2. Secure Development Lifecycle

Prof. Dr. Marc Rennhard
Institut für angewandte Informationstechnologie InIT
ZHAW School of Engineering
rema@zhaw.ch

Content

- Overview of secure software development processes
- Security Activities during secure software development
- How to start adopting a secure software development process

Goals

- You have an overview of software security activities (best practices) and see how they fit together in a secure development lifecycle
- You know the presented security activities and during which phases they are applied and can provide brief explanations about their purpose
- You realize that the discussed approach to build secure software does not introduce a novel software development process, but that it can be applied to any process to transform it into a secure development lifecycle

Overview of Secure Software Development Processes

The Need for Secure Software Development

- We have seen in chapter 1 that "traditionally" employed security measures don't work well
  - Measures to secure software and systems are often reactive (penetrate and patch, filtering devices...)
  - When thinking about integrating security during software development, the focus is on security functions (e.g. TLS) but not on preventing vulnerabilities in general

- The only solution to really make software more secure (reducing the number of vulnerabilities) is by putting a stronger focus on security during the entire development process
  - This means employing a secure development lifecycle (SDL)
  - An SDL in general means that security activities are applied during different phases of the software development process

Secure Software Development Processes (1)

Some secure software development processes that have been proposed:

- Microsoft SDL: Microsoft Security Development Lifecycle
  - Microsoft-internal mandatory since 2004, as a response to security-related problems in Microsoft product
- BSIMM: The Building Security In Maturity Model
  - Defined by a consortium of several companies, based on analysing projects of > 30 companies
- CLASP: Comprehensive Lightweight Application Security Process
  - OWASP project concerned with security during the software development lifecycle (basically a collection best practices)
- SAMM: Software Assurance Maturity Model
  - OWASP project that additionally defines maturity levels for the different disciplines

Secure Software Development Processes (2)

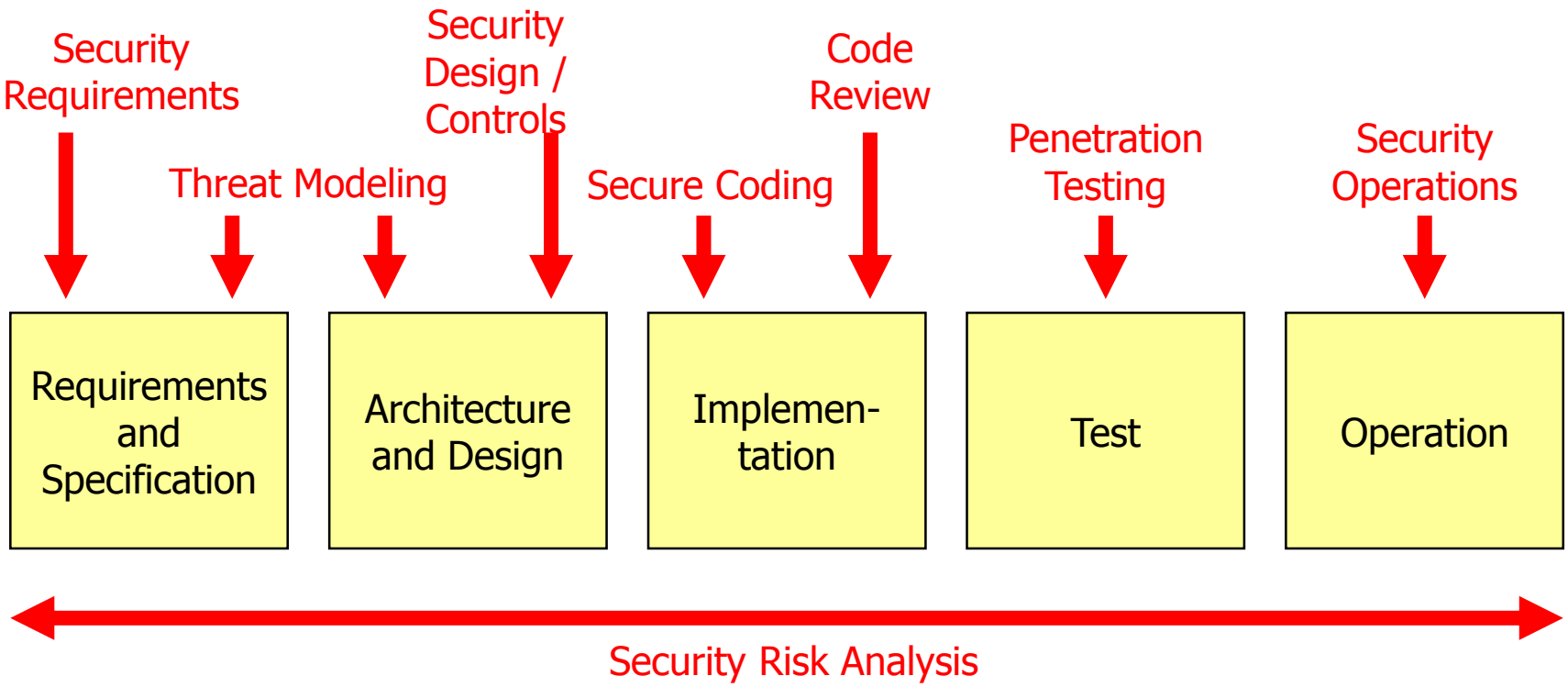
- All of these processes are similar to each other and propose similar security activities to be performed during software development
  - The differences are in the actual details about how individual activities should be performed, e.g. what checklists to use during threat modeling
  - This also means that one does not have to stick to a particular process and one can easily mix components from different processes

- In this course, we won't follow a particular one of these processes, but focus on the security activities (which can be found in any of them)
  - For all activities, we then discuss best practices (sometimes borrowed from the processes above or from somewhere else) that have proven to work well in practice
  - Combined, the presented activities provide all building blocks for a secure development process

Security Activities during the Software Lifecycle

Security-related Activities during the Software Lifecycle

- A secure software lifecycle means that different security activities are carried out during different phases



How can this be Applied to my Development Process (1)

- The picture on the previous slide is laid out just like the traditional waterfall model
  - But today, iterative approaches to software development are preferred
  - In general, there are many different development processes

- As a result, the security activities do not focus on a specific process, but can be applied to any software development process
  - This is achieved by applying the security activities to artifacts that are generated by virtually every development process
  - Artifacts include requirements documents, system architecture, code, running system etc.

How can this be Applied to my Development Process (2)

- With iterative development processes, the security activities are applied appropriately during each iteration
  - But just like with functional aspects, only "to a degree" as it is reasonable during a particular iteration

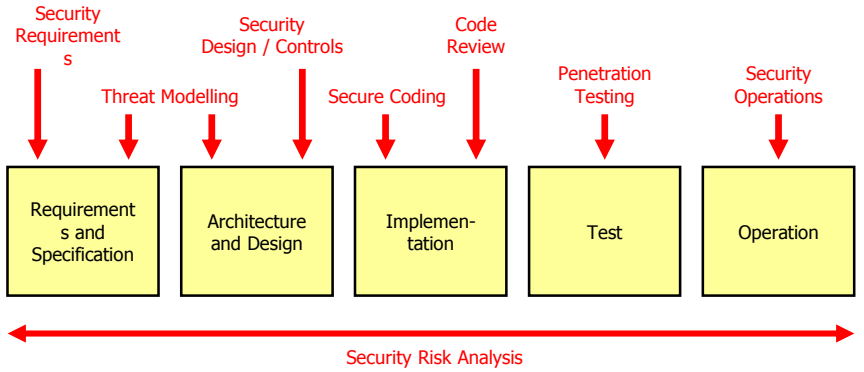
- Example: Unified Process (UP)
  - Security requirements are defined during iterations where "normal" requirements are defined (mainly inception and elaboration phases)
  - Threat Modeling is based on the currently available system design/description
  - The security design and controls will grow and be refined as the system architecture and design develops during multiple iterations
  - Code reviews are carried out along the development of the code

So you basically take your favorite software development process and transform it into a secure development lifecycle (SDL) by applying the security activities appropriately

Overview of the Security Activities

Security Risk Analysis

- Security risk analysis is a "horizontal" activity that accompanies the other "vertical" activities

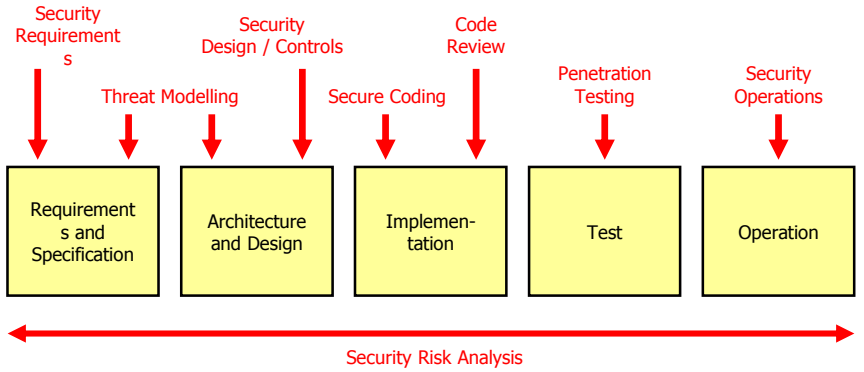


- Examples:
  - When threats are defined during threat modeling, risk analysis serves to assess the criticality of the individual threats
  - When a vulnerability is detected during penetration testing, risk analysis serves to quantify the criticality of the vulnerability

- While risk analysis is important during many activities, it is especially important at the end of the architecture & design phase
  - This is when threats have been identified and security controls and countermeasures have been defined
  - Security risk analysis should demonstrate that the selected architecture and design are "secure enough"

Security Requirements

- Defining security requirements is the first security-related activity that is carried out

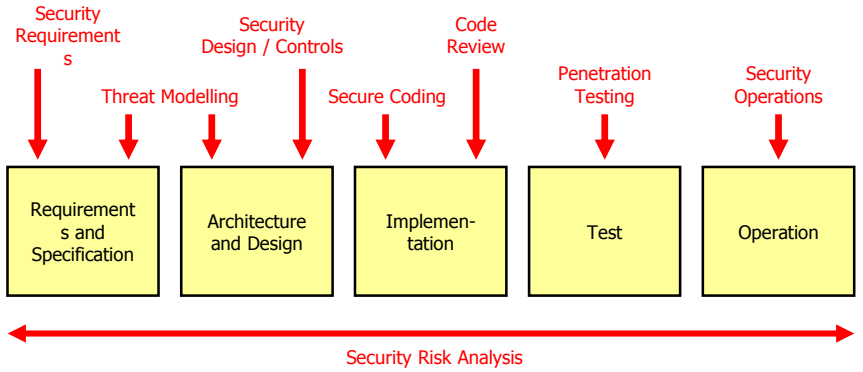


- At first, they are based on the functional requirements and the goal is to define "some obvious security requirements", e.g.:
  - Credit card information is transmitted → client and server must communicate via a cryptographically protected channel
  - There are different areas for different user groups → an access control mechanism must be employed

- Initial security requirements often focus on functional security
  - During threat modeling, additional security requirements are defined and refined (also during further iterations)

Threat Modeling

- Threat modeling is carried out during the requirements and architecture and design phases

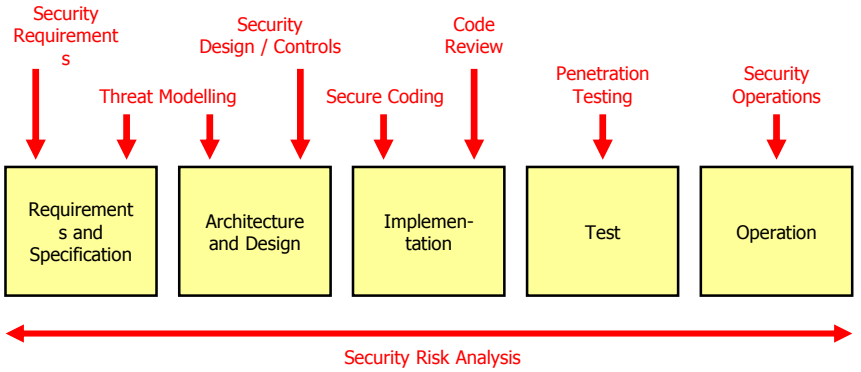


- The goals of threat modeling are the following:
  - Identify possible threats against the system (its assets)
  - Identify weaknesses in the architecture and design
  - Provide the basis for reasonable countermeasures (additional security requirements)

- Threat modeling is a critical and powerful activity in a secure development process
  - Only the threats one has identified are likely to be prevented by adequate security requirements / security controls

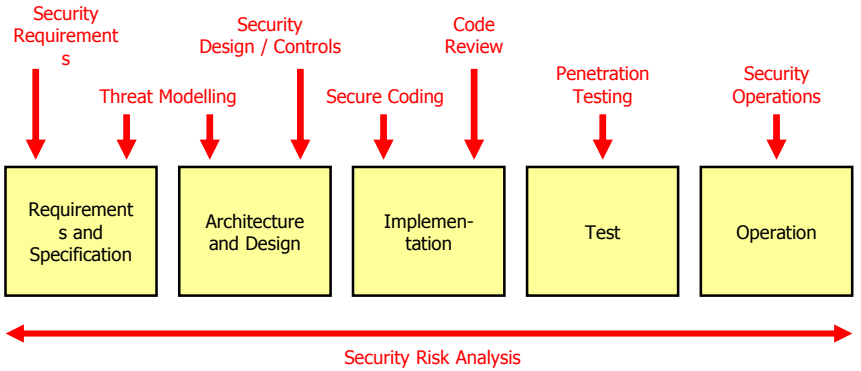
Security Design / Controls

- The security design encompasses all security measurements that are required to secure the system adequately



- It is driven by the security requirements and (implicitly) the threats and vulnerabilities identified during threat modeling
- The goal of the security design is to fulfill the security requirements and reduce the risks to an acceptable level
- Security controls should be chosen reasonably with respect to the threat / risk they reduce
  - E.g.: It is pointless to spend CHF 10'000 per year to protect from a threat that is expected to result in CHF 2'000 damage a year

Secure Coding



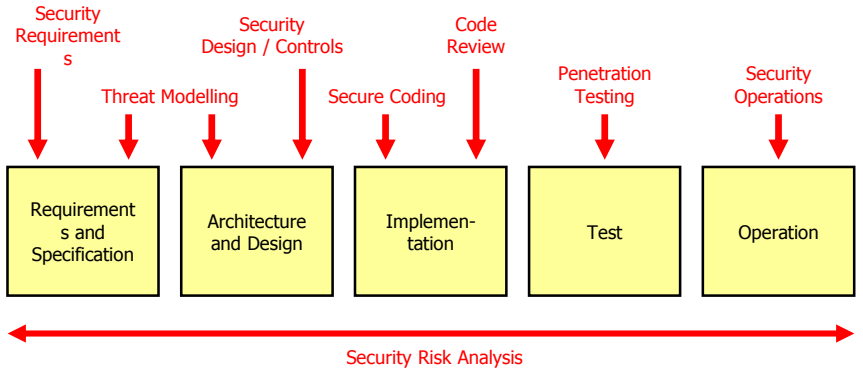
- Secure Coding means implementing the specified design such that no "new" defects are introduced by making programming mistakes

- It's important to use the right technology and third party libraries that can provide the functionality to provide a secure implementation
  - This implies that you understand the technology you are using in to make correct judgments whether it can actually provide the "necessary security"

- Secure coding goes beyond implementing security functions (e.g. encryption and access control)
  - It also means preventing vulnerabilities, e.g. by handling unexpected situations due to non-conforming user input

Code Review

- Code Review means inspecting the code and search for security problems

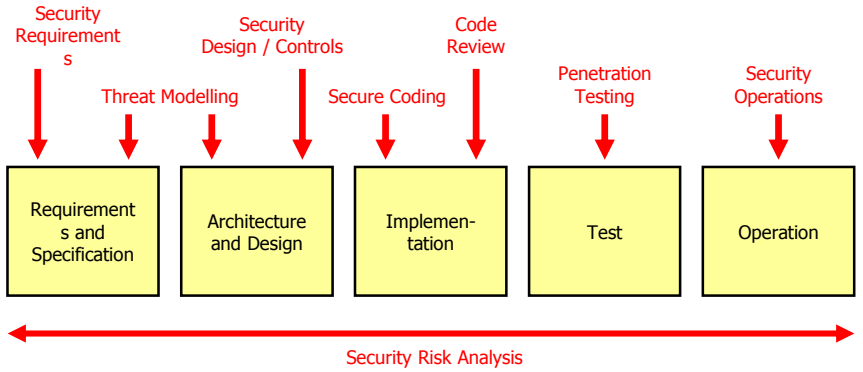


- The goal of code reviews is to detect implementation bugs
  - As these account for approx. 50% of software security problems, good code review can uncover up to about 50% of all security problems
  - The other 50% are design flaws that are virtually impossible to uncover by looking at code (and that should be uncovered during threat modeling)

- Code review is usually done using automated code analysis tools
  - Manual code review can be reasonable for some "very security critical" sections, but is usually too expensive for large amounts of code

Penetration Testing

- Penetration Testing means taking the attackers view and trying to find and exploit vulnerabilities in the running system



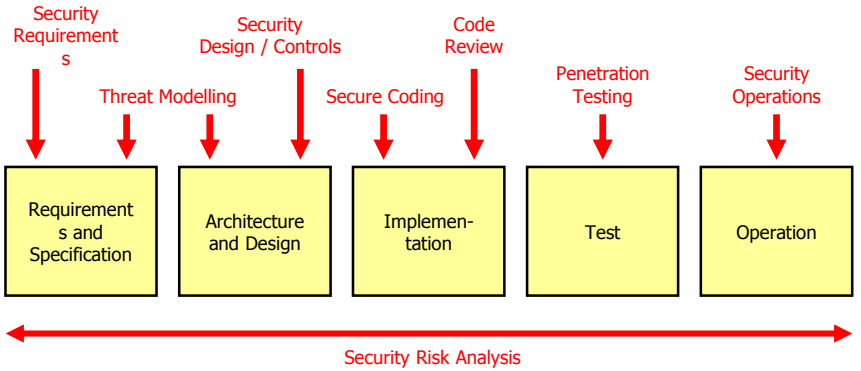
- Valuable as it provides information of the security of a complete system "in reality"

- Ideally, a penetration test takes the information about previous risk analysis results into account
  - I.e. put more effort in the areas where major risks were identified to check whether the countermeasures are sufficient

- Automated tools are available, but they likely will only uncover easy-to-spot vulnerabilities

Security Operations

- Security Operations includes all security-related activities that take place while the system is in operation



- Once a system is in operation, it's almost guaranteed that attacks (or at least attempts) will happen

- Monitoring a fielded system is therefore important for various reasons
  - To both learn about possible attacks and vulnerable areas and to feed back the findings into the development process
  - To detect a system compromise

Security Activities – Example (1)

- As an example, consider the following piece of code:

```
1 read(fd, userEntry, sizeof(userEntry));
2 comp = memcmp(userEntry, correctPasswd, strlen(userEntry));
3 if (comp != 0)
4    return (BAD_PASSWORD);
```

- Identify the different problems in this code and think about "with the help of which" security activity the problem may be identified

Security Activities – Example (2)

```
1 read(fd, userEntry, sizeof(userEntry));
2 comp = memcmp(userEntry, correctPasswd, strlen(userEntry));
3 if (comp != 0)
4    return (BAD_PASSWORD);
```

How to Start Moving Towards an SDL

Incremental Adoption of an SDL

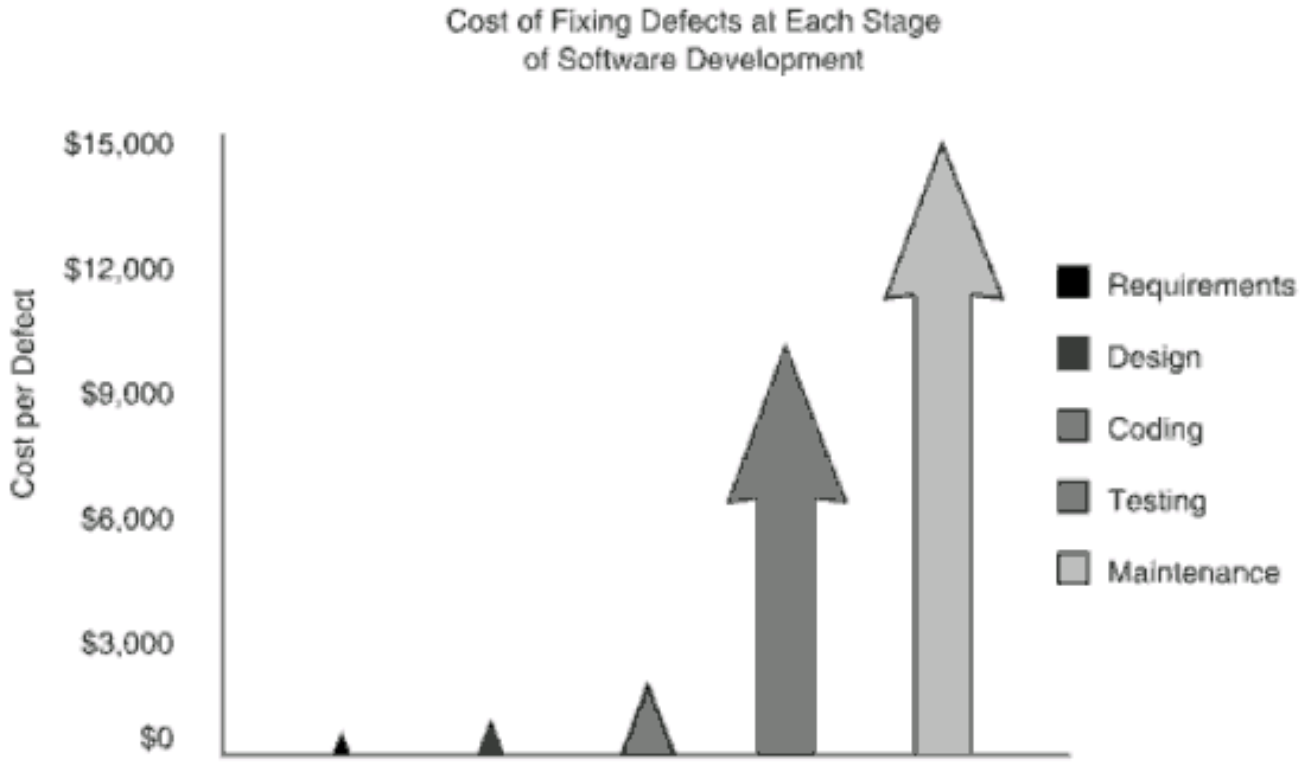
- When moving towards a more secure development process, there's no need to start applying all security activities at once
  - You can start by employing code reviews (which should already help a lot as it may detect up to 50% of all security problems)
  - A reasonable next step would be performing threat modeling together with risk analysis (as it also covers design flaws)
  - Penetration tests can also be used on their own and provide you with immediate feedback about the "real" security of a system

- Every added security activity will increase the security of the resulting software

The long-term goal should be to adopt all security activities because they complement each other

Fixing Earlier is Better (1)

- Just like with functional software defects, finding and fixing security-related defects early in the lifecycle is cheaper



- Therefore, the "early security activities" are especially important
  - They help to not only to detect defects, but can prevent them

Fixing Earlier is Better (2)

- But in reality, the focus on software security is very often only (if at all) at the end of the lifecycle
  - Many companies "start" with a penetration test just prior to (or even after) fielding the system
  - They "hope" that not much is found so they "get the green light" for productive operation
  - Sometimes, this is simply done so one can point to the (external) penetration testers in case a security breach happens during operation
    - This is also known as CYA Security: Cover your Ass Security

- Doing only a penetration test is better than not doing anything at all, but what happens if serious (design-) problems are uncovered?
  - This usually means that the underlying defects won't be fixed adequately, especially if the necessary effort (time, money) is significant

Fixing Earlier is Better (3)

- Typically, the following is done instead of truly fixing the underlying problem
  - Implement a "quick fix", which usually does not really solve the underlying problem
    - Which likely means it will show up again (somewhere else in the code or at the same place through a variation of the attack)
  - Don't fix it at all, but employ network-based security measures such as IPS or WAF etc. to prevent attacks from reaching the targeted system

- Conclusion: Looking at security only late in the lifecycle results in reactive security and supports "penetrate and patch"
  - This is why adopting security activities early in the lifecycle is paramount for truly secure software!

Security Activities in this Course (1)

- Security Risk Analysis:
  - Chapter 10: Security Risk Analysis
- Security Requirements:
  - Chapter 9: Security Requirements Engineering and Threat Modeling
- Threat Modeling:
  - Chapter 9: Security Requirements Engineering and Threat Modeling
- Security Design / Controls:
  - Chapter 5: Security Controls
  - Chapter 6: Secure Design Principles
  - Chapter 8: Java Web Application Security

Security Activities in this Course (2)

- Secure Coding:
  - Chapter 3: Typical Programming Errors
  - Chapter 4: Java Security
  - Chapter 8: Java Web Application Security
- Code Review:
  - No lecture, will be discussed in Security Lab
- Penetration Testing:
  - Chapter 7: Finding and Exploiting Vulnerabilities in Web Applications
  - Chapter 11: Penetration Testing
- Security Operations:
  - Not part of this lecture

Summary

- Employing a Secure Development Lifecycle (SDL) is the only reasonable way towards secure (enough) software
  - Experience tells us that reactive security measures don't work well
- SDL is not a new software development process but consists of a set of security activities can be applied to virtually any existing development process
  - By applying the security activities to artifacts that are generated by virtually every development process
- The security activities cover all phases of typical software development processes (repeatedly with iterative processes)
  - Requirements, specification, architecture, design, implementation, test, operation
- An SDL can be adopted incrementally with the goal to eventually employ all security-related activities