

TDDE49

Information security modeling

Gurjot Singh Gaba
gurjot.singh@liu.se

Information security modeling

Security – non-technical example

- House rules:
 - No strangers alone in the house
 - Only roommates may remove things from the house
- Someone leaves the door unlocked
- A stranger enters the house through the door
- The stranger steals the TV

Security example – in IT security terms

- House rules (no strangers alone in the house, only roommates may remove things from the house) ~ **Security policy**
- Stranger alone in the house ~ **Security violation**
- Valuables (TV, laptop, ...) ~ **Assets**
- Robber ~ **Threat agent, adversary** -> later **attacker**
- Unlocked door ~ **Vulnerability**
- Someone wants to steal the TV and sell it for cash ~ **Threat**
(Existence of a vulnerability and an adversary)
- Entering through unlocked door ~ **Attack vector**
- Stranger enters through the door and removes the TV ~ **Attack**

Information security modeling notions

- **Asset** – anything useful or valuable worth protecting
 - E.g., data, systems and infrastructure, human resources
- **Vulnerability** – an insufficient protection of an asset
 - Design/implementation flaw – e.g., missing input validation
 - Deployment/configuration issue – e.g., default passwords
 - Feature misuse – HTML in email to disguise a phishing link
- **Threat** – an event with the potential to harm an asset
 - Existence of a capable and incentivised adversary and a vulnerability; not necessary that the attack will occur
 - E.g., acts of malicious internal/ext. users, accidents, disasters

Information security modeling notions

- **Security policy** – formally defines security
 - E.g., the goals, rules, and practices; what is and isn't allowed
- **Security violation** – system is in an unauthorized state
- **Adversary, threat agent, attacker**
- **Attack** – a realization of a threat
 - E.g., by an attacker, due to the existence of a vulnerability
- **Attack vector** – steps to carry out an attack
- **Countermeasures, security mechanisms** – processes and measures that help enforce the policy; prevent violations, detect violations and limit damage; handle recovery

Risk assessment

Risk assessment

- **Risk** – determined by combination of the probability of an attack and the damage caused by the attack

$$R = T * V * C \text{ (risk equation)}$$

- T – probability that the threat is instantiated, V – existence of vulnerabilities, C – cost of an attack (for victim)

$$R = P * C$$

- P – combined probability – adversary exploits a vulnerability
- **Security measure** – a mechanism for decreasing or eliminating risk
 - E.g., checking IDs at a reception, a fire suppression system

Quantitative risk assesment

- Numerically estimating the **expected losses**
- **Cost-benefit analysis**
 - The total cost of a defense should not exceed the anticipated benefit (i.e., the expected loss)
- Disadvantages:
 - Rare incidents are difficult to estimate
 - Vulnerabilities evolve and remain undiscovered
 - Attacker actions are hard to predict
 - Unknown value of intangible assets (e.g., know-how, reputation)

Qualitative risk assesment

- **Categorical rating of risks**
- Determine the order in which assets require attention
- Example matrix combining probability and impact:

C: Cost (impact)	P: Probability				
	Rare	Unlikely	Possible	Likely	Certain
Negligible (very low)	1	1	1	1	1
Limited (low)	1	2	2	2	2
Serious (moderate)	1	2	3	3	3
Severe (high)	2	2	3	4	4
Catastrophic (very high)	2	3	4	5	5

Risk assessment – addressing threats

- **Mitigating threats**

- Making it harder to take advantage of a threat – raises cost/effort for attacker
- E.g., password composition rules to make guessing harder

- **Eliminating threats**

- Typically eliminate features, decommission systems

- **Tranferring threats**

- Letting someone else handle the risk
- E.g., insurance, OAuth “Sign in with Google”

- **Accepting risks**

- If the costs of the countermeasures exceed the expected loss

Security analysis & adversary modelling

Security analysis

- From design & development to testing & deployment
- Find design vulnerabilities and overlooked threats
- “What’s your threat model?”
 - What are we protecting? What assets have value?
 - What can go wrong? What attacks put the assets at risk?
 - How can we stop or manage damaging actions?
 - How well did we perform the analysis? (Iterate the process)
- Centered on attackers, assests
 - E.g., assets: Things the attackers want, Things you want to protect, Stepping stones to get either of those

Security analysis methods

- Vulnerability assessment
 - Finding weaknesses in deployed systems
 - Penetration testing (pen testing)
 - Black-box vs. White-box (e.g., source code review)
 - Independent formal security evaluation and certification
- Threat modeling (second part of the lecture)
 - Threat model – threats, threat agents, attack vectors
 - Adversary model – attributes of the adversary
 - Assumptions – about the system, environment, and attackers
 - Good model also specifies what is *out of scope*

Adversary attributes – Know your enemy

- **Goals and objectives** – intent and motivation
 - Example named motivation levels: 1. curiosity, 2. personal fame, 3. personal gain, 4. national interests
- **Methods** – expected types of attacks
- **Capabilities** – tools and skills, computing resources, opportunity (e.g., physical access), personnel
 - Example named skill level: 1. script kiddie, 2. undergraduate, 3. expert, 4. specialist
- **Funding level** – influences the attributes above
- **Insider or outsider** – starting advantage
 - E.g., initial level of access to the system, knowledge of system

Adversary classes (Paul C. van Oorschot)

1. Foreign intelligence (e.g., nation state attackers, government-funded agencies)
2. Cyber-terrorists and politically motivated attackers (e.g., hacktivists)
3. Industrial espionage (e.g., competitors)
4. Organized crime (structured groups)
5. Lesser criminals and crackers (script kiddies)
6. Malicious insiders (e.g., disgruntled employees)
7. Non-malicious employees (e.g., security unaware or curious users)

Adversary classes (according to me)

- 
1. Foreign intelligence (e.g., nation state attackers, government-funded agencies)
 - ~~4.~~ **2.** Organized crime (structured groups)
 3. Industrial espionage (e.g., competitors)
 - ~~2.~~ **4.** Cyber-terrorists and politically motivated attackers (e.g., hacktivists)
 5. Lesser criminals and crackers (script kiddies)
 - ~~(6.)~~ **1?** Malicious insiders (e.g., disgruntled employees)
 7. Non-malicious employees (e.g., security unaware or curious users)

Threat modelling

Threat modeling

- Identify
 - Assets
 - Potential vulnerabilities
 - Threat agents
- Allows estimation of likelihoods and consequences of attacks $\Rightarrow$ *Risk modelling*
- Threat and risk modeling is used as basis for proposing security measures and mitigations
- Method: *structured brainstorming*
 - Many tools/approaches exist for aiding in this process

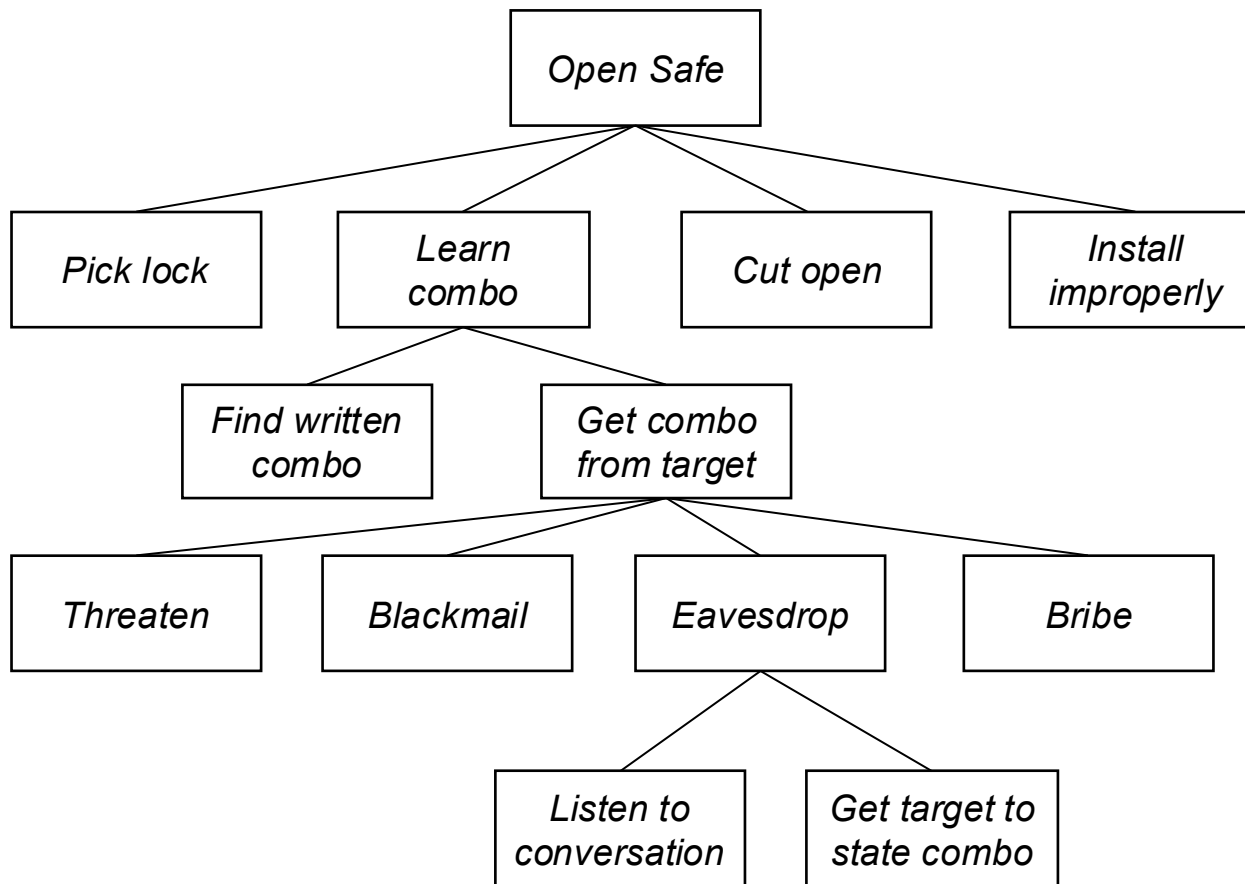
Threat modeling – checklists

- Lists of well known threats compiled by larger communities from years of experience
- More accessible for beginners
- Disadvantages:
 - General vs. specific deployment of your system
 - False sense of security?
- Cross-reference and combined with other methods

Threat modeling – attack trees

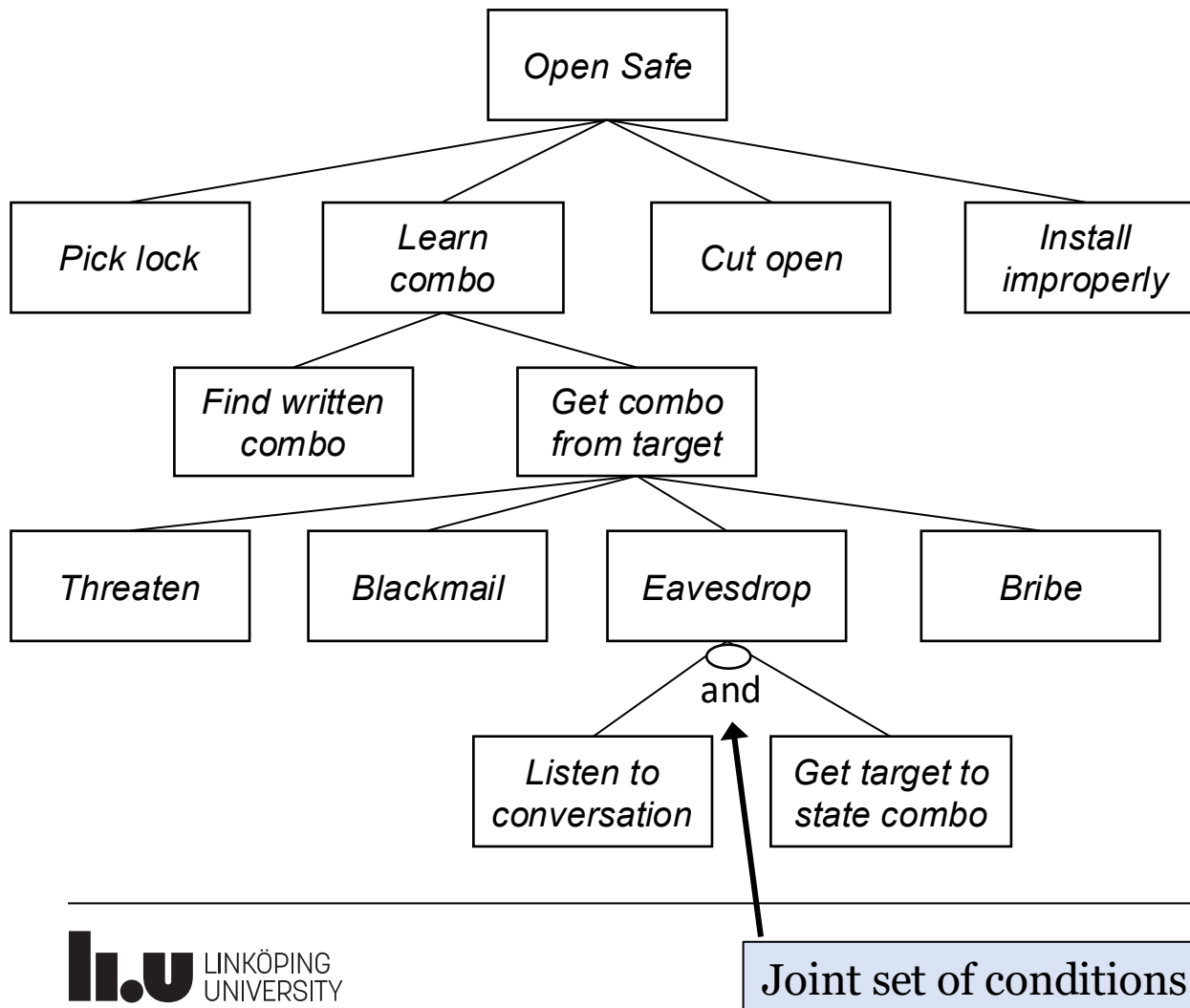
- Attacks on a system in a tree structure
 - *Root node* at the top represents the goal
 - Internal nodes are subtrees ending in *leaf nodes*
 - Lower nodes show *alternative ways* (i.e. OR) of reaching the parent, some can be marked as AND (joint set of conditions)
- Nodes can be annotated
 - Legal/illegal, cost, probability of success, likelihood of attack, required skills and equipment, etc.
 - Mark out infeasible nodes (but keep in the model)
- Think like an attacker, brainstorm and review with colleagues
- Revisit the tree and study the attack vectors and defenses

Attack tree example



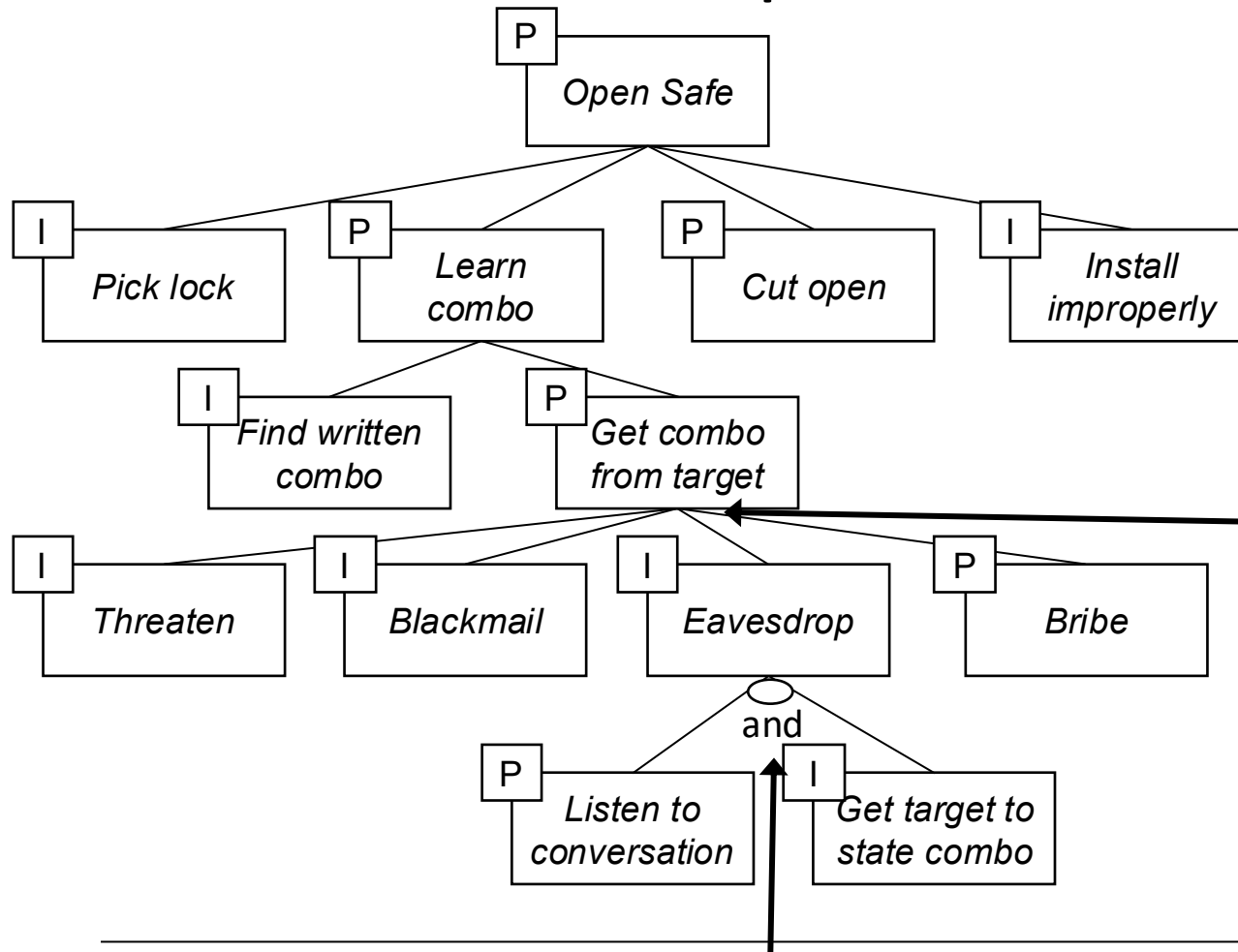
- Root node, overall attack goal: get assets from a safe
- Alternative ways of reaching the goal; Possible threats posed by the attacker
- Security measures must address leaf nodes of the tree

Attack tree example



- Root node, overall attack goal: get assets from a safe
- Alternative ways of reaching the goal; Possible threats posed by the attacker
- Security measures must address leaf nodes of the tree

Attack tree example

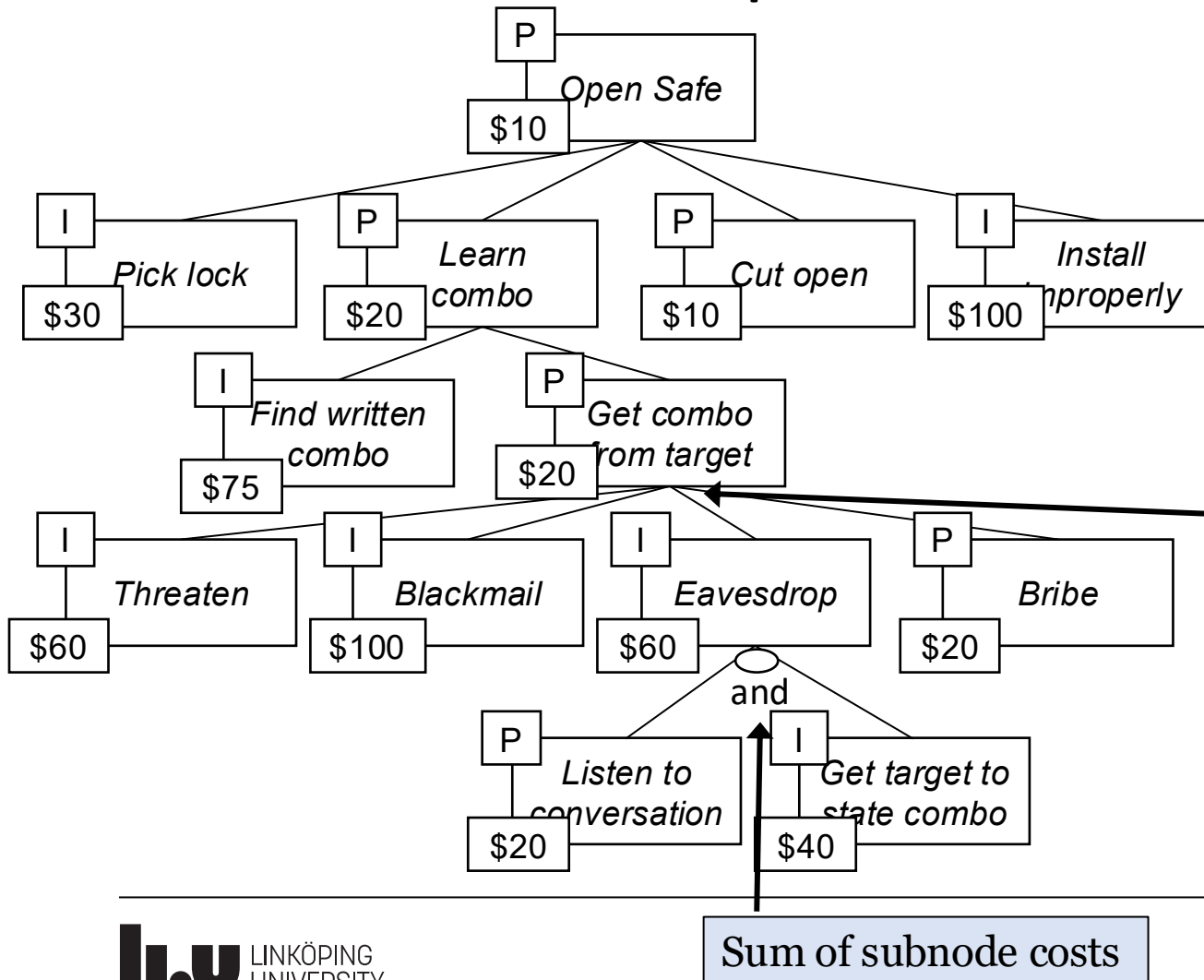


- Nodes can be annotated with various attributes
- Attributes propagate upwards in tree
- For example:
 - **Likelihood**

Logical OR of "possible"/"impossible" (or probability of at least one attack in subnodes succeeding)

Logical AND of "possible"/"impossible" (or compound probability of numerical likelihood)

Attack tree example



- Nodes can be annotated with various attributes
- Attributes propagate upwards in tree
- For example:
 - Likelihood
 - **Cost**

Minimum of
subnode costs

Sum of subnode costs

Threat modeling – STRIDE 1/3

- **STRIDE** – keywords to stimulate brainstorming

1. Spoofing

- Impersonating someone, pretending to be someone else
- E.g., faking the sender field of an e-mail; impersonating a customer or a website
- Violates authentication

2. Tampering

- Modifying data (in storage or in transit)
- E.g., changing files or DB entries, dropping network packets
- Violates integrity

Threat modeling – STRIDE 2/3

3. Repudiation

- Denial of an action, not acknowledging responsibility
- E.g., denying approving an expense report
- Violates non-repudiation (actions of users cannot be refuted)

4. Information disclosure

- Allowing access to data to unauthorized users
- E.g., selling company secrets, failing to set up authorization for a database
- Violates confidentiality

Threat modeling – STRIDE 3/3

5. Denial of service (DoS)

- Preventing a system from providing a service
- E.g., by consuming system resources; a distributed DoS attack uses up all available network connections
- Violates availability

6. Elevation of privilege

- Doing something not allowed at the current level of authorization
- E.g., user code running with admin privileges; accessing the business logic directly instead of through the web interface
- Violates authorization

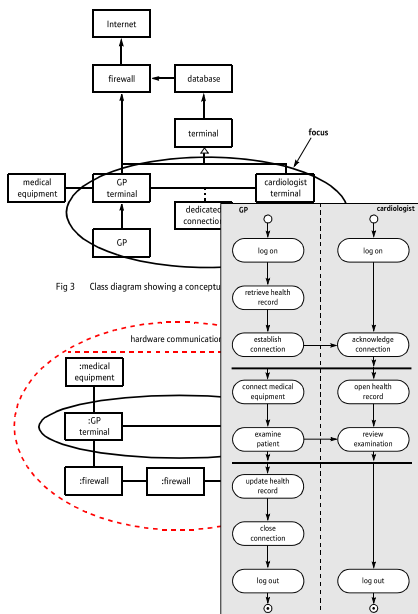
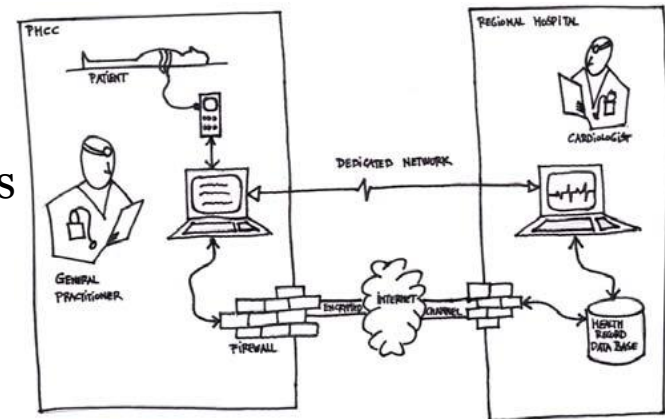
CORAS

- Method for threat and risk modeling
 - Not necessarily the most well-known method, but good illustration of the steps involved in a security analysis
- Consists of 7 steps
- Read in preparation for the teaching session and the security modeling assignment:

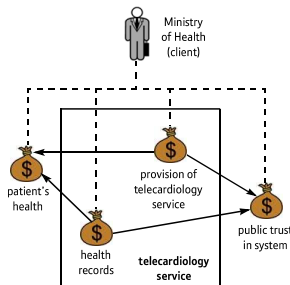
den Braber et al. (2007). Model-based security analysis in seven steps — a guided tour to the CORAS method. *BT Technology Journal*, 25(1)
<https://link.springer.com/content/pdf/10.1007/s10550-007-0013-9.pdf>

CORAS – overview

Step 1 – Experts and clients decide upon which system is to be analyzed and what parts of the system that should be focused upon.



Step 2 – The system to be analyzed is formalized, assets are identified, high-level risk analysis.



Who/what causes it?	How? What is the incident? What does it harm?	What makes it possible?
Hacker Employee	Breaks into the system and steals health records Sloppiness compromises confidentiality of health records	Insufficient security Insufficient training
Eavesdropper	Eavesdropping on dedicated connection	Insufficient protection of connection
System failure	System goes down during examination	Unstable connection/immature technology
Employee	Sloppiness compromises integrity of health record	Prose-based health records (i.e. natural language)
Network failure	Transmission problems compromise integrity of medical data	Unstable connection/immature technology
Employee	Health records leak out by accident — compromises their confidentiality and damages the trust in the system	Possibility of irregular handling of health records

CORAS – overview

Consequence value	Description
Catastrophic	1000+ health records (HRs) are affected
Major	100-1000 HRs are affected
Moderate	10-100 HRs are affected
Minor	1-10 HRs are affected
Insignificant	No HR is affected

Likelihood value	Description ³
Certain	Five times or more per year (50-∞: 10y = 5-∞: 1y)
Likely	Two to five times per year (21-49: 10y = 2,1-4,9: 1y)
Possible	Once a year (6-20: 10y = 0,6-2: 1y)
Unlikely	Less than once per year (2-5: 10y = 0,2-0,5: 1y)
Rare	Less than once per ten years (0-1:10y = 0-0,1:1y)

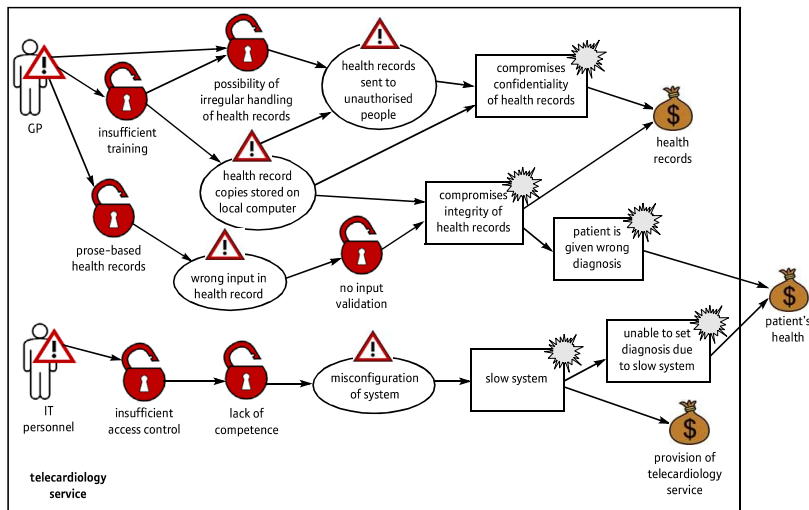
Asset	Importance	Type
Health records	2	Direct asset
Provision of telecardiology service	3	Direct asset
Public's trust in system	(Scoped out)	Indirect asset
Patient's health	1	Indirect asset

Step 3 – Prioritize assets, create scales for consequence and likelihood values, create risk evaluation matrix.

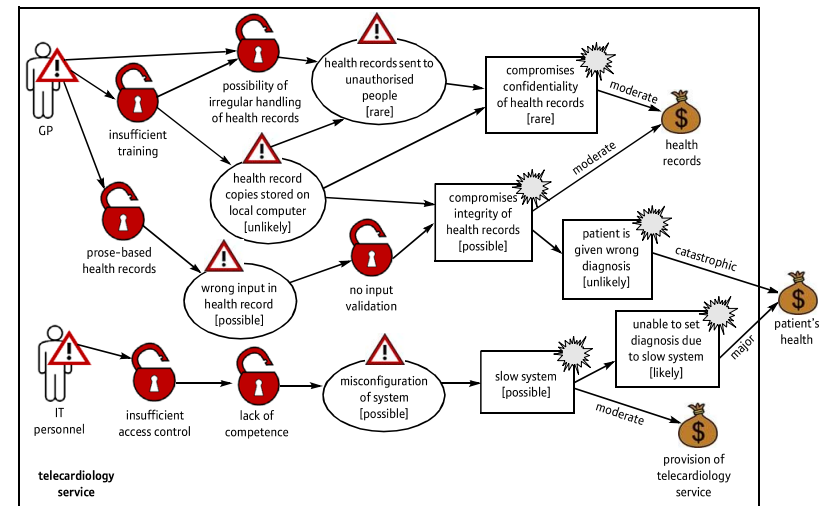
		Consequence				
		Insignificant	Minor	Moderate	Major	Catastrophic
Frequency	Rare	Acceptable	Acceptable	Acceptable	Acceptable	Must be evaluated
	Unlikely	Acceptable	Acceptable	Acceptable	Must be evaluated	Must be evaluated
	Possible	Acceptable	Acceptable	Must be evaluated	Must be evaluated	Must be evaluated
	Likely	Acceptable	Must be evaluated	Must be evaluated	Must be evaluated	Must be evaluated
	Certain	Must be evaluated	Must be evaluated	Must be evaluated	Must be evaluated	Must be evaluated

CORAS – overview

Step 4 – Create threat diagrams through structured brainstorming (workshop).



Step 5 – Estimate risks (consequence and likelihood)

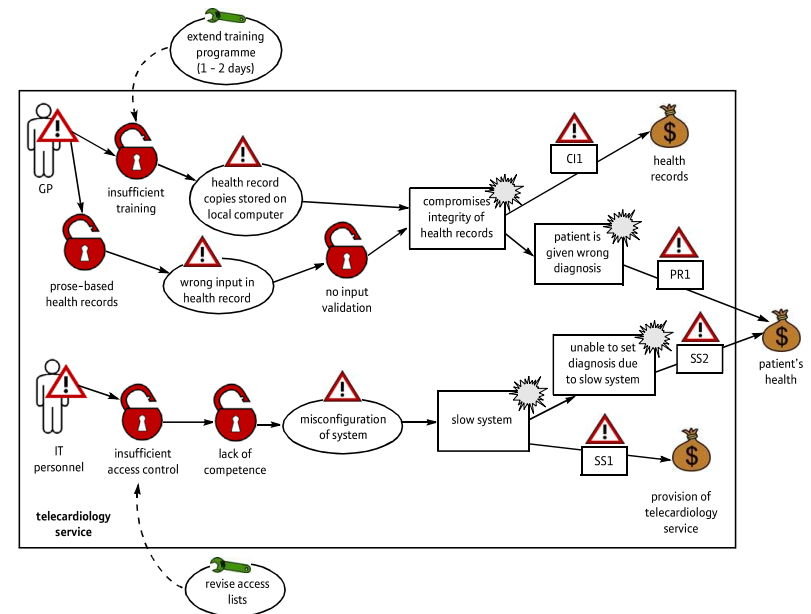


CORAS – overview

Step 6 – Risk evaluation,
estimates are confirmed or
adjusted.

		Consequence				
Likelihood		Insignificant	Minor	Moderate	Major	Catastrophic
	Rare			CC1		
	Unlikely					PR1
	Possible			CI1, SS2		
	Likely				SS1	
	Certain					

Step 7 – Risk treatment



Checking the model

- Modeling – iterative process, evolving threat models
- Invalid assumptions and focusing on wrong threats
- Real world outcomes of a security policy:
 - Defenses prevent policy violations, and the policy is complete w.r.t. the security needs of the organization
 - Defenses fail to support the policy, goals are not met
 - The policy fails to capture the actual security needs of the organization, even correctly implemented defenses may be insufficient

Security design principles

Security design principles

- **Verify first** (before trusting)
 - “Trust, but verify”
- **Security by design**
 - Security awareness since early design stage
 - Specify design goals and assumptions, who is trusted and not trusted, what is out of scope
- **Design for evolution**
 - Re-evaluate effectiveness of security mechanisms, be ready to update designs as needed
 - Algorithm agility – upgrading crypto algorithms



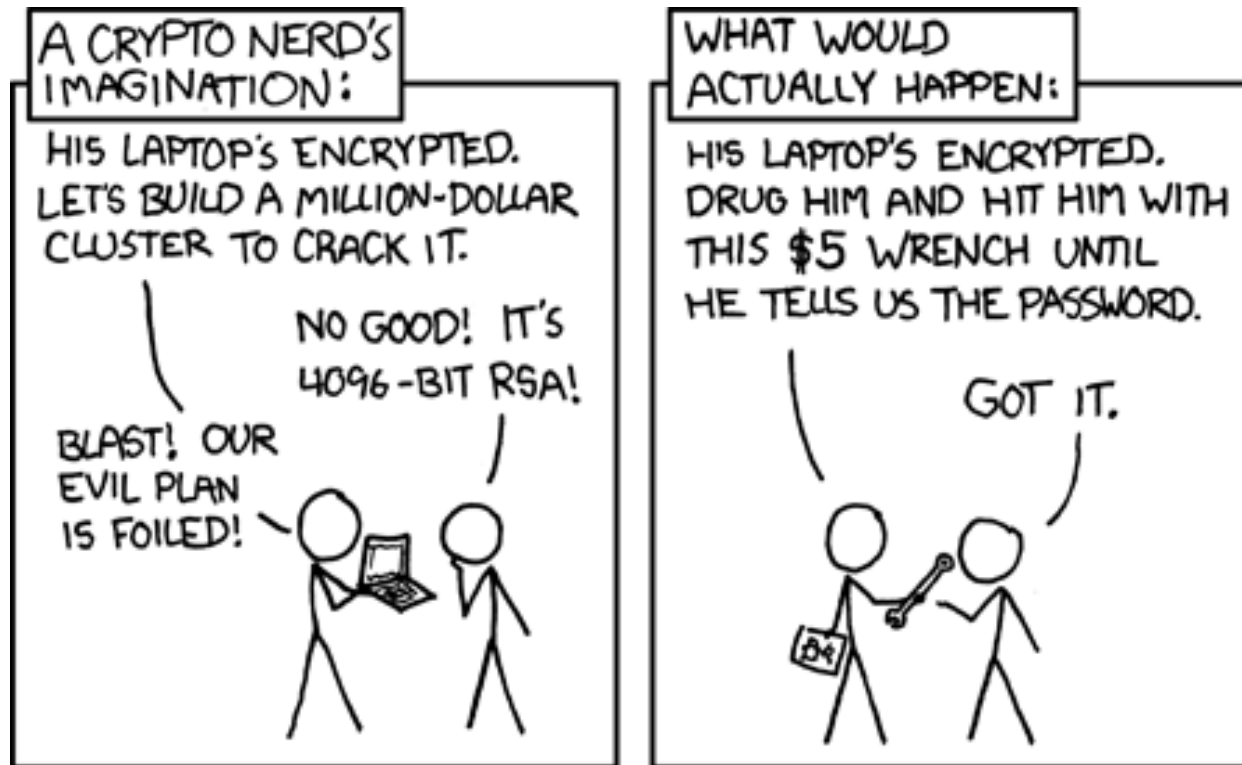
Selected security design principles 1/2

- **Simplicity and necessity** – keep designs as simple & small as possible; keep only essential functionality
 - Minimizes attack surface (possible attack vectors)
- **Safe defaults** – defaults often go unchanged
 - Access control deny-by-default (whitelist over blacklist)
 - Fail-safe systems – “closed” when they fail
- **Open design** – don’t rely on secret designs, attacker ignorance, “security by obscurity”
- **Least privilege** – allocate the fewest privileges needed for the shortest duration possible

Selected security design principles 2/2

- **Time tested tools** – expert-built security tools
 - Don't implement custom crypto primitives and protocols
- **Least surprise** – security mechanisms should behave as users expect (users' mental models)
- **User buy-in** – users should be motivated to use security mechanisms
 - User experience, convenient, clearly beneficial
- **Defense in depth** – defenses built in multiple layers
 - Avoid single point of failure; strengthen the weakest link first
- More principles in Ch. 1.7 of *Computer Security and the Internet*

Weakest link: “Rubber-hose cryptanalysis”



<https://xkcd.com/538/>

