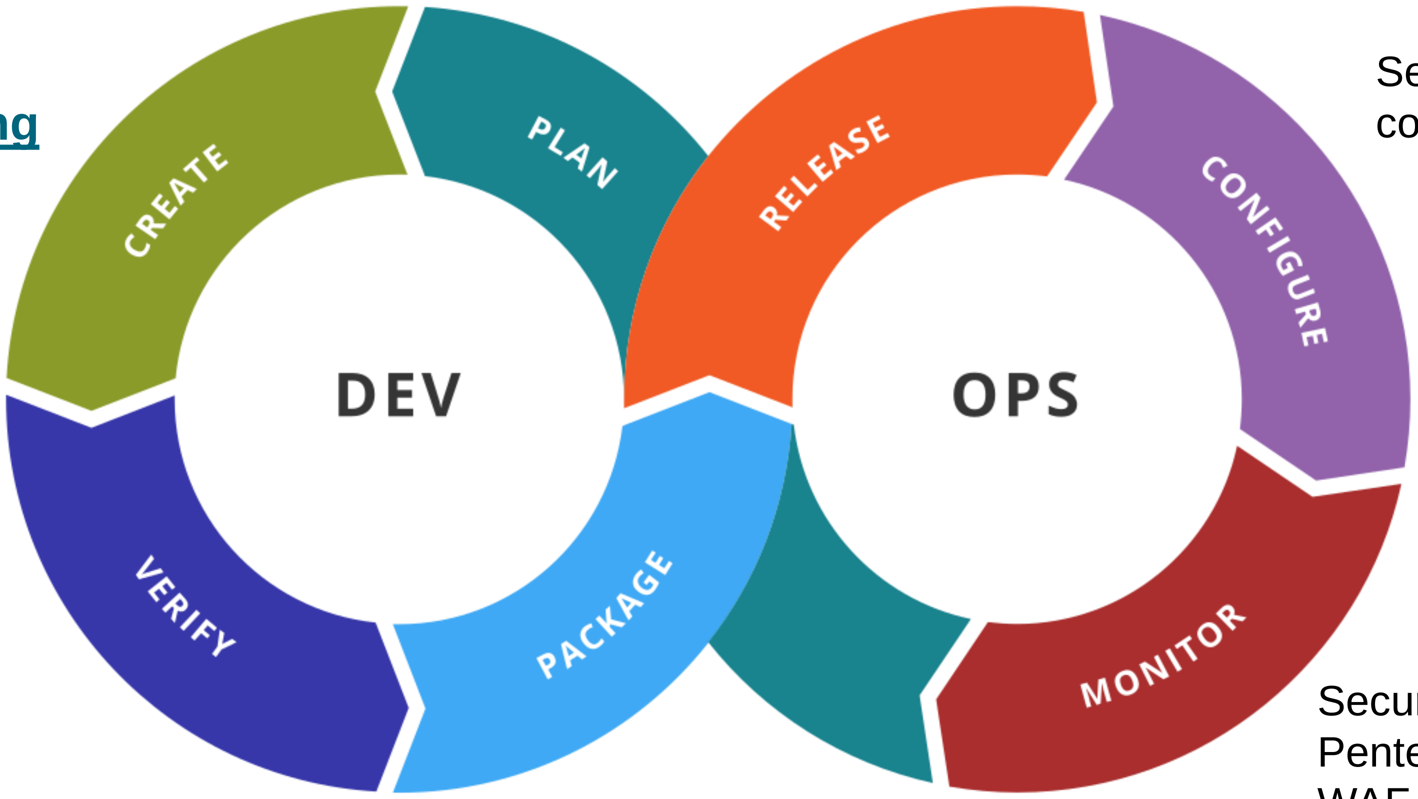
| Question | Waterfall | Agile |
|---|---|---|
| Q1: What are we building? | Big scope, all up front design | Small iterative and incremental design |
| Q2: What can go wrong? | Brainstorm<br>STRIDE<br>Attack trees<br>Attack libraries | Brainstorm<br>STRIDE<br>Attack trees<br>Attack libraries |
| Q3: What are we going to do about it? | Controls<br>Mitigation<br>Test cases | Same but put in the backlog (or epic) |
| Q4: Did we do a good enough job? | Test plans | Automated testing |

## Secure DevOps

Security
Requirements
Awareness - training

Threat Modeling
Secure Coding

Secure
configuration



DAST, SAST

Security monitoring
Pentesting, Bug Bounties
WAF

Dependency
checking

## Threat Modeling in DevOps

There are many benefits to help implement threat modeling in a DevOps environment:

- Most stakeholders are already part of the team
- Follow up through / integration with ticketing system
- ‘shift left’ is an enabler for threat modeling activities
- Speed gains through CI/CD pipeline

Threat modeling is one step in the ‘continuous security cycle’

## Lessons learned: what can go wrong

- Start complex threat modeling from the first time
- Wrong people in the workshops
- Sessions too long, without focus
- Sessions, reporting stretched too far in time
- Starting too early in the project
- Tool only approach

## Lessons learned: pleasant surprises

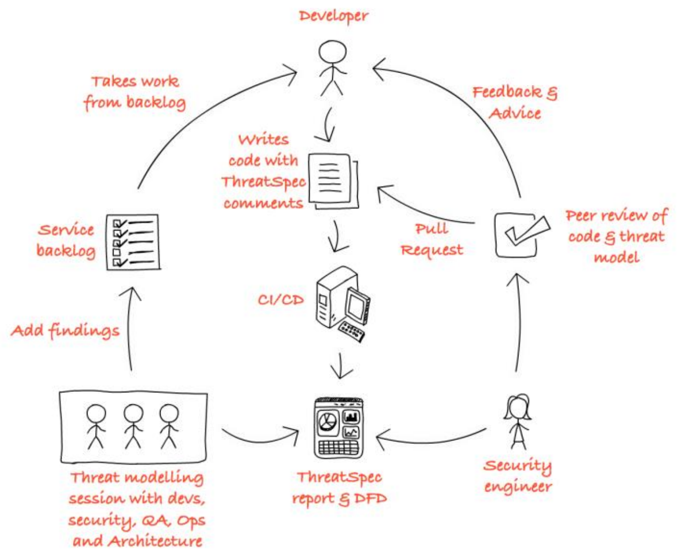
- Better common understanding of the project
- Feature based, short whiteboard sessions
- Applied to other domains: e.g. industrial control systems
- Ideal to scope and target security testing
- Great foundation for Privacy by Design

## Open source or free tools

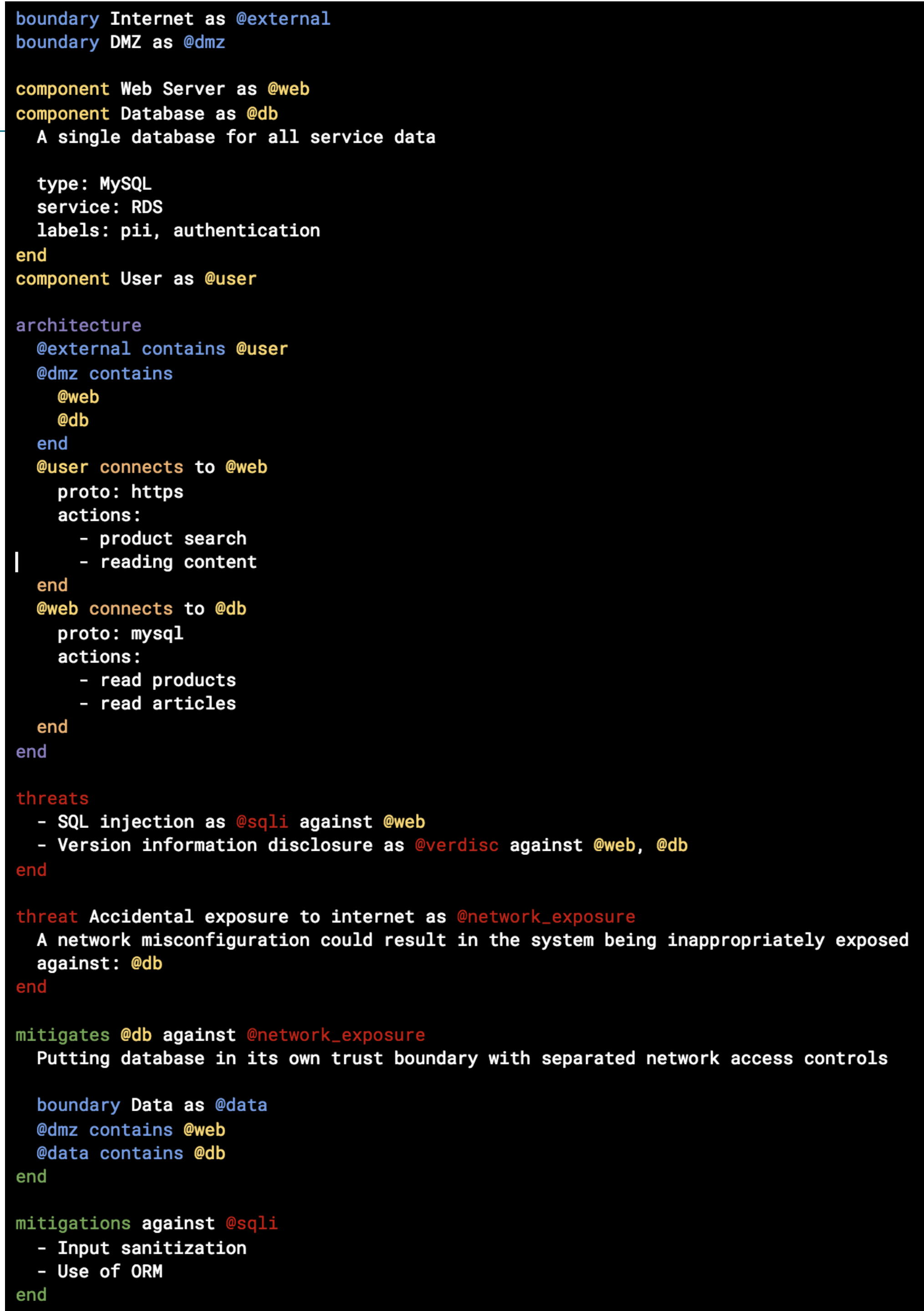
- Whiteboards!
- Mind-mapping diagramming tools such as FreeMind
- Microsoft threat modeling tool 2016 https://www.microsoft.com/en-us/download/details.aspx?id=49168
- OWASP threat dragon project https://www.owasp.org/index.php/OWASP_Threat_Dragon
- ThreatSpec - https://threatspec.org/
- Elevation of Privilege (EoP) card game https://www.microsoft.com/en-us/sdl/adopt/eop.aspx

## ThreatSpec

- Threat modeling through annotations
- Automatically generate DFDs
- Peer review and shared language developers and security


Developer
Takes work from backlog
Feedback & Advice
Writes code with ThreatSpec comments
Pull Request
Peer review of code & threat model
Service backlog
CI/CD
Add findings
Threat modelling session with devs, security, QA, Ops and Architecture
ThreatSpec report & DFD
Security engineer

https://threatspec.org/



```
boundary Internet as @external
boundary DMZ as @dmz

component Web Server as @web
component Database as @db
  A single database for all service data

  type: MySQL
  service: RDS
  labels: pii, authentication
end
component User as @user

architecture
  @external contains @user
  @dmz contains
    @web
    @db
  end
  @user connects to @web
    proto: https
    actions:
      - product search
      - reading content
  end
  @web connects to @db
    proto: mysql
    actions:
      - read products
      - read articles
  end
end

threats
  - SQL injection as @sqli against @web
  - Version information disclosure as @verdisc against @web, @db
end

threat Accidental exposure to internet as @network_exposure
  A network misconfiguration could result in the system being inappropriately exposed
  against: @db
end

mitigates @db against @network_exposure
  Putting database in its own trust boundary with separated network access controls

  boundary Data as @data
  @dmz contains @web
  @data contains @db
end

mitigations against @sqli
  - Input sanitization
  - Use of ORM
end
```

## Commercial tools (no particular order)

- Microsoft Visio (Windows)
- ConceptDraw Pro (MacOS)

- IriusRisk by Continuum Security https://iriusrisk.continuumsecurity.net/
- MyAppSecurity ThreatModeler http://myappsecurity.com/threatmodeler/
- Security Compass SD Elements https://www.securitycompass.com/threatmodeling/

## Books

- Threat modeling, designing for security (Adam Shostack, MS)
- Securing systems, applied security architecture and threat models (Brook Schoenfield)
- Risk centric threat modeling: process for attack simulation and threat analysis (P.A.S.T.A) (Marco Morana and Tony “UV.”)

## Toreon Threat Modeling Newsletter

- Monthly newsletter
- Threat modeling news digest
- Toreon and other resources
- Direct access to:
  - whitepapers,
  - whiteboard hacking survival guide
  - templates
  - presentations

**http://eepurl.com/ghAr8b**

## Thank you

## That’s All Folks!

You can contact me through:

| | |
|---|---|
| • Toreon: | seba@toreon.com |
| • Private/OWASP: | seba@owasp.org |
| • Twitter: | @SebaDele |

- OWASP TM Slack channel