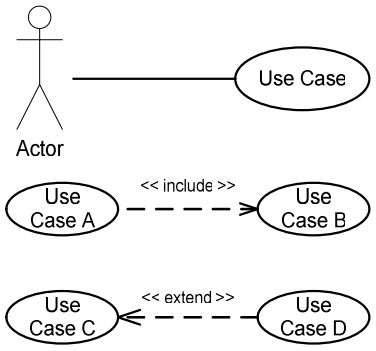
From Use Cases to Abuse Cases

- In general, use cases are a well-suited as a basis for security requirements engineering / threat modeling
  - Because use cases help to understand the purpose of a system and are therefore also suited to think about threats
- If UML use case diagrams are used, they can directly be used to think about threats threats → abuse cases
  - The threats can even be directly drawn in the diagram
- Reminder of UML use case notation
  - An actor interacts with a use cases, which can by any function of the system
  - Use case A includes use case B, meaning that whenever A is executed, B is executed as part of it
  - Use case D extends use case C, meaning that when executing C, D may be executed as part of it (depending on conditions)



UML

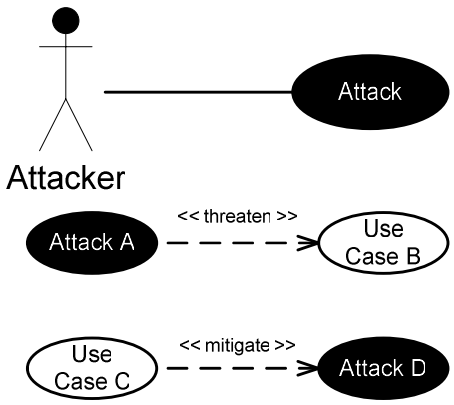
A very good basic introduction to UML can be found here (in German): *http://www.highscore.de/uml*

Abuse Cases and UML Diagrams (2)

Abuse case notation introduces further diagram types:

- Not an official UML standard, just a proposal how it can be extended

- We have a new type of actor, the attacker (drawn as inverted actor)
- The attacker executes attacks (drawn as inverted use case)
- An attack A threatens a use case B
- The use case C can mitigate attack D



Abuse Cases and UML Diagrams

This has been discussed in several papers, including the following:

Guttorm Sindre and Andreas Opdahl, Eliciting Security Requirements by Misuse Cases. In Proceedings of 37th International Conference on Technology of Object-Oriented Languages and Systems, 2000, TOOLS-Pacific 2000.

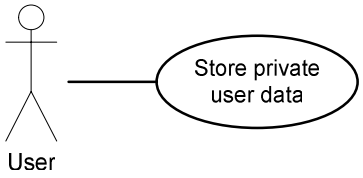
Joshua Pauli and Dianxiang Xu, Threat-Driven Architectural Design of Secure Information Systems. Avaliable at http://cs.ndsu.edu/~dxu/publications/pauli-xu-ICEIS05.pdf

Lilian Rostad, An extended misuse case notation: Including vulnerabilities and the insider threat. Available at http://www.di.unipi.it/REFSQ06/Papers/01%20Rostad.pdf

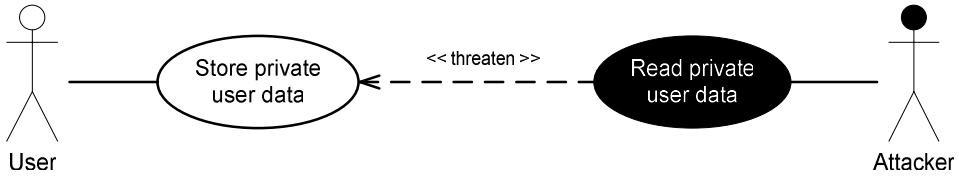
The notation we are using here is closely related to the one in the paper by Lilian Rostad.

Abuse Cases – Simple Example (1)

- Consider a system that stores private user data, e.g. a health care application
- To keep it very simply, we consider one use case: store private user data

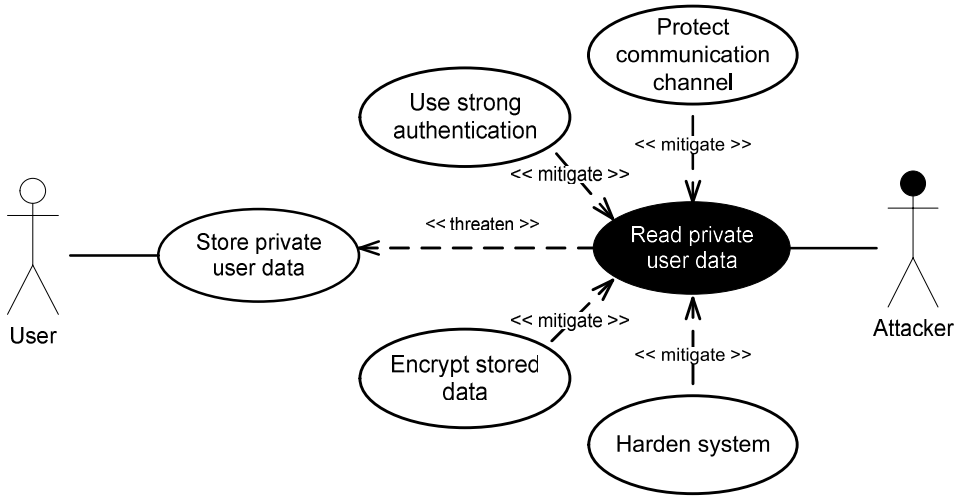


- If this use case is used to think about attacks, then we can identify an attacker that wants to read the private user data
  - So we have an attack (abuse case) «read private user data» that threatens the stored private user data



Abuse Cases – A Simple Example (2)

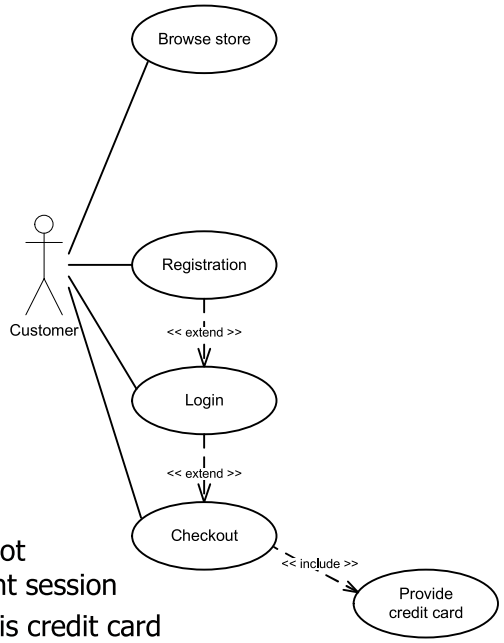
- Based on this scenario, we can now think about security requirements (drawn as use cases) that can mitigate the attack



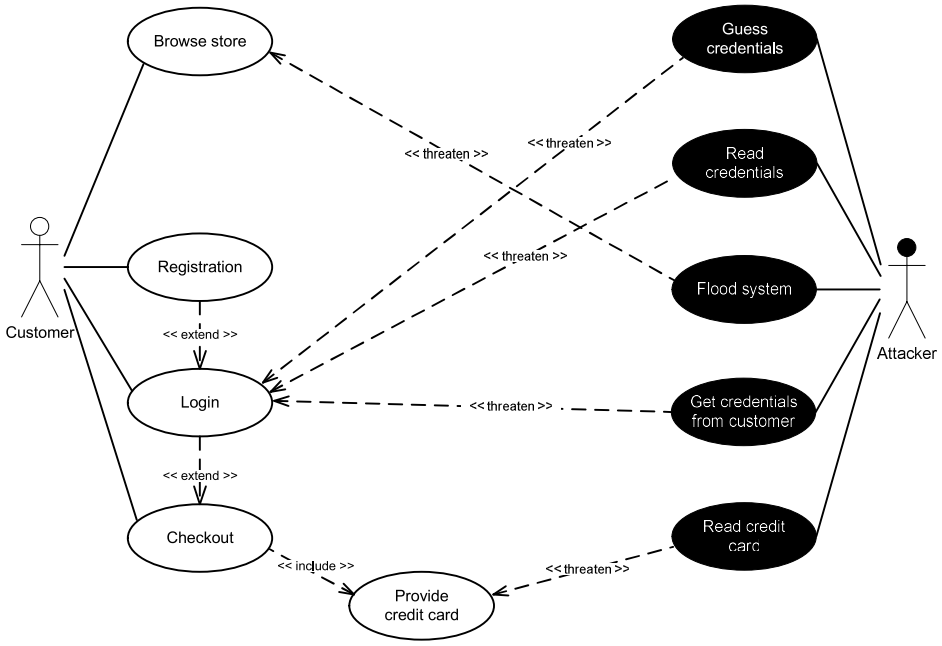
A more Elaborate Example – Basic Use Cases

Consider an e-shop with some basic functions (use cases)

- Browsing the store
- Customer registration
- Login
  - May require registration if customer has not registered before
- Checkout
  - May require login if customer has not performed a login during the current session
  - Requires the customer to provide his credit card



A more Elaborate Example – Abuse Cases

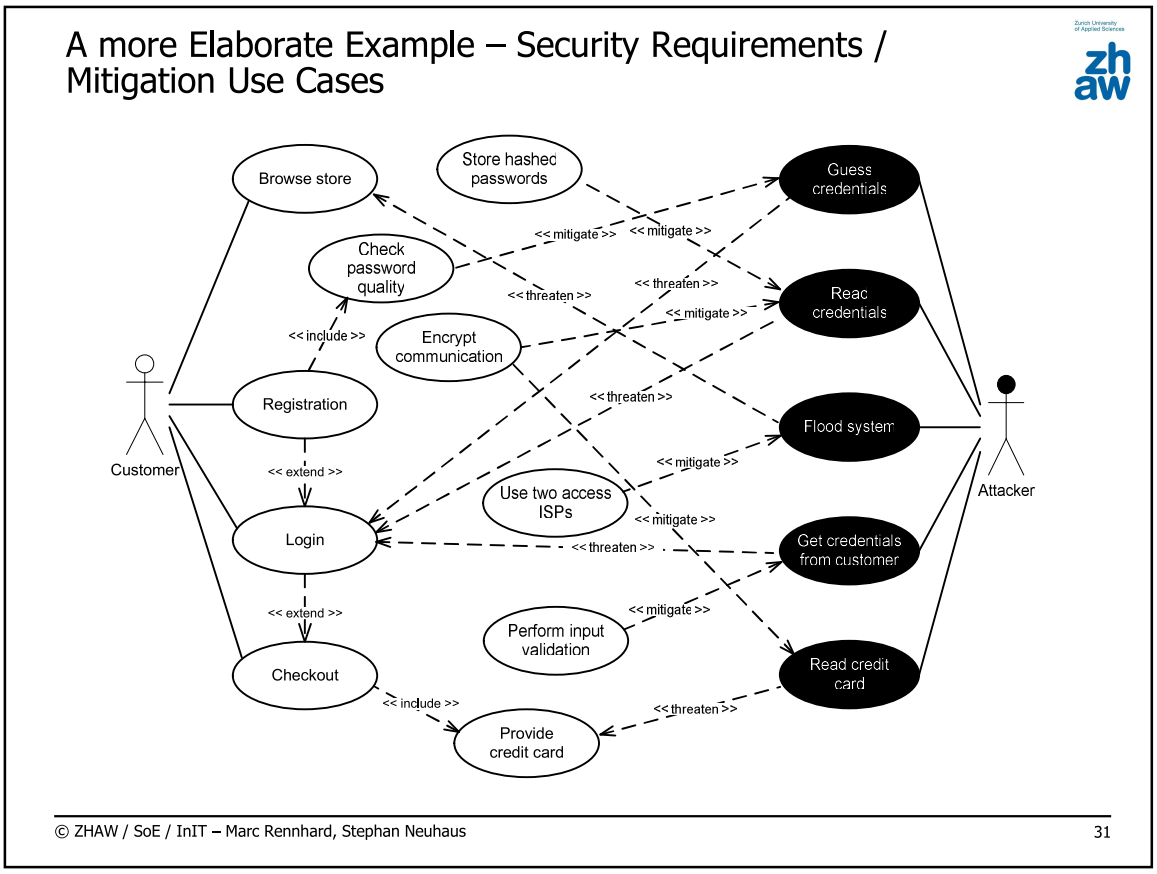


Abuse Cases

- Read credentials: E.g. via SQL injection, reading unencrypted network data, or by stealing the server disk
- Get credentials from customer: E.g. via XSS or phishing

Completeness

Note that not all associations have been drawn into the diagram above for clarity. For instance, “flood system” does not only target the use case “browse store”, but will most likely affect the availability of the entire store and should therefore affect virtually all use cases.



Mitigation Use Cases

- Perform input validation: To mitigate “get credentials from customer” via XSS

Of course, many more mitigation use cases are possible and also reasonable, e.g. data sanitation against XSS, prepared statements against SQLi, 2-factor authentication against phishing etc.

What about Malicious Insiders?

Normal use case actors (e.g. customer, administrator) can also be attackers. In this case, they exist on the actor side and on the attacker side (e.g. malicious administrator).

A more Elaborate Example – Summary

- A UML use case diagram is used as the basis
  - This helps to think about attacks against use cases, which are included in the diagram as abuse cases
  - Which then helps to identify appropriate security requirements (mitigation use cases), which are also included in the diagram
- Not all mitigation use case are noticeable as functions by the users
  - «Password quality check» and «encrypt communication» will be visible
  - «Store hashed passwords», «use two access ISPs», and «perform input validation» typically won't be noticed
- Often, the non-visible mitigation use cases are also less obvious and are more likely to be forgotten during software development
  - Even developers that are not security-aware will likely realize that «Password quality check» and «encrypt communication» should be incorporated into the application
  - But the others are more likely to be forgotten, especially when the developer is not aware of the corresponding threats

Abuse Cases and UML Use Cases Diagrams – Discussion

- Supplementing use case diagrams with abuses cases has several benefits
  - Visualizing use cases, mitigation use cases, and attacks helps to identify missing security requirements
  - It can effectively show what security requirements mitigate what attacks and also what requirements mitigate many attacks
  - The diagram is also understandable by non-experts, so it can be well used when talking e.g. to the management about security issues
    - E.g. because you want to get more resources for security work
  - And it even helps documenting the security requirements

- However, there are also drawbacks
  - Suffers from the same problem as pure UML use case diagrams: they may grow very large and complex
  - It is therefore usually required to decompose the entire system and apply abuse cases to a subset of all use cases

A diagram with use cases and abuse cases is most likely much better understandable by the management than a rather technical Data Flow Diagram.