

Threat Modeling

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

Overview

- Introduction
- Key Concepts
 - Threats, Vulnerabilities, Countermeasures
 - Example
- Microsoft's Threat Modeling Process
- Conclusion

Software engineering & security

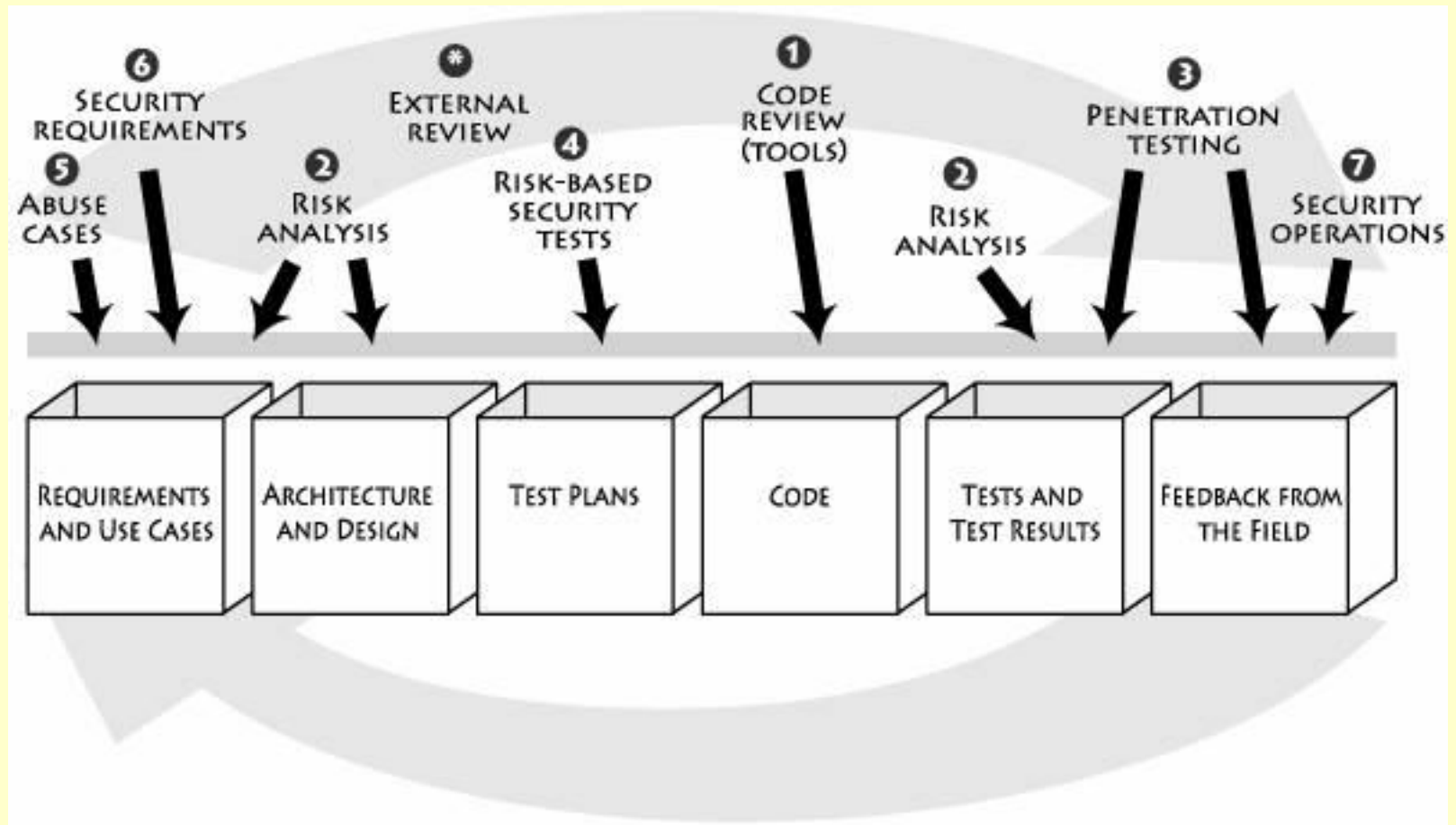
- Different SE processes
 - Waterfall, Spiral, Prototyping, UP, XP, Adaptive programming, ...
 - Different phases
- None of the processes *really* supports security
 - Security is highly demanding wrt. SE process
 - No good and overall support for security in standard processes

Software engineering & security

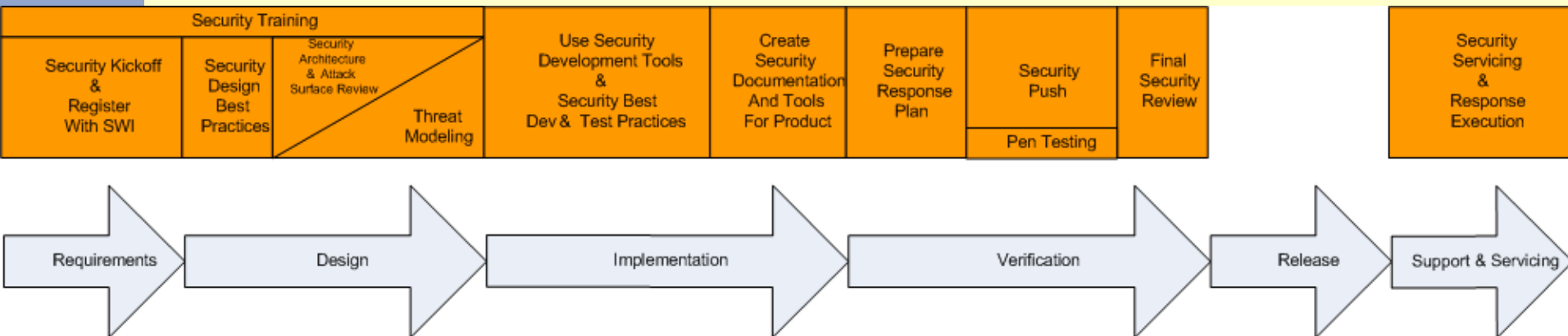
- *in practice* -

- Often ad hoc (typically in later phases) or not covered at all
- Issues:
 - Non systematic approach
 - Bad early decisions can make your life *very* difficult
 - Hard to manage and modify
- First attempts at systematic integration in SW engineering process in big software houses
 - Too early to evaluate, but first indications promising
 - E.g. MS IIS 5 -> MS IIS 6

Example: Cigital's touchpoints



Example: Microsoft's SDL



Key Activities

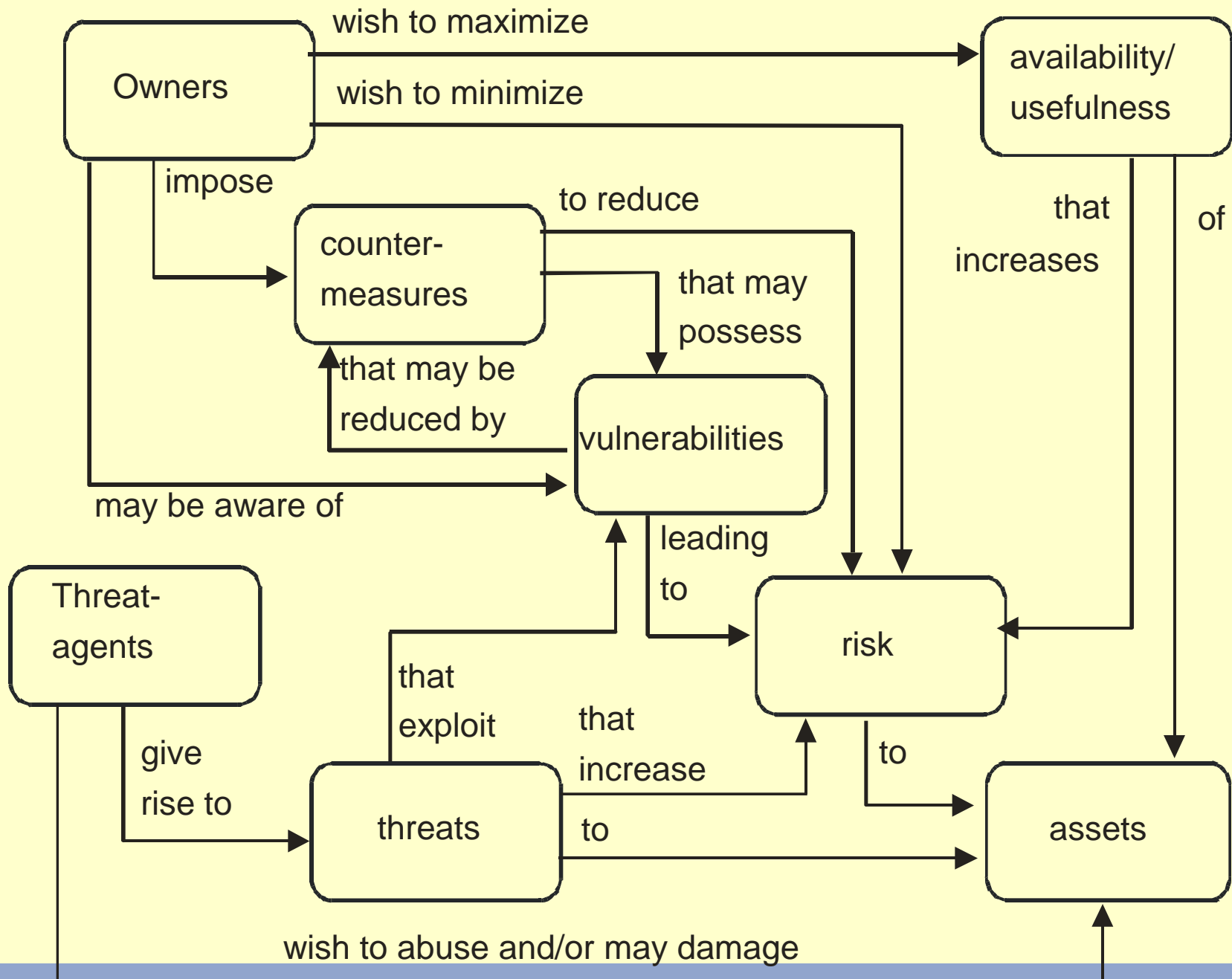
- Both process models rank the following two activities as key:
 - Threat modeling as a late form of security requirements engineering
 - (Tool-assisted) code review to deal with implementation level vulnerabilities
- This lecture will discuss threat modeling from a practical perspective

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

Overview

- Introduction
- Key Concepts
 - Threats, Vulnerabilities, Countermeasures
 - Example
- Microsoft's Threat Modeling Process
- Conclusion



Threats versus Security Goals

- In a first approximation, threats and security goals are each others negation:
 - A security goal is a statement of intent to counter identified threats
 - A threat is the intention of a threat agent to break a security goal

Typical Threats

- Information disclosure
 - Threat: Information leaks to threat agents that should not be able to get access to that information
 - Corresponding Security Goals:
 - **Data Confidentiality**: protecting data against unauthorized reading
 - **Access Control**: preventing unauthorized access to software functions
 - Example: stealing of credit card numbers

Typical Threats

- Information tampering
 - Threat: threat agents tamper with data they should not be able to modify
 - Corresponding Security Goals:
 - **Data Integrity**: protecting data against unauthorized writing
 - **Data Origin Authentication**: verifying the claimed identity of the originator of a message
 - **Access Control**: preventing unauthorized access to software functions
 - Example: changing the price of purchased goods

Typical Threats

- Repudiation
 - Threat: threat agent denies involvement in an event
 - Corresponding Security Goals:
 - **Non-repudiation**: the provision of irrefutable evidence about what happened and who was involved
 - **Audit**: chronological record of system activities to enable the reconstruction of events
 - Example: denying placement of a stock order

Typical Threats

- Denial of Service
 - Threat: threat agent destroys the usefulness of the system for legitimate users
 - Corresponding Security Goals:
 - **Availability:** ensuring availability of the system to legitimate users
 - Example: Distributed Flood Attack

Typical Threats

- Elevation of privilege
 - Threat: threat agent gets more access to information/communication/processing resources than he is authorized for
 - Corresponding Security Goals:
 - **Access Control**: preventing unauthorized access to software functions
 - **Entity Authentication**: verifying the claimed identity of a party one is interacting with
 - Example: “getting root”

Typical Threats

- Spoofing
 - Threat: Threat agent pretends to be someone/something he is not
 - Corresponding Security Goals:
 - **Entity Authentication:** verifying the claimed identity of a party one is interacting with
 - **Data Origin Authentication:** verifying the originator of a message
 - Example: phishing

Vulnerabilities

- A vulnerability is an aspect of (a component of) the system that allows a threat agent to realize a threat (i.e. break a security goal)
- Hence, vulnerabilities are relative to the capabilities of the threat agent
 - E.g. the system can be vulnerable to insiders but not to outsiders

Vulnerabilities

- Vulnerabilities arise through failures in:
 - Requirements,
 - Failure to identify all relevant assets, or specific threats
 - E.g. protecting against the threat of downloaded malicious code was not recognized as a requirement for OS security in the eighties
 - Construction,
 - Vulnerabilities in security components
 - E.g. a weak cryptographic algorithm
 - Vulnerabilities in application functionality
 - E.g. buffer overflows, SQL injection, ...
 - Operation
 - E.g. Setting an incorrect policy

Vulnerabilities

- Vulnerabilities can exist in the different layers of a software system
 - Application, e.g. SQL injection
 - Programming language runtime, e.g. type confusion
 - Middleware, e.g. open management interface
 - Operating system, e.g. FAT32 file system
 - Hardware, e.g. side-channels

Countermeasures

- Countermeasures are the mechanisms used to:
 - Reduce vulnerabilities, and hence:
 - Realize security goals
 - Counter threats

Countermeasures

- Countermeasures can be:
 - Preventive: avoid vulnerability
 - Detective: detect vulnerability exploitation
 - Reactive: handle incidents
- Countermeasures can be taken by:
 - Software engineers, programmers who develop software
 - Administrators, system managers who deploy software
 - ...

Software Engineer Countermeasures

- For vulnerabilities introduced in the requirements phase:
 - Security reqs engineering, threat analysis and modeling
- To address threats collected during requirements engineering
 - Security technologies
 - cryptography, access control mechanisms, authentication mechanisms, ...
- For vulnerabilities introduced during construction:
 - Secure programming, static analysis, safe languages, ...
- For vulnerabilities introduced during operation:
 - Documentation, operational procedures, secure defaults, ...

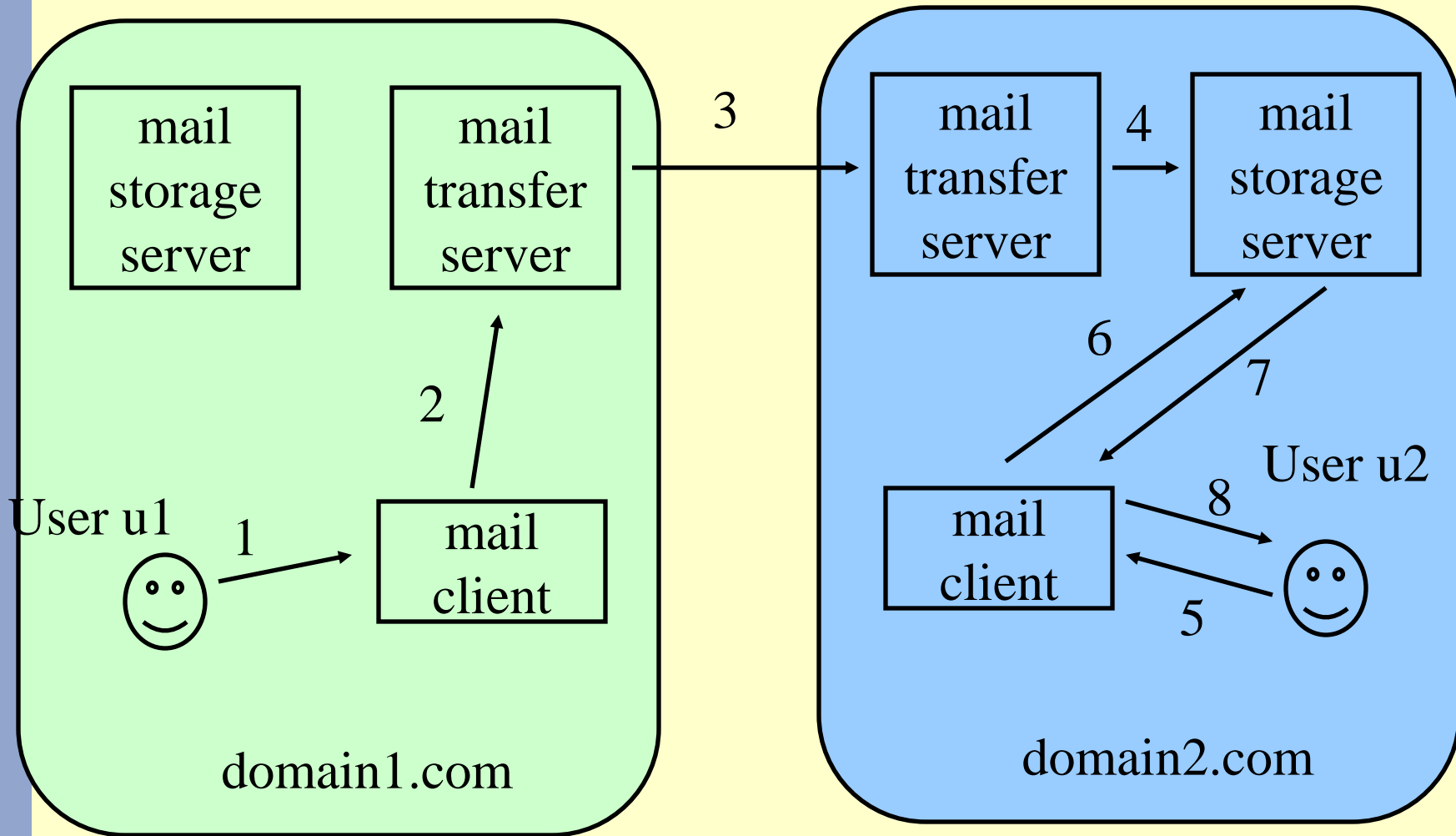
Administrator Countermeasures

- Preventive countermeasures:
 - deployment of additional protection: Firewalls, VPN's, ...
 - patching weaknesses where possible: CERT advisories, Windows Update, ...
- Detective countermeasures:
 - Intrusion Detection software or Fraud Detection software
 - Virus scanning
- Security solutions should be managed, supporting reactive countermeasures

Overview

- Introduction
- Key Concepts
 - Threats, Vulnerabilities, Countermeasures
 - Example
- Microsoft's Threat Modeling Process
- Conclusion

Simplified e-mail system



Assets

- E-mail messages
 - header
 - body
- Address book / contact information
- Client and Server machines
 - Storage space
 - Computational resources
 - Mail delivery service
 - Infrastructural software (OS,...)
- Network infrastructure
- ...

Threats

- E-mail message
 - Unauthorized reading of message body
 - Define what is “authorized”!
 - Tampering with message
 - In transit
 - In storage
 - E-mail spoofing (tampering with the from: field)
 - Repudiation
 - Of sending
 - Of receipt
 - ...

Threats

- Client and Server machines
 - Denial of service
 - E.g. Inbox storage space
 - Elevation of privilege
 - Server is a “trusted software layer”, making a limited functionality (sending/receiving mail) available to clients
 - If you can break the server out of this limited functionality (e.g. because of bugs), you can attack the server machine
 - E-mail viruses
 - E.g. through executable attachments
 - Unauthorized use of mail delivery service
 - E.g. Spam e-mail
 - ...

Threats

- Not all “undesirable behavior” can easily be related to assets:
 - Detecting when somebody reads his mail
 - Asset = “privacy of the reader”?
 - ...

Vulnerabilities

- Requirements: Security was not a concern in the original design of Internet e-mail
 - As a consequence, plain e-mail system is vulnerable to **all** identified threats
 - Countermeasures have been designed and integrated over the past decades
- Construction:
 - Implementation bugs in all e-mail components have been exploited
 - Sometimes amplified because of vulnerabilities in OS
- Operation:
 - Insecure configuration of e-mail: open mail relays, sendmail debug mode, ...

Possible Countermeasures

- Public key encryption and signing of e-mail, counters:
 - e-mail spoofing
 - tampering / reading messages in transit
 - repudiation of sending
- Password based authentication and access control
 - Counters tampering / reading messages in storage
- Sandboxing of attachments
 - Counters virus-spreading
- Code review or static analysis of server code
 - Counters elevation-of-privilege on server
- ...

Vulnerabilities in Countermeasures

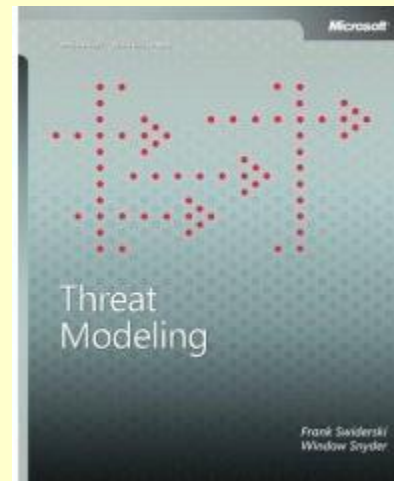
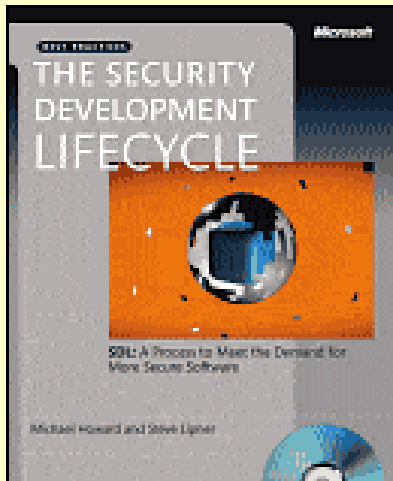
- Weak encryption algorithms
- Predictable generation of encryption keys
- Guessable passwords
- False negatives in static analysis for bugs
- ...

Overview

- Introduction
- Key Concepts
 - Threats, Vulnerabilities, Countermeasures
 - Example
- Microsoft's Threat Modeling Process
- Conclusion

Microsoft's Threat Modeling process

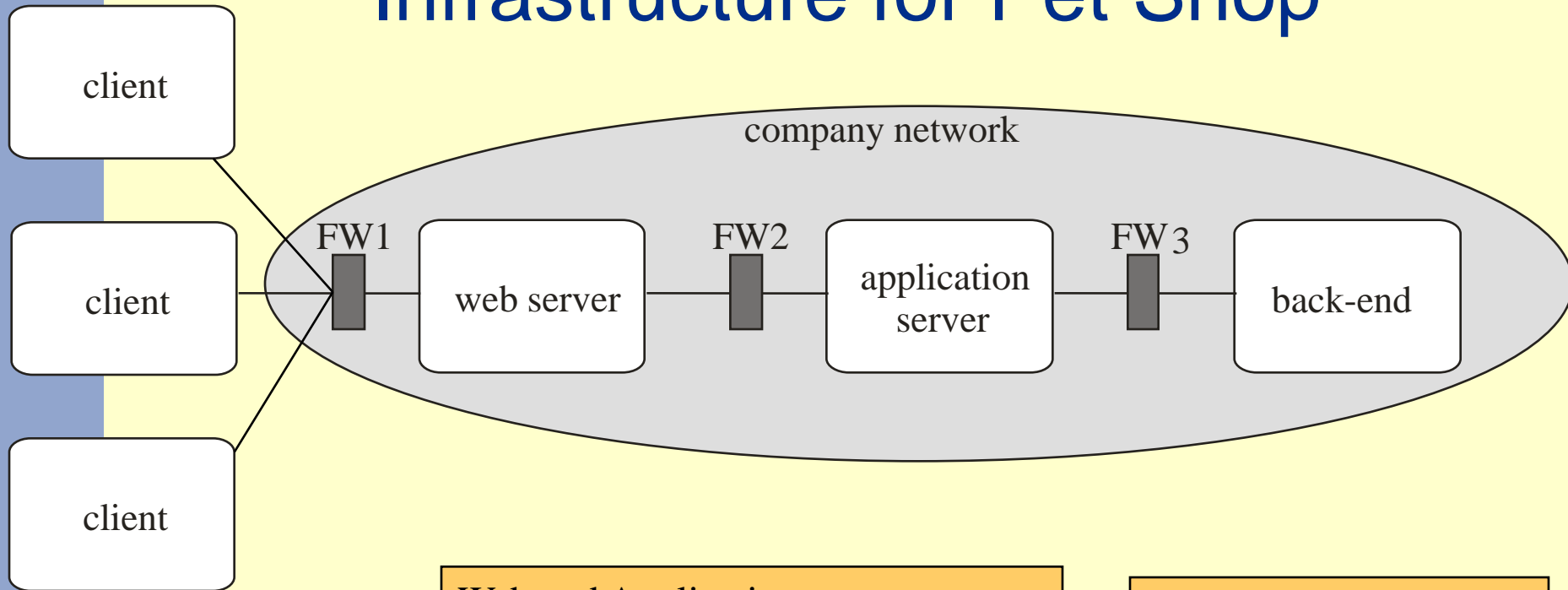
- As part of it's Secure Development Life Cycle, Microsoft has defined a threat modeling activity
 - Supported by documentation and tools
- We discuss it as an example:
 - Discuss the structure of this Threat Modeling process
 - Illustrate it on example application (taken from the left book)



Example application: Pet Shop



Infrastructure for Pet Shop



Browsers:

- Internet Explorer
- Firefox
- ...

Web and Application server :

- Windows Server 2003 with:
 - MS IIS
 - .NET Framework 2.0
 - ...

Back-end:

- SQL Server 2005

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

1. Define Use Scenarios

- Define how the system will be used, and how it won't be used
 - Bound the threat modeling discussion
- What threat scenarios are in scope?
 - E.g. do we care about insider threat agents?

2. List External Dependencies

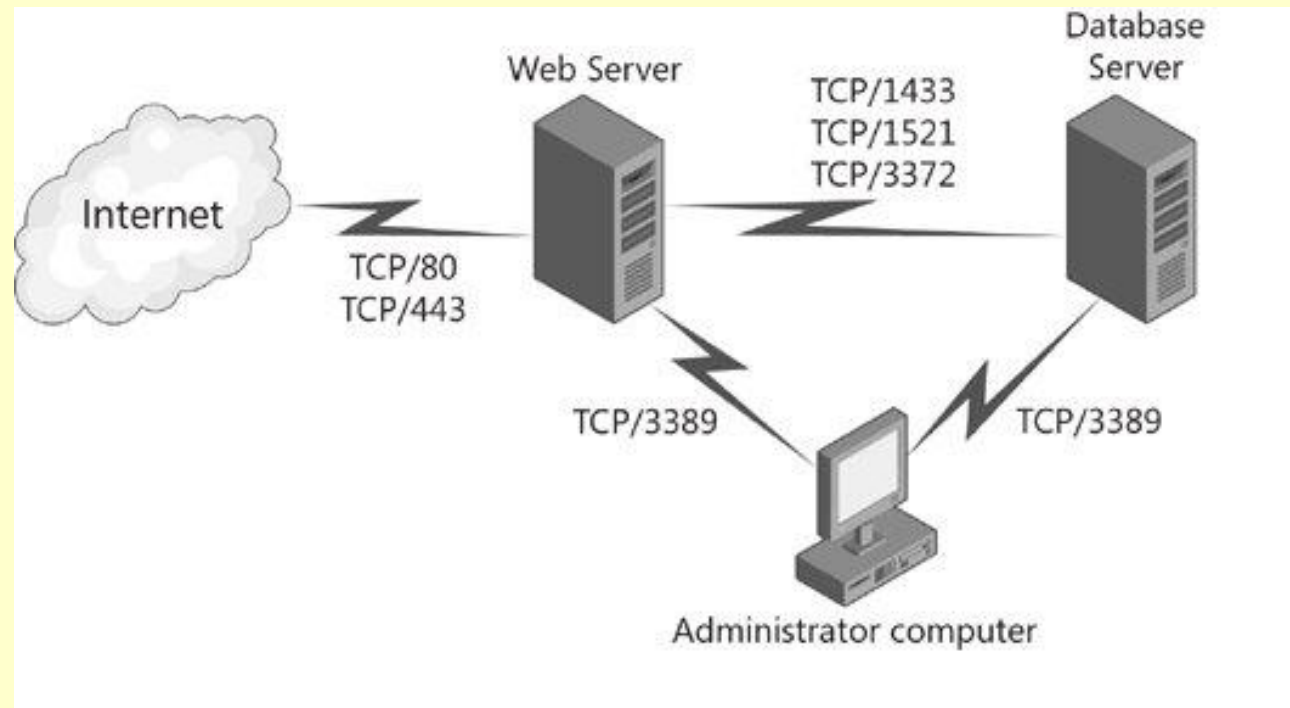
- All application software relies on infrastructural software for its correct functioning
- Document these dependencies
- E.g. Pet Shop depends on:
 - Web-based client s.a. Internet Explorer or Firefox
 - MS Windows Server 2003 operating system
 - MS IIS
 - SQL Server 2005
 - .NET Framework 2.0
 - ...

3. Define Security Assumptions

- What security guarantees do you expect from your external dependencies?
- E.g. for Pet Shop:
 - IIS and ASP.NET enforce authentication correctly
 - Physical access to the servers is restricted to administrators
 - Configuration files are protected using operating system access controls
 - ...

4. Create External Security Notes

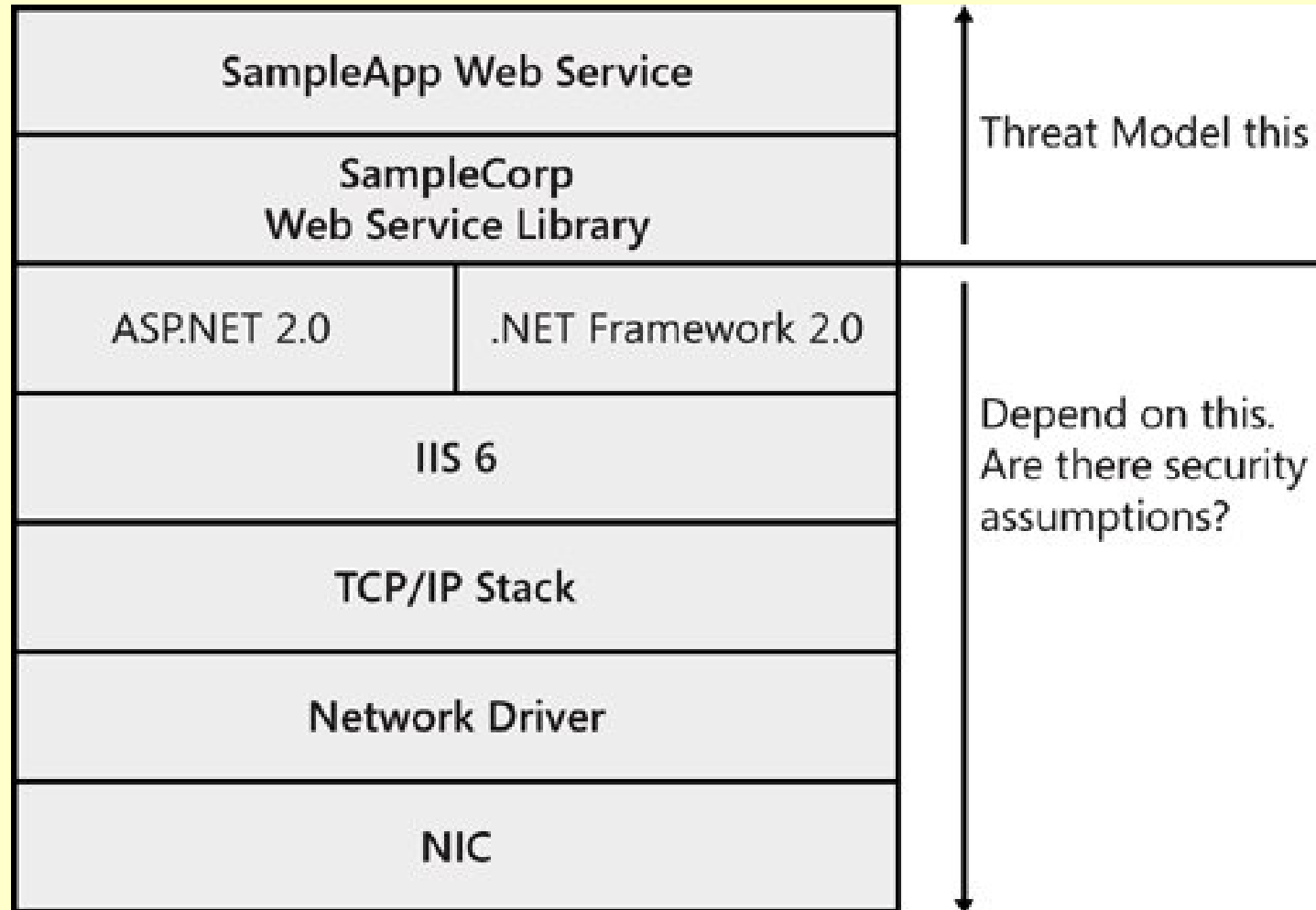
- Document security-relevant information for users of the application
- E.g.



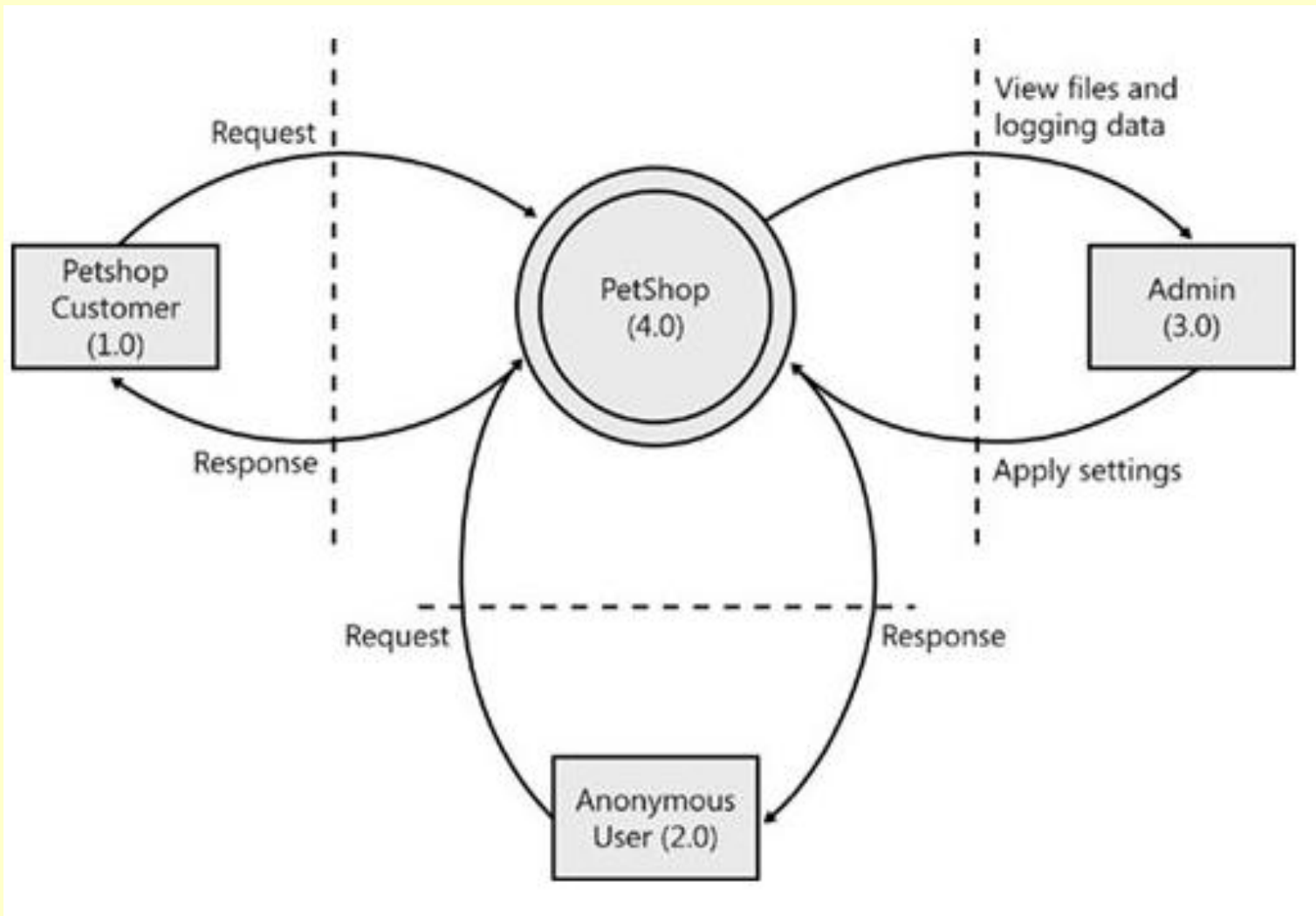
5. Model the Application

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

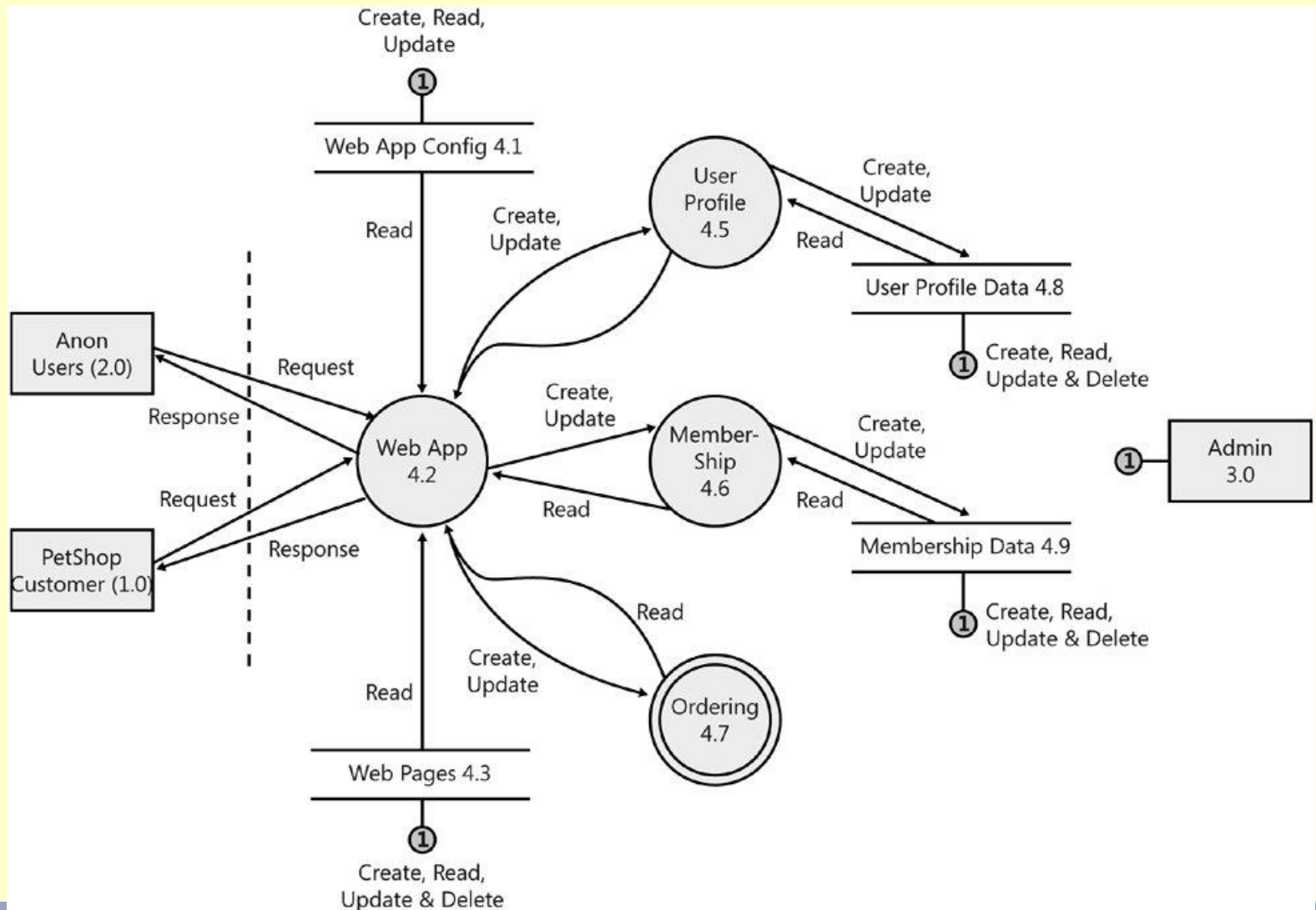
To model or not to model?



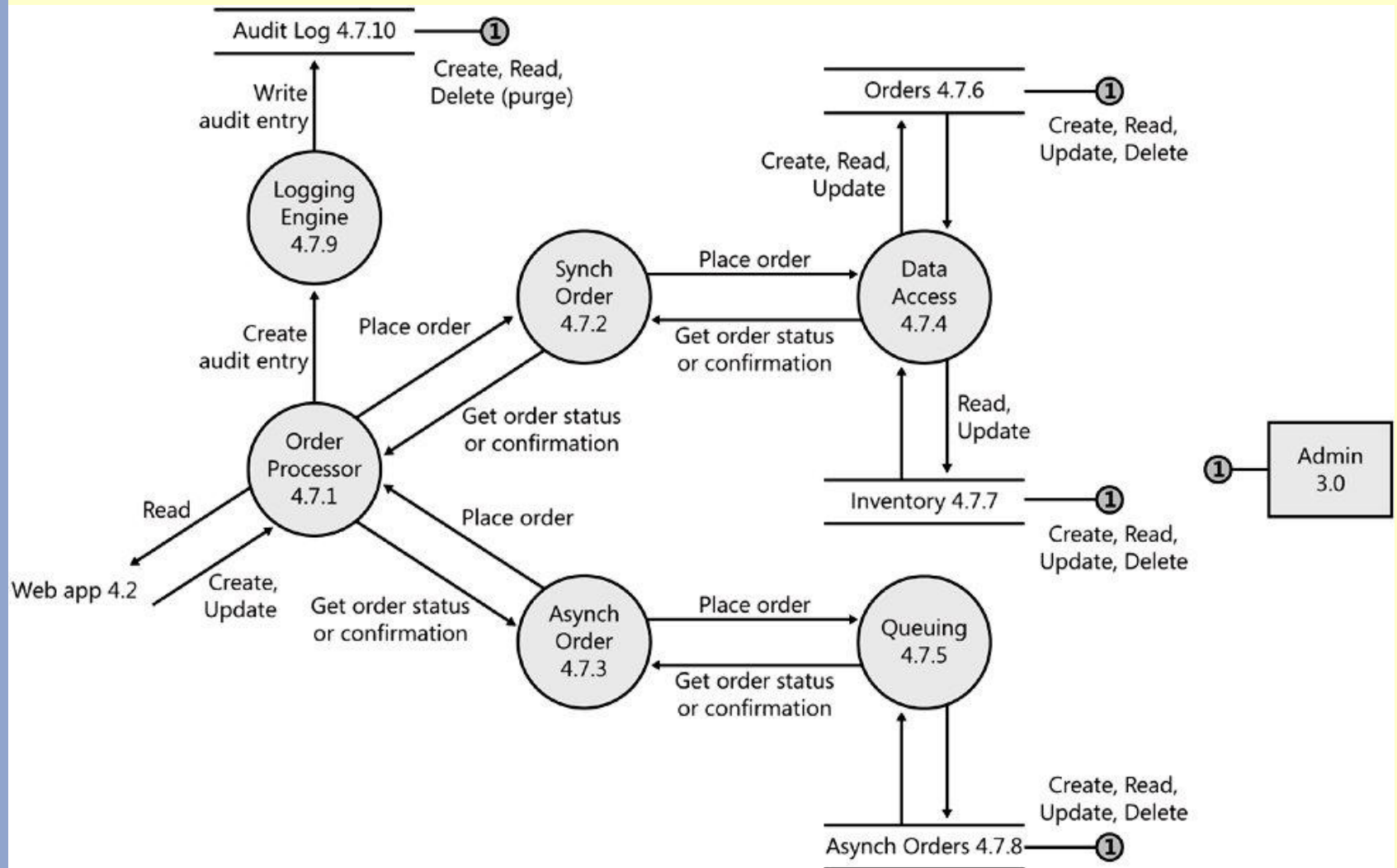
Example Model for Pet Shop



Example Model for Pet Shop



Example Model for Pet Shop



6. Determine Threat Types

- Microsoft's process uses the STRIDE threat types:
 - Spoofing
 - Tampering
 - Repudiation
 - Information Disclosure
 - Denial of Service
 - Elevation of Privilege

7. Identify Threats

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

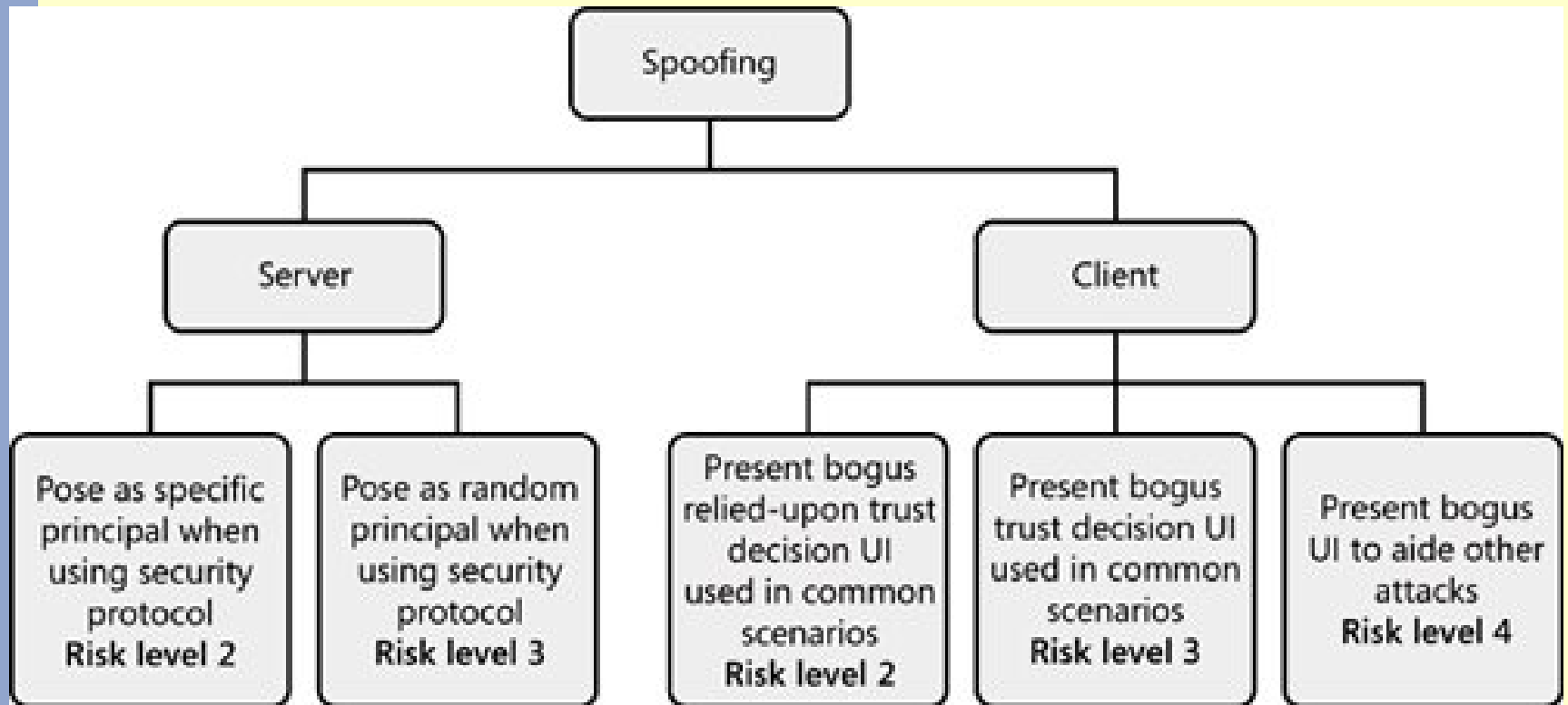
Table 9-5. Mapping STRIDE to DFD Element Types

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

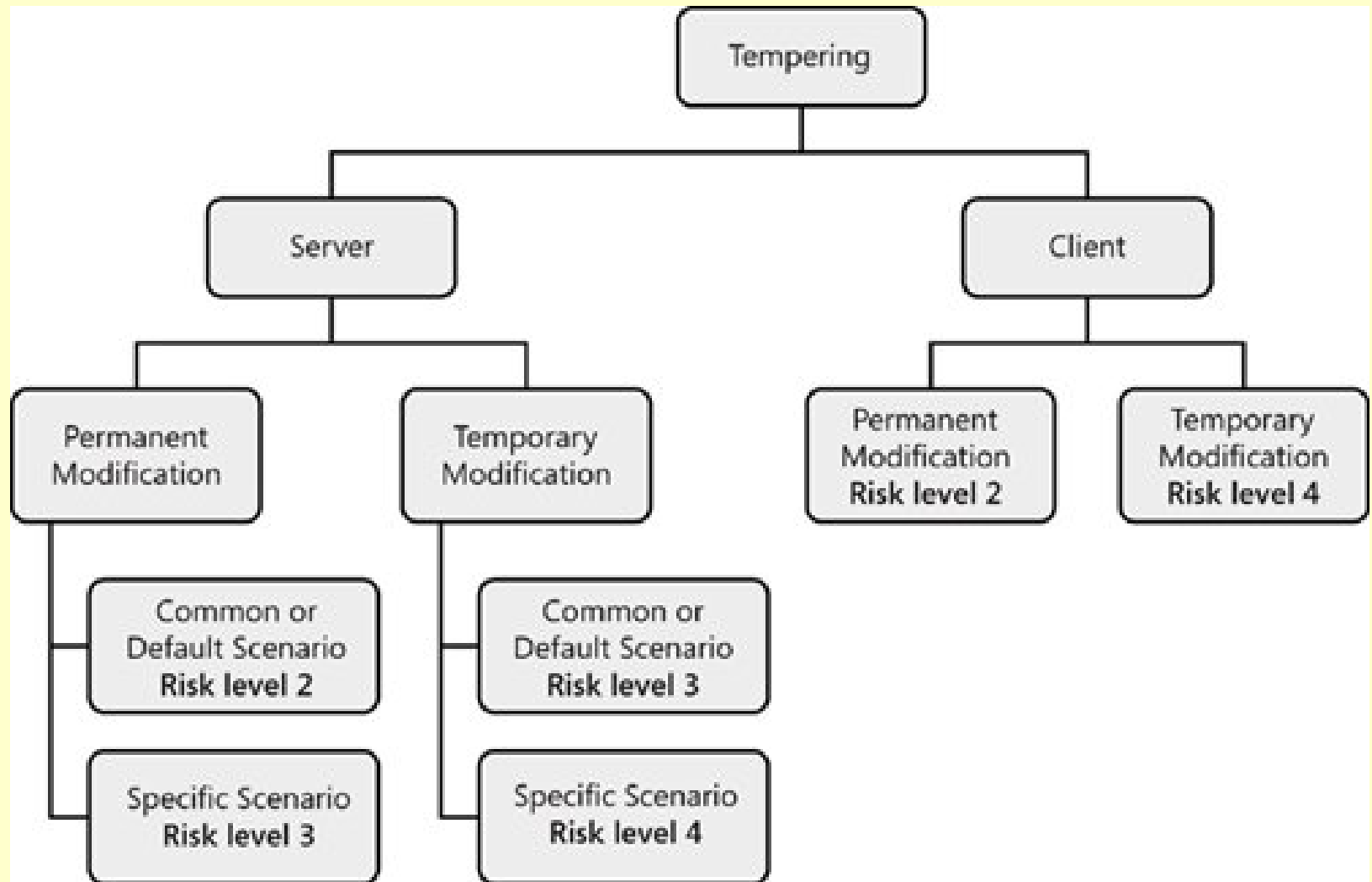
8. Determine Risk

- 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
 - ...

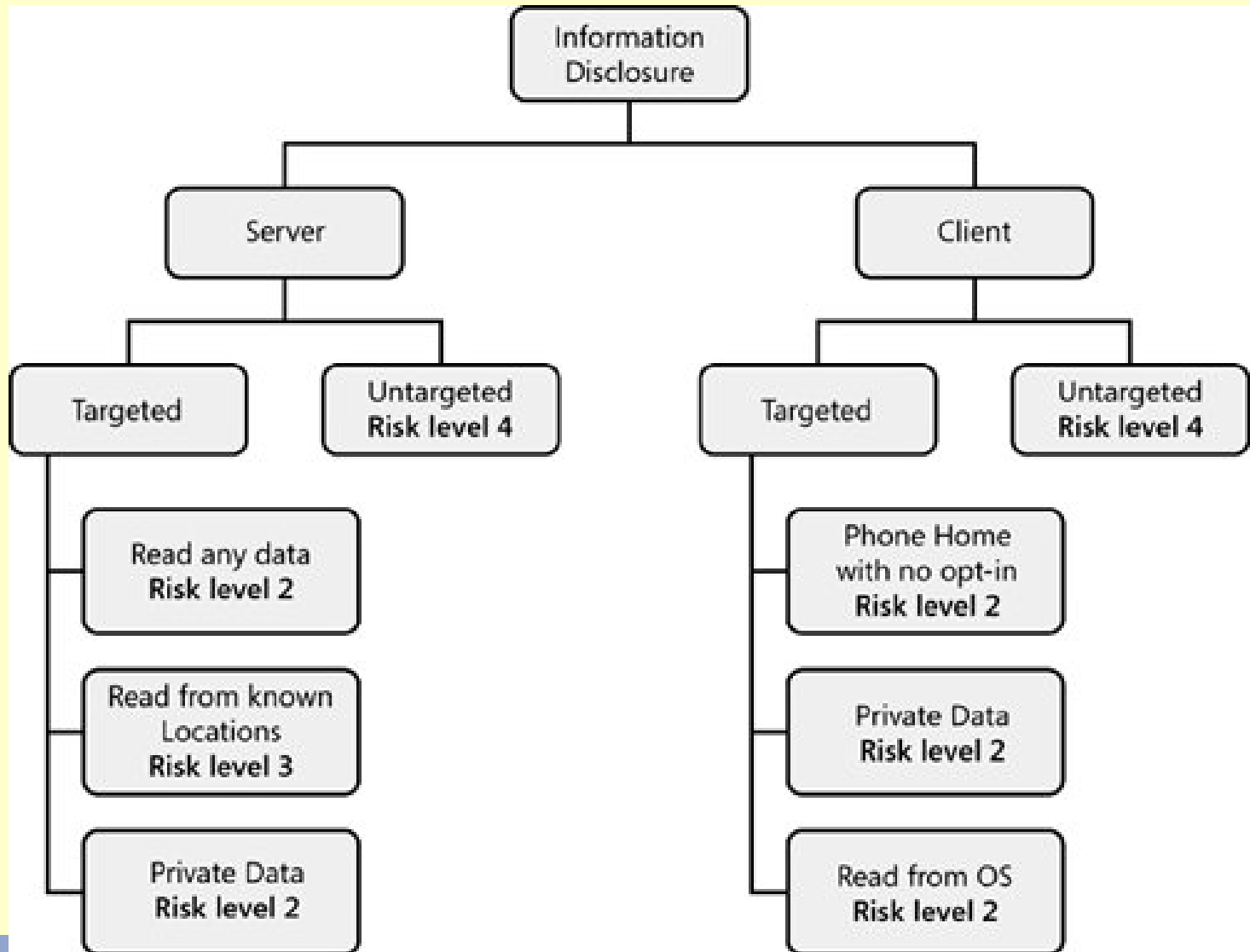
Spoofing threats risk ranking



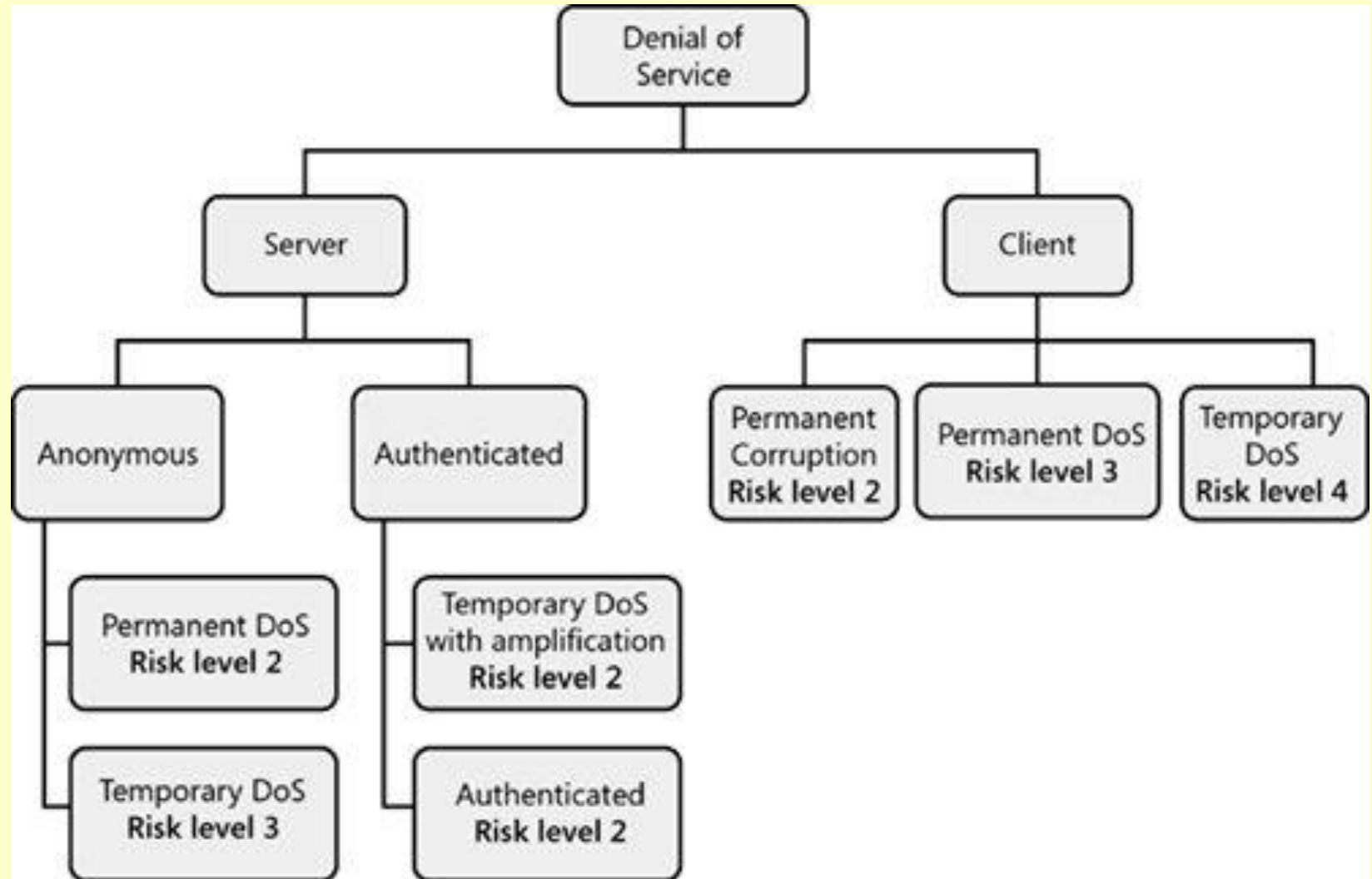
Tampering threats risk ranking



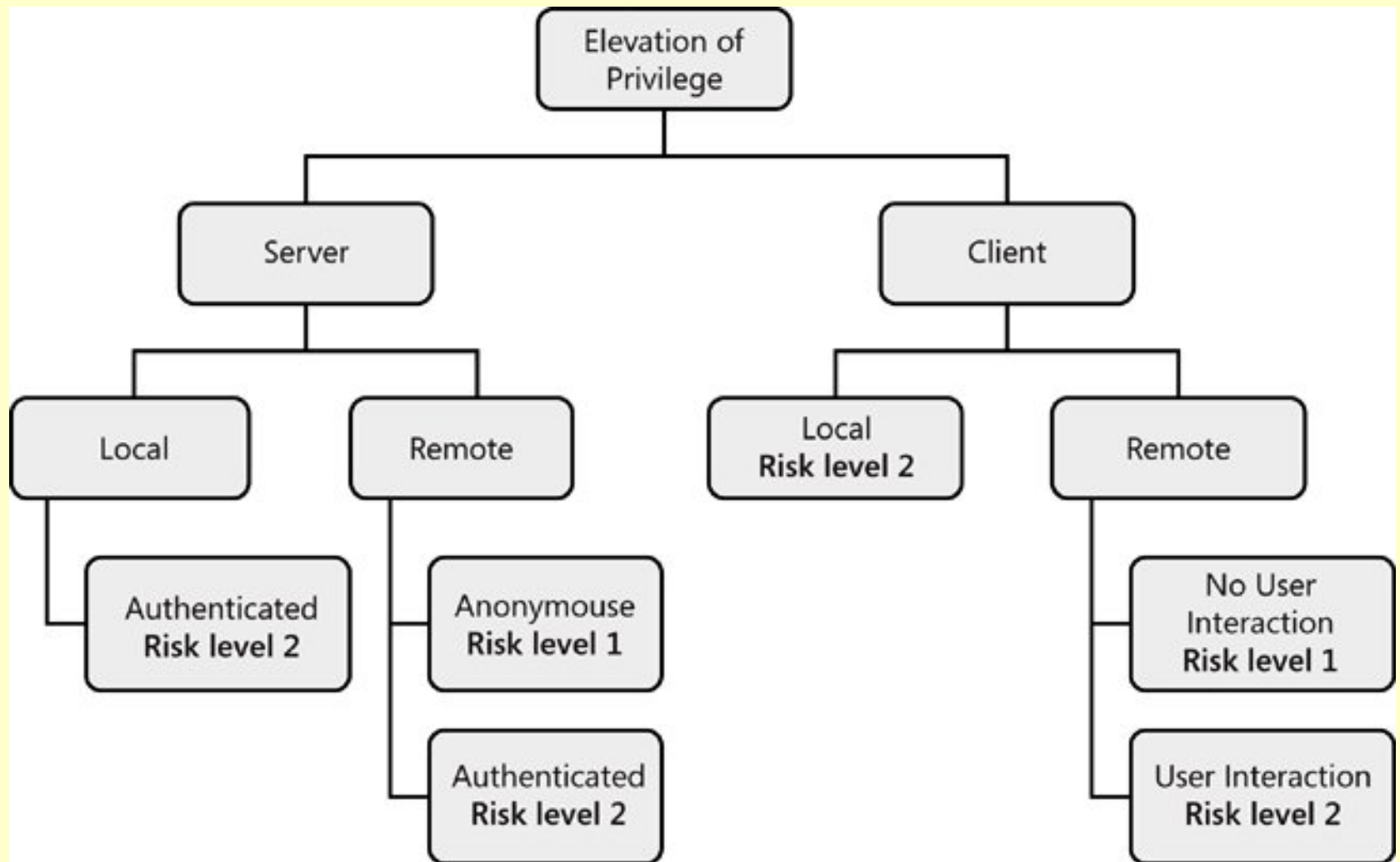
Information Disclosure risk ranking



DoS threats risk ranking



EoP threats risk ranking



9. Plan Mitigations

- Mitigation strategies
 - Do nothing
 - External security note!
 - Remove the feature
 - Turn off the feature
 - Warn the user
 - Dancing pigs syndrome!
 - Counter the threat with technology

Example countermeasures

Table 9-11. Defenses Used in Portions of Pet Shop 4.0

Asset	Asset Type	Example Threat	Example Mitigation
(1.0 → 4.2 → 1.0)	Data flow from Pet Shop user to Web application and back	I	SSL/TLS (also mitigates the tampering threat)
4.7.10	Audit log data store	T	ACL and MAC
4.7.1	Order processor process	T and E	ACL, MAC, and reduced process privilege

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

Overview

- Introduction
- Key Concepts
 - Threats, Vulnerabilities, Countermeasures
 - Example
- Microsoft's Threat Modeling Process
- Conclusion

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