

Secure Software: Conclusions

Frank Piessens
(Frank.Piessens@cs.kuleuven.be)

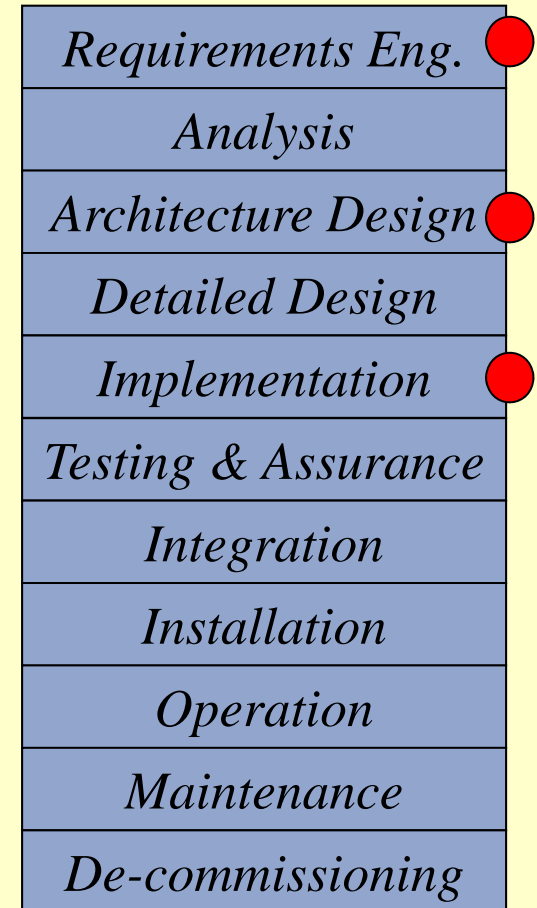
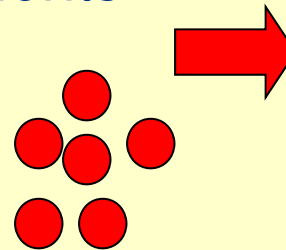
Security in the software lifecycle

- Vulnerabilities can enter a system in any phase of the SW lifecycle
- It is important to understand the nature of all these classes of vulnerabilities, and to use countermeasures at each phase
 - This course has focused on the implementation level (with a bit of design)

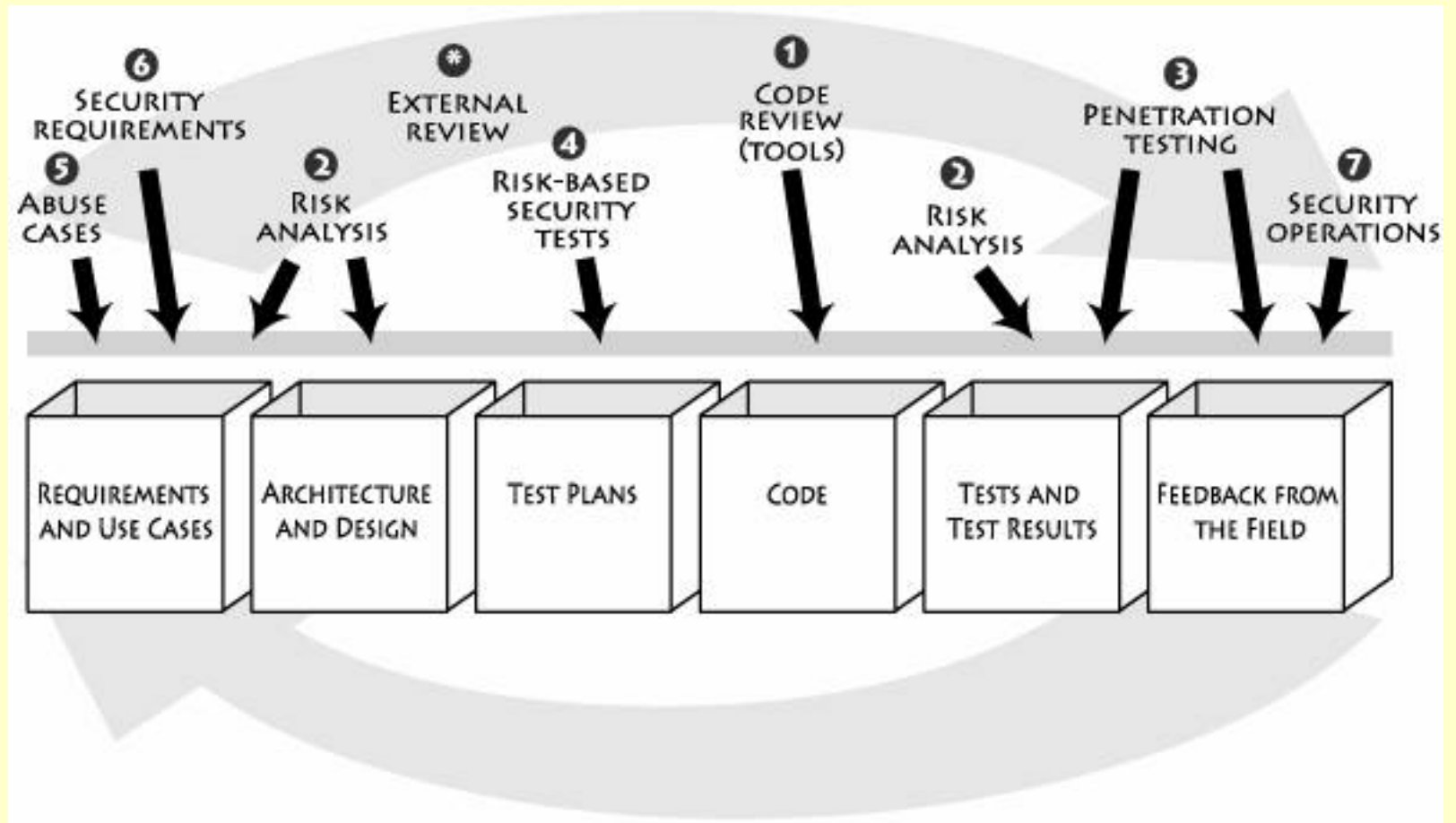
<i>Requirements Eng.</i>
<i>Analysis</i>
<i>Architecture Design</i>
<i>Detailed Design</i>
<i>Implementation</i>
<i>Testing & Assurance</i>
<i>Integration</i>
<i>Installation</i>
<i>Operation</i>
<i>Maintenance</i>
<i>De-commissioning</i>

Security in the software lifecycle

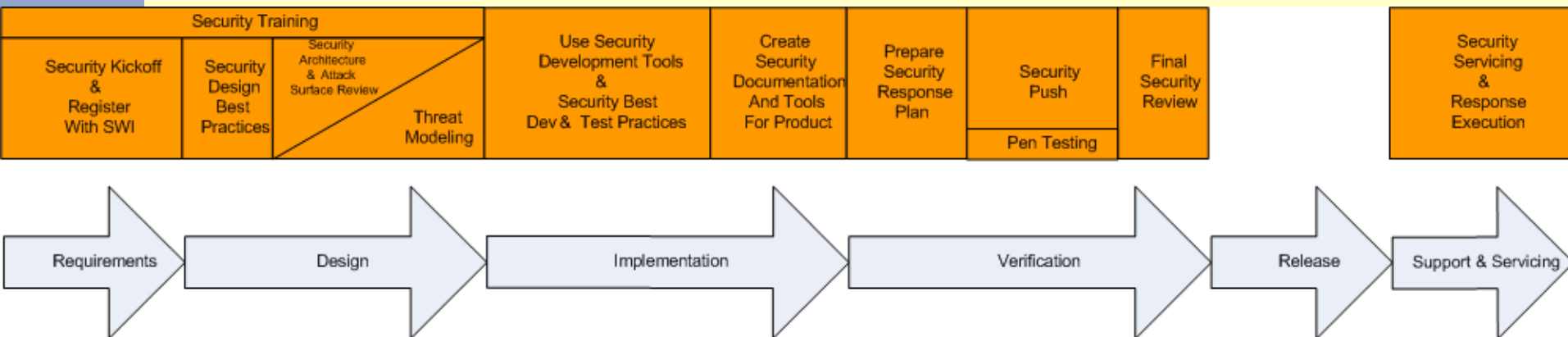
- BUT: such countermeasures should not require a complete redesign of the SW development lifecycle
 - “Touchpoints”
 - “SW process enrichments”



Example: Cigital's touchpoints



Example: Microsoft's SDL



Example: Threat modeling

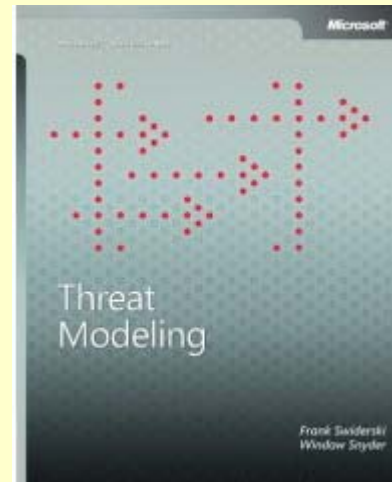
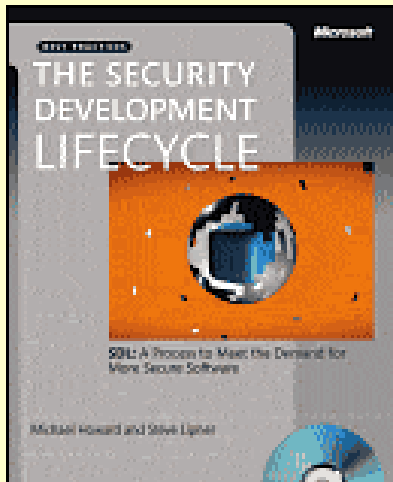
- Threat modeling is an activity early in the software development lifecycle
 - Primary goal: get a good view on possible threats to the system being developed
- Threat modeling can be done on various levels of abstraction
 - System level
 - E.g. Threat modeling of the Internet e-mail system
 - Application or component level
 - E.g. Threat modeling of the e-mail client software

Threats

- Threat categories:
 - Spoofing
 - Information Tampering
 - Repudiation
 - Information disclosure
 - Denial of service
 - Elevation of privilege
- Eliciting threats needs:
 - Good understanding of assets
 - Good understanding of threat agents (motivation and power)

Microsoft's Threat Modeling process

- As part of it's Secure Development Life Cycle, Microsoft has defined a threat modeling activity
 - Supported by documentation and tools



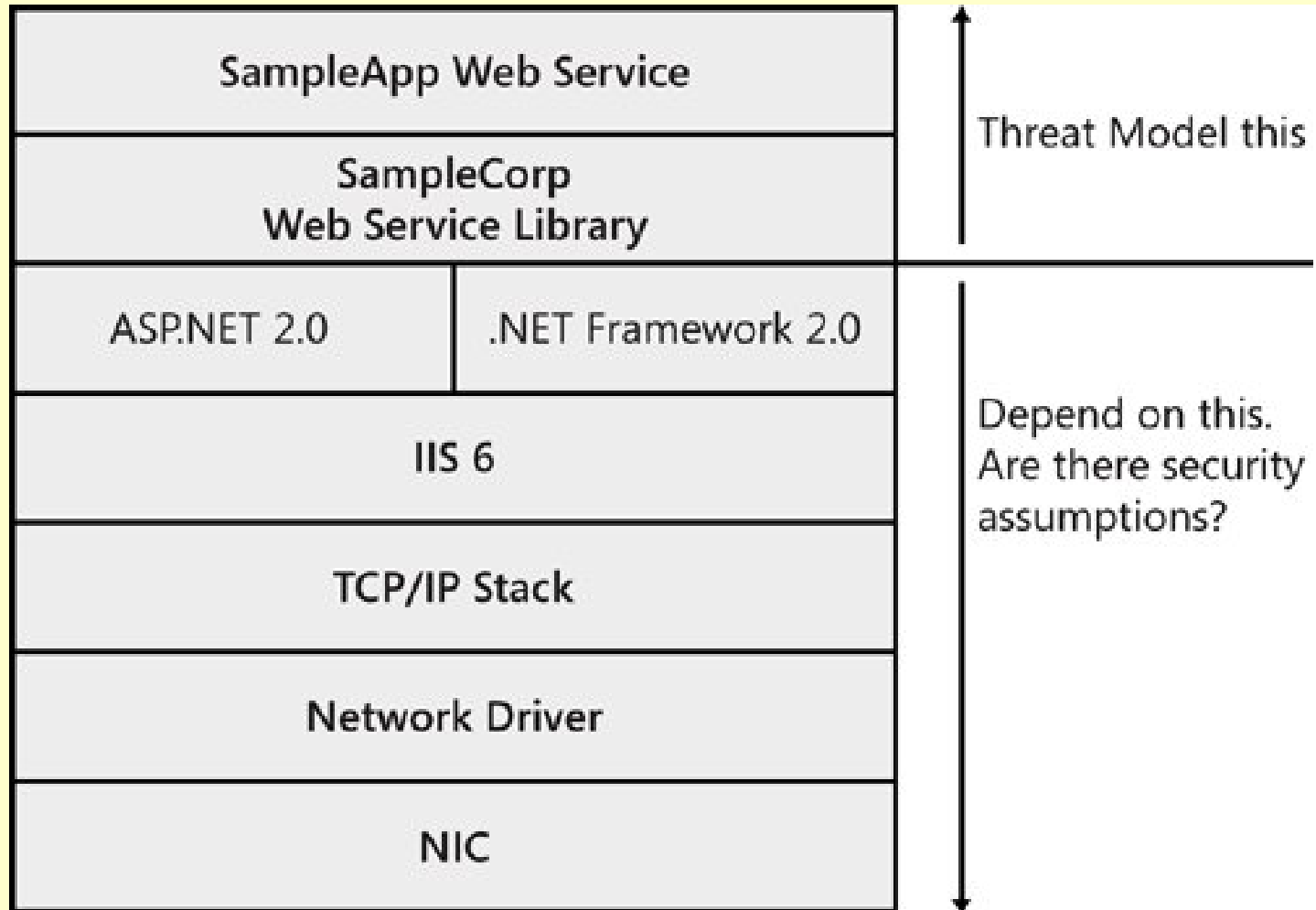
Overall Structure of the Threat Modeling Process

1. Define Use Scenarios
2. List External Dependencies
3. Define Security Assumptions
4. Create External Security Notes
5. Model the Application
6. Determine Threat Types
7. Identify Threats
8. Determine Risk
9. Plan Mitigations

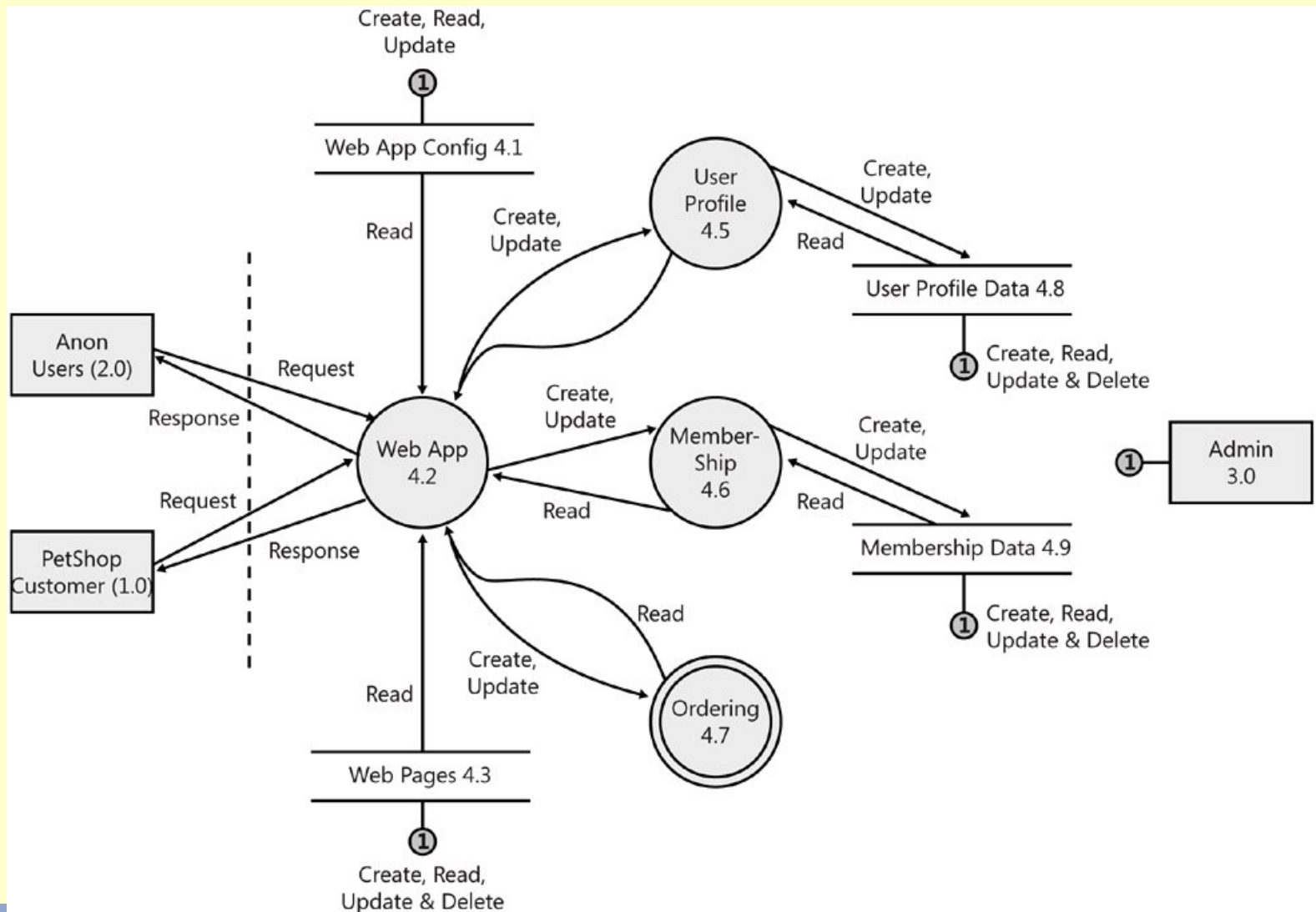
5. Model the Application

- Model the system
 - Typically modeled using dataflow diagrams
 - External entities
 - Processes
 - Data stores
 - Data flow
 - Privilege boundaries

To model or not to model?



Example Model for Pet Shop



7. Identify Threats

- Identify assets
 - By default, all DFD elements are considered an asset

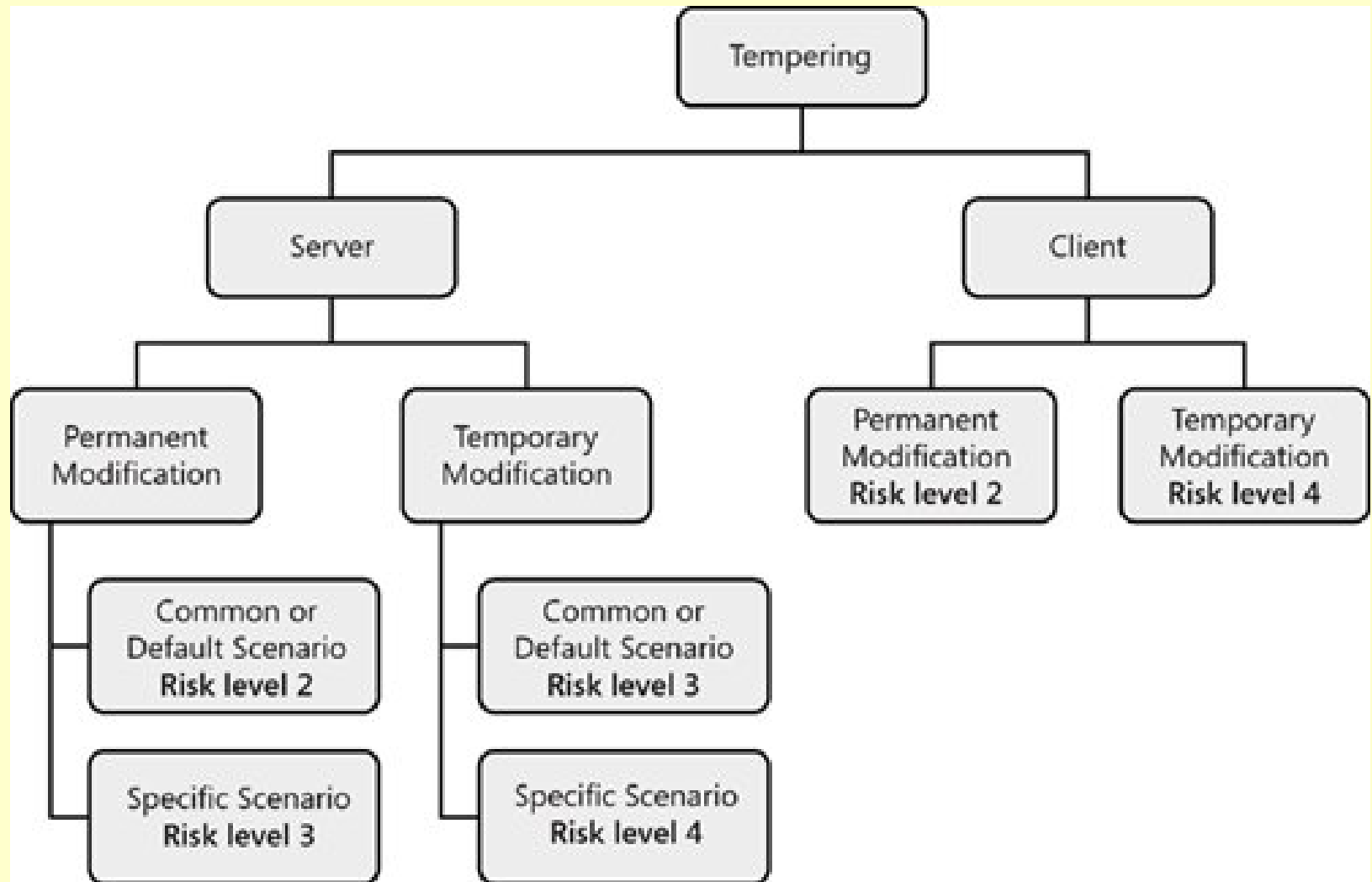
Table 9-5. Mapping STRIDE to DFD Element Types

DFD Element Type	S	T	R	I	D	E
External Entity	X		X			
Data Flow		X		X	X	
Data Store		X	†	X	X	
Process	X	X	X	X	X	X

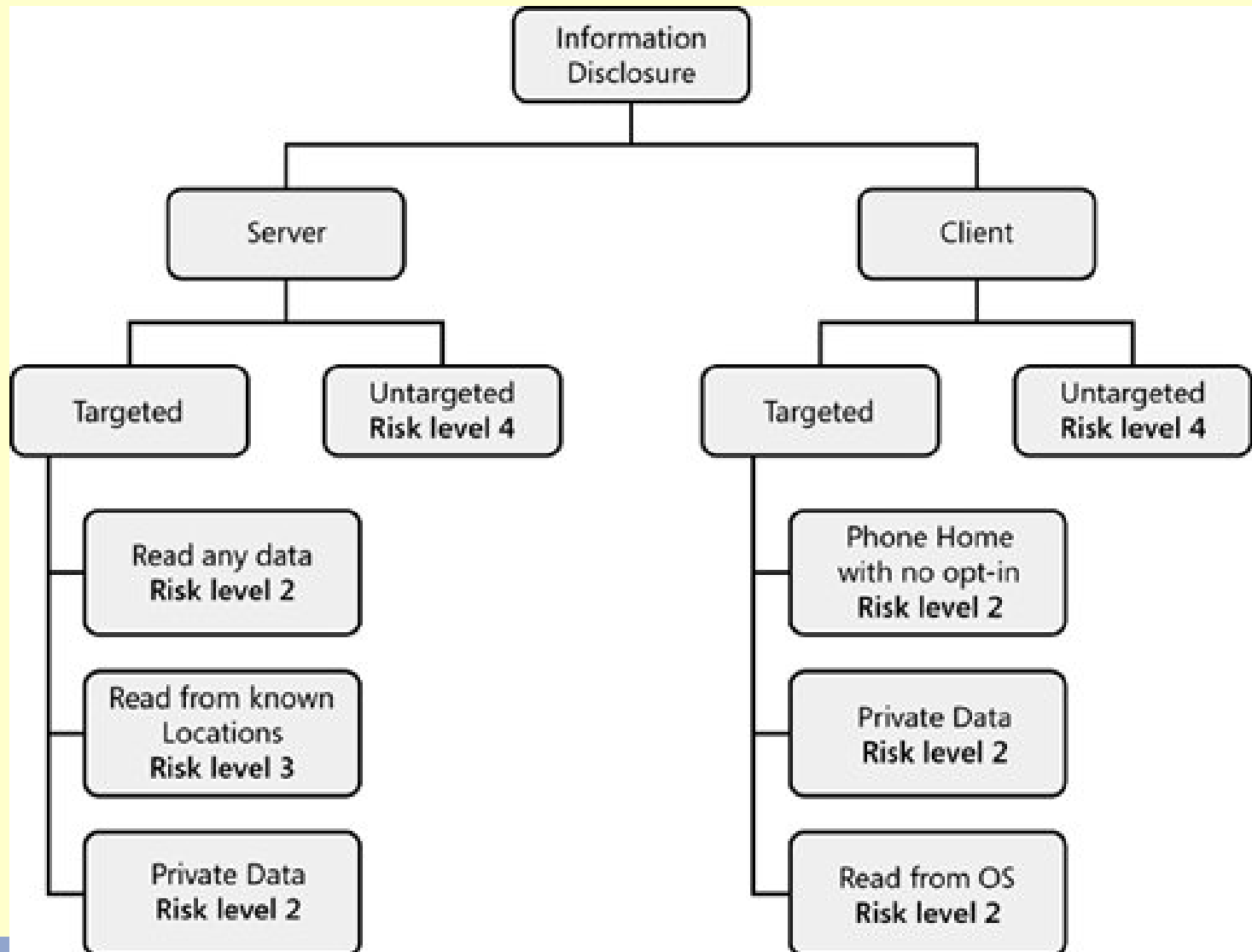
8. Determine Risk

- Subjective numerical estimation techniques have been abandoned, e.g. DREAD
- Current guideline is to use objective thread characteristics:
 - Server app versus client app
 - Local versus remote accessibility
 - Accessible to anonymous or remote users
 - ...

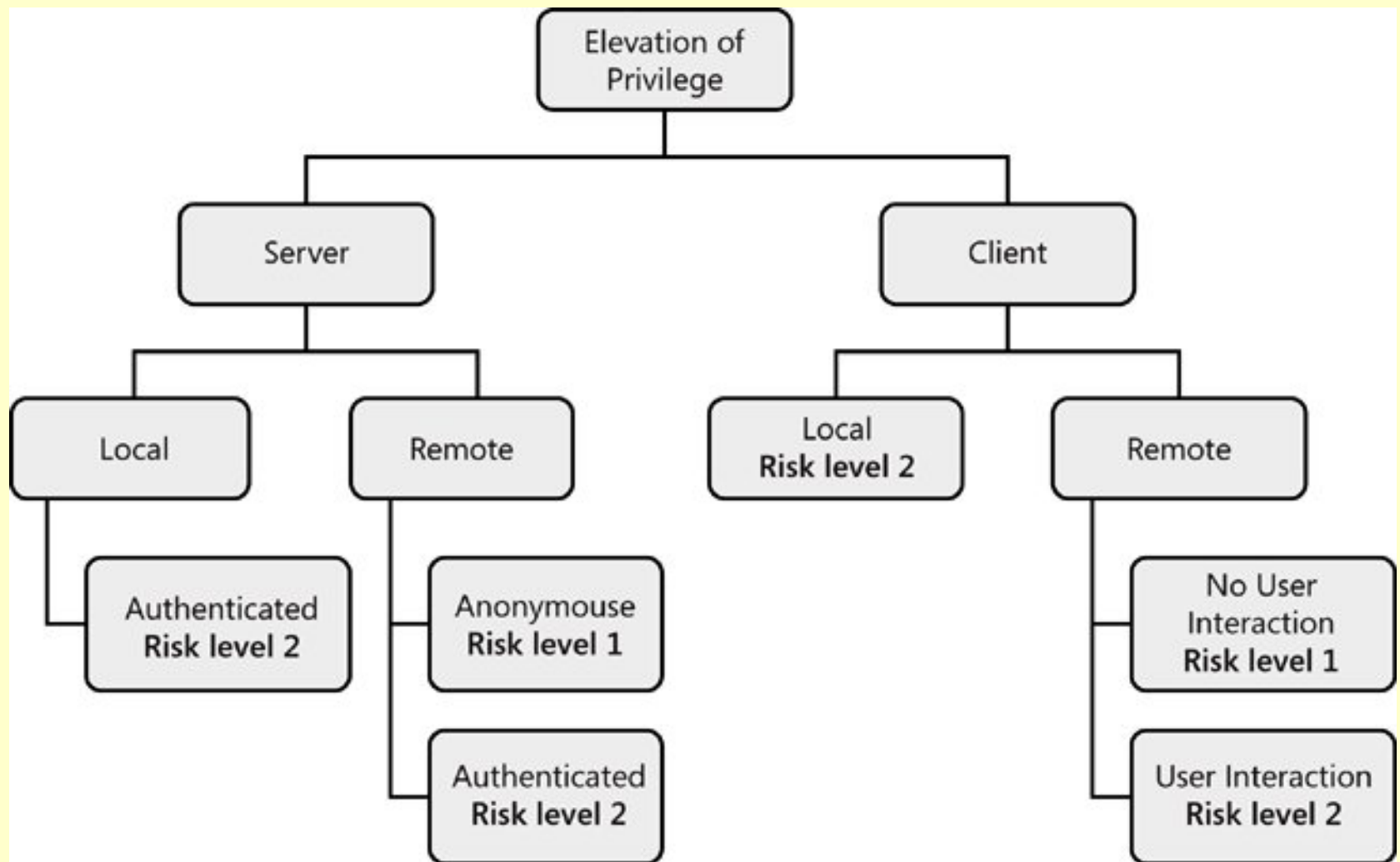
Tampering threats risk ranking



Information Disclosure risk ranking



EoP threats risk ranking



Conclusion

- Threat modeling is an activity to be performed in the early phases of the software life cycle
 - In order to understand the “threat profile” of the software
 - In order to select countermeasures in an informed way
- Microsoft places Threat Modeling in the top 2 of most important security related activities
 - Other one is static analysis for implementation vulnerabilities

Conclusion

- In today's networked world, security is a critical success factor for software projects
- Security engineer has a complex toolbox of security technologies
- Security engineering activities should be enrichments of the software engineering process, reusable in the variety of existing processes.
- Many challenging research tasks remain:
 - Improvements in programming languages and composition
 - Improvements in the software engineering process