UNIVERSITÄT PASSAU

Fakultät für Informatik und Mathematik

6090: Security of Computer and Embedded Systems

Week 3: Secure Software Development Lifecycle (SSDL); Threat Modelling

Elif Bilge Kavun

elif.kavun@uni-passau.de

November 02, 2021

This Week's Outline

- "Building Secure Systems"
- Secure Software Development Lifecycle (SSDL)
  - Path Towards Secure(r) Software
- Threat Modelling
  - Real-world approaches and needs
  - STRIDE

“Building Secure Systems”

- Focus of the next three weeks
  - How to build secure systems
  - How to build systems securely

- Focus of the next three weeks
  - How to build secure systems
  - How to build systems securely

- In other words: Providing secure(r) software

- Focus of the next three weeks
  - How to build secure systems
  - How to build systems securely
- In other words: Providing secure(r) software
- Focus later
  - Security systems
  - How to build them correctly

Secure Software Development Lifecycle (SSDL)

Vault 7: CIA Hacking Tools Revealed

March 7, 2017 – https://wikileaks.org/ciav7p1/

- Summary
  - It is a leak about the CIA's hacking arsenal used against foreign governments and citizens both domestically and abroad
- Origin
  - Allegedly from the CIA's Center for Cyber Intelligence unit in Langley, Virginia USA
- Volume
  - 7,818 web pages with 943 attachments
- Time-frame
  - Documents are from 2013-2016
- First analysis
  - No evidence that crypto is broken!
  - Very effective techniques for exploiting software bugs!
    - Use of known bugs as well as unknown bugs (zero days)

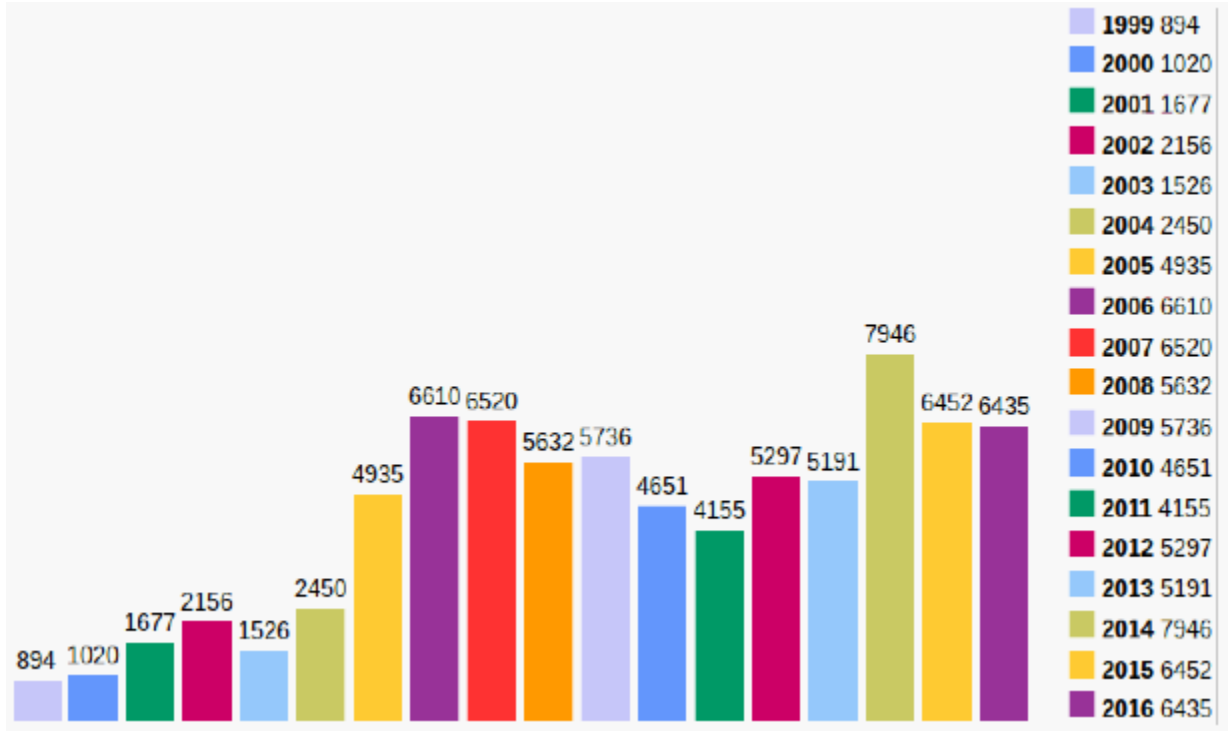
Vault 7: CIA Hacking Tools Revealed

March 7, 2017 – https://wikileaks.org/ciav7p1/

- Summary
  - It is a leak about the CIA's hacking arsenal used against foreign governments and citizens both domestically and abroad
- Origin
  - Allegedly from the CIA's Center for Cyber Intelligence unit in Langley, Virginia USA
- Volume
  - 7,818 web pages with 943 attachments
- Time-frame
  - Documents are from 2013-2016
- First analysis
  - No evidence that crypto is broken!
  - Very effective techniques for exploiting software bugs!
    - Use of known bugs as well as unknown bugs (zero days)
- First conclusion
  - *The quality (security, correctness, . . .) of the developed software needs to be improved!*

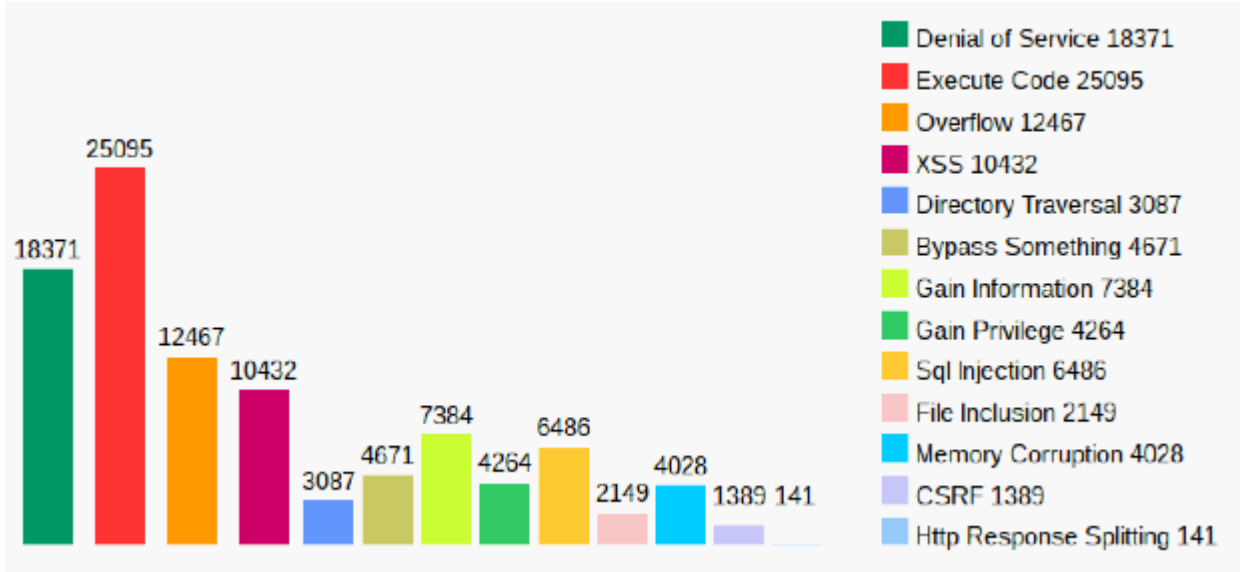
Not a New Problem: Vulnerability Distribution (Since 1999)

www.cvedetails.com



Vulnerability by Year

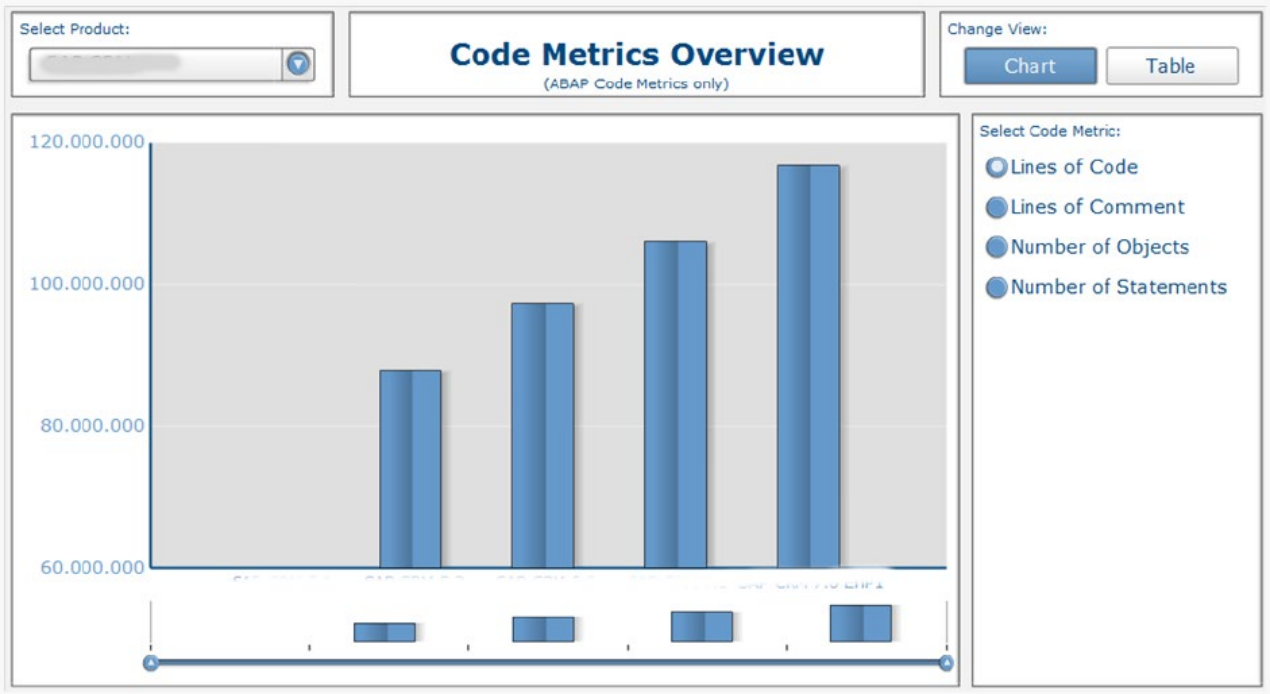
Vulnerability Types



- *Still*, attacks on crypto not a common problem – but a lot of "rotten" software...

Why Is It So Hard to Get Software "Right"?

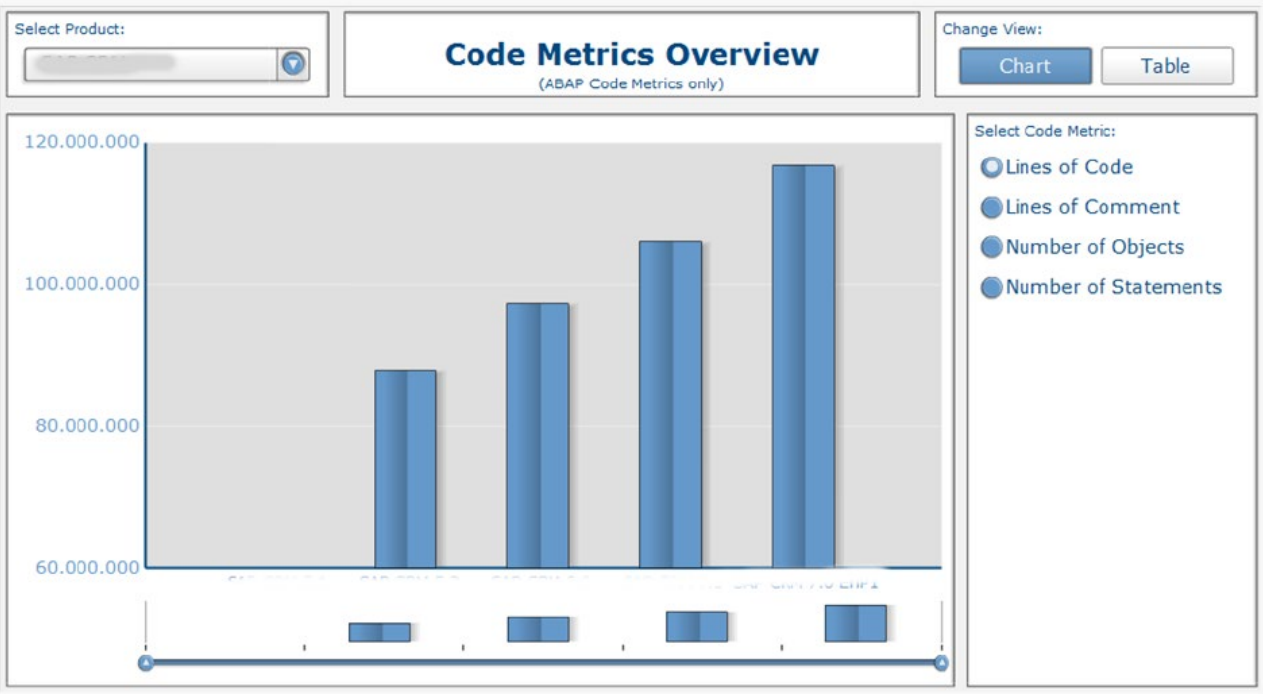
- Evolution of Source Code



- Increase in
  - Code size
  - Code complexity
  - Number of products
  - Product versions
  - Used technologies (programming languages, frameworks)

Why Is It So Hard to Get Software "Right"?

- Evolution of Source Code



- Increase in
  - Code size
  - Code complexity
  - Number of products
  - Product versions
  - Used technologies (programming languages, frameworks)

- Software Maintenance Challenge
  - Increased user demand → Increased number of systems → Longer maintenance periods required

A Path Towards (More) Secure Software

Secure Software Development Lifecycle (SSDL)

A Path Towards (More) Secure Software

Secure Software Development Lifecycle (SSDL)

- Security awareness
- Secure programming
- Threat modelling
- Security testing
- Data protection and privacy
- Security expert curriculum ("Masters")

A Path Towards (More) Secure Software

Secure Software Development Lifecycle (SSDL)

- Risk identification ("high-level threat modelling")
- Threat modelling
- Data privacy impact assessment

A Path Towards (More) Secure Software

Secure Software Development Lifecycle (SSDL)

- Plan product standard compliance
- Plan security features
- Plan security tests
- Plan security response

A Path Towards (More) Secure Software

Secure Software Development Lifecycle (SSDL)

- Secure programming
- Static code analysis (SAST)
- Code review
- Dynamic testing (e.g., IAST, DAST)
- Manual testing
- External security assessment

A Path Towards (More) Secure Software

Secure Software Development Lifecycle (SSDL)

- Check for "flaws" in the implementation of the SSDL
- Ideally, security validation finds
  - No issues that can be fixed/detected earlier
  - Only issues that cannot be detected earlier (e.g., insecure default configurations, missing security documentation)

Penetration tests in productive environments are different

- They test the actual configuration
- They test the productive environment (e.g., cloud/hosting)

A Path Towards (More) Secure Software

Secure Software Development Lifecycle (SSDL)

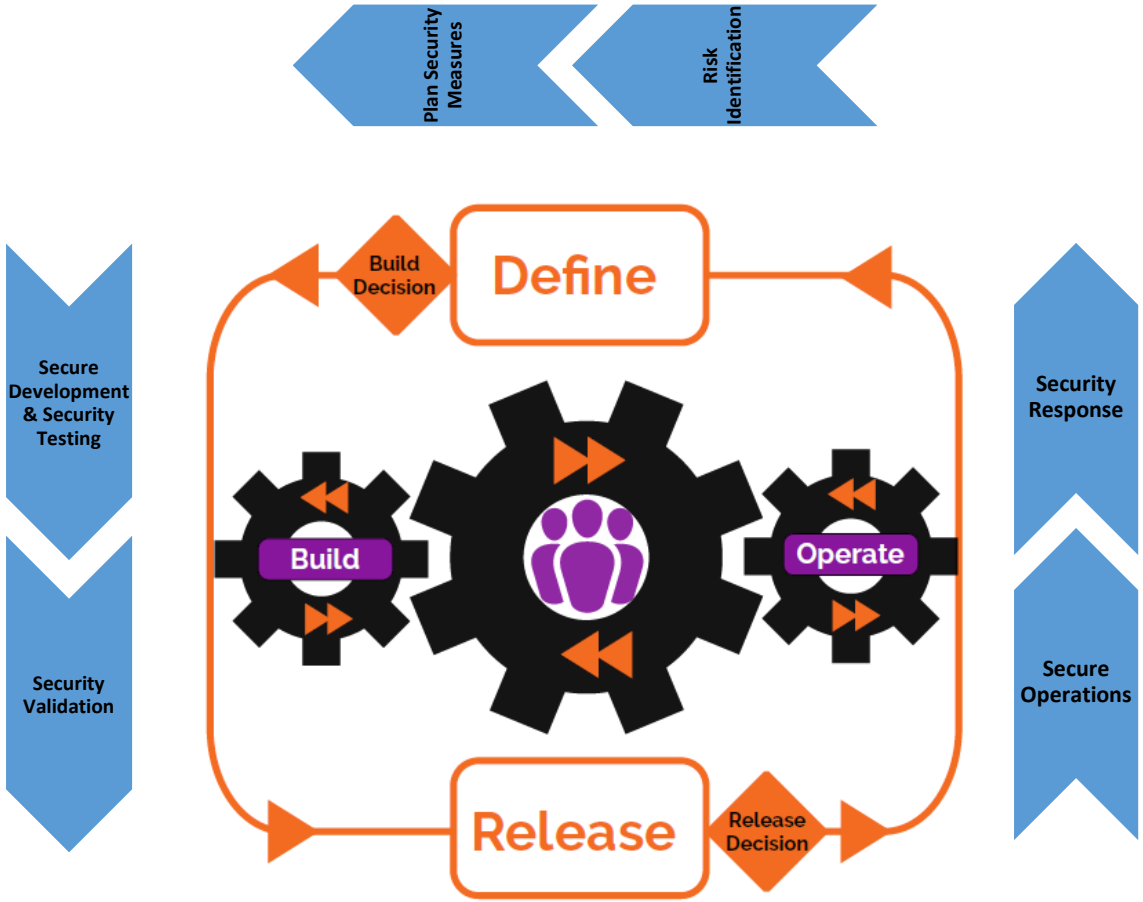
- Understanding security operations concepts
- Need-to-know/least privilege
- Separation of duties and responsibilities
- Monitor special privileges (e.g., operators, administrators)
- Marking, handling, storing, and destroying of sensitive information

A Path Towards (More) Secure Software

Secure Software Development Lifecycle (SSDL)

- Execute the security response plan
- Security related external communication
- Incident handling
- Security patches
- Monitoring of third party components

Secure Software Development Lifecycle for Cloud/Agile



Summary

- Application/software security is important
- Attackers (and governments) exploit software vulnerabilities
  - Recall the famous iPhone case: *FBI paid more than $1 Million to access iPhone*
  - Attacks on actual crypto (i.e., the math) are seldom (but equally effective)
- Application/software security is mostly a software engineering/programming problem!

Threat Modelling

Motivation

Observation		
Securing systems is expensive		Not all systems are equally rewarding for attackers

Motivation

Observation		
Securing systems is expensive	⟷	Not all systems are equally rewarding for attackers

- Let’s consider you want to secure your bike

Motivation

Observation		
Securing systems is expensive	⟷	Not all systems are equally rewarding for attackers

- Let’s consider you want to secure your bike
  - What do you want to protect?

Motivation

Observation		
Securing systems is expensive	⟷	Not all systems are equally rewarding for attackers

- Let’s consider you want to secure your bike

- What do you want to protect?
  - Your old city bike

Motivation

Observation		
Securing systems is expensive	⟷	Not all systems are equally rewarding for attackers

- Let’s consider you want to secure your bike

- What do you want to protect?
  - Your old city bike
  - Your new stylish bike

Motivation

Observation		
Securing systems is expensive	⟷	Not all systems are equally rewarding for attackers

- Let’s consider you want to secure your bike

- What do you want to protect?
  - Your old city bike
  - Your new stylish bike
- Against whom?

Motivation

Observation		
Securing systems is expensive	⟷	Not all systems are equally rewarding for attackers

- Let’s consider you want to secure your bike

- What do you want to protect?
  - Your old city bike
  - Your new stylish bike
- Against whom?
  - The casual attacker

Motivation

Observation		
Securing systems is expensive	⟷	Not all systems are equally rewarding for attackers

- Let’s consider you want to secure your bike

- What do you want to protect?
  - Your old city bike
  - Your new stylish bike
- Against whom?
  - The casual attacker
  - Targeted attack

Motivation

Observation	
Securing systems is expensive	Not all systems are equally rewarding for attackers

- Let’s consider you want to secure your bike

- What do you want to protect?
  - Your old city bike
  - Your new stylish bike
- Against whom?
  - The casual attacker
  - Targeted attack
- Available countermeasures

Motivation

Observation		
Securing systems is expensive	⟷	Not all systems are equally rewarding for attackers

- Let’s consider you want to secure your bike

- What do you want to protect?
  - Your old city bike
  - Your new stylish bike
- Against whom?
  - The casual attacker
  - Targeted attack
- Available countermeasures
  - A cheap bike lock

Motivation

Observation		
Securing systems is expensive	⟷	Not all systems are equally rewarding for attackers

- Let’s consider you want to secure your bike

- What do you want to protect?
  - Your old city bike
  - Your new stylish bike
- Against whom?
  - The casual attacker
  - Targeted attack
- Available countermeasures
  - A cheap bike lock
  - An expensive lock

Motivation

Observation	
Securing systems is expensive	Not all systems are equally rewarding for attackers

- Let’s consider you want to secure your bike

- What do you want to protect?
  - Your old city bike
  - Your new stylish bike
- Against whom?
  - The casual attacker
  - Targeted attack
- Available countermeasures
  - A cheap bike lock
  - An expensive lock
- Most vulnerable points

Motivation

Observation		
Securing systems is expensive	⟷	Not all systems are equally rewarding for attackers

- Let’s consider you want to secure your bike

- What do you want to protect?
  - Your old city bike
  - Your new stylish bike
- Against whom?
  - The casual attacker
  - Targeted attack
- Available countermeasures
  - A cheap bike lock
  - An expensive lock
- Most vulnerable points
  - Locking the front wheel only

Motivation

Observation		
Securing systems is expensive	⟷	Not all systems are equally rewarding for attackers

- Let’s consider you want to secure your bike

- What do you want to protect?
  - Your old city bike
  - Your new stylish bike
- Against whom?
  - The casual attacker
  - Targeted attack
- Available countermeasures
  - A cheap bike lock
  - An expensive lock
- Most vulnerable points
  - Locking the front wheel only
  - Locking the frame

Motivation

Observation		
Securing systems is expensive	⟷	Not all systems are equally rewarding for attackers

- Let’s consider you want to secure your bike

- What do you want to protect?
  - Your old city bike
  - Your new stylish bike
- Against whom?
  - The casual attacker
  - Targeted attack
- Available countermeasures
  - A cheap bike lock
  - An expensive lock
- Most vulnerable points
  - Locking the front wheel only
  - Locking the frame

Threat Modelling As Part of A SSDL

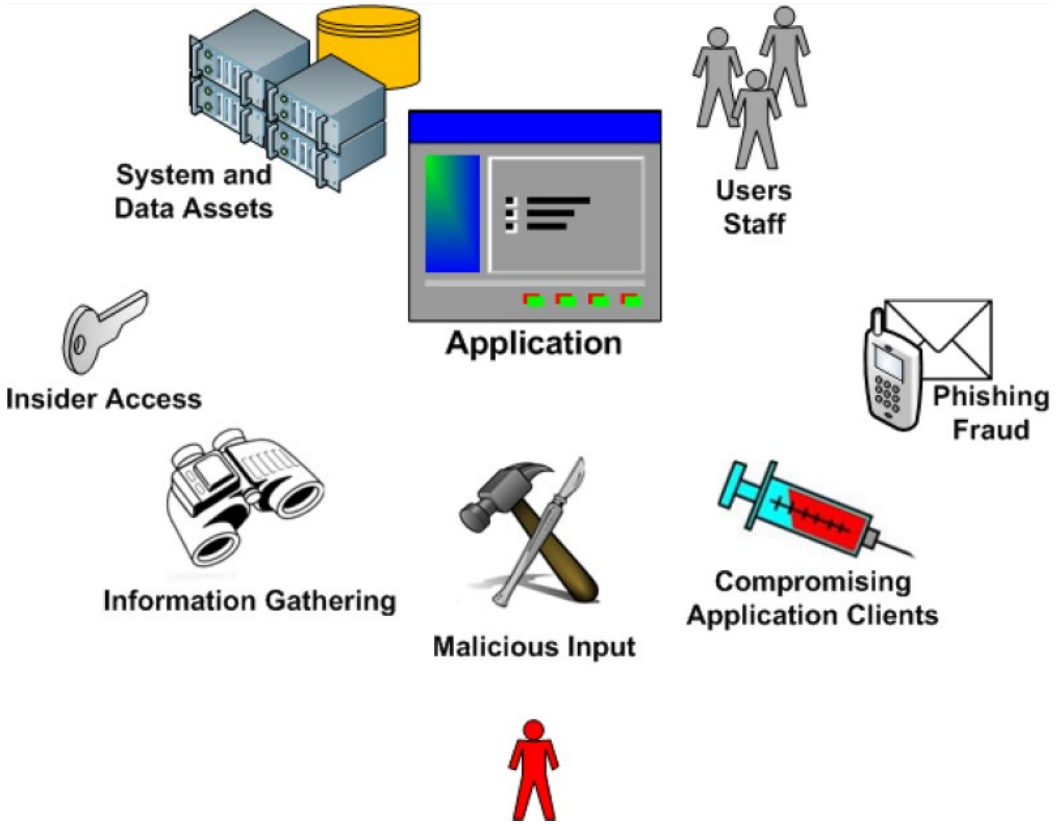
- **Threat modelling is a process, usually as part of the early steps of software development, by which potential threats are identified, enumerated, and prioritized**

Threat Modelling As Part of A SSDL

- **Threat modelling is a process, usually as part of the early steps of software development, by which potential threats are identified, enumerated, and prioritized**

- Think like an attacker
  - Where are the high-value assets?
  - Where am I most vulnerable to attack?
  - What are the most relevant threats?
  - Is there an attack vector that might go unnoticed?

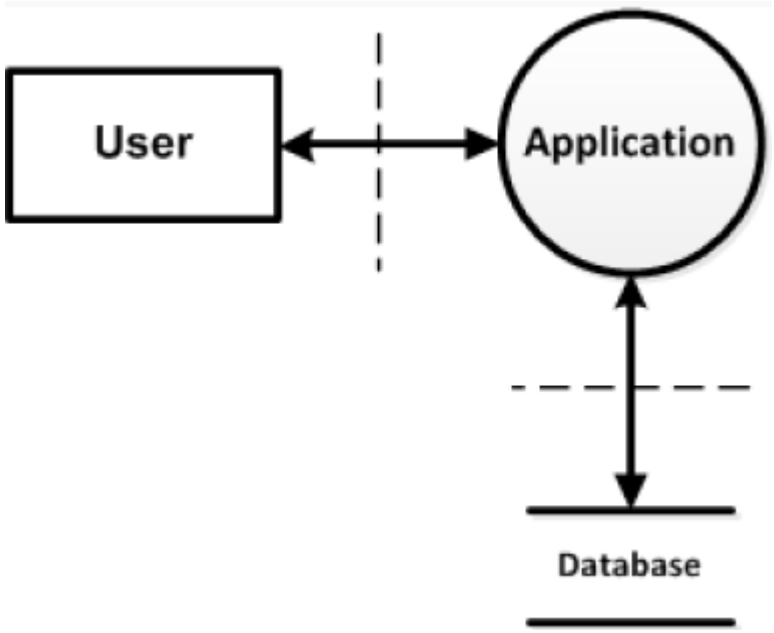
Understanding the Threats and Risks



- High-level attack vectors
  - Defeating a security mechanism
  - Abusing an application feature
  - Exploiting the insufficient security or poor implementation
- Remember, your application is part of a larger system

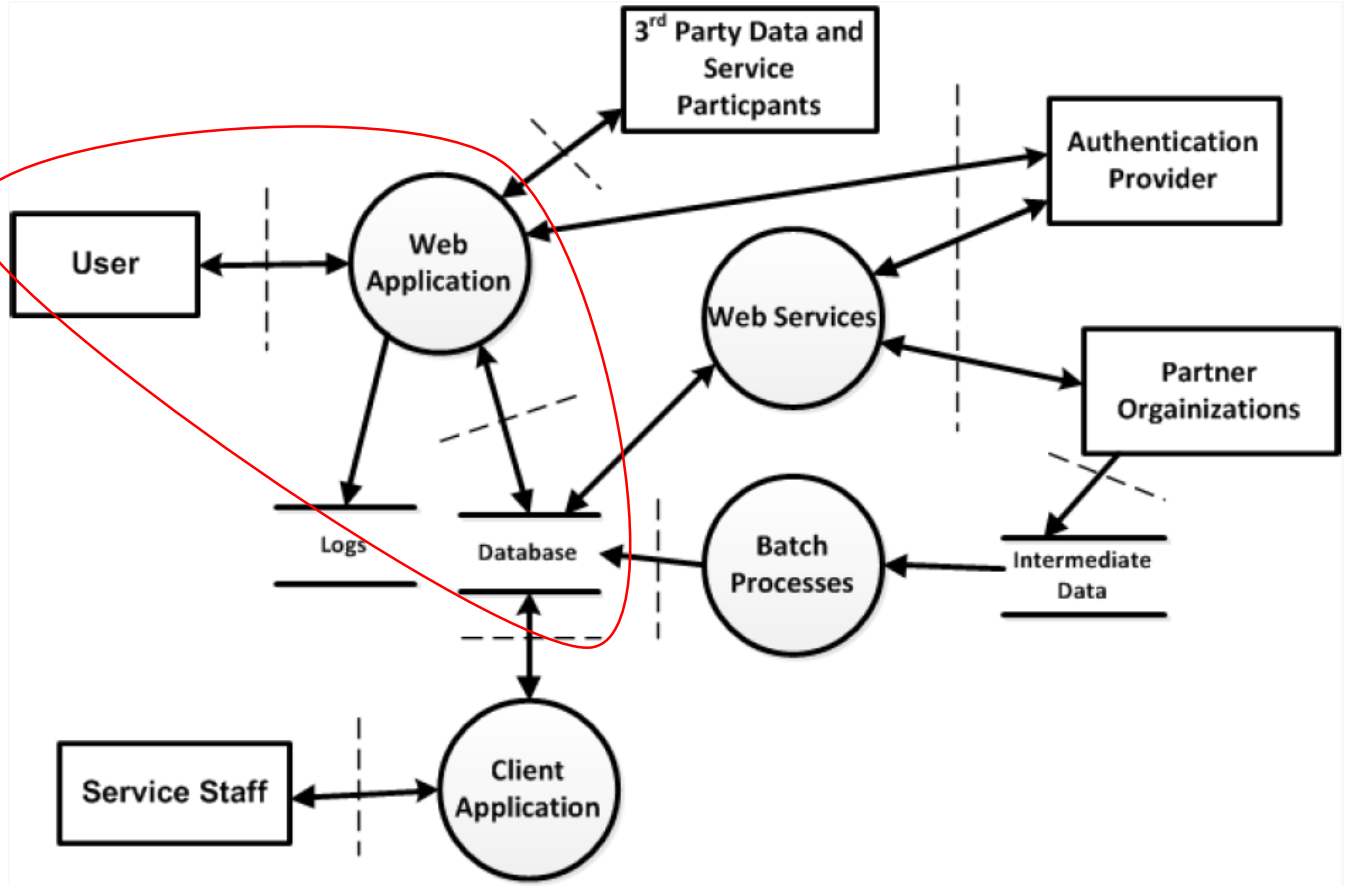
Understanding the Threats and Risks

- A simple application



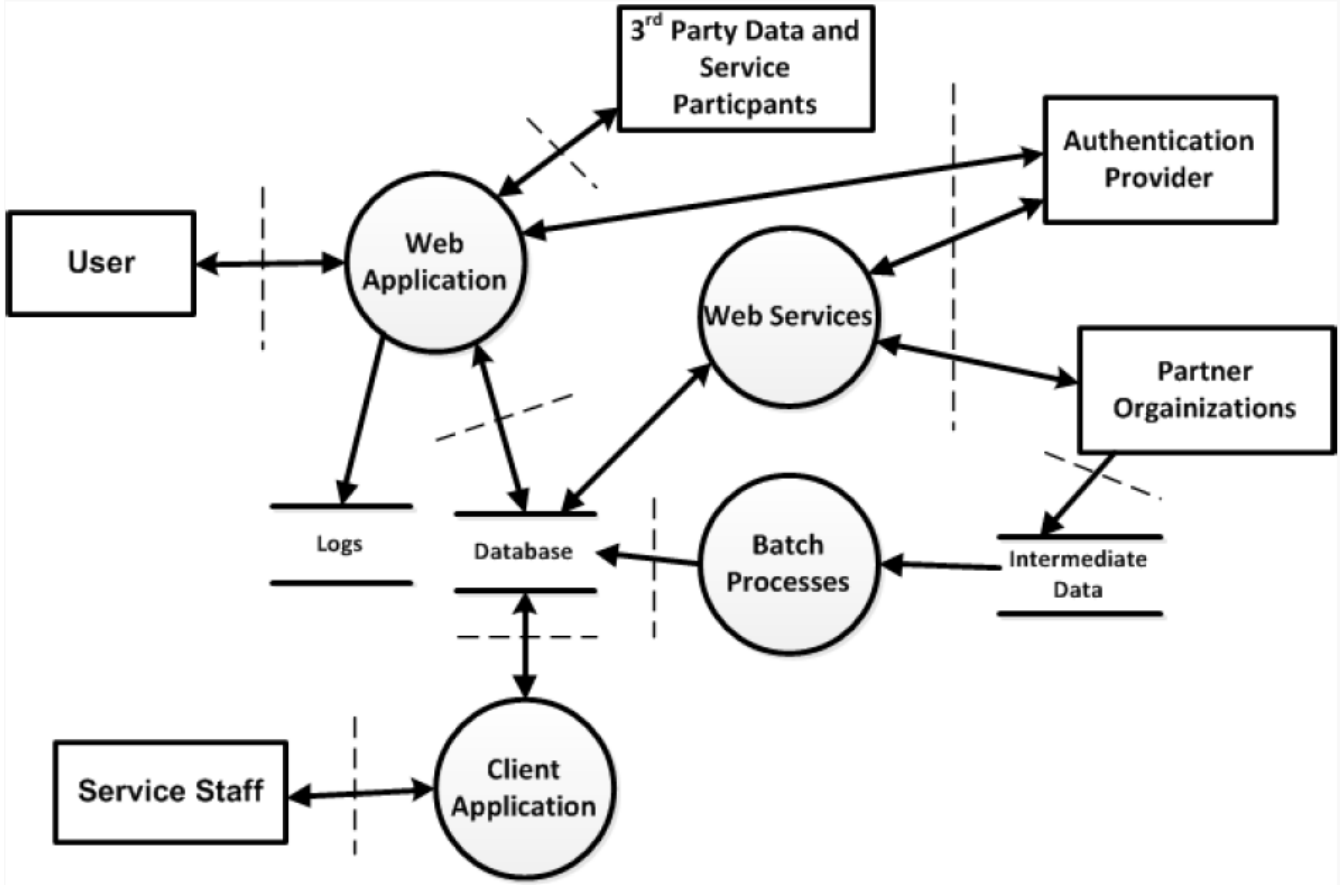
Understanding the Threats and Risks

- A simple application explodes quickly into something complex



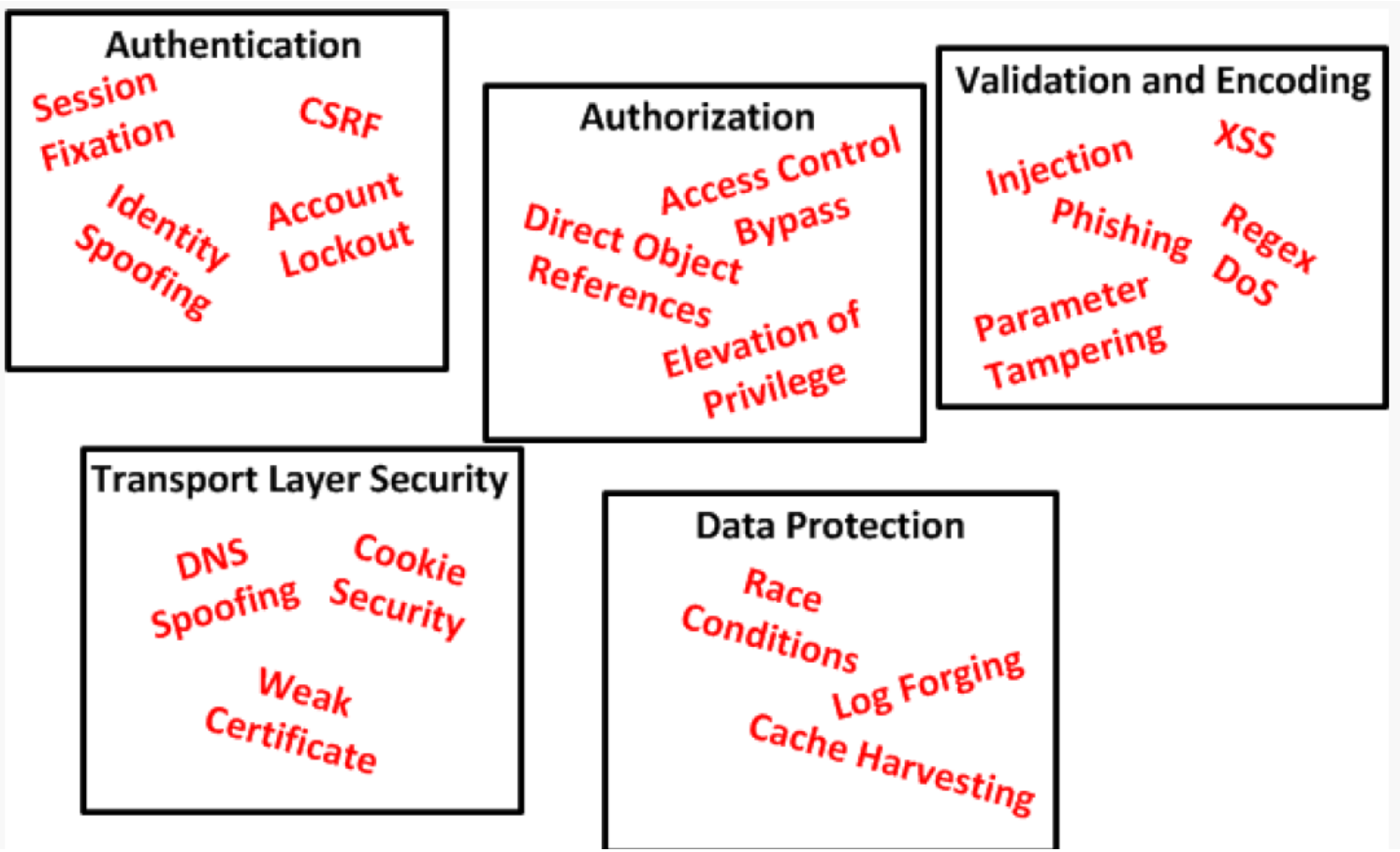
Understanding the Threats and Risks

- Try not to decide the scope of an architecture review or security assessment before thinking of the big picture
- The weakest point in a system may not be what you think
- With the right information on-hand, discovering vulnerabilities can be a simple matter of Q&A



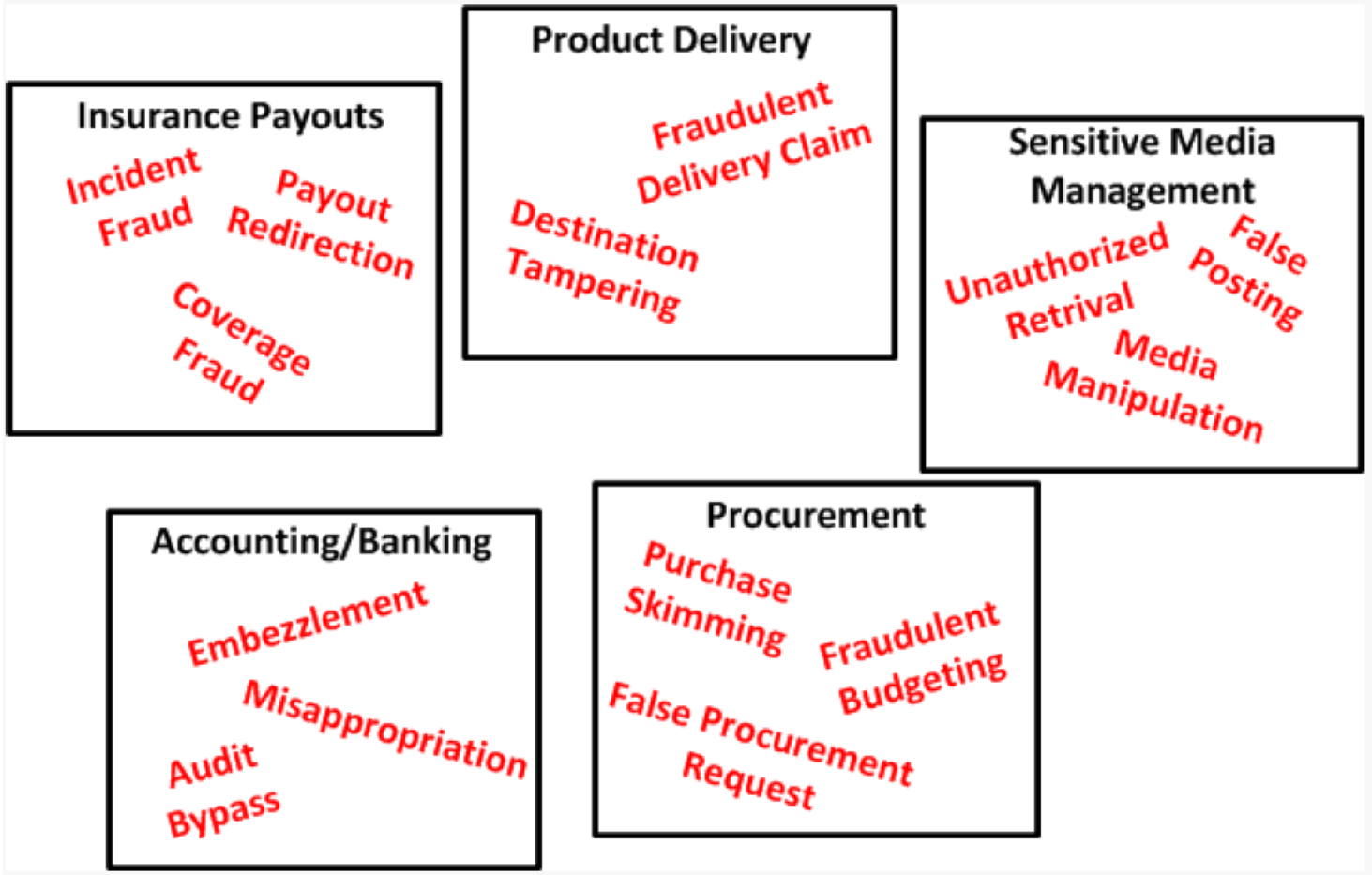
Understanding the Threats and Risks

- Poor functional security



Understanding the Threats and Risks

- Insecure features



Threat Modelling: What We Need

Threat Modelling: What We Need

- *Business*: Knowledge what the system should do, e.g., in terms of
  - Scenarios
  - Use cases

Threat Modelling: What We Need

- *Business*: Knowledge what the system should do, e.g., in terms of
  - Scenarios
  - Use cases
- *Architectural*: Knowledge how information/data "flows" in the system, e.g., in terms of
  - Block/component diagrams
  - Data-flow diagrams

Threat Modelling: What We Need

- *Business*: Knowledge what the system should do, e.g., in terms of
  - Scenarios
  - Use cases
- *Architectural*: Knowledge how information/data "flows" in the system, e.g., in terms of
  - Block/component diagrams
  - Data-flow diagrams
- *Functional Security*: How to defeat an attack, e.g., in terms of
  - Planned security technologies/checks/processes

Threat Modelling: What We Need

- *Business*: Knowledge what the system should do, e.g., in terms of
  - Scenarios
  - Use cases
- *Architectural*: Knowledge how information/data "flows" in the system, e.g., in terms of
  - Block/component diagrams
  - Data-flow diagrams
- *Functional Security*: How to defeat an attack, e.g., in terms of
  - Planned security technologies/checks/processes
- *Attackers' Goals*: Knowledge what an attacker might want to achieve, e.g., in terms of
  - Attack Trees
  - Threat Trees

Threat Modelling: What We Need

- *Business*: Knowledge what the system should do, e.g., in terms of
  - Scenarios
  - Use cases
- *Architectural*: Knowledge how information/data "flows" in the system, e.g., in terms of
  - Block/component diagrams
  - Data-flow diagrams
- *Functional Security*: How to defeat an attack, e.g., in terms of
  - Planned security technologies/checks/processes
- *Attackers' Goals*: Knowledge what an attacker might want to achieve, e.g., in terms of
  - Attack Trees
  - Threat Trees
- *A team of experts*, e.g.,
  - Software architect
  - Product owner
  - Lead developer
  - Security experts
  - Domain experts

Threat Modelling: What We Need

- *Business*: Knowledge what the system should do, e.g., in terms of
  - Scenarios
  - Use cases
- *Architectural*: Knowledge how information/data "flows" in the system, e.g., in terms of
  - Block/component diagrams
  - Data-flow diagrams
- *Functional Security*: How to defeat an attack, e.g., in terms of
  - Planned security technologies/checks/processes
- *Attackers' Goals*: Knowledge what an attacker might want to achieve, e.g., in terms of
  - Attack Trees
  - Threat Trees
- *A team of experts*, e.g.,
  - Software architect
  - Product owner
  - Lead developer
  - Security experts
  - Domain experts
- *A "structured" process* to
  - Ensure that no important aspects got forgotten
  - Results are prioritized and documented

Identifying Threats: STRIDE

- STRIDE is expansion of the common CIA threat types
  - Confidentiality
  - Integrity
  - Availability

- STRIDE
  - **S**poofing Identity
  - **T**ampering with Data
  - **R**epudiation
  - **I**nformation Disclosure
  - **D**enial of Service
  - **E**levation of Privilege

Summary

- Threat modelling often a structured way of brain-storming

- Results should be documented
  - Containing the identified threats (with priorities!)
    - Either acknowledging that a threat/risk is accepted
      - Ideally with justification why the risk is acceptable
    - Or the planned countermeasures for an identified threat
      - Ideally with information how to test that the countermeasure is implemented correctly

Reading List

- Ross J. Anderson. Security Engineering: A Guide to Building Dependable Distributed Systems. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 2001.
  - The complete book is available at: http://www.cl.cam.ac.uk/~rja14/book.html

Thanks for your attention!

- Any questions or remarks?