

## 9. Security Requirements Engineering and Threat Modeling – Part 1

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

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 1

### Content

- Security requirements engineering / threat modeling overview – why they are fundamental activities
- The basic process of security requirements engineering / threat modeling based on the STRIDE approach
- Attack trees as another method to think about threats against a system
- The importance of knowing your enemy
- Documenting security requirements
- Further security requirements engineering / threat modeling methods
  - Abuse cases, attack patterns

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 2

This topic is covered, e.g., in the following books:

- Michael Howard and Steve Lipner. *The Security Development Lifecycle*, ISBN-13: 978-0735622142 (Chapter 9: Stage 4: Risk Analysis)
- Gary Mc Graw. *Software Security: Building Security In*. ISBN-13: 978-0321356703 (Chapter 5: Risk Analysis, Chapter 8: Abuse Cases)

There are also some online resources:

- OWASP. Threat Risk Modeling. [https://www.owasp.org/index.php/Threat\\_Risk\\_Modeling](https://www.owasp.org/index.php/Threat_Risk_Modeling)
- OWASP. Application Threat Modeling. [http://www.owasp.org/index.php/Application\\_Threat\\_Modeling](http://www.owasp.org/index.php/Application_Threat_Modeling)
- Shawn Hernan, Scott Lambert, Tomasz Ostwald, and Adam Shostack. *Uncover Security Design Flaws Using The STRIDE Approach*. <http://msdn.microsoft.com/en-us/magazine/cc163519.aspx>
- Tom Olzak. *A Practical Approach to Threat Modeling*. [http://adventuresinsecurity.com/blog/wp-content/uploads/2006/03/A\\_Practical\\_Approach\\_to\\_Threat\\_Modeling.pdf](http://adventuresinsecurity.com/blog/wp-content/uploads/2006/03/A_Practical_Approach_to_Threat_Modeling.pdf)
- The SQUARE Methodology. <http://www.cert.org/sse/square/>

The slides in this chapter partly contain information from these books and articles.

## Goals



- You understand the **importance of security requirements engineering and threat modeling during early phases** of the secure software development process
- You know and understand the **security requirements engineering / threat modeling process** and can apply it to software systems or applications in development yourself
- You know and can apply two fundamental methods to think about threats / attacks on a system: **STRIDE** and **attack trees**
- You understand the **importance of knowing your enemy** to base your security requirements engineering / threat modeling process on realistic assumptions
- You know how to **document security requirements**

---

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 3

## Software Security (SSI)



### Introduction

---

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 4

## The Need for Security Requirements Engineering (1)



- Security Requirements Engineering means **determining security requirements early** in the development lifecycle
- Just like "normal" requirements engineering, it's a fundamental and important activity
  - Missed security requirements are often **difficult and expensive to "fix"** later
- In practice, security requirements engineering is **often neglected** today
  - The focus of requirements engineering is mainly on **functional aspects**
- If security requirements are identified, they are **often very general and taken from standard lists**, e.g.
  - Use passwords, use encrypted communication channels, use firewalls etc.
  - This is usually not enough to specify the "right level of security" for a particular system

---

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 5

## The Need for Security Requirements Engineering (2)



- If security requirements are **not properly defined**, the following problems arise:
  - There is no basis for integrating appropriate **security controls** into the system architecture and design
  - There is no basis for **implementing appropriate security controls**
  - There is no basis to perform focused **security tests**, which should be driven by security requirements
- → Without adequate security requirements, it's therefore **very unlikely the resulting system is going to be secure**
- Proper security requirements engineering has the goal to prevent all this by providing **foundation for all subsequent security activities**

---

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 6

## Threat Modeling (1)



- Threat modeling is **closely associated with security requirements engineering**
  - Security requirements engineering without threat modeling is hardly going to produce good results
  - In fact, security requirements engineering mainly consists of threat modeling
- The goal of threat modeling is to find out whether the specified security requirements or security controls are **appropriate**
- Applied to a secure development process, threat modeling consist of the following fundamental steps:
  - **Identify possible threats** against the system (its assets)
  - Based on these threats, **identify vulnerabilities**
    - Taking into account already defined security requirements or security controls
  - The identified vulnerabilities provide the basis for **additional security requirements**

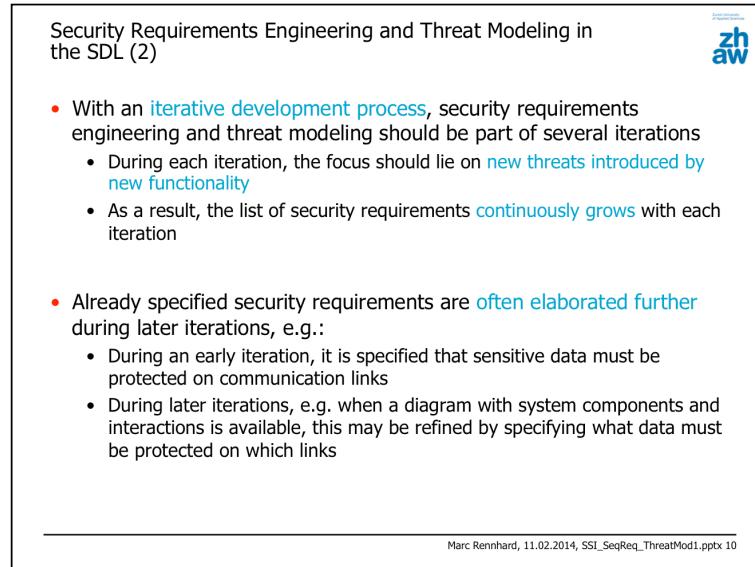
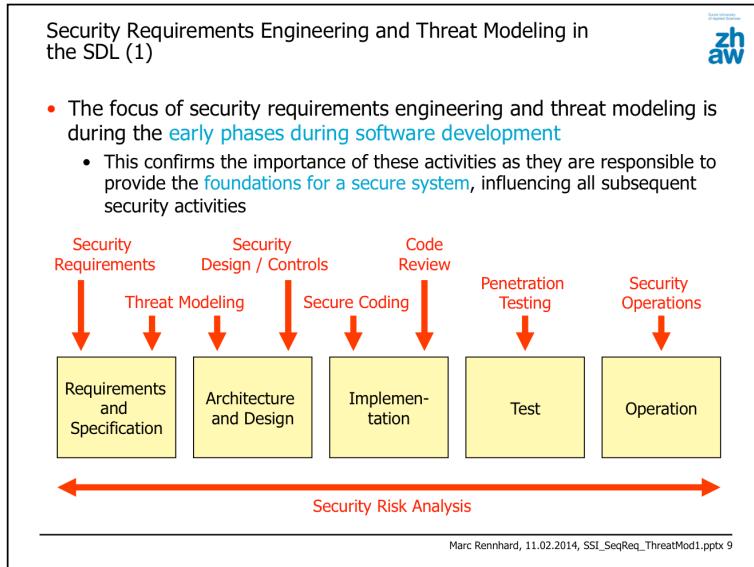
Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 7

## Threat Modeling (2)



- Besides as an activity during a secure development process, threat modeling can also be **used as an activity on its own**
  - E.g. to perform an analysis of a completed system (design) as an external security expert
- But the result will be typically the same: A list of **identified system vulnerabilities**
  - Usually rated with respect to their risk / criticality (see chapter on security risk analysis)

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 8



## Functional Security Features – and Beyond (1)



- When starting to think about security requirements in a project, **some "obvious" basic security requirements** will be defined right away
  - So threat modeling is not necessary to define every single security requirement
- Examples of "**obvious**" security requirements of an e-shop:
  - Sensitive information such as payment details must be transmitted via a cryptographically secured communication channel
  - Only sales personnel is allowed to view and modify the orders of all customers
  - Customers can only see their own orders

---

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 11

## Functional Security Features – and Beyond (2)



- However, these "obvious" security requirements are often mainly **driven by the functional aspects** of the system to be developed
  - "We transfer credit card numbers, so we must encrypt communications"
  - "There's a special area for sales personnel, so we need access control to make sure other users cannot access this area"
  - "A customer can see his own orders, so we need again access control"
- If we want the software to be really secure and robust, such a **functional view is not enough**
  - Typically, attackers do directly attack security functions, but they look for "places" where security was forgotten
  - → Attackers look for ways to **circumvent** the security measures

---

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 12

## Functional Security Features – and Beyond (3)



- So to access credit card information, the attacker **most likely won't try to break TLS**, but could attempt the following:
  - He'll ask himself where this data may be accessible in non-encrypted form?
  - E.g. in the database (where it might be accessible using SQL injection or by compromising the database server)
- And to access the data that is protected via access control mechanisms, the attacker **most likely won't attempt to directly break the access control mechanism** itself, but may do the following:
  - He'll likely check whether access control was forgotten in certain parts of the application (e.g. via forced browsing)
  - Or he may try to access the data via other means, e.g.:
    - Via injected JavaScripts which reads the data in the user's browser
    - Again via SQL injection to directly read the data from the database
    - By getting the credentials of a user that has the necessary access rights via various means (guessing, social engineering, via XSS / SQL injection...)

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 13

## About Circumventing Security...



Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 14

## Functional Security Features – and Beyond (4)



- The most important goal of threat modeling is to **uncover these “not so obvious” threats**
- This also requires you to start thinking like an attacker, i.e. to **“put on the black hat”**
- During threat modeling, ask yourself: **If I were attacking** the system...
  - **What would I want?** (steal money, get administrator access, take the system offline, deface the website...)
  - **How could I accomplish this?** (flood the target, elevate privileges, guess password, perform SQL injection...)
    - Be creative and consider a wide range of attacks!
- → This will help you to look at software (and security) **from an entirely different side**
  - It takes your focus away from existing (functional) security measures and helps you to consider a **wide range of possibilities** to achieve an attack goal



Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 15

## “What”, not “How”



- Security Requirements should describe **“what” has to be done**, not **“how” is has to be done**
  - How a requirement is going to implemented (technical details) will be determined when defining the **system architecture and design**
- A **reasonable security requirement**:
  - “The web application must employ an authorization mechanism, authorization must be checked on every single request”
  - The choice of the mechanism is not determined by the requirement
- But security requirements should also be **precise** and not leave too much room for “security interpretation”
  - “Passwords must have at least the following complexity:  $\geq 10$  characters, at least one digit and at least one special character” is better than
  - “The application must use strong passwords”

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 16

## Security Requirements Engineering – The Process

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 17

- Security Requirements Engineering is a young discipline and far from being a “scientific method”
- Several method have been proposed, but fundamentally they are similar to each other
  - There’s also not a “currently best method” or a dominating “industry standard” one should follow
- We do not follow one specific proposed method here, but use a combination of proposed approaches
  - It is mainly influenced by the OWASP threat risk modeling process and Microsoft’s threat modeling process

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 18

### OWASP Threat Risk Modeling

[https://www.owasp.org/index.php/Threat\\_Risk\\_Modeling](https://www.owasp.org/index.php/Threat_Risk_Modeling)

### Microsoft’s Threat Modeling Approach

It's part of the Microsoft Security Development Lifecycle process (<http://www.microsoft.com/security/sdl/discover/default.aspx>)

Security Requirements Engineering – The Process (2)

The security requirements engineering process consists of **6 steps**:

1. Identify the **business and security goals** of the system
2. **Collect information** about the system so you get a good understanding about its purpose and functions
3. **Decompose the system** to understand its internal workings and as a basis for the following steps
4. **Identify threats** that are relevant for the system
5. **Identify vulnerabilities** in the system, given the threats
6. **Mitigate the threats** where necessary by proposing adequate security requirements

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 19

mitigate = entschärfen

**Step 1: Identify Business and Security Goals**

- The first step when performing security requirements engineering is **identifying the business goal and the security goals**
  - This helps to understand the overall goals and purpose of the system and provides the basis for all subsequent steps
- Try to express the **business goal in one or at most a few sentences**
  - This forces you to really think about the main business goal
- Driven by the business goal, **identify a few security goals**
  - The security goals should be important to achieve the **business goal**, but they may also be required due to general **policies or regulations**
  - Again, focus on the most important, overall security goals and don't go too much into the details
  - Something between 3 and 6 security goals is usually reasonable
- Business and security goals are usually defined by performing **interviews with the stakeholder**
  - E.g. the external client or internal project sponsor

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 20

#### Determining Business and Security Goals

Of course, it may be the software development team that is performing security requirements engineering / threat modeling is well aware of business and security goals. However, it may also be that special internal teams are consulted for this phase or an external company is given the task to perform threat modeling. In these cases, the persons performing the process may not be aware of these goals, so it's important to consult the correct resources (e.g. stakeholders) to get the information. In any case, it's important that all personnel involved in threat modeling has a clear understanding of the business and security goals.

## The University Library Application (1)



- As an example to illustrate the entire process, we use a fictional web application, the "University Library Application"
- Current state of the project:
  - Several functional requirements and even some security requirements have already been defined
  - Designing the system has started
  - The team is suddenly unsure whether security is considered adequately, which is where you are hired as a security expert
- As a security expert, you understand the **importance of security requirements engineering**, so this is where you start
  - As some security requirements have already been defined, this is used as the basis for your analysis

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 21



## The University Library Application (2)

- According to the process, the first step is **identifying the business and security goals**
- Carrying out **interviews with the stakeholder** results in the following:

### Business Goal:

The system allows its users efficient online access to the university library

### Security Goals:

- The integrity of the system and its data shall be maintained
- The confidentiality of personal user data and credentials shall be guaranteed
- Any system activity is logged and can be linked to a user

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 22

### The University Library Application

It is based on an example by OWASP and has been modified for this course.

Source: [http://www.owasp.org/index.php/Application\\_Threat\\_Modeling](http://www.owasp.org/index.php/Application_Threat_Modeling)

### Security Goals

- The first security goal is clearly driven by the business goal: Only a correctly functioning system (system integrity) and correct data (data integrity) will allow efficient online access.
- The second security goal is mainly driven by university policies: Personal user data shall be kept private and credentials must not be leaked, but it's also needed to support the business goal as leaked credentials of librarians allow abusing the system (e.g. create a huge number of user accounts until the system becomes non-operational).
- The third goal was added to detect potential abuse of the system, e.g. by a malicious librarian or an attacker that managed to steal the credentials of a user, which helps to identify the cause of a security breach and which therefore helps to resume normal operation after an incident, which in turn supports the business goal.

## Step 2: Collect Information (1)



- The second step is getting a **good understanding** about the system
- This includes the following:
  - What **functions** does the system provide in detail?
  - Who **uses** the system (users, administrators, other systems...)?
  - What **data** does the system process?
  - What are the most important **assets**?
    - Should be driven by the business and security goals
    - E.g. data that are directly necessary to achieve the business goal are more valuable than others
  - How does the system **work** (components, technologies, interactions...)?
  - What are the **external dependencies** that must be considered?
    - This is important as they may have security implications that are not directly under control of the developing team
  - What is the **service level** the system must comply to (e.g. 24/7)?
  - What **security requirements or controls** have already been defined
  - ...

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 23

## Step 2: Collect Information (2)



- This can be achieved by:
  - Analyzing available **artifacts**
    - Specifications, security requirements, design & architecture documents etc.  
→ any documentation available (as pure text, UML diagrams or whatever)
  - Carrying out **interviews** with involved people (typically engineers)
- The more information that is available, **the more thorough** the subsequent steps can be performed
  - In early phases, there often exists only a general description and some functional requirements
    - Only basic security requirements can be derived (with limited time effort)
  - As the system architecture and design advances, more information will be available allowing for a more detailed analysis
    - Which likely will produce further and more specific security requirements
- Security requirements engineering is **therefore usually applied repeatedly** while the overall system design is developed
  - Which fits well with an iterative software development process

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 24

### The University Library Application (3)



Collecting information on our example application reveals the following:

- General information of the application:
  - The university library application is a web application to provide librarians and library users (students and staff) with online services
    - As this is the first version, the functionality will be limited
  - There will be three types of main users of the application:
    - Students, Staff, Librarians
  - Offered functionality:
    - Students and staff will be able to log in and search for and reserve books
    - Staff members can additionally request new books
    - Librarians will be able to log in, search for books, maintain the list of available books, view book reservations, and add and delete users
  - Users have individual login credentials, librarians share one account
  - High availability is not needed, downtimes during the weekend or even partial downtimes during workdays are not critical

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 25

### The University Library Application (4)



Some security requirements have also been defined and documented:

- Users and librarians can access the web application server only via cryptographically protected channels (HTTPS / TCP port 443)
- When TLS is used, servers are authenticated using Extended Validation (EV) certificates
- Direct access to the web and database servers for configuration and log monitoring is only allowed to administrators and only via SSH
- Administrators must have at least security clearance "medium" (employed  $\geq$  5 years, regular background checks)
- All performed actions are logged locally on the web application server
- All queries are logged on the database server
- The SSH daemons log successful logins and login failures

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 26

## The University Library Application (5)



There are also some [external dependencies](#) for the application:

- The university library application runs [on standard university servers for web applications](#)
  - Linux, Java, Tomcat
  - Hardened according to the university's server hardening standards, which are considered adequate for typical university services
    - Includes latest operating system and software security patches
- The database server will be [MySQL on a Linux server](#), also hardened according to university standards
- The server components are [physically located](#) in the same network as servers of other applications with similar security requirements, access to the network is protected by [packet filtering firewalls](#)

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 27

## The University Library Application (6)



We also determine the [assets \("the crown jewels"\) of the application](#), which helps to identify the most important system components:

- [Both server-side systems](#) (integrity of systems and data)
  - Otherwise, efficient system usage is not possible
- [Personal user data](#) (account details, reservations)
- [User and librarian credentials](#) (username & password)
  - To keep personal user data confidential and to prevent system abuse
- [Administrator credentials](#), which would provide attackers with powerful access to the system
- [The logs](#), as they allow identify problems quickly and resume normal operation

Don't just pick the assets at random, but pick the ones that are important with respect to defined business and security goals!

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 28

### Assets are driven by Business and Security Goals

The assets listed above are driven by the business and security goals we defined earlier:

- The business goal ("The system allows its users efficient online access to the university library") and first security goal ("The integrity of the system and its data shall be maintained") imply that the server-side system components (integrity of systems and data) are important assets.
- The second security goal ("The confidentiality of personal user data and credentials shall be guaranteed") implies that personal user data and user and librarian credentials are important assets.
- Administrator credentials are important assets, as owning them allows to threaten the business goal and all three security goals, because with the administrator credentials, access to all application components and all data in the database is likely possible.
- Finally, the logs are assets as they are needed to fulfill the third security goal ("Any system activity is logged and can be linked to a user").

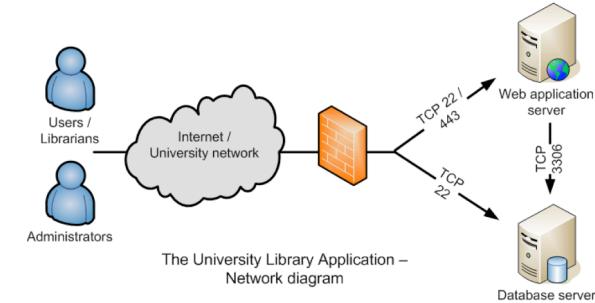
### Step 3: Decompose the System

- The next step is to **decompose the system into its component**
- There are **multiple ways to do so**, e.g. by drawing a network diagram or a data flow diagram
- It's important to include all relevant information**
  - System components such as servers
  - Different users
  - Interactions between users and system components and between system components
  - Interactions with external systems or applications
- Choose a **reasonable level of detail** that allows to perform a sound and efficient threat modeling process
  - Often, it's **not necessary to include very many details** as this will only make threat modeling much more complex without producing significantly better results

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 29

### Network Diagram (1)

- A **network diagram** typically includes the most important users, system components, communication relationships, and protocols / ports
  - Suited as a **first overview**, also to make sure all involved persons have the same and correct understanding of the system
  - If threat modeling is done by an external team, this can be verified by showing the network diagram to the development team



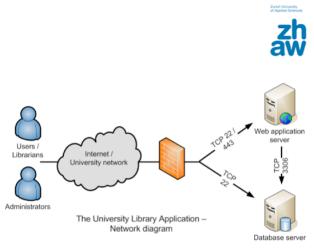
Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 30

### Network Diagram

It may of course be well possible that such diagrams are available as system documentation and have been collected during step 1. In general, it's always a good start to begin with a network diagram as it shows – although on a high-level – the most important components and how they interact.

## Network Diagram (2)

- This network diagram **can already be used** to think about threats and security requirements:
  - The web application server is the entry point into the application, so it must **authenticate users** before allowing access to restricted information
  - Credentials are sent across the network, which could be sniffed by attackers, so this must be **protected** (as is already the case with HTTPS/SSH)
  - All servers in the network can access the database server, so it must **authenticate the web application server** before granting access
- But in general, network diagrams **only serve as a first overview** how system components interact
- To really perform threat modeling, one uses more detailed diagrams, e.g. **data flow diagrams**



Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 31

## Data Flow Diagrams (1)

- **Data Flow Diagrams (DFD)** are well suited to decompose a system at various levels of details
  - The system components are depicted with different **elements**
- **Process:** represents a **task** within the system; a task usually receives and processes data and forwards data to other components
  - E.g. the authentication process in an application or the method that processes orders made by customers
- **Multiple Process:** used to represent a **collection of processes**
  - Examples: an entire mail server or a web application server



Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 32

## Data Flow Diagrams (2)

- External Entity:** used to represent any entity outside the system that interacts with the system
  - E.g. users, administrators, but also external components used by / interacting with the system (e.g. a 3rd party web service)
- Data Store:** used to represent locations where data is stored
  - Examples: database files, configuration files, log files
- Data Flow:** represents interactions within the system; the arrow indicates the direction of the data flow
  - Examples: data sent over a network or data exchanged within components of a mail server

```

graph TD
    Users[Users] --> OrderData[Order Data]
    OrderData --> PlaceOrder[Place Order]
  
```

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 33

## Data Flow Diagrams (3)

- Privilege Boundary (or Trust Boundary):** used to mark the change of privilege / trust levels as the data flows through the system
  - Example 1: Data flow between external users and a web application
    - Depending on their rights, the users are typically only allowed to use a subset of all functions offered by the web application
    - This is a change of privilege levels because data from (typically) low-privileged external users are processed by the high-privileged web application
  - Example 2: Data flow between web application server and database server
    - The web application is likely not allowed to use all functions (CRUD) available on the database server
  - In general, privilege boundaries usually imply that some sort of access control must be performed:
    - E.g. to guarantee low-privileged users can only access the allowed functions
  - When identifying threats and vulnerabilities (steps 4 & 5), it's usually a good idea to analyze DFD elements "close to privilege boundaries" with extra care

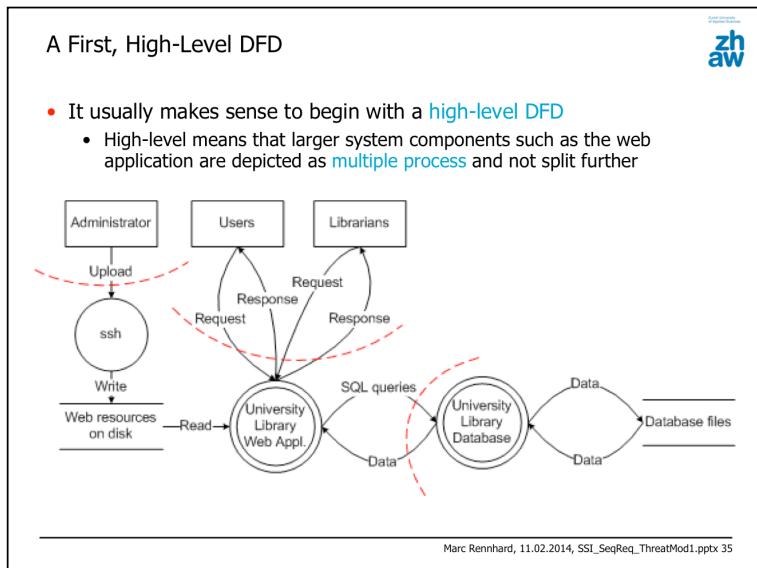
Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 34

### Privilege Boundary

They usually delineate data moving from "low to high privilege" areas and vice versa. For instance, an anonymous user creates a request to a higher-privilege process (e.g. a web application) for consumption. As the web application most likely has powerful (high) privileges (e.g. perform a shopping transaction in a e-shop system), it is necessary to closely inspect this trust boundary to make sure that anonymous users can only do what they should be allowed to do. But the "return channel" must also be considered to make sure the web application does not leak sensitive (high trust) information to anonymous users, e.g. showing the shopping history of a specific user to an anonymous user. External entry points are therefore typical candidates for trust boundaries, except in the case where the system serves purely informational purposes without distinguishing between different categories of users.

Privilege Boundaries also exist between systems. For instance, the web application process may access a database process but most likely, the web application should not be able to do everything (e.g. changing the DB schema) and there may be auditing data in the DB that should not be leaked. So again, a privilege boundary makes sense to "keep this in mind" during the subsequent analysis.

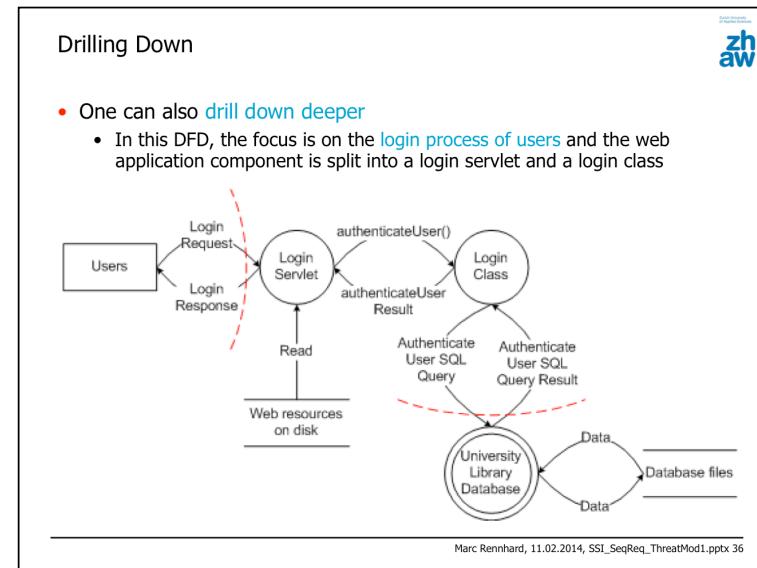
In general, privilege boundaries usually imply that some sort of access control must take place and that measures to prevent information leakage are provided. And in general, DFD elements "close to privilege boundaries" should be analyzed with extra care as they correspond to "entry points from the outside" into the application, where attackers likely will invest a lot of effort to find vulnerabilities.



Note that the DFD above has already left out some details, e.g. administrative access to the database via SSH, the SSH logs and also the logs on the web application and database servers.

#### Privilege Boundaries

There are obvious privilege boundaries between the users and the system components, illustrated by the top two boundaries. But there's another one between the web server and the database server, which makes sense as to access the database, credentials are needed, which (hopefully) are only in possession of the web server and of no other server in the network (or anywhere else)



#### Drilling Down

DFDs can be drawn at every possible detail level. Use the level that is most suited for your threat modeling effort but in general, starting at a high-level and making sure that the important trust boundaries (and data flows across these boundaries) are present is a good idea. Once threat modeling has been done at this level, you can always drill down further (especially with respect to some security-relevant core functions) if necessary.

## Step 4: Identify Threats



- The next step is to identify threats that are relevant for the system under analysis
- A good way to do this is by using the Microsoft STRIDE approach
- STRIDE basically is a checklist of six attack / threat categories:
  - Spoofing
  - Tampering
  - Repudiation
  - Information disclosure
  - Denial of service
  - Elevation of privilege

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 37

## STRIDE (1)



- **Spoofing**
  - Spoofing means an attacker pretends to be somebody or something else (e.g. another user or another server)
  - Examples: Accessing a system with another user's identity, e-mail spoofing, e-banking server spoofing to collect credentials during a phishing attack...
- **Tampering**
  - Attacker modifies data or code in a malicious way (at rest or in transit)
  - Examples: Modifying a file with insufficient access protection (e.g. world-writeable), modifying data sent across unprotected network links (e.g. replies from a DNS server)...
- **Repudiation**
  - Attacker can deny having performed an action because there's no evidence to link the action to the attacker
  - Examples: Manipulating log files after a successful attack to remove traces...

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 38

## STRIDE

For more on STRIDE, see

- *Shawn Hernan, Scott Lambert, Tomasz Ostwald, and Adam Shostack. Uncover Security Design Flaws Using The STRIDE Approach.* <http://msdn.microsoft.com/en-us/magazine/cc163519.aspx>
- <http://msdn.microsoft.com/en-us/library/ms954176.aspx>

## STRIDE (2)

- Information disclosure

- Attacker gets access to information he's not authorized to read
- Examples: Reading files with plaintext passwords, reading credit card data sent across unprotected network links, reading data from the database using SQL injection...

- Denial of service

- Attacker denies access to legitimate users
- Examples: Crashing an application or a system (triggering an exception, causing a buffer overflow), flooding a system with large amounts of network traffic...

- Elevation of privilege

- Attacker gets more privileges than he is entitled to
- Examples: Exploiting a vulnerable program that runs with administrator rights, circumventing poor access control mechanism...

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 39

## Applying STRIDE to Data Flow Diagrams

- The STRIDE categories are mapped to the DFD elements according to the following table:

DFD Element Type	S	T	R	I	D	E
External Entity	X		X			
Data Flow		X		X	X	
Data Store		X	X*	X	X	
(Multiple) Process	X	X	X	X	X	X

- Privilege boundaries are not considered as they primarily serve to identify interesting areas in a DFD

- Examples:

- Spoofing threats affect external entities or processes
  - But one cannot directly spoof a data flow as it always originates from a spoofed external entity or a process
- Tampering threats affect data flows/stores (manipulating the data) and processes (manipulating them so they behave differently)

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 40

\* If the data store contains logging or audit data, repudiation is a potential threat because by manipulating the data, the attacker could cover his tracks.

### Relation between STRIDE threats and DFD elements

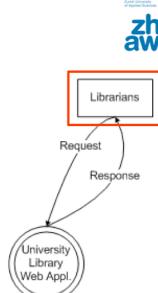
- Spoofing threats affect external entities (e.g. an attacker masquerading as a legitimate user) or processes (e.g. an attacker pretending to be an e-banking server).
- Tampering threats affect data flows (manipulating data in transit), data stores (manipulating stored data) and processes (manipulating a process such that it performs differently from being indented, e.g. by sending credentials to the attacker after a successful user login). Note that Tampering external entities is of course possible (e.g. manipulating a 3<sup>rd</sup> party system), but it's out of scope as "nothing can be done" about it. So either you have to trust that system "to be secure enough" or you shouldn't use it.
- Repudiation threats can stem from external entities (users that don't have to authenticate to perform critical activities or users that share credentials) and processes (processes (typically compromised ones) that can access other systems without authentication to perform critical activities or that no longer perform logging correctly). If audit or log data is stored in a data store, it also gets affected by repudiation threats as manipulating the stored data may allow an attacker to cover his tracks.
- Information disclosure threats affect data flows (getting access to sensitive data while in transit), data stores (accessing sensitive data in storage) and processes (non-validated input parameters that are processed).
- Denial of service threats affect data flows (legitimate data flows are prevented due to flooding by an attacker or by an insider who physically interrupts the network link), data stores (exhaust a log file by an attacker that triggers many log entries) and processes (an attacker sending a malformed packet to a server so it crashes).
- Elevation of privilege threats affect processes (e.g. a bug in an authentication routine or a missing access control check).

## Step 5: Identify Vulnerabilities

- Identifying vulnerabilities can be done as follows:
  - Go through all DFD elements...
  - ... and think about how the relevant STRIDE threat categories may be used to attack the system
- For each relevant combination, do the following:
  - Identify possible threat / attack scenarios
  - Check if countermeasures are in place that protect from the threat
  - If no sufficient countermeasures are in place, a vulnerability has been identified
- In the following, we consider some examples using the high-level DFD
  - This DFD is well suited to perform threat modeling of the overall system

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 41

## External Entity Example (SR): Librarians (1)



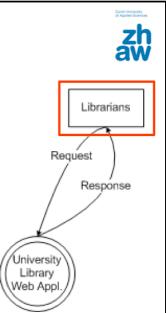
- Spoofing Threat
  - There's a spoofing threat if the attacker manages to learn the librarians' password (by guessing, stealing...)
- Countermeasures
  - Password-based authentication is used to prevent non-authorized access, which is basically good
- Rating
  - The requirements do not specify a minimal password strength, weak password will likely be selected
  - In addition, using a shared account among librarians increases the probability that passwords will be written down next to the librarians' computers
  - Overall, this is not considered sufficient to adequately protect from the threat → We identify this as a vulnerability (V1)

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 42

To increase efficiency of this step, put a stronger focus on threats that target the assets of the system and on DFD elements "close to privilege boundaries"

## External Entity Example (SR): Librarians (2)

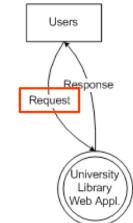
- **Repudiation Threat**
  - There's a **repudiation threat** if librarians can perform malicious actions and can deny having done this
- **Countermeasures**
  - Librarians use a special account, which allows assigning actions to librarians
- **Rating**
  - But as the librarians **share an account**, the logged actions cannot be unambiguously assigned to individual librarians
  - → We identify this as a vulnerability (**V2**)



Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 43

## Data Flow Example (TID): Request (from Users) (1)

- **Information Disclosure Threat**
  - There's an **information disclosure threat** if an attacker can read the password being transmitted
- **Countermeasures**
  - Usage of **TLS** with server-side **certificates** prevents this
- **Rating**
  - → We do not identify a vulnerability
- **Tampering Threat**
  - There's a **tampering threat** if an attacker manages to modify the data without being detected (although it's unclear what his gain would be)
- **Countermeasures**
  - Usage of **TLS** allows to detect any tampering
- **Rating**
  - → We do not identify a vulnerability



Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 44

## Assets and Vulnerabilities

Note that V1 directly targets an asset we identified earlier (credentials of users and librarians), so it's certainly an important vulnerability. Similarly, V2 also directly targets an identified asset (the logs), so it's also an important vulnerability.

## Data Flow Example (TID): Request (from Users) (2)

- Denial of Service Threat

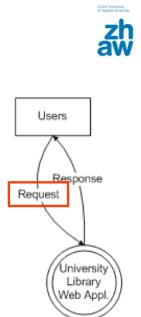
- There's a **denial of service threat** if an attacker manages to prevent data from legitimate users from reaching the web application (e.g. by exhausting a communication link or network device by flooding)

- Countermeasures

- There are no specific countermeasures to prevent this

- Rating

- As this is not a high-availability system, this is acceptable  
→ We do not identify a vulnerability



Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 45

## Data Store Example (TR\*ID): Web Resources on Disk (1)

- Tampering Threat

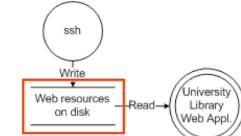
- There's a **tampering threat** if an attacker manages to modify the web resources, e.g. to deface the website or to modify the code so that credentials are sent to the attacker

- Countermeasures

- The university's server **hardening standards** that are considered adequate for typical university services
- Only trusted administrators can log in and upload web resources

- Rating

- → We do not identify a vulnerability



Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 46

### Denial of Service Threat against Data Flows

This threat means that the data flow from users to the web application is no longer possible. When considering DoS threats to data flows, then think about threats that prevent the data of other users to reach the end system (the web application), e.g. by flooding the communication link to exhaust a resource (e.g. a network device such as a router or a link). It is also possible to simply physically disrupt a data flow by cutting a cable.

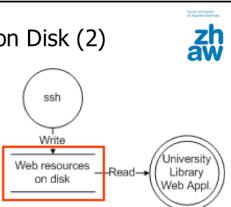
### Repudiation Threats

\*This data store does not contain logging or audit data, so repudiation threats are no issue.

## Data Store Example (TR\*ID): Web Resources on Disk (2)

- **Information Disclosure Threat**

- There's an **information disclosure threat** if an attacker manages to read the web resource (e.g. to get valuable information from them)
- This could help the attacker to identify vulnerabilities in the web application code (e.g. servlets)



- **Countermeasures / Rating**

- → For the same reasons as with tampering, we do not identify a vulnerability

- **Denial of Service Threat**

- There's a **denial of service threat** if an attacker can exhaust storage

- **Countermeasures**

- None

- **Rating**

- → We do not identify a vulnerability, as the application does not continuously write this data store (unlike, e.g., a log file)

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 47

## Process Example: University Library Web Appl. (STRIDE) (1)

- **Spoofing Threat**

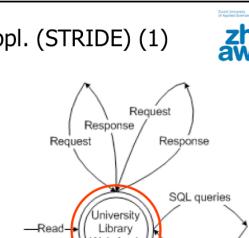
- There's a **spoofing threat** if the attacker operates his own version of the library web application to collect credentials

- **Countermeasures**

- An **EV certificate** is used to authenticate the web application, which is considered sufficient to protect from the threat

- **Rating**

- → We do not identify a vulnerability



Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 48

## Process Example: University Library Web Appl. (STRIDE) (2)

- Tampering Threat**

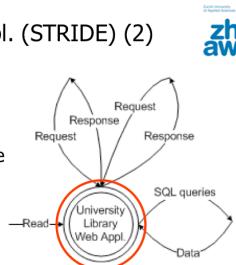
- There's a **tampering threat** if an attacker manages to modify the running web application or the Java/Tomcat software to collect credentials

- Countermeasures**

- The university's server **hardening standards** are considered adequate to protect from external attackers that attempt to get access and modify the code
- Using Java makes it unlikely to inject code by exploiting, e.g., a buffer-overflow vulnerability

- Rating**

- We do not identify a vulnerability



Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 49

## Data Store Tampering vs. Web Appl. Code Tampering

- You may have noticed that the attack goal "[collecting user credentials by modifying the web application](#)" has already occurred twice

- This is reasonable as we have identified [two ways to achieve this goal](#):

- Data store: By modifying the [web resources on the disk](#) (servlets, JSPs...)
- Process: By modifying the [running web application](#) by injecting code or by modifying the underlying [Java/Tomcat software](#)
- In both cases, one analyses whether appropriate [countermeasures](#) are in place or whether there are [vulnerabilities](#)

- In fact it is important to [identify all ways to achieve an attack goal](#) as the attacker just needs one vulnerability to be successful

- Therefore, apply an attack goal in the context of all relevant DFD elements

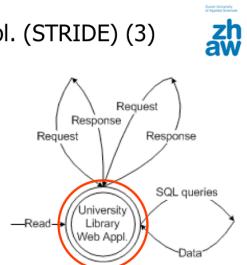
Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 50

## Web Application Code Tampering vs. Data Store Tampering

On the previous slide, we talked about tampering the web resources on disk and here we have again the resources to tamper with the process, so this is obviously an overlap. This sometimes happens with STRIDE when there are dependencies between elements, but it's not a problem. As the web resources are in the data store, it makes sense to consider them when analyzing the data store, but as they are of course needed to run the application (the process), we have a dependency between the elements and as a result, the same resources are considered here as well. Note, however, that different vulnerabilities may be uncovered when considering the different dependent elements: With the data store, you focus on possible attacks that may manipulate the data (resources) when they reside on disk. With the process, one thinks about options to manipulate the code during execution, e.g. by exploiting a buffer-overflow vulnerability to inject code.

### Process Example: University Library Web Appl. (STRIDE) (3)

- **Repudiation Threat**
  - There's a **repudiation threat** if the web application – after being compromised and modified to stop logging – performs non-legitimate transactions



#### Countermeasures

- Logging on the web application server will no longer work, but all queries are **also logged on the database** (which runs on a separate host)
- So the attacker would need to compromise two systems to completely remove any traces

#### Rating

- → We do not identify a vulnerability

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 51

### Process Example: University Library Web Appl. (STRIDE) (4)

#### Information Disclosure Threat

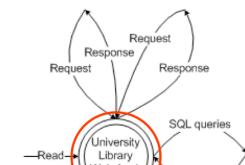
- There's an **information disclosure** threat if the web application allows accessing data via SQL injection

#### Countermeasures

- The security requirements do not make any statements to prevent this

#### Rating

- → We identify this as a vulnerability (**V3**)



Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 52

### Information Disclosure and Processes

The corresponding threats affect both the web application code itself (e.g. the program that is running) but especially also the data provided by the process. In the case of a web application, this includes all data in the database as it is the task of the web application to provide controlled access to this data, depending on the user rights, and thereby preventing attacks such as SQL injection. Similarly, when thinking about information disclosure threats against web applications, one should also consider attacks to get user credentials or session IDs via Cross-Site Scripting (XSS), as both are sensitive pieces of information maintained by the web application. Although these attacks eventually "happen" in the browser of the user interacting with the web application, the underlying threat/attack targets the web application to insert malicious JavaScripts into web pages.

Process Example: University Library Web Appl. (STRIDE) (5)

**Denial of Service Threat**

- There's a **denial of service threat** if an attacker manages to prevent the web application from offering its services

**Countermeasures**

- The university's server **hardening standards** are considered adequate to prevent from some DoS threats (e.g. crashing the system/service by sending a malformed packet)

**Rating**

- Because of the hardening standards and as this is not a high-availability system, this is acceptable → We do not identify a vulnerability

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 53

#### Assets and Vulnerabilities

Just like V1 and V2, the newly identified vulnerabilities V3 and V4 are also important vulnerabilities, as both directly target assets we identified earlier. V3 may be exploited to access personal user data and maybe also credentials (assuming they are stored in the database in plaintext) and V4 may also be used to access personal user data

#### Reputation Threat – Entity, process or data Store?

The Reputation threat is a good example that shows that it is sometimes not obvious to which element a specific threat "belongs". According to STRIDE, reputation threats can be associated with entities, processes and data stores. Assuming we have users that use an application which itself generates logs, three different element types are associated in the logging process, so how should one divide different threats among the three elements? Here are some examples that may help in making a reasonable separation of the element types:

- Data store:** To determine reputation threats here, you should think about scenarios that directly affect the data store (i.e. the logged data), without considering the application or the system that produces the logs. Typical threats are a system administrator that often has the necessary (too many?) rights to remove his traces by removing some local log file entries.
- Process:** To determine reputation threats here, think about flaws in the process (e.g. the application) that may result in manipulating the log files. For instance, there may be an input validation problem that allows the user accessing the application to write arbitrary system files, which also allows him to modify log files.
- Entity (user):** Finally, to determine reputation threats, think about fundamental logging problems, e.g. no logging is done at all, no timestamps are associated with the logged data, multiple user sharing accounts etc. One could also argue that this "belongs" to the process – as it is a flaw in the process that no logging is done at all, but in practice, such fundamental logging flaws are often associated with the entity

In general, when thinking about STRIDE and analyzing an element, always ask yourself what could go wrong HERE (with this element), which usually results in making the right decision.

#### Denial of Service Threat against Processes

Here, think about threats that take the process "offline", e.g. by flooding the web application so it can no longer serve legitimate user or by sending the server a malformed packet so it crashes.

Process Example: University Library Web Appl. (STRIDE) (6)

**Elevation of Privilege Threat**

- There are **elevation of privilege** threats if an attacker manages to circumvent access control checks to get access as a (e.g.) librarian

**Countermeasures**

- The security requirements do not state anything about whether **access control** must be performed

**Rating**

- We identify this as a vulnerability (V4)

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 54

#### Assets and Vulnerabilities

Just like V1 and V2, the newly identified vulnerabilities V3 and V4 are also important vulnerabilities, as both directly target assets we identified earlier. V3 may be exploited to access personal user data and maybe also credentials (assuming they are stored in the database in plaintext) and V4 may also be used to access personal user data

#### Reputation Threat – Entity, process or data Store?

The Reputation threat is a good example that shows that it is sometimes not obvious to which element a specific threat "belongs". According to STRIDE, reputation threats can be associated with entities, processes and data stores. Assuming we have users that use an application which itself generates logs, three different element types are associated in the logging process, so how should one divide different threats among the three elements? Here are some examples that may help in making a reasonable separation of the element types:

- Data store:** To determine reputation threats here, you should think about scenarios that directly affect the data store (i.e. the logged data), without considering the application or the system that produces the logs. Typical threats are a system administrator that often has the necessary (too many?) rights to remove his traces by removing some local log file entries.
- Process:** To determine reputation threats here, think about flaws in the process (e.g. the application) that may result in manipulating the log files. For instance, there may be an input validation problem that allows the user accessing the application to write arbitrary system files, which also allows him to modify log files.
- Entity (user):** Finally, to determine reputation threats, think about fundamental logging problems, e.g. no logging is done at all, no timestamps are associated with the logged data, multiple user sharing accounts etc. One could also argue that this "belongs" to the process – as it is a flaw in the process that no logging is done at all, but in practice, such fundamental logging flaws are often associated with the entity

In general, when thinking about STRIDE and analyzing an element, always ask yourself what could go wrong HERE (with this element), which usually results in making the right decision.

#### Denial of Service Threat against Processes

Here, think about threats that take the process "offline", e.g. by flooding the web application so it can no longer serve legitimate user or by sending the server a malformed packet so it crashes.

## Step 6: Mitigating the Threats

- The last step of threat modeling is to [define new security requirements](#) that eliminate or reduce the identified vulnerabilities
- A nice property about STRIDE is that each threat directly [maps to a security properties](#)

STRIDE Threat	Security Property
Spoofing	Authentication
Tampering	Integrity
Repudiation	Non-repudiation
Information disclosure	Confidentiality
Denial of service	Availability
Elevation of privilege	Authorization

- Mitigating the threats therefore means [proposing appropriate security requirements](#) that can provide the desired security property

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 55

## Mitigating the Identified Threats (1)

- Vulnerability V1: [spoofing librarians](#) because
  - A shared account is used
  - Password strength is not enforced by the system
- According to the table of security properties, [this affects authentication](#) and corresponding security requirements should be defined
- In this case, this could result in the following:
  - Use [individual accounts for librarians](#) with dedicated passwords
  - Enforce a [minimal password quality](#) (length, character mix, compare with dictionary...)
  - Optional: integrate into [university-wide identity management system](#) (if such a system is available)
- The same proposal [also mitigates the reputation threat \(V2\)](#) by librarians

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 56

## Mitigating the Identified Threats (2)



- Vulnerability V3: **Information disclosure of the web application** because the security requirements do not make any statements with respect to preventing SQL injection attacks
- According to the table of security properties, **this affects confidentiality** and corresponding security requirements should be defined
  - Confidentiality is not just about encrypting data, it's about providing information only to legitimate users and systems
- In this case, proposed security requirements could include:
  - Accessing the database is only allowed via **prepared statements**
  - All data received from users should be considered non-trusted and **should first be validated** before processed further
  - Use a technical database user with **minimal privileges**

---

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 57

## Mitigating the Identified Threats (3)



- Vulnerability V4: **Elevation of privilege when using the web application** because the security requirements do not make any statements with respect to access control
- According to the table of security properties, **this concerns authorization** and corresponding security requirements should be defined
- In this case, defined security requirements could include:
  - The web application must **employ an authorization mechanism**
  - Authorization must be **checked on every single request** (complete mediation)

---

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 58

## Security Requirements Engineering / Threat Modeling – Remarks (1)



- As said before, this is not a scientific method but rather a **creative process**
  - As a result, this is **best be done in a team**, where different views and knowledge areas of the participants can be considered
- As long as a system is developed further, security requirements engineering / threat modeling **never really stops**
  - It's an **iterative process** – and therefore fits well with iterative software development in general
  - When the system is **extended**, new security requirements may have to be defined
- Even without any system extension, it may be necessary to adapt security requirements, e.g. when **new types of attack** surface
  - In these cases, a new type of attack can be the trigger for a new iteration

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 59

## Security Requirements Engineering / Threat Modeling – Remarks (2)



- What's the **appropriate level of detail** to use for the DFD?
  - A **high-level DFD** as we used it here is usually well suited to analyze a system with "typical security requirements"
  - In some cases, a **more detailed view** is appropriate
    - To analyze **very security-critical processes**, it may be reasonable to identify all involved components in detail
    - E.g. the entire payment process in an e-banking system
- "Tricks" to simplify a complex DFD
  - **Combine elements** with the same privilege level
    - Consequently, we did not split the web application multiple process into smaller processes in the University library application example as all these processes have the same privilege level
  - Two **data flows between the same two elements** that use the same protocols can also often be combined into one bi-directional data flow
    - In our example this could have been done with the request / response data flows between users and the web application

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 60

## Security Requirements Engineering / Threat Modeling – Remarks (3)



- Often, threats are not independent: a **successful attack** means other attacks can be carried out
  - If the attacker manages to **spoof the administrator**, he likely gets access to the system, where he can **view and tamper data**, delete data and stop processes (**DoS**), remove log data to cover his traces (**repudiation**) etc.
  - Don't think too far**, simply consider each threat against each element on its own (otherwise, it gets quickly very complicated)
  - As positive side effect, this "forces" you to think about **defense in depth**
    - Because for each threat and each component (DFD element), you have to ask yourself what you can do HERE to protect from this threat
- Don't make security assumptions!**
  - You analyze what you see and not what "you would like to see"
  - If it's not in the documentation and if no one has told you about a security requirements or control in an interview, **assume it's not there**
  - It's good to be on the conservative side, if someone tells you afterwards that "this requirement has been defined or this control is in place", no great harm was done

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 61



## Security Requirements Engineering / Threat Modeling – Remarks (4)

- Don't forget insiders**
  - Often, security analyses focus on external attackers, but in reality, **internal attacks are more likely**
    - This includes **malicious insiders** (disgruntled employees)...
    - ...but also "**accidents by insiders**" (e.g. an administrator who connects an infected laptop to the network in the server room)
- Document** your findings and refine / extend them during each iteration
  - No really good tools available, **a spreadsheet is a good start**
  - Microsoft has made available the **SDL Threat Modeling Tool**, which allows drawing DFDs and applying STRIDE to the elements
- Even the best security requirements engineering / threat modeling process can **never prove a system is secure**
  - But you can significantly increase the likelihood that the system does not contain major security design flaws

Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 62

### Defense in Depth

Thinking about threats against each DFD element and defining appropriate security requirements will likely result in multiple security requirements that ultimately protect from the same attack scenario. But that's OK from a defense in depth approach because defense in depth is usually a desired property. When reviewing the complete list of security requirements, it is always possible to reduce the list of security requirements if you think that you have "too much security", e.g. by reducing the number of security requirements that help to protect from a specific attack from 3 to 2.

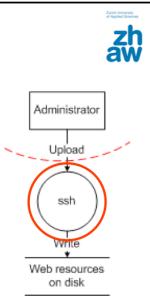
### SDL Threat Modeling Tools

More information – and the tool itself – is available at <http://www.microsoft.com/security/sdl/adopt/threatmodeling.aspx>

The tool requires Visio 2007 and – as a result – only runs under Windows.

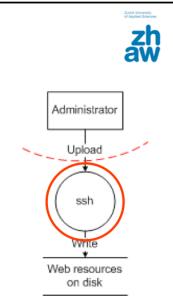
### Exercise (1)

Consider the ssh process, identify one threat for each STRIDE category and determine whether appropriate countermeasures are in place. If a vulnerability is identified, propose new appropriate security requirements.



Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 63

### Exercise (2)



Marc Rennhard, 11.02.2014, SSI\_SeqReq\_ThreatMod1.pptx 64