DC-Anonymity & Security Engineering

Software Security

Steffen Helke
Chair of Software Engineering

14th November 2018

Brandenburgische Technische Universität Cottbus - Senftenberg

Objectives of today's lecture

- ➜ *Dining Cryptographers* protocol and DC networks
- ➜ Understanding processes and methods for *Security Engineering* and basic *modeling notations*
- ➜ Being able to identify *protection goals* for each actor and applying *Misuse Cases* for a security analyses
- ➜ Being able to describe *Threats* using Attack Trees

Dining Cryptographers

Source: First published by David L. Chaum in *The dining cryptographers problem: Unconditional sender and recipient untraceability*, Journal of Cryptology 1.1, S. 65-75, 1988.

Protocol for Anoymity: Dining Cryptographers

Problem Description

- 3 Cryptographers sitting at a dining table
- Waiter says: It's already paid,
  1. either *one* of the cryptographers or
  2. the NSA (National Security Agency)

  has paid.

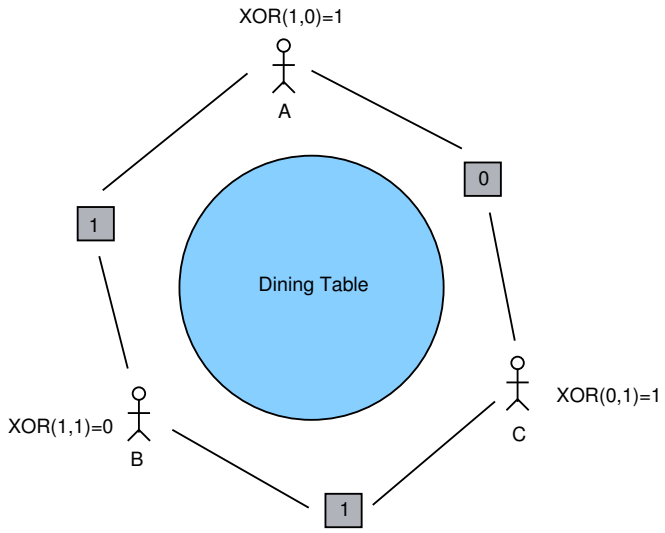
Question

➜ How can we determine, whether the NSA have paid or not without uncovering the anonymity of the cryptographers?

Note

- Protocol also works for more than 3 cryptographers

How works the Protocol for 3 Cryptographers?



XOR(1,0,0)=1 : A cryptograph has paid for the dinner
XOR(1,0,1)=0 : The NSA has paid for the dinner

How is the DC Protocol generalized?

➜ The dinner story is just a textbook example and only supports the anonymous sending of 1-bit messages for 3 persons

Idea

- Two neighbours no longer flip just a single coin, but instead *use a random n-bit key*
- For sending the message $n = b_1 \dots b_n$ anonymously, a sender combines n with the keys known to him
- All other participants only combine the keys known to them
- Furthermore, the protocol can be extended from 3 participants to any number of participants (cf. next slide)

Dining Cryptographers: Generalization

Assumption

- There are k cryptographers at the table
- The i th cryptograph flips coins with his two neighbours and knows the results, called LC_i and RC_i

Procedure

1. Every i th cryptograph calculates $EC_i = LC_i \oplus RC_i$ and announces AC_i, where
   - $AC_i = \neg\, EC_i$, if the cryptograph has paid and
   - $AC_i = EC_i$ otherwise
2. The truth is determined with $E = AC_1 \oplus \dots \oplus AC_k$
   - if $E = 0$, then the NSA has paid
   - otherwise one of the cryptographers

Note: Operator $\oplus$ again represents the logical operation XOR (exclusive disjunction)

Dining Cryptographers: Limits

Revealing the Truth

- The result $E = 0$ can also occur if ...
  an *even number* of cryptographers have paid together
- Result $E = 1$ arises if ...
  an *odd number* of cryptographers have paid together

Attack Possibility

- Two cryptographers can uncover the identity of another person sitting directly between them by sharing their results

Countermeasure

- Each participant must flip a coin with each other and combine the results using XOR

➜ Consequence: Two neighbours alone cannot successfully attack another person

Practical Applications for DC Networks

Conclusions

- Method guarantees *information theoretical* security for the anonymity of the sender
- DC-based anonymity has also been evaluated in so-called DC networks with a focus on scalability (the ability of a computing process to be used or produced in a range of capabilities.)
- Main problem: Performance is limited by overhead in communication

➜ Consequently, there are only a few practical examples of DC networks compared to probabilistic anonymity using mixes or onion routing

Example Application: Coordination of Dates

- Benjamin Kellermann: *Dudle: Mehrseitig sichere Web 2.0 Terminabstimmung*[1], Dissertation, TU-Dresden, 2011.

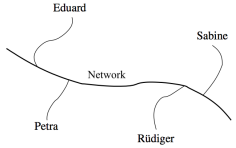
1) https://dudle.inf.tu-dresden.de/

Example for Sending a Message within a DC Network

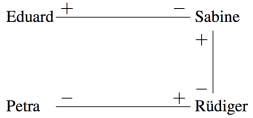
Example for DC Net: How to send a Message?

➜ Assuming Sabine wants to send the word *Secret* anonymously

1. DC net consists of 4 participants with the following *topology*



2. *Key graph* specifies which participants have agreed on a key



➜ We have 3 keys, called K_{ES}, K_{SR}, K_{PR}, which are generated randomly

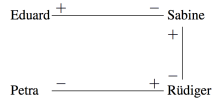
3. Participants have agreed on the following *coding alphabet*

_	A	B	C	D	E	F	G	H	I	J	K	L	M	N
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

O	P	Q	R	S	T	U	V	W	X	Y	Z	.	!	?
15	16	17	18	19	20	21	22	23	24	25	26	27	28	29

Example for DC Net: How to send a Message?

K_{ES}	$\widehat{=}$	G	_	S	H	_	L	$\widehat{=}$	7	0	19	8	0	12
K_{SR}	$\widehat{=}$	H	.	A	?	C	.	$\widehat{=}$	8	27	1	29	3	27
K_{PR}	$\widehat{=}$	_	M	W	X	Q	S	$\widehat{=}$	0	13	23	24	17	19
M	$\widehat{=}$	S	E	C	R	E	T	$\widehat{=}$	19	5	3	18	5	20



1. Sabine calculates her intermediate result $R_S \mathrel{\widehat{=}} M \oplus K_{SR} \ominus K_{ES}$ and distributes R_S to all other participants

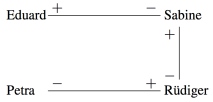
	19	5	3	18	5	20
$\oplus$	8	27	1	29	3	27
	27	2	4	17	8	17
$\ominus$	7	0	19	8	0	12
	20	2	15	9	8	5

2. Eduard calculates his intermediate result $R_E \mathrel{\widehat{=}} NE \oplus K_{ES}$ using the neutral element NE and distributes R_E to all other participants

	0	0	0	0	0	0
$\oplus$	7	0	19	8	0	12
	7	0	19	8	0	12

Example for DC Net: How to send a Message?

K_{ES}	$\widehat{=}$	G	_	S	H	_	L	$\widehat{=}$	7	0	19	8	0	12
K_{SR}	$\widehat{=}$	H	.	A	?	C	.	$\widehat{=}$	8	27	1	29	3	27
K_{PR}	$\widehat{=}$	_	M	W	X	Q	S	$\widehat{=}$	0	13	23	24	17	19
M	$\widehat{=}$	S	E	C	R	E	T	$\widehat{=}$	19	5	3	18	5	20



3 Rüdiger calculates his intermediate result $R_R \mathrel{\widehat{=}} NE \oplus K_{PR} \ominus K_{SR}$ and distributes R_R to all other participants

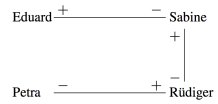
	0	0	0	0	0	0
$\oplus$	0	13	23	24	17	19
	0	13	23	24	17	19
$\ominus$	8	27	1	29	3	27
	22	16	22	25	14	22

4 Petra calculates her intermediate result $R_P \mathrel{\widehat{=}} NE \ominus K_{PR}$ using the neutral element NE and distributes R_P to all other participants

	0	0	0	0	0	0
$\ominus$	0	13	23	24	17	19
	0	17	7	6	13	11

Example for DC Net: How to send a Message?

K_{ES}	$\widehat{=}$	G	_	S	H	_	L	$\widehat{=}$	7	0	19	8	0	12
K_{SR}	$\widehat{=}$	H	.	A	?	C	.	$\widehat{=}$	8	27	1	29	3	27
K_{PR}	$\widehat{=}$	_	M	W	X	Q	S	$\widehat{=}$	0	13	23	24	17	19
M	$\widehat{=}$	S	E	C	R	E	T	$\widehat{=}$	19	5	3	18	5	20



5 Finally, all participants add up the intermediate results and receive the anonymous message

	0	17	7	6	13	11
	22	16	22	25	14	22
	7	0	19	8	0	12
$\oplus$	20	2	15	9	8	5
	19	5	3	18	5	20

➜ Note, these 5 steps have just described the procedure of one round

➜ If you want to send further messages anonymously, for each new round new keys must be negotiated

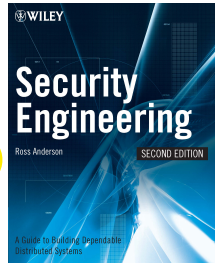
➜ It is also important to ensure that only one participant can send a message in each round, which can be implemented e.g. by collision avoidance algorithms

Security Engineering

What is Security Engineering?

General Remarks

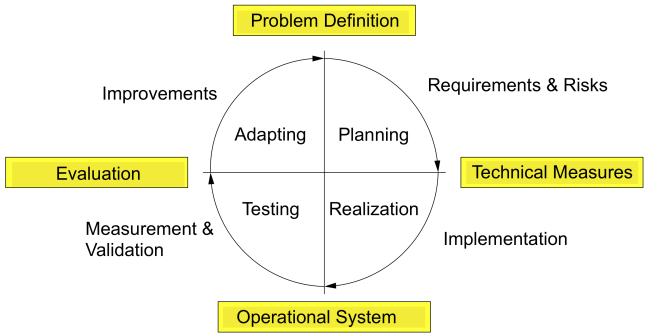
- Analysis of security requirements and threats of a software system
- Defining technical measures to reduce risk
- Systematic procedure for the selection and use of established methods



Questions

- Specifying protection goals: *What should be protected?*
- Identifying threats: *Who could attack?*
- Check the context: *What organizational measures are required?*

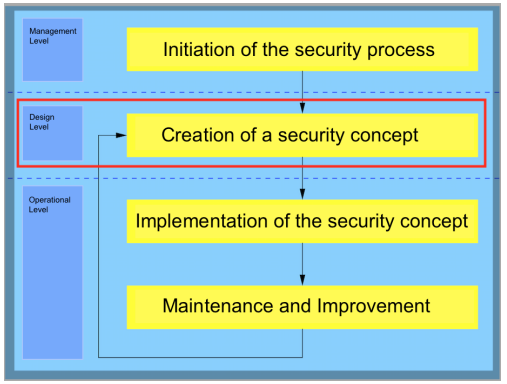
Activities of a Security Engineering Process



- ➜ Continuous monitoring of the protection objectives is required
- ➜ The achievable level of security is strongly influenced by developments in technology

Security Engineering Process of the BSI

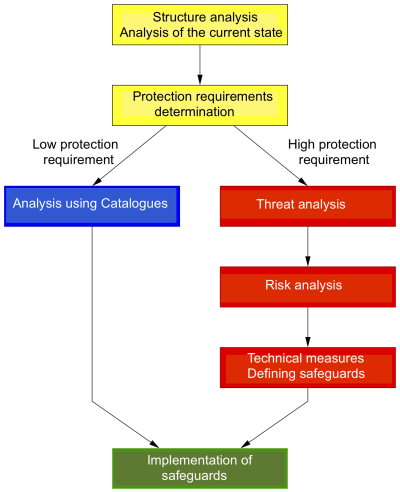
- ➜ IT-Grundschutz Catalogues of the BSI[1] proposes a specific security process
- ➜ Methodology is mainly designed to protect existing IT infrastructures



1) The Federal Office for Information Security (German: Bundesamt für Sicherheit in der Informationstechnik, BSI)

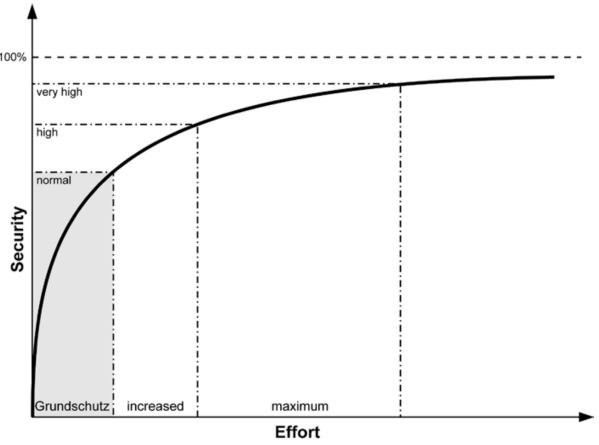
Refinement of the Design Level of the BSI

- ➜ Creation of an IT security concept is regulated by law
- ➜ e.g. Bundesdatenschutzgesetz or Telemediengesetz



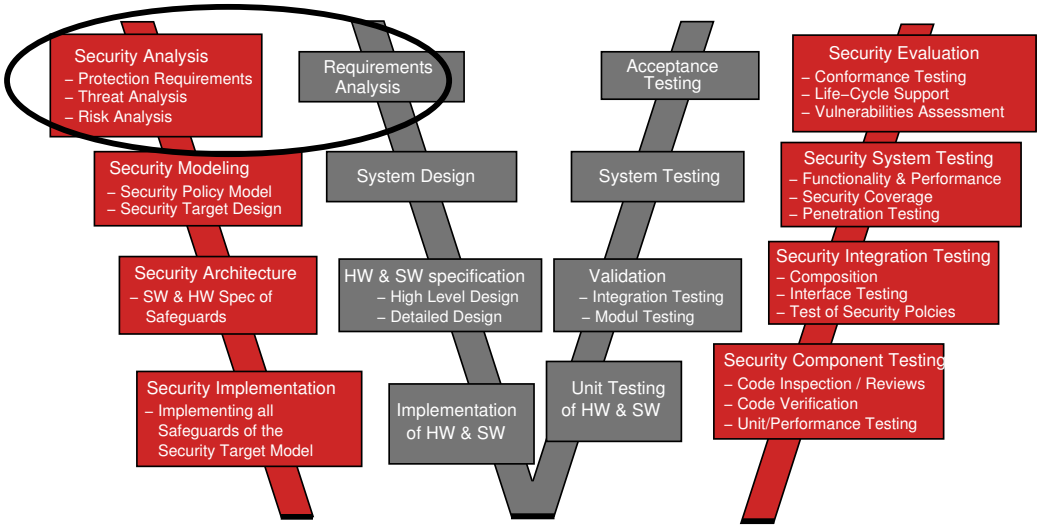
Cost/Benefit Relation for Information Security

- ➜ Note, it is not possible to achieve perfect information security
- ➜ if only a normal protection is required, *IT-Grundschutz* is a nice and rather inexpensive option



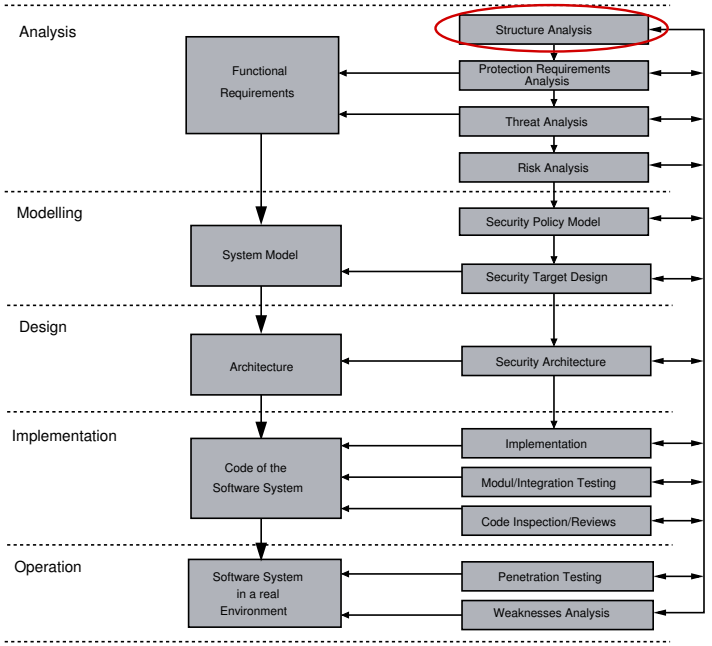
Source: *BSI-Standard 100-2, IT-Grundschutz Methodology*, BSI, 2008.

Security Engineering for the V-Model



Quelle: In Anlehnung an H. Hintze, R.God: *Marko Wolf, Embedded Security Engineering, ESCRYPT Munich*, 2012.

Activities of a Security Engineering Process



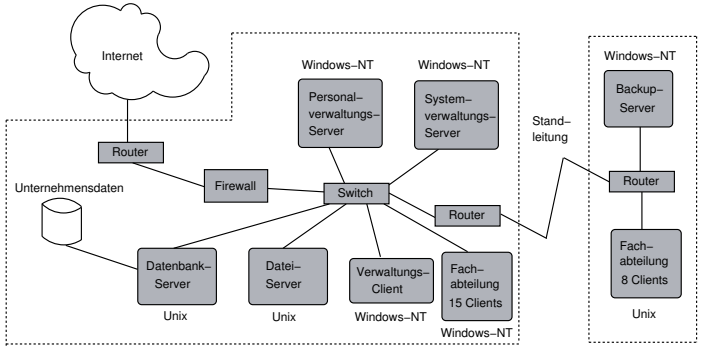
1. Structure Analysis

Objectives

- Operating environment and intended use
- Functional characteristics
- Security requirements

Artifacts

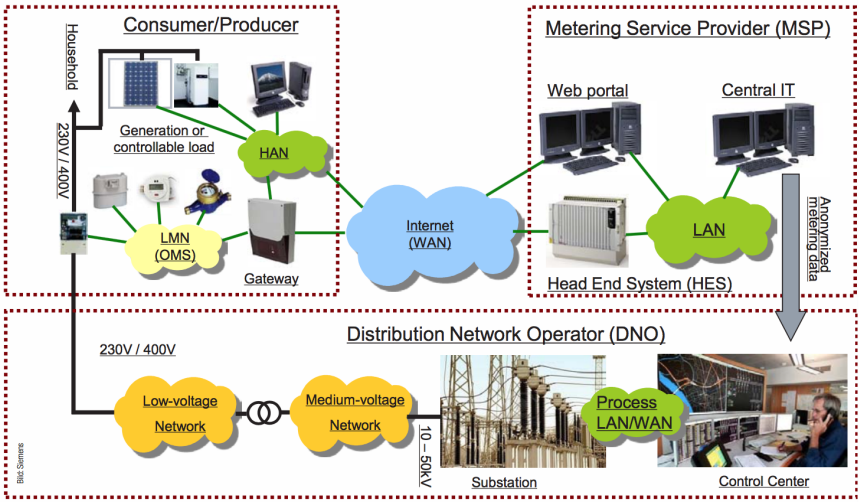
1. Network topology model
2. Table with attributes for each component
3. Initial requirement specification



Example: Smart Metering System

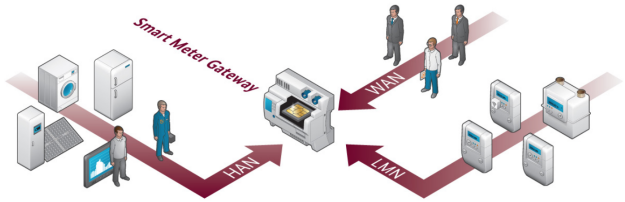
Source: EVB Energie AG, de.wikipedia, CC BY-SA 3.0

Example: Network Topology Smart Metering System



Source: D. von Oheimb: *Stellenweise Sicher – Kritische Betrachtung der IT-Security-Anforderungen fürs Smart Metering*, elektronikjournal 03/2013

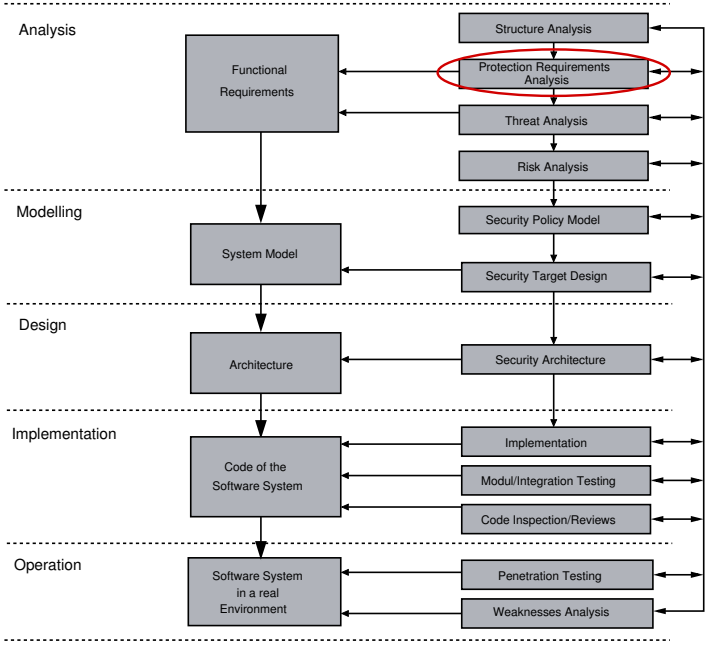
Example: Network Topology Smart Meter Gateway



Source: BSI: *Das Smart-Meter-Gateway – Sicherheit für intelligente Netze*, 2015.

- **HAN**: *Home Area Network*
  ➜ for final consumer, connection for controllable devices
- **LMN**: *Local Metrological Network*
  ➜ for obtaining network status data, e.g. consumer data
- **WAN**: *Wide Area Network*
  ➜ for communication with all external market participants

Activities of a Security Engineering Process



2. Protection Requirements Analysis

➜ *What could be attacked in principle and how bad would it be?*
But don't ask if it is actually possible!

- Determining *damage scenarios* and *protection goals*
- Evaluation and identifying the required protection level

Sample Scenarios
1 The right to self-determination of information is violated
2 It is an attack that breaks laws, regulations or contracts
3 Threats to the physical or psychic integrity of persons

Protection Level	Damage Impact ...
normal	... is limited and calculable
high	... may be considerable
very high	... may be of catastrophic proportions

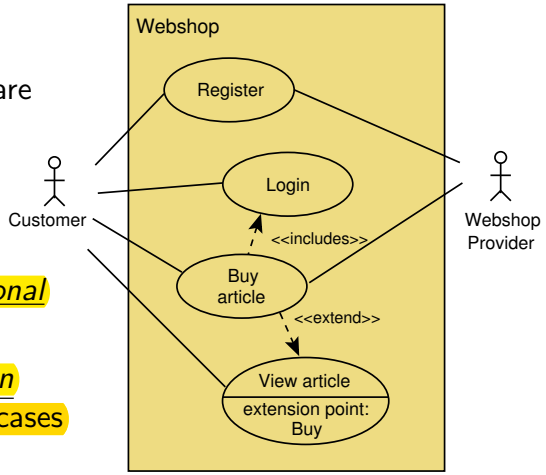
Modeling of Actors and Use Cases

Use Cases

- Modeling of *functional requirements* of a software system

Extensions for protection requirement analyses?

- Notation for *non-functional requirements*
- Assignment of *protection goals*, later also misuse cases and countermeasures



Protection Goals for Requirements Analysis

➜ **Confidentiality**
unauthorized gain of information

➜ **Integrity**
unauthorized modification of information

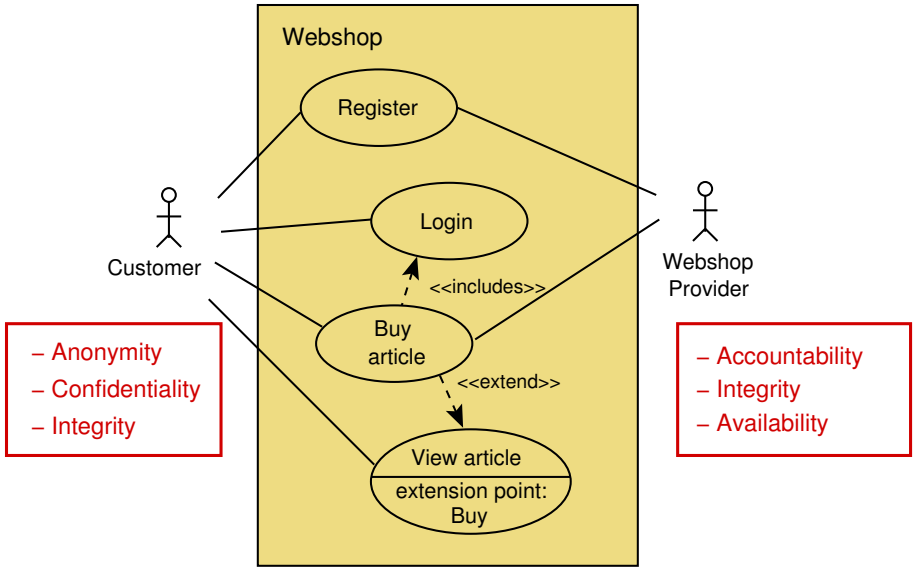
➜ **Availability**
unauthorized impairment of functionality

➜ **Anonymity**
covering identity when using resources

➜ **Accountability**
sender and recipient of messages can be responsible to third parties in the case of damage

➜ ...

Which protection goals can be identified in this example?



Correlations between Protection Goals

Tasks for a Protection Requirement Analysis

➜ Check the feasibility of the collected protection goals

➜ Suggest possible compromises in case of conflicts (e.g. pseudonymity)

Basic Theory

➜ *Multi Lateral Security*

➜ Correlations and monotony behaviour of protection goals

Precise Definitions of Protection Goals (1)

Confidentiality ensures that nobody apart from the communicants can discover the content of the communication

Hiding ensures the confidentiality of the transfer of confidential user data. This means that nobody apart from the communicants can discover the existence of confidential communication

Anonymity ensures that a user can use a resource or service without disclosing his/her identity, not even the communicants can discover the identity of each other

Unobservability ensures that a user can use a resource or service without others being able to observe that the resource or service is being used. Parties not involved in the communication can observe neither the sending nor the receiving of messages

Integrity ensures that modifications of communicated content (including the sender's name, if one is provided) are detected by the recipient(s)

Precise Definitions of Protection Goals (2)

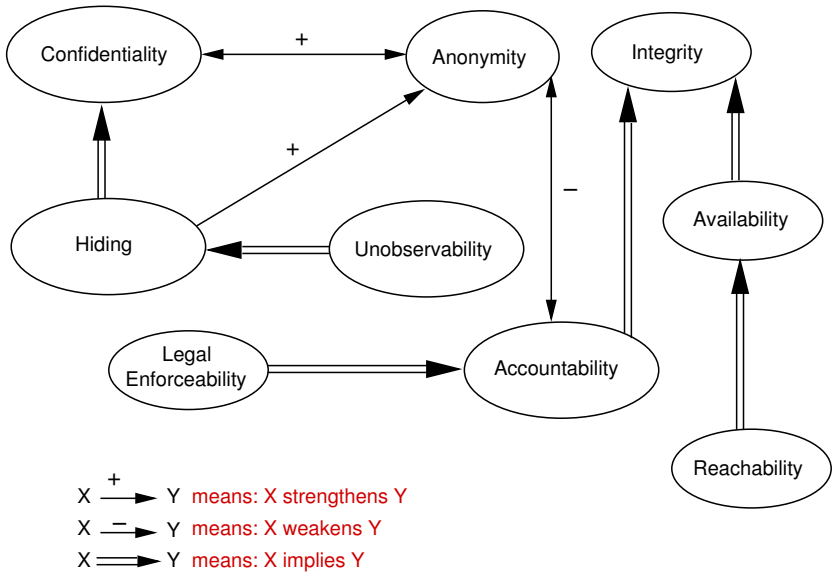
Accountability ensures that sender and recipients of information cannot successfully deny having sent or received the information. This means that communication takes place in a provable way

Availability ensures that communicated messages are available when the user wants to use them

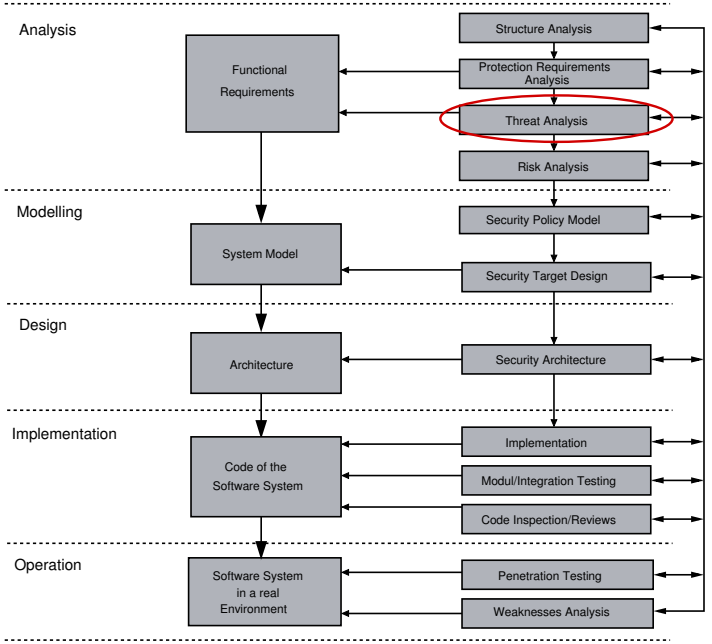
Reachability ensures that a peer entity (user, machine, etc.) either can or cannot be contacted depending on user interests.

Legal Enforceability ensures that a user can be held liable to fulfill his/her legal responsibilities within a reasonable period of time.

Correlations between Protection Goals



Activities of a Security Engineering Process



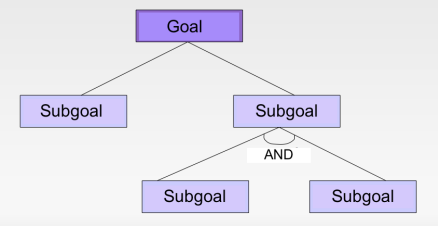
3. Threat Analysis

Procedure

- Systematic identification of the *causes of threats*
- *Expert knowledge* of current security issues and vulnerabilities must be taken into account

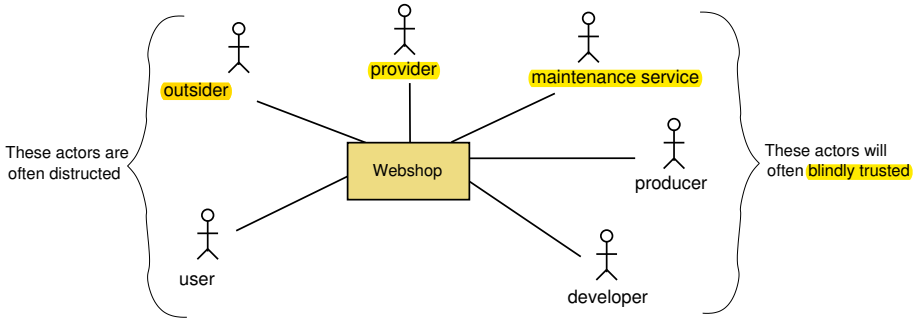
Models and Notations

- (Threat matrix)
- Misuse cases
- Attack trees ➜



Threat Analysis

Where are potential attackers?

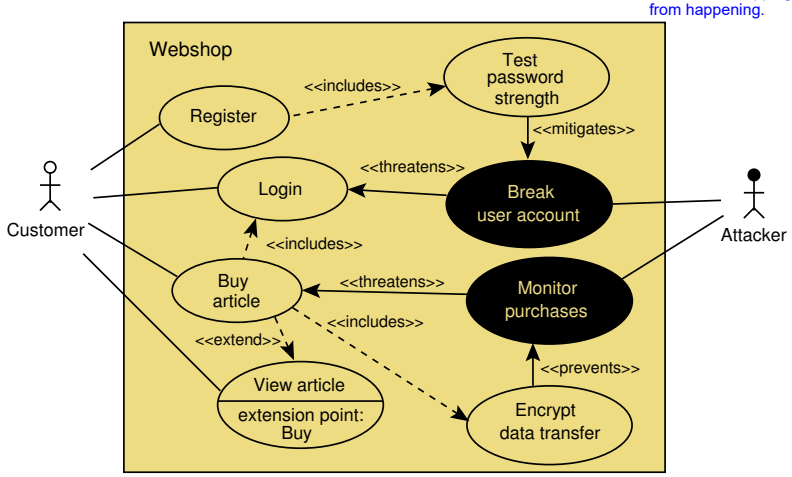


➜ Check all stakeholders, including developers, producers and even other IT systems

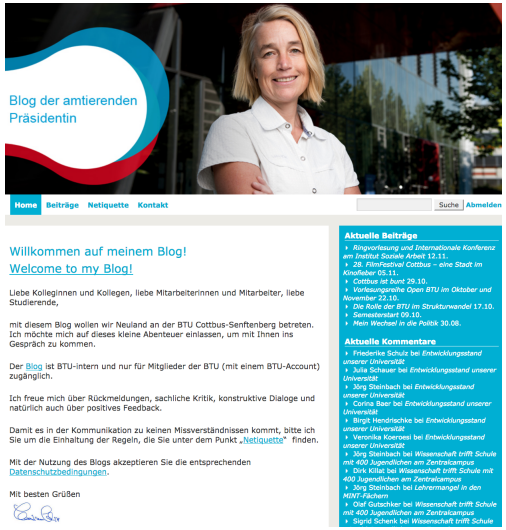
Modeling using Misuse Cases

- Misuse cases model attacker behaviour
- Attacker is also called *Misactor*
- Relationships: *threatens*, *mitigates* & *prevents*

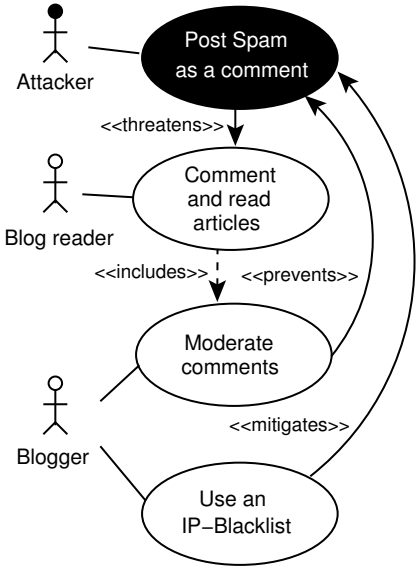
mitigation is reducing the impact of the disaster while prevention is focused on stopping the disaster from happening.



Exercise: Presidential Blog Threat Analysis (Misuse Case Model)



Example for a Misuse Case Model



NON-FUNCTIONAL REQUIREMENTS
The definition of a non-functional requirement is:

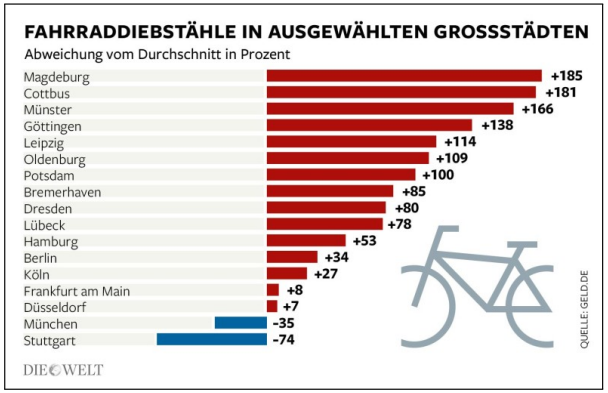
Any requirement which specifies how the system performs a certain function.

In other words, a non-functional requirement will describe how a system should behave and what limits there are on its functionality.

Non-functional requirements generally specify the system's quality attributes or characteristics, for example: "Modified data in a database should be updated for all users accessing it within 2 seconds."

A non-functional requirement for the cup mentioned
previously would be: "contain hot liquid without heating
up to more than 45 °C".

Example: Threat Analysis Bicycle Stealing

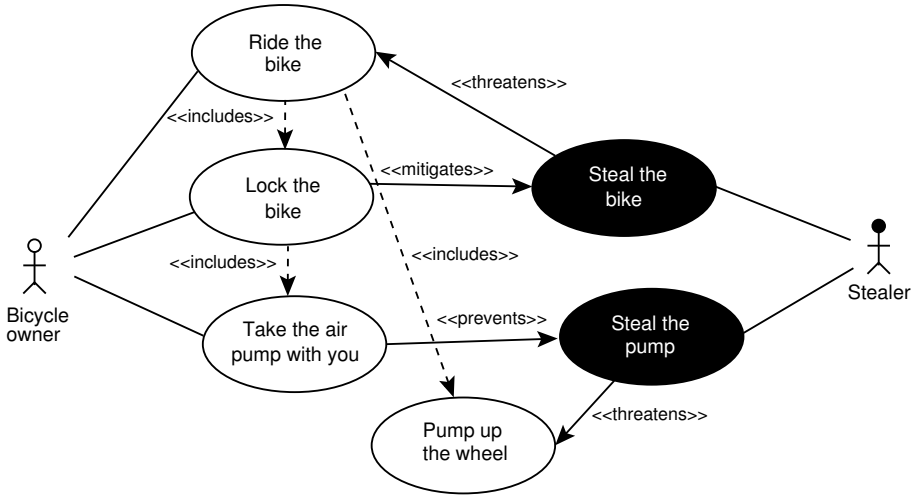


Source: *Die Welt* 26.6.2014

FUNCTIONAL REQUIREMENTS
The definition of a functional requirement is: Any requirement which specifies what the system should do.
In other words, a functional requirement will describe a particular behaviour of function of the system when certain conditions are met, for example: "Send email when a new customer signs up" or "Open a new account".

A functional requirement for an everyday object like a cup would be: "ability to contain tea or coffee without leaking".

Example: Threat Analysis Bicycle Stealing



typically non-functional requirements fall into areas such as:

Accessibility
Capacity, current and forecast
Compliance
Documentation
Disaster recovery
Efficiency
Effectiveness
Extensibility
Fault tolerance
Interoperability
Maintainability
Privacy
Portability
Quality
Reliability
Resilience
Response time
Robustness
Scalability
Security
Stability
Supportability
Testability

Evaluation of the Misuse Case Notation

Advantages

+ Provides a detailed analysis of attack scenarios
+ Is supported by a distinct methodology for describing functional and non-functional requirements, e.g. by dealing with external threats
+ Similar to the popular UML notation

Disadvantages

(make trivial or insignificant)

– Trivialisation of security requirements
– Models can quickly become confusing

Typical functional requirements include:

Business Rules
Transaction corrections, adjustments and cancellations
Administrative functions
Authentication
Authorization levels
Audit Tracking
External Interfaces
Certification Requirements
Reporting Requirements
Historical Data
Legal or Regulatory Requirements

A functional requirement describes what a software system should do, while non-functional requirements place constraints on how the system will do so.

Let me elaborate.

An example of a functional requirement would be:

A system must send an email whenever a certain condition is met (e.g. an order is placed, a customer signs up, etc).
A related non-functional requirement for the system may be:

Emails should be sent with a latency of no greater than 12 hours from such an activity.
The functional requirement is describing the behavior of the system as it relates to the system's functionality. The non-functional requirement elaborates a performance characteristic of the system.