Threat Modeling: A Guided Tour

Koen Yskout

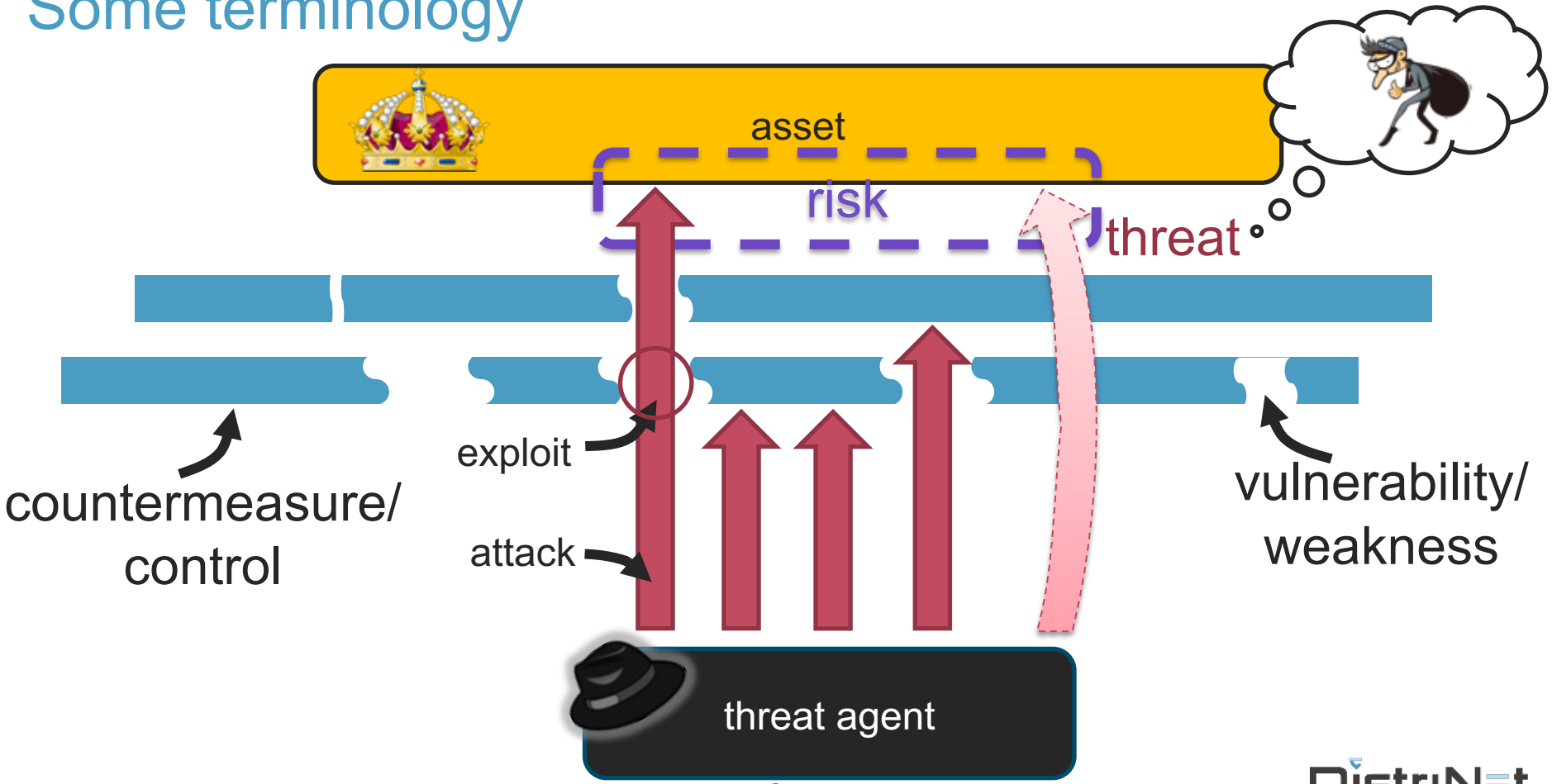
April 8, 2020 – CIF seminar

DistriNet

Goal: secure your home

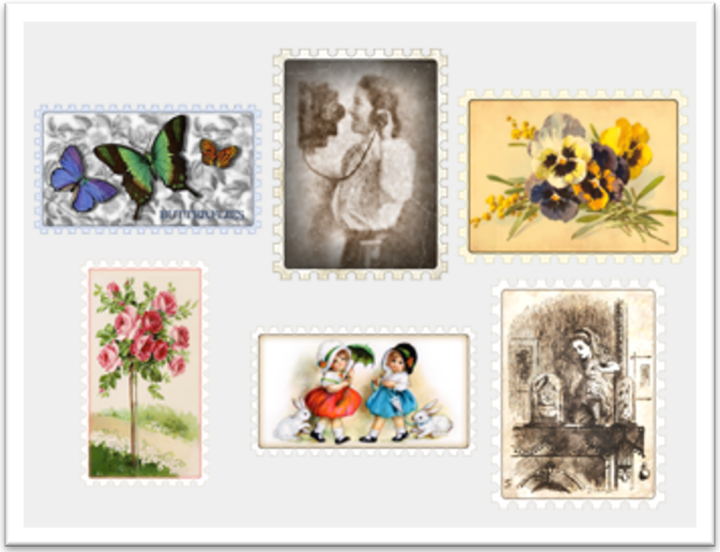
- **Option 1.** Wait for security incident, then fix it
  - Penetrate & patch
- **Option 2.** Invite ex-burglar to point out weaknesses, then install defenses
  - Security as an afterthought
- **Option 3.** Think about security before house is being built, involving security professionals
  - Security by design

Some terminology



DistriNet

Terminology example



Asset: stamp collection

Threat: collection is stolen

Threat agent: burglar who needs money

Risk: value of collection × likelihood of being stolen

Attack: successful theft

Countermeasure: locked door

Weakness: lock with pin tumbler

Exploit: pick the lock with standard tools

Countermeasure: stored in safe

Weakness: safe not anchored

Exploit: remove entire safe

Terminology example

bal_date	cust_id	eod_cust_bal_usd
12/1/2013	233270	0.03
12/1/2013	324490	0.04
12/1/2013	338470	839.6
12/1/2013	793440	103.810005
12/1/2013	800980	2080556.4
12/1/2013	854210	1981.33
12/1/2013	1080910	130742.94
12/1/2013	1080920	34938248
12/1/2013	1605310	5352583.5
12/1/2013	1619050	1033902340
12/1/2013	2311320	0
12/1/2013	2990820	31795

Asset: customer data

Threat: data leaks

Threat agent: script kiddie

Risk: value of data × likelihood of being stolen

Attack: attacker obtains a copy of the data by hacking the application

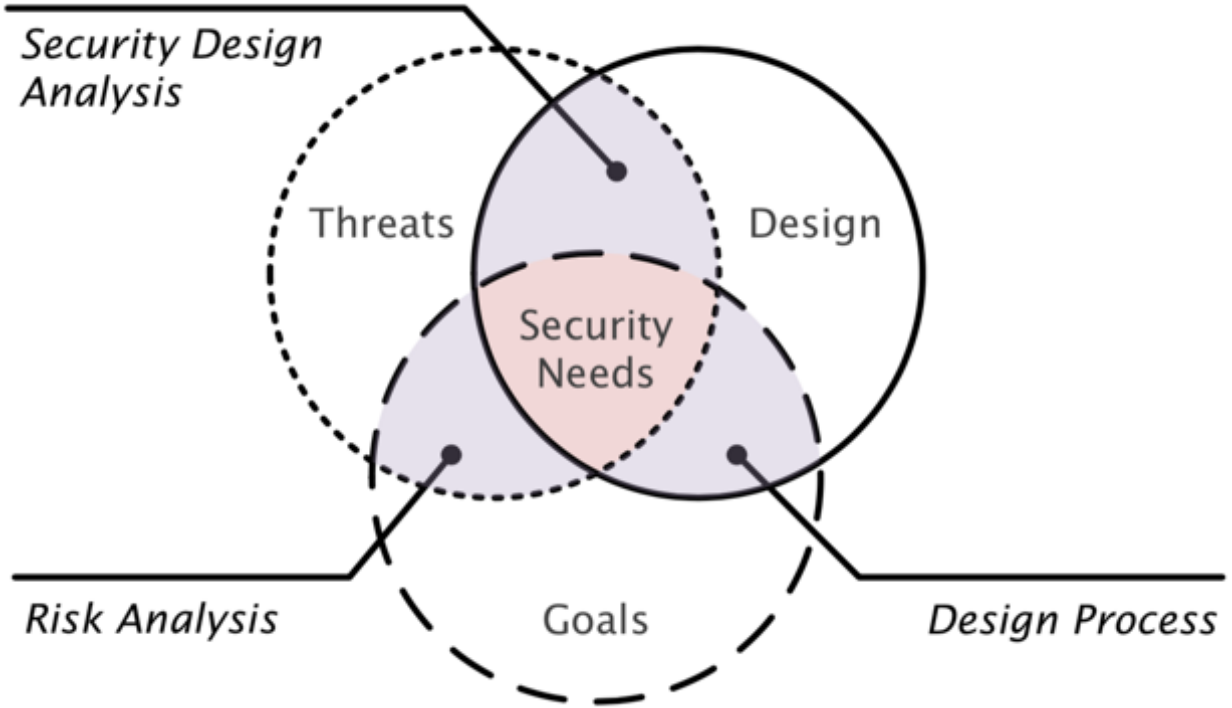
Countermeasure: firewall

Weakness: only protects at network level — **Exploit**: application-level attack

Countermeasure: access control

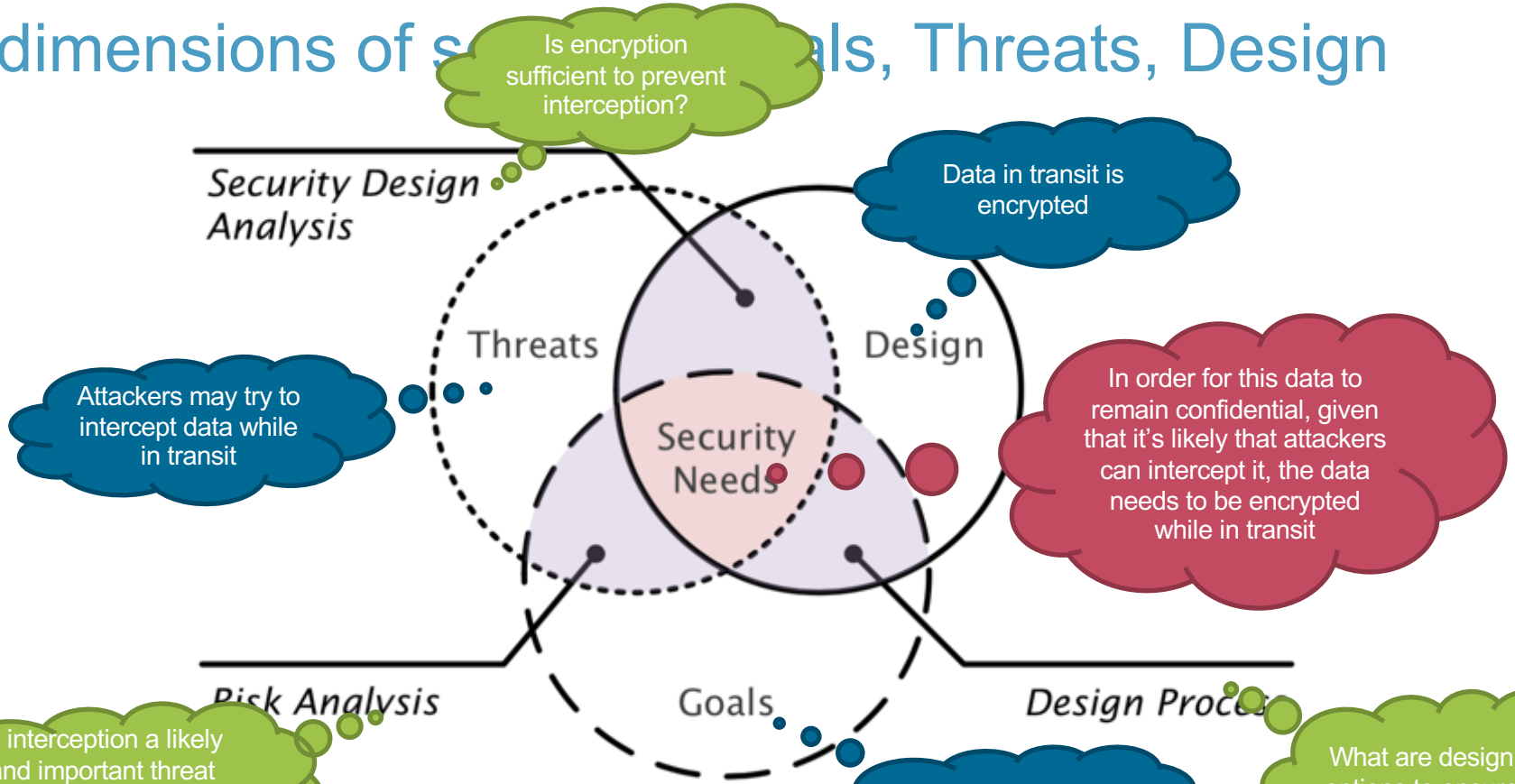
Weakness: application has full DB access — **Exploit**: SQL injection

3 dimensions of security: Goals, Threats, Design



TÜRPE, S., 2017. The Trouble With Security Requirements. In *25th IEEE International Requirements Engineering Conference*. IEEE Computer Society.

3 dimensions of s…als, Threats, Design



…ble With Security Requirements. In *25th IEE… …ents Engin…*
Co… …EE Computer Society.

DistriNet

GOALS

- CIA triad
  - **Confidentiality**
  - **Integrity**
  - **Availability**
- Extensions (top-level?)
  - Authentication
  - Accountability / Nonrepudiation
  - Auditability
  - Assurance

THREATS: STRIDE

- Spoofing
- Tampering
- Repudiation
- Information disclosure
- Denial of service
- Elevation of privilege

This is only one way to categorize threats, many others exist…

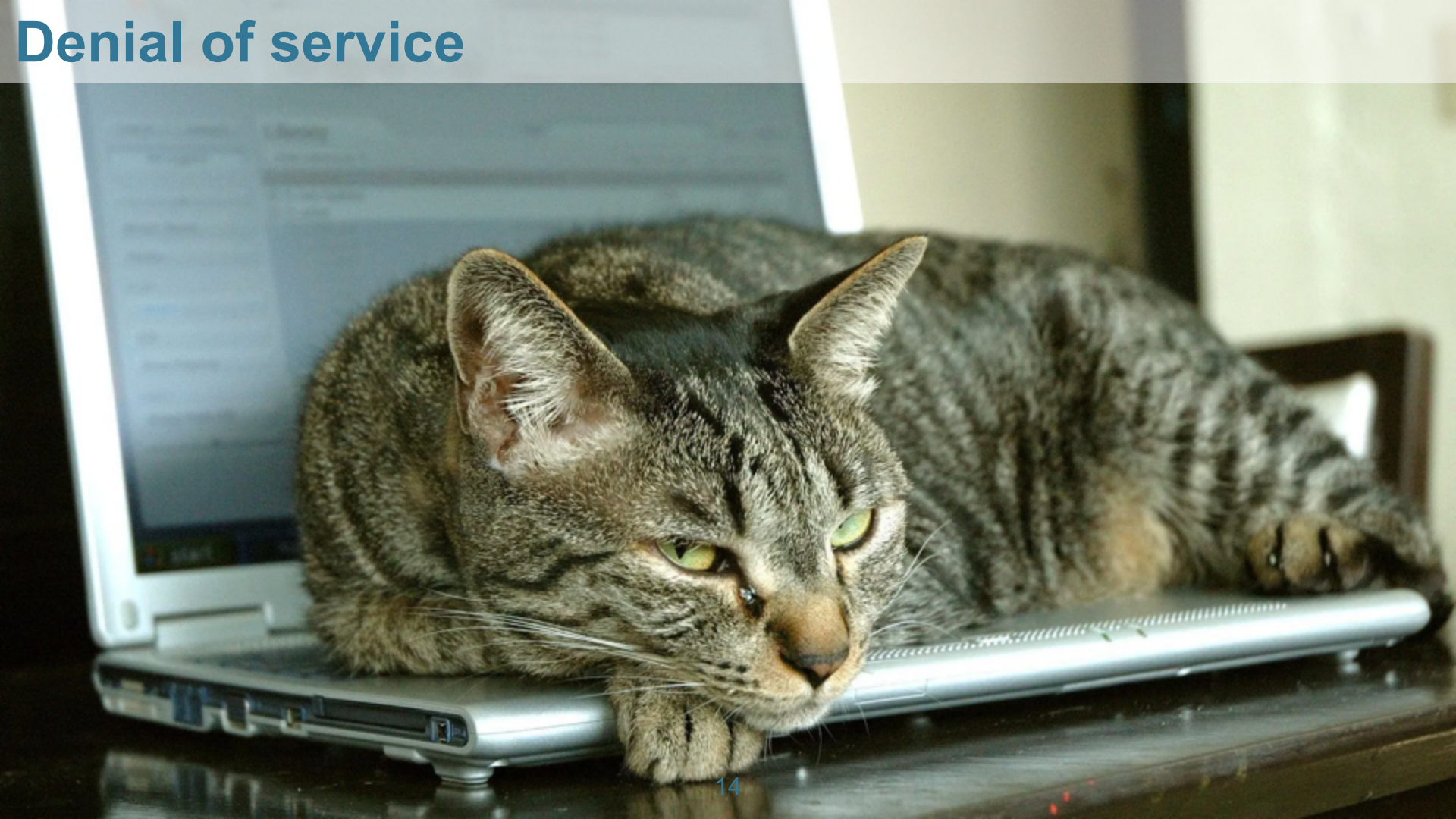
Spoofing

Tampering

Repudiation

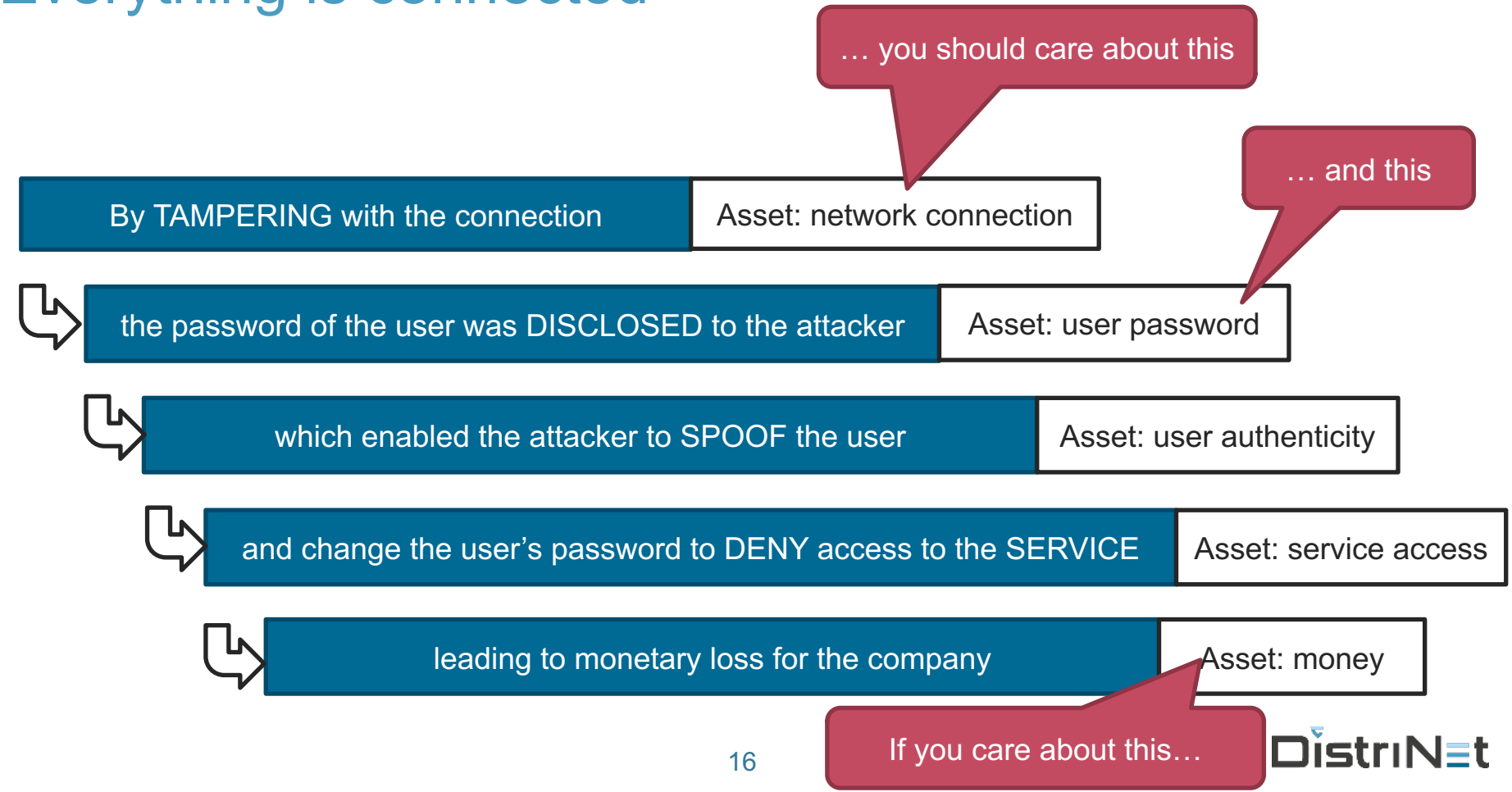
Information disclosure

Denial of service



Elevation of Privilege

Everything is connected



DistriNet

Everything is connected

By TAMPERING with the connection

the password of the user was DISCLOSED to the attacker

which enabled the attacker to SPOOF the user at another service

leading to monetary loss for the other company

The main cause of security problems?

Wrong assumptions!

- About **data formats**
  - Names do not contain special characters
- About **guarantees** provided by other components
  - Access control will be dealt with later in the process
- About **trustworthiness** of components
  - JavaScript sent to a web browser will always run as expected
- About **capabilities** of attacker
  - An attacker will never discover this
- About **behavior** of users
  - A user would never try to do circumvent this

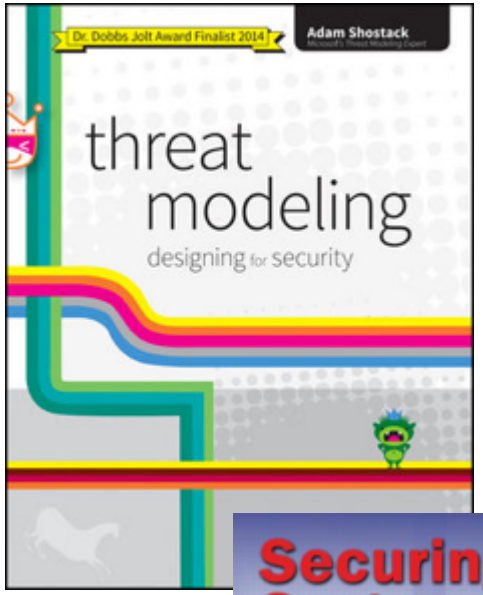
Threat modeling

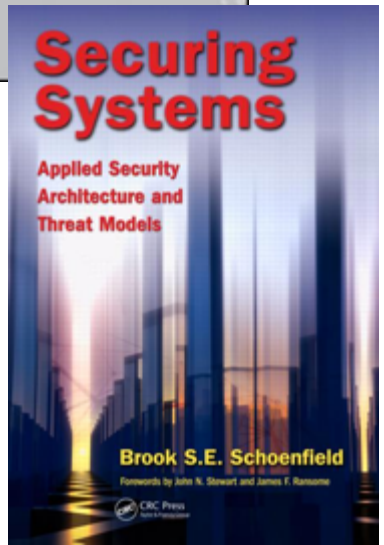
Threat modeling

What?

"In short, threat modeling is the use of abstractions to aid in thinking about risks."

Shostack, A., 2014. Threat Modeling. Wiley.





Also known as:

architectural risk analysis (ARA)

3 approaches to threat modeling

- **Attacker-based**
  - Who are possible attackers?
  - What would the attacker do?
- **Asset-based**
  - What assets do I have to protect?
- **System-based**
  - What is the system I'm protecting?

Attacker-based

“Think like an attacker!”

“Preparing a meal?
Think like a chef!”

Can you list all your (potential) attackers?

Do you know what they’re after?

Asset-based

- List your assets
  - What do you want to protect?
  - What does the attacker want?
    - Think like an attacker!
- How to protect them?

  = what are the threats to these assets?

  Back to #1…

System-based

4-questions model (Shostack)

- What are we building?
- What can go wrong?
  - And do we care?
- What to do about it?
- Did we do a good job?

System-based

ATASM (Schoenfield)

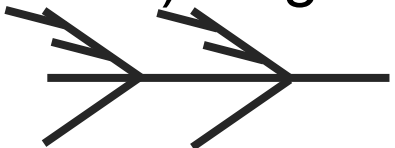
Architecture → Threats → Attack surfaces → Mitigations

- Architecture (system details)
- Threats (threat agents)
- Attack surfaces (potential attacks)
- Mitigations (security controls)

Attack trees

(Threat trees)

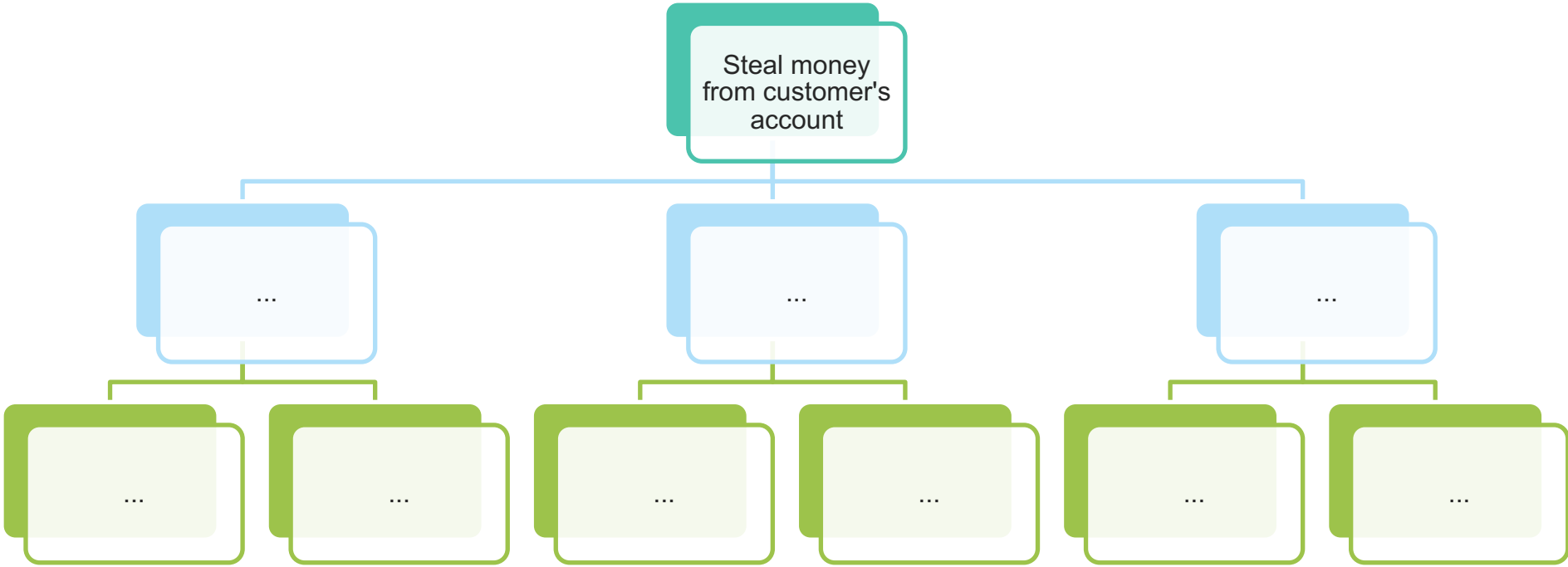
- Similar to *fault trees* (safety & reliability)
  - RUIJTERS, E. AND STOELINGA, M., 2015. Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. *Computer Science Review*, 15–16, pp.29–62.
- Root = the attacker's goal (hence, attacker-based)
- Hierarchically describe different conditions (cause/effect) under which the parent may occur; AND/OR decompositions
- Vizualisation: tree, cause/effect (fishbone) diagram, …



SCHNEIER, B., 1999. Attack trees. *Dr. Dobb's Journal*, (December).
https://www.schneier.com/academic/archives/1999/12/attack_trees.html

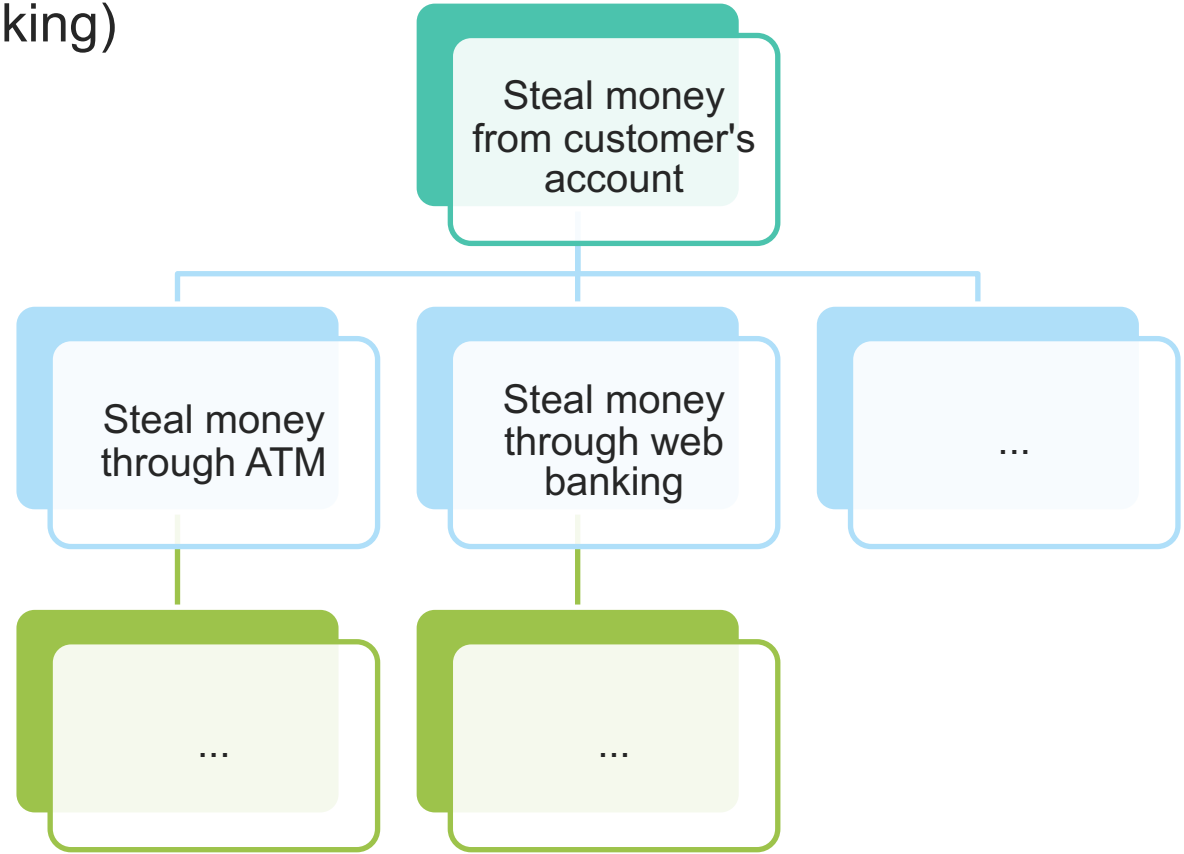
Attack trees

Example (banking)



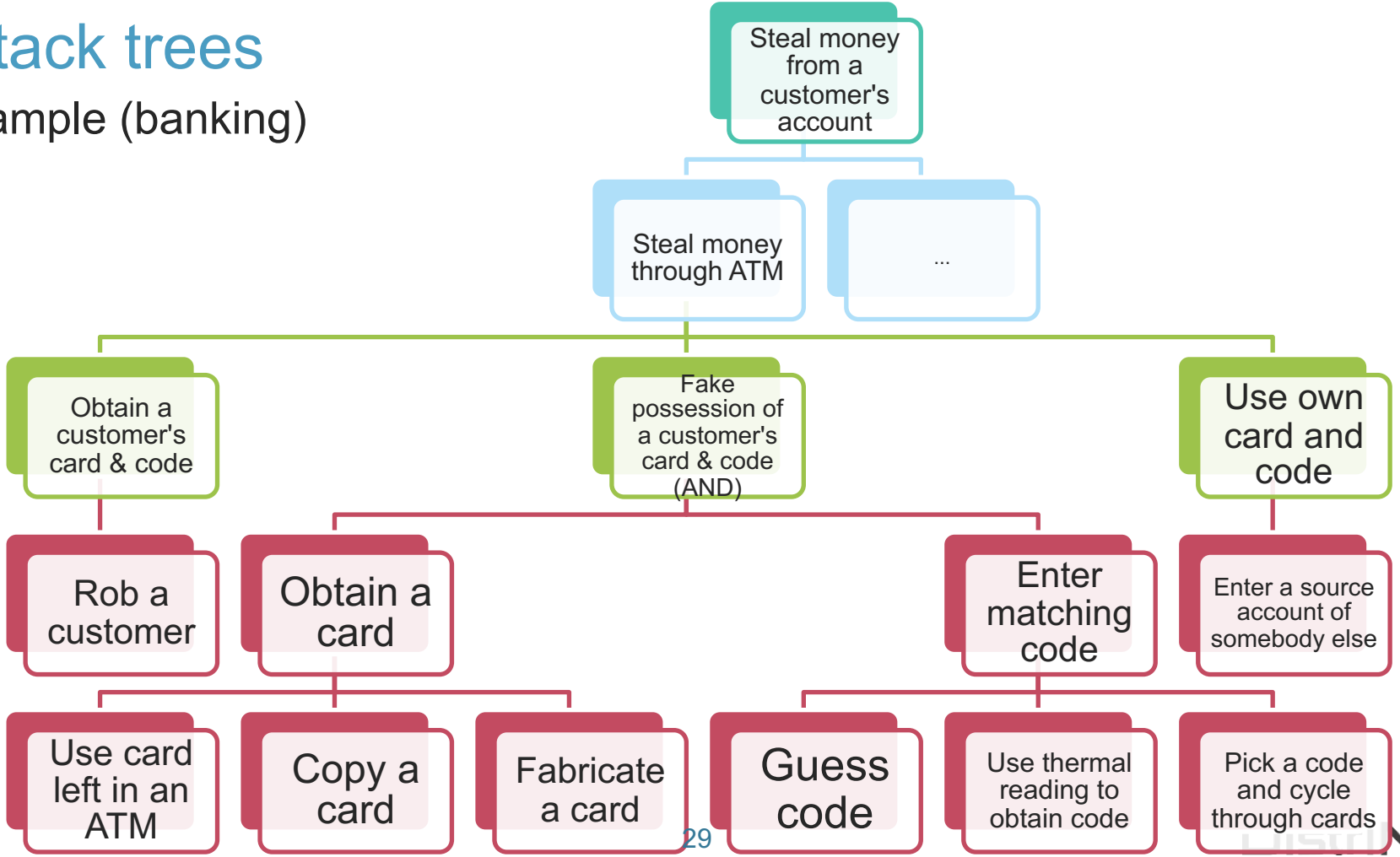
Attack trees

Example (banking)



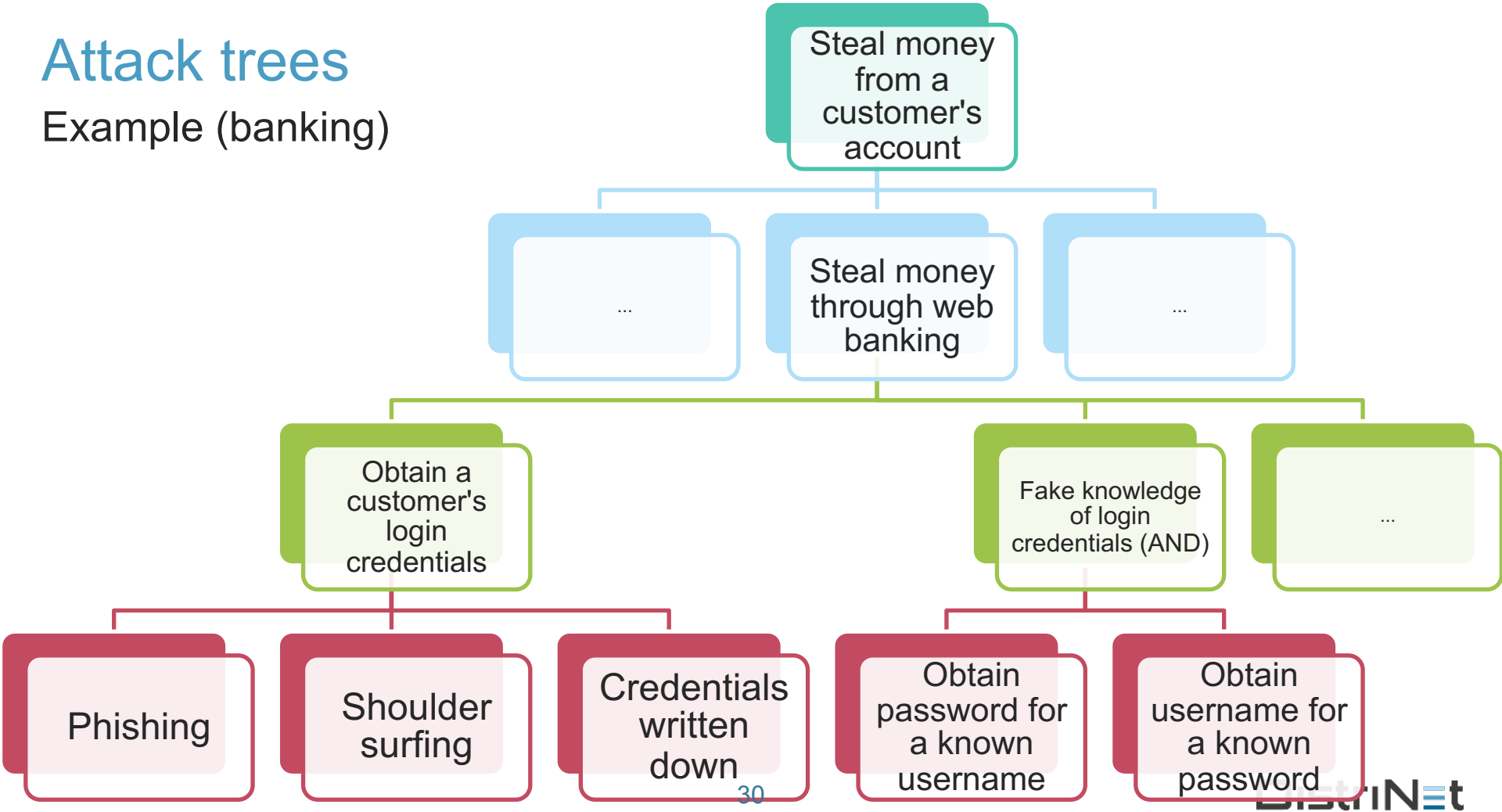
Attack trees

Example (banking)



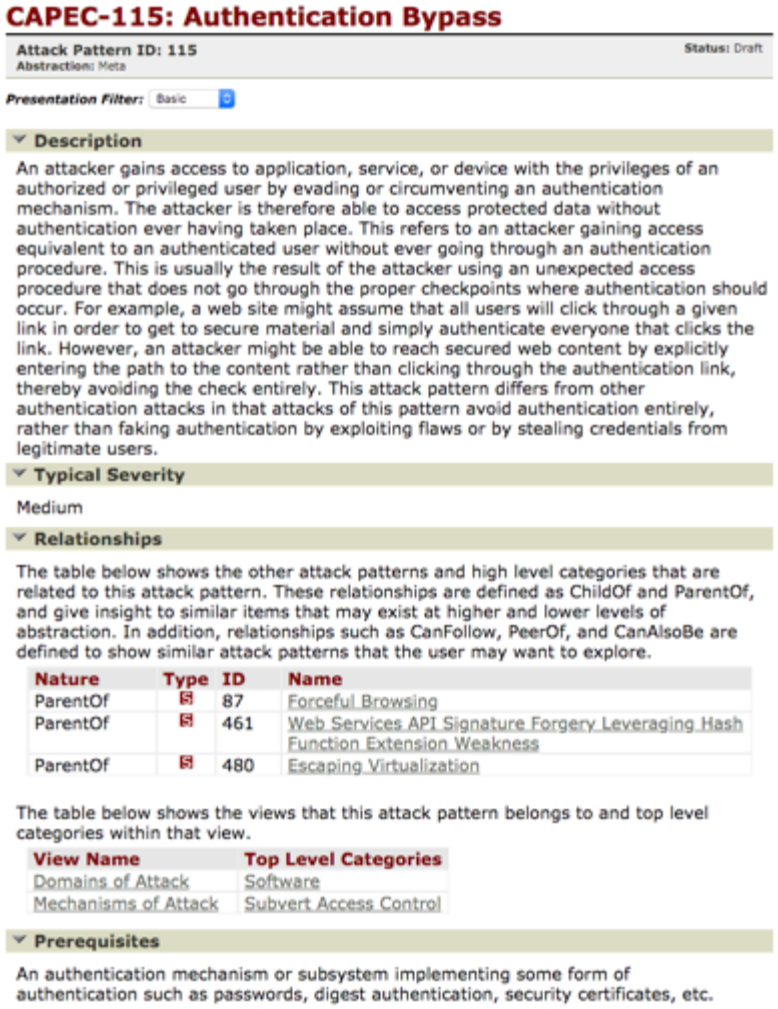
Attack trees

Example (banking)



Attack libraries: CAPEC

- MITRE **CAPEC** (Common Attack Pattern Enumeration and Classification) http://capec.mitre.org/
  - Structured collection of **attack** patterns (e.g., CAPEC-115 Authentication bypass);
- 26 listed use cases (https://capec.mitre.org/about/use_cases.html)
  - From requirements to evaluation
  - Common theme: compose a checklist



CAPEC-115: Authentication Bypass

Attack Pattern ID: 115 **Status:** Draft
Abstraction: Meta

Presentation Filter: Basic

Description

An attacker gains access to application, service, or device with the privileges of an authorized or privileged user by evading or circumventing an authentication mechanism. The attacker is therefore able to access protected data without authentication ever having taken place. This refers to an attacker gaining access equivalent to an authenticated user without ever going through an authentication procedure. This is usually the result of the attacker using an unexpected access procedure that does not go through the proper checkpoints where authentication should occur. For example, a web site might assume that all users will click through a given link in order to get to secure material and simply authenticate everyone that clicks the link. However, an attacker might be able to reach secured web content by explicitly entering the path to the content rather than clicking through the authentication link, thereby avoiding the check entirely. This attack pattern differs from other authentication attacks in that attacks of this pattern avoid authentication entirely, rather than faking authentication by exploiting flaws or by stealing credentials from legitimate users.

Typical Severity

Medium

Relationships

The table below shows the other attack patterns and high level categories that are related to this attack pattern. These relationships are defined as ChildOf and ParentOf, and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as CanFollow, PeerOf, and CanAlsoBe are defined to show similar attack patterns that the user may want to explore.

Nature	Type	ID	Name
ParentOf	S	87	Forceful Browsing
ParentOf	S	461	Web Services API Signature Forgery Leveraging Hash Function Extension Weakness
ParentOf	S	480	Escaping Virtualization

The table below shows the views that this attack pattern belongs to and top level categories within that view.

View Name	Top Level Categories
Domains of Attack	Software
Mechanisms of Attack	Subvert Access Control

Prerequisites

An authentication mechanism or subsystem implementing some form of authentication such as passwords, digest authentication, security certificates, etc.

Checklist-based evaluation

https://github.com/OWASP/ASVS

- XSS and SQL injection

5.3.3	Verify that context-aware, preferably automated - or at worst, manual - output escaping protects against reflected, stored, and DOM based XSS. (C4)	✓	✓	✓	79
5.3.4	Verify that data selection or database queries (e.g. SQL, HQL, ORM, NoSQL) use parameterized queries, ORMs, entity frameworks, or are otherwise protected from database injection attacks. (C3)	✓	✓	✓	89

- Custom crypto

6.2.2	Verify that industry proven or government approved cryptographic algorithms, modes, and libraries are used, instead of custom coded cryptography. (C8)		✓	✓	327
6.2.3	Verify that encryption initialization vector, cipher configuration, and block modes are configured securely using the latest advice.		✓	✓	326

STRIDE

- Originated at Microsoft in 1999
- **S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service, **E**levation of privilege
- Nowadays a basis for a lot of practical threat modeling

Applying STRIDE

- Option 1: Use STRIDE mnemonic when looking for threats
  - Brainstorming, EoP card game, …
  - Focus on assets, attackers, **software**
- Option 2: More systematic variants (~ algorithmic)
  - STRIDE per element
  - STRIDE per interaction (implemented in Microsoft's tool)
- No completeness guarantees!
- Only the discovery of a threat matters, not its precise categorization!

Applying STRIDE systematically

- Create a model (diagram) of your software
- Apply knowledge base to the model to elicit threats

Applying STRIDE systematically: an analogy

	Forced entry	Stolen key	Observe inhabitants
Door	X	X	
Window	X		X
Garage door	X	X	
Fence	X		X

Possible threats:

- Forced entry through front door
- Enter through front door using stolen key
- Forced entry through back door
- Enter through back door using stolen key
- Forced entry through kitchen window
- Observe inhabitants through kitchen window
- Forced entry through garage door
- …

Applying STRIDE systematically: an analogy

	Forced entry	Stolen key	Observe inhabitants
Door	X	X	
Window	X		X
Garage door	X	X	
Fence	X		X

Possible threats:

- Forced entry through front door
- Enter through front door using stolen key
- Forced entry through back door
- Enter through back door using stolen key
- Forced entry through kitchen window
- Observe inhabitants through kitchen window
- Forced entry through garage door
- …

Applying STRIDE systematically: an analogy

	Forced entry	Stolen key	Observe inhabitants
Door	X	X	
Window	X		X
Garage door	X	X	
Fence	X		X

Possible threats:

- Forced entry through front door
- Enter through front door using stolen key
- Forced entry through back door
- Enter through back door using stolen key
- Forced entry through kitchen window
- Observe inhabitants through kitchen window
- Forced entry through garage door
- …

Applying STRIDE systematically: an analogy

	Forced entry	Stolen key	Observe inhabitants
Door	X	X	
Window	X		X
Garage door	X	X	
Fence	X		X

Possible threats:

- Forced entry through front door
- Enter through front door using stolen key
- Forced entry through back door
- Enter through back door using stolen key
- Forced entry through kitchen window
- Observe inhabitants through kitchen window
- Forced entry through garage door
- …

Applying STRIDE systematically: an analogy

	Forced entry	Stolen key	Observe inhabitants
Door	X	X	
Window	X		X
Garage door	X	X	
Fence	X		X

Possible threats:

- Forced entry through front door
- Enter through front door using stolen key
- Forced entry through back door
- Enter through back door using stolen key
- Forced entry through kitchen window
- Observe inhabitants through kitchen window
- Forced entry through garage door
- …

Applying STRIDE systematically: an analogy

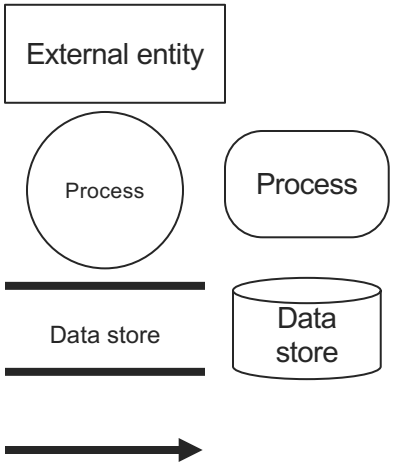
	Forced entry	Stolen key	Observe inhabitants
Door	X	X	
Window	X		X
Garage door	X	X	
Fence	X		X

Possible threats:

- Forced entry through front door
- Enter through front door using stolen key
- Forced entry through back door
- Enter through back door using stolen key
- Forced entry through kitchen window
- Observe inhabitants through kitchen window
- Forced entry through garage door
- …

STRIDE input: data flow diagram (DFD)

- External entity
- Process
- Data store
- Data flow
- Trust boundary – – – – = a place where principals (with different privileges) interact

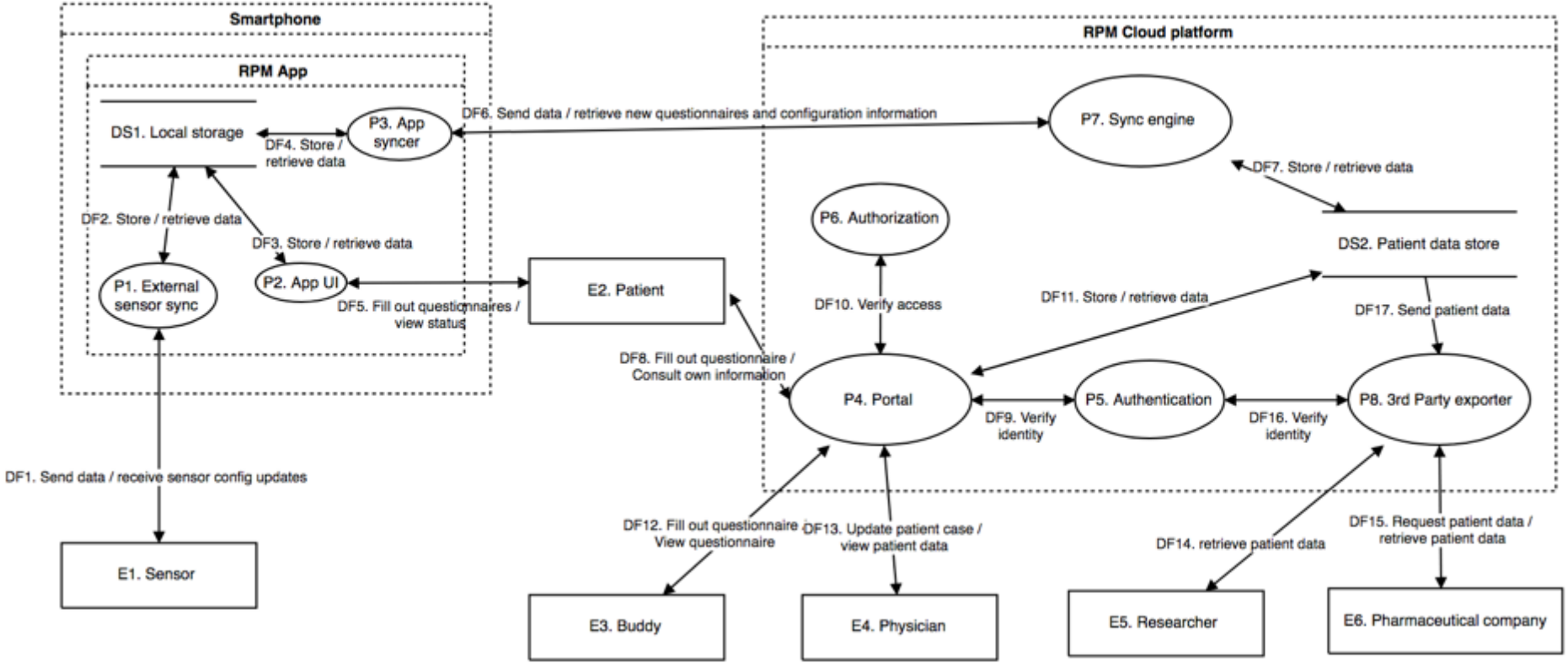


ELEMENT	APPEARANCE	MEANING	EXAMPLES
Process	Rounded rectangle, circle, or concentric circles	Any running code	Code written in C, C#, Python, or PHP
Data flow	Arrow	Communication between processes, or between processes and data stores	Network connections, HTTP, RPC, LPC
Data store	Two parallel lines with a label between them	Things that store data	Files, databases, the Windows Registry, shared memory segments
External entity	Rectangle with sharp corners	People, or code outside your control	Your customer, Microsoft.com

Shostack, A., 2014. Threat Modeling. Wiley.

STRIDE input: data flow diagram (DFD)

Example of a DFD



STRIDE

per element

	S	T	R	I	D	E
External Entity	x		x			
Process	x	x	x	x	x	x
Data Flow		x		x	x	
Data Store		x	?	x	x	

For each DFD element:

For each STRIDE category:

If table contains an ‘x’ at intersection, you’ve found a (potential) threat

STRIDE

per element

	S	T	R	I	D	E
External Entity	x		x			
Process	x	x	x	x	x	x
Data Flow		x		x	x	
Data Store		x	?	x	x	

Element type	Element	Threat
External entity	Sensor	Spoofing of sensor
		Repudiation by sensor
	Patient	Spoofing of patient
		Repudiation by patient
	…	
…		
Data store	Local storage	Tampering with local storage
		Information disclosure through local storage
		…

STRIDE

per element

	S	T	R	I	D	E
External Entity	x		x			
Process	x	x	x	x	x	x
Data Flow		x		x	x	
Data Store		x	?	x	x	

Element type	Element	Threat
External entity	Sensor	Spoofing of sensor
		Repudiation by sensor
	Patient	Spoofing of patient
		Repudiation by patient
	…	
…		
Data store	Local storage	Tampering with local storage
		Information disclosure through local storage
		…

STRIDE

per element

	S	T	R	I	D	E
External Entity	x		x			
Process	x	x	x	x	x	x
Data Flow		x		x	x	
Data Store		x	?	x	x	

Element type	Element	Threat
External entity	Sensor	Spoofing of sensor
		Repudiation by sensor
	Patient	Spoofing of patient
		Repudiation by patient
	…	
…		
Data store	Local storage	Tampering with local storage
		Information disclosure through local storage
		…

STRIDE: threat trees

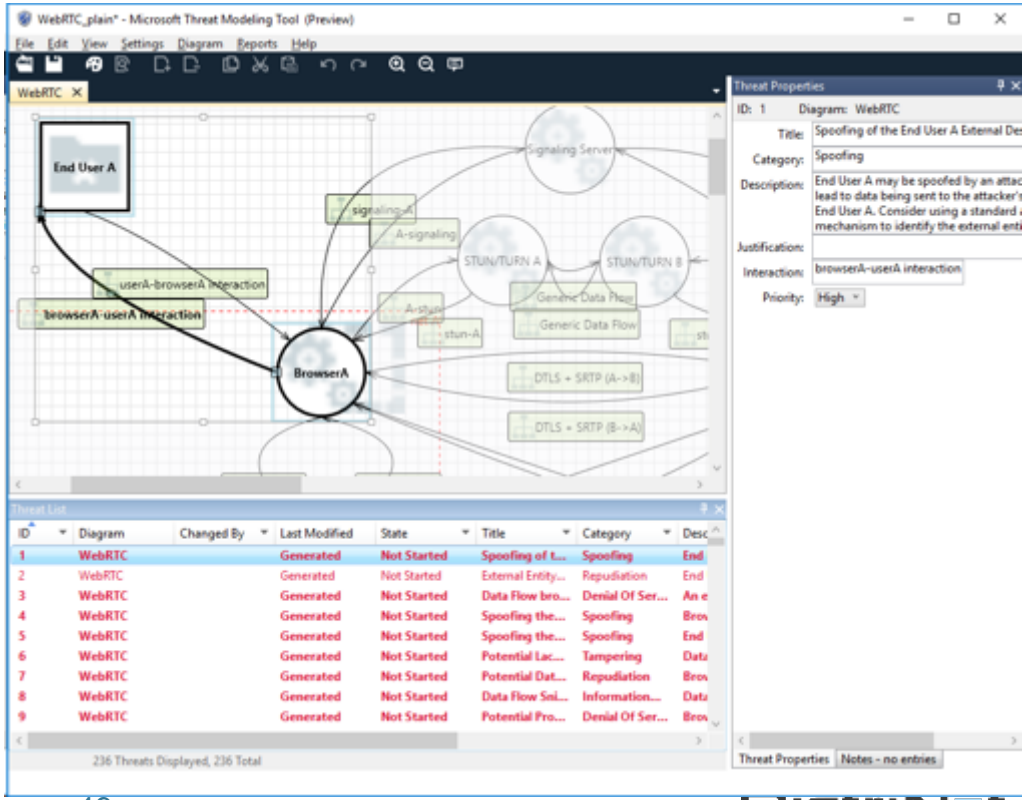
- STRIDE threats are very generic
- Threat tree: refinement of threats

	S	T	R	I	D	E
External Entity	x		x			
Process	x	x	x	x	x	x
Data Flow		x		x	x	
Data Store		x	?	x	x	

Figure B-1: Spoofing an external entity (client)

Microsoft Threat Modeling Tool

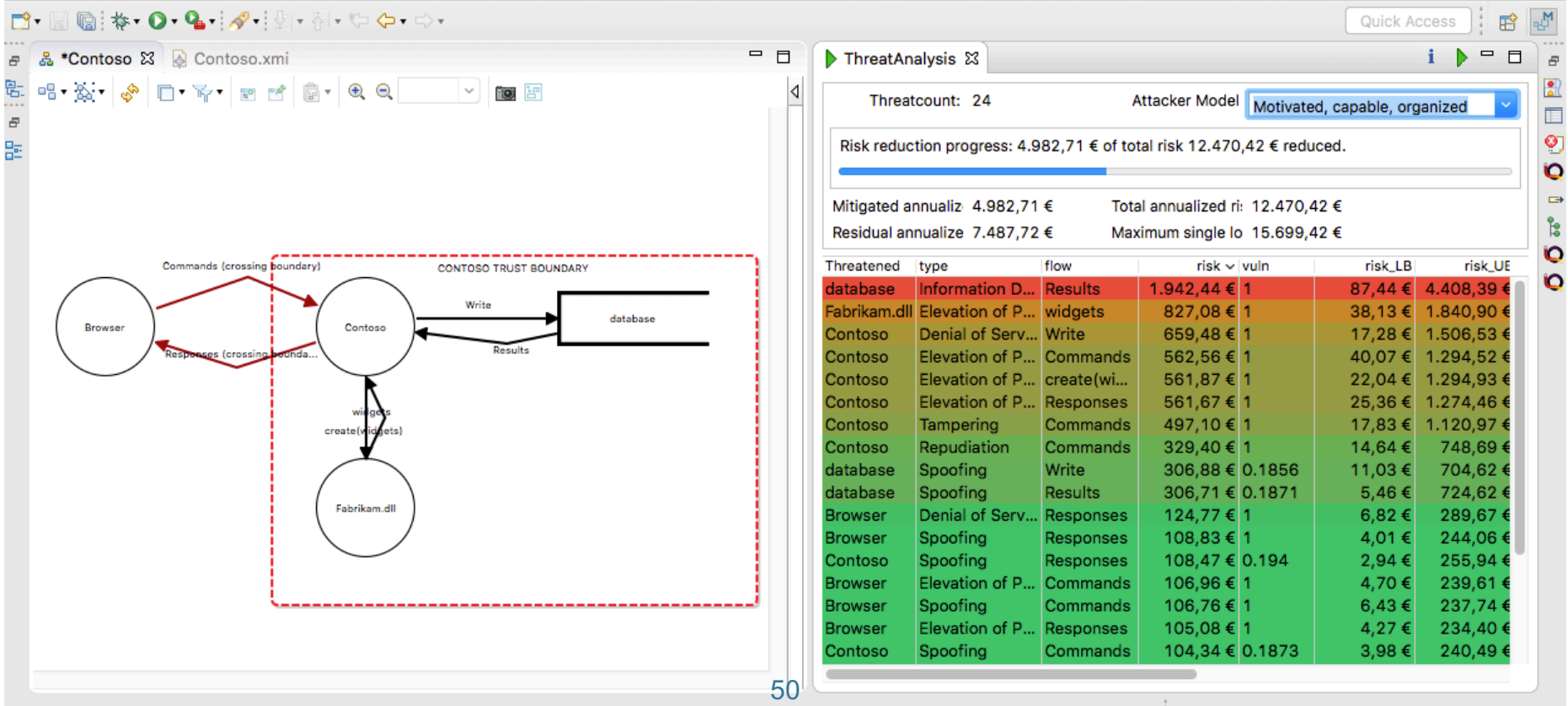
- Catalogs with types of
  - processes, data stores, external entities, data flows
  - Threats
- In practice: generates lots of irrelevant threats



https://blogs.msdn.microsoft.com/secdevblog/2018/09/12/microsoft-threat-modeling-tool-ga-release/

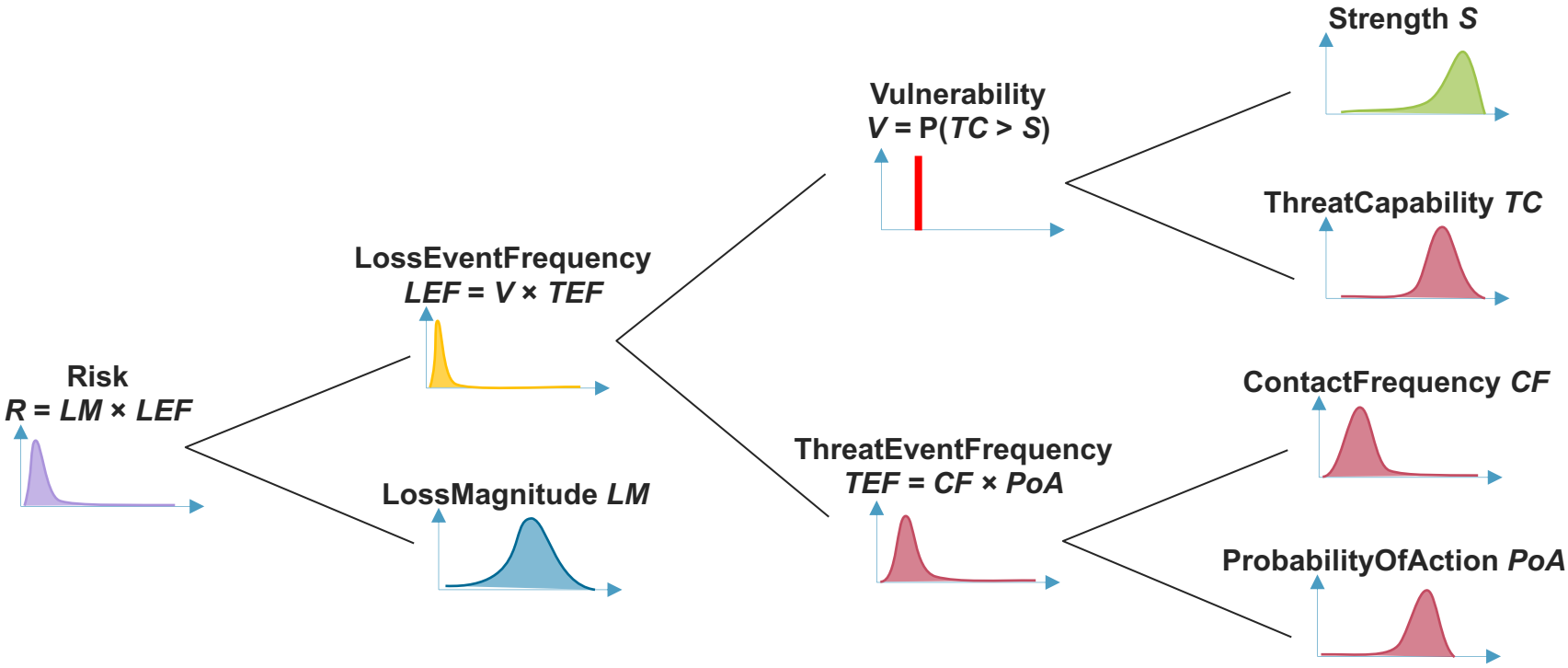
SPARTA (threat modeling + risk analysis)

WiP by Laurens Sion @ DistriNet https://distrinet.cs.kuleuven.be/software/sparta/



Threatened	type	flow	risk	vuln	risk_LB	risk_UE
database	Information D...	Results	1.942,44 €	1	87,44 €	4.408,39 €
Fabrikam.dll	Elevation of P...	widgets	827,08 €	1	38,13 €	1.840,90 €
Contoso	Denial of Serv...	Write	659,48 €	1	17,28 €	1.506,53 €
Contoso	Elevation of P...	Commands	562,56 €	1	40,07 €	1.294,52 €
Contoso	Elevation of P...	create(wi...	561,87 €	1	22,04 €	1.294,93 €
Contoso	Elevation of P...	Responses	561,67 €	1	25,36 €	1.274,46 €
Contoso	Tampering	Commands	497,10 €	1	17,83 €	1.120,97 €
Contoso	Repudiation	Commands	329,40 €	1	14,64 €	748,69 €
database	Spoofing	Write	306,88 €	0.1856	11,03 €	704,62 €
database	Spoofing	Results	306,71 €	0.1871	5,46 €	724,62 €
Browser	Denial of Serv...	Responses	124,77 €	1	6,82 €	289,67 €
Browser	Spoofing	Responses	108,83 €	1	4,01 €	244,06 €
Contoso	Spoofing	Responses	108,47 €	0.194	2,94 €	255,94 €
Browser	Elevation of P...	Commands	106,96 €	1	4,70 €	239,61 €
Browser	Spoofing	Commands	106,76 €	1	6,43 €	237,74 €
Browser	Elevation of P...	Responses	105,08 €	1	4,27 €	234,40 €
Contoso	Spoofing	Commands	104,34 €	0.1873	3,98 €	240,49 €

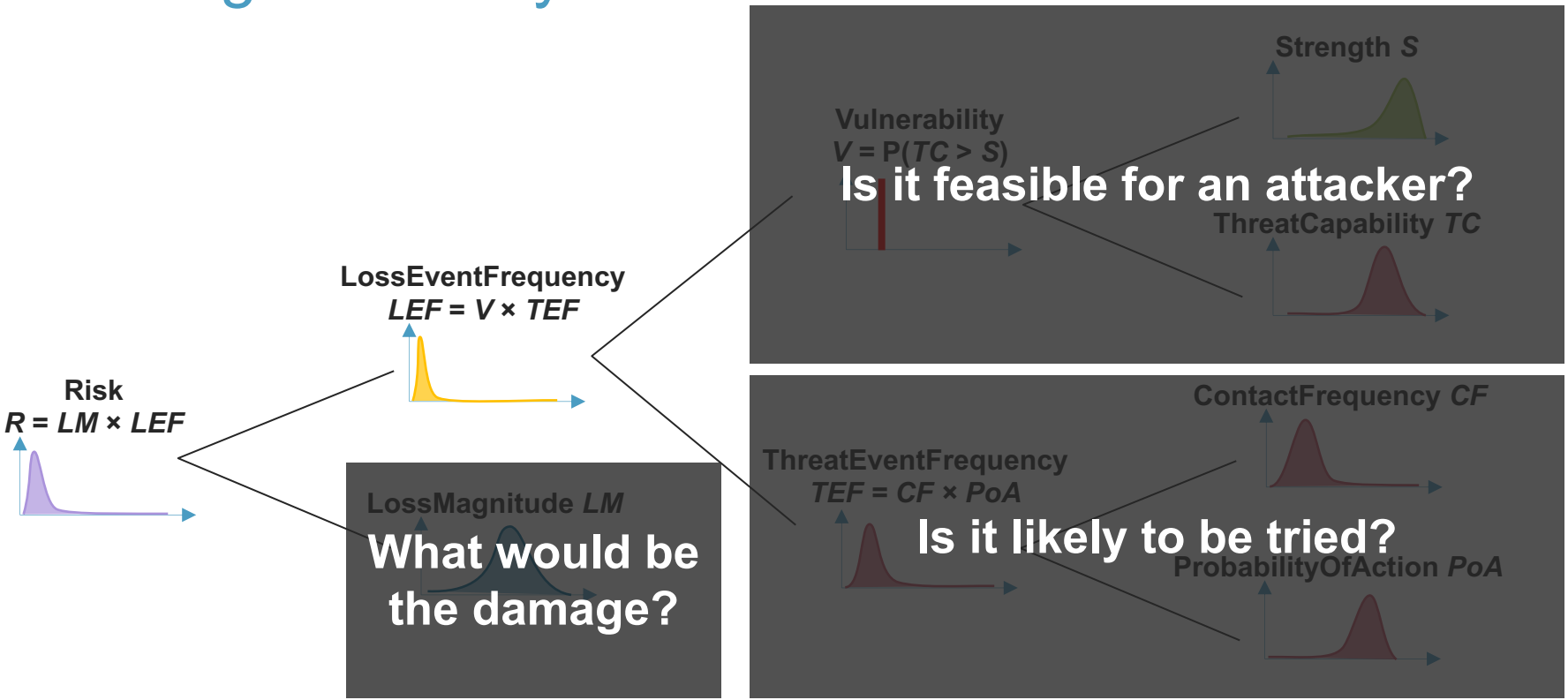
Prioritizing threats by risk: SPARTA



SION, L., YSKOUT, K., VAN LANDUYT, D. AND JOOSEN, W., 2018. Risk-based design security analysis. In *Proceedings of the 1st International Workshop on Security Awareness from Design to Deployment - SEAD '18*. New York, New York, USA: ACM Press, pp. 11–18.

DistriNet

Prioritizing threats by risk: SPARTA



Sion, L., Yskout, K., Van Landuyt, D. and Joosen, W., 2018. Risk-based design security analysis. In *Proceedings of the 1st International Workshop on Security Awareness from Design to Deployment - SEAD '18*. New York, New York, USA: ACM Press, pp. 11–18.

DistriNet

LINDDUN

- “STRIDE for privacy”
  - Linkability, Identifiability, Non-repudiation, Detectability, Disclosure of information, Unawareness, Non-compliance
- Kim Wuyts @ DistriNet
- See https://linddun.org

Threat modeling in practice

Why not defend against everything?

DistriNet

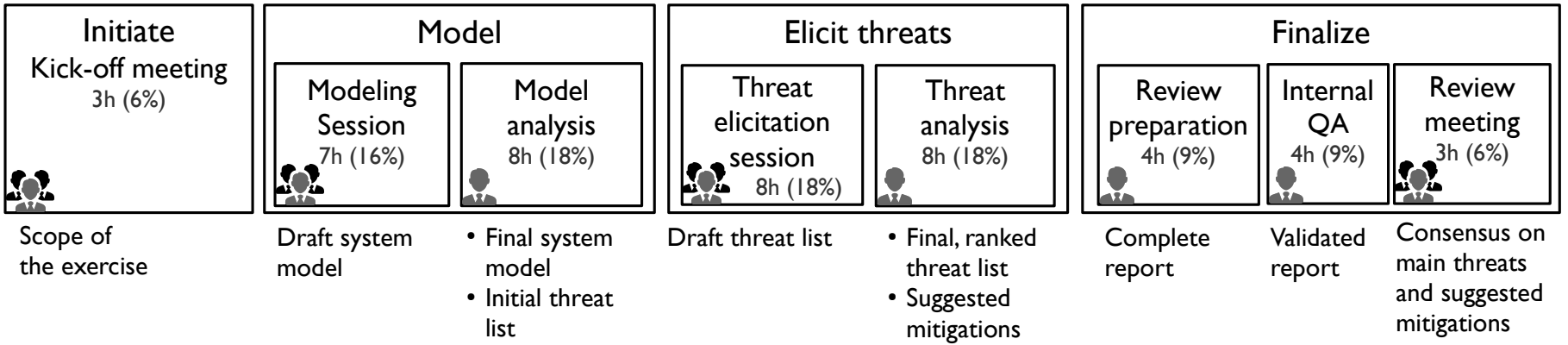
Characteristics of modern software development

- Agile
  - 2-week sprints, most valuable feature first, working prototypes
  - Not much focus on architecture and design (code-first)
- Continuous integration, continuous deployment
  - New software is deployed multiple times per week/day/…
  - Automated testing
- DevOps
  - Integration of developers and operational people (sysadmins)
  - Infrastructure as code, far-reaching automation

Where does threat modeling fit?

Threat modeling in practice

Process of Toreon



Total effort for one project (without background noise):
45 hours

DistriNet

Other challenges for threat modeling adoption

- Management buy-in
  - Requires translating technical threats/risk to **business risk**!
  - **Compliance** requirements might help as well (cfr. safety)
- **Scaling** the process to an entire organization
  - Training
  - Lack of security expertise
- (Lack of) security **culture**
  - Security department is often seen as ‘necessary evil’

It’s not (only) a technical problem, but also a people/resources one!

The end.

Q&A