Threat and Risk Analysis

Riccardo Scandariato

Institute of Software Security, TUHH, Germany

ric***do . scanda***to @ tuhh.de

Master Course “Secure Software Engineering”
Summer Semester 2022

Learning objectives

- Learning fundamentals/terminology of risk analysis

- Introduction to STRIDE

Reading material
See project

- Knowledge repositories of security attacks

What is risk?

Definition from ISO 31000 ISO (2018)

„Effect of uncertainty on objectives“

- Objectives can have different aspects, and can be applied at different levels
- An effect is a deviation from the expected
  - Can be positive or negative (or both)
  - Can result in opportunities and threats
- Risk is usually expressed in terms of risk sources, potential events, their consequences and their likelihood

Key Terminology (for software security)

- **Asset:** An asset is something to which a party assigns value and hence for which the party requires protection
- **Risk:** A risk is the likelihood of an unwanted incident and its consequence for a specific asset
- **Threat:** A threat is a potential cause of an unwanted incident
- **Unwanted Incident:** An unwanted incident is an event that harms or reduces the value of an asset
- **Vulnerability**: A vulnerability is a weakness, flaw or deficiency that opens for, or may be exploited by, a threat to cause harm to or reduce the value of an asset

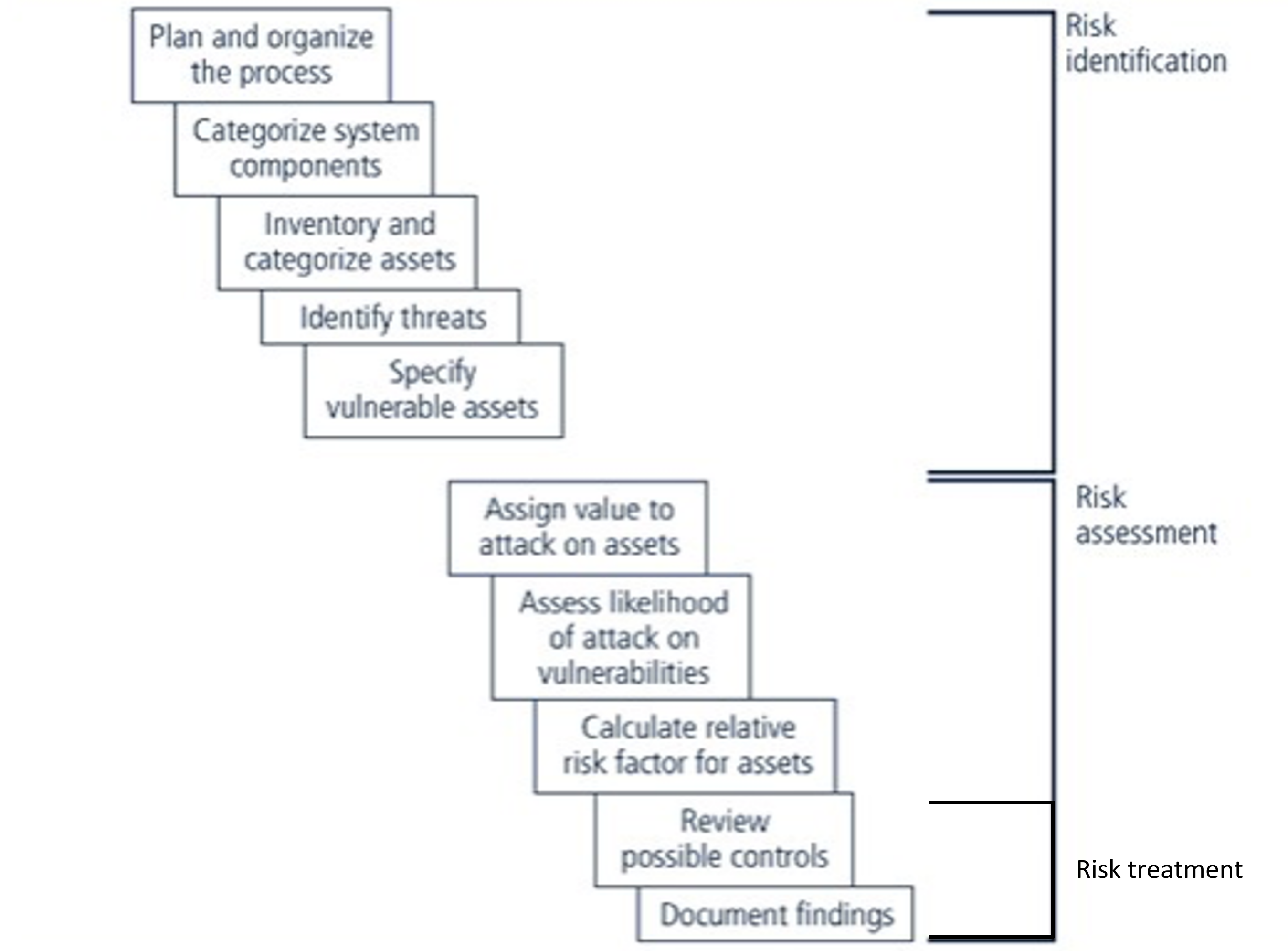
Risk Management (in secure software engineering)

- Risk management is primarily concerned with identifying and assessing the risks for the software system when in operation
- As well as treating those risks to reduce their impact on the software and its environment
- Questions
  - *What can go wrong?*
  - *What is the likelihood of it going wrong?*
  - *What will the damage be?*
  - *What can we do about it?*

Risk management can be applied also to:

- security risks for the IT infrastructure
- security risks for the software development infrstructure (supply chain),
- Etc…

Components of Risk Management


Plan and organize the process
Categorize system components
Inventory and categorize assets
Identify threats
Specify vulnerable assets
Assign value to attack on assets
Assess likelihood of attack on vulnerabilities
Calculate relative risk factor for assets
Review possible controls
Document findings
Risk identification
Risk assessment
Risk treatment

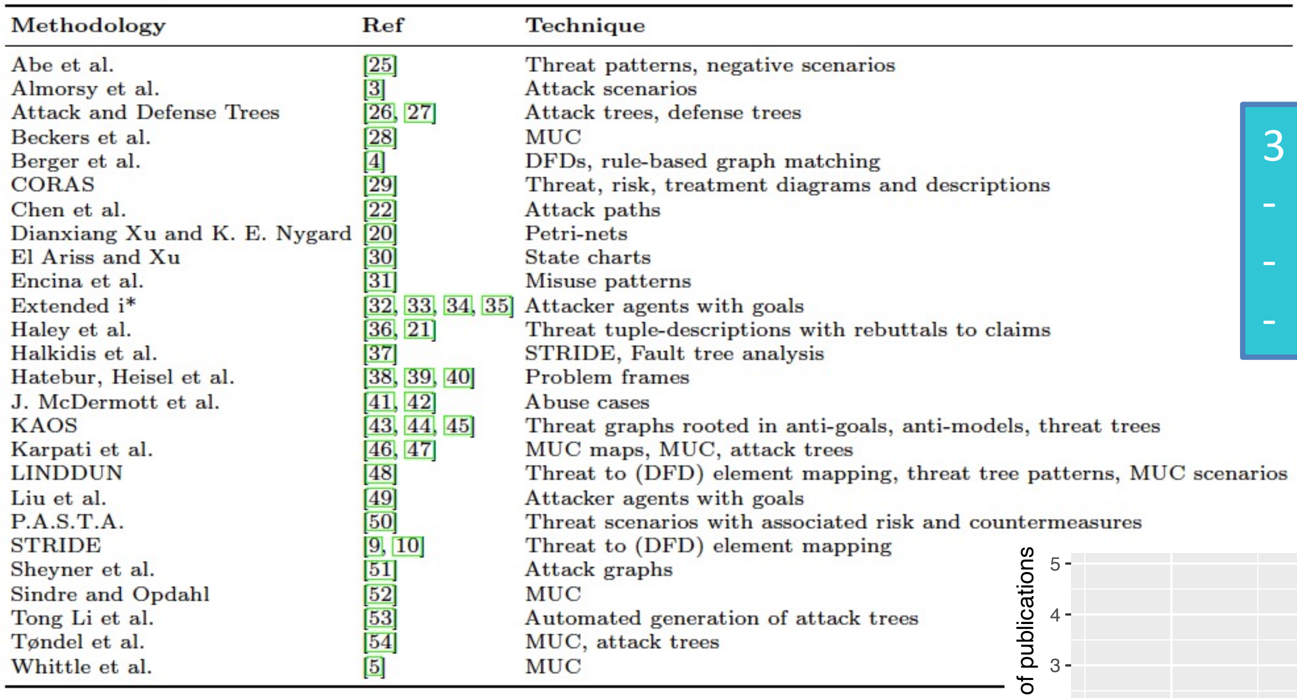
Whitman, Michael E., and Herbert J. Mattord. "Principles of Information Security." (2007).

TECHNIQUES FOR THREAT & RISK IDENTIFICATION

Several TARA techniques available

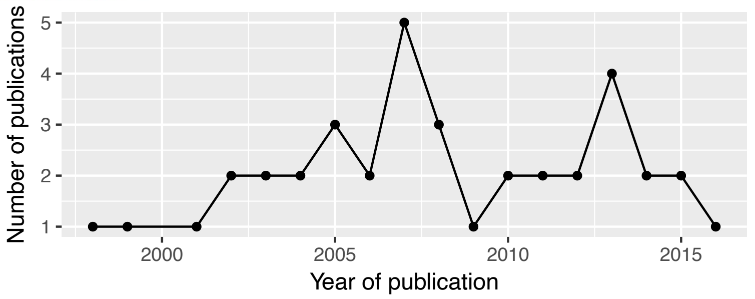
Katja Tuma, Gul Calikli, Riccardo Scandariato, **Threat analysis of software systems: A systematic literature review**, Journal of Systems and Software, 2018

38 papers grouped into 26 techniques



Methodology	Ref	Technique
Abe et al.	[25]	Threat patterns, negative scenarios
Almorsy et al.	[3]	Attack scenarios
Attack and Defense Trees	[26, 27]	Attack trees, defense trees
Beckers et al.	[28]	MUC
Berger et al.	[4]	DFDs, rule-based graph matching
CORAS	[29]	Threat, risk, treatment diagrams and descriptions
Chen et al.	[22]	Attack paths
Dianxiang Xu and K. E. Nygard	[20]	Petri-nets
El Ariss and Xu	[30]	State charts
Encina et al.	[31]	Misuse patterns
Extended i*	[32, 33, 34, 35]	Attacker agents with goals
Haley et al.	[36, 21]	Threat tuple-descriptions with rebuttals to claims
Halkidis et al.	[37]	STRIDE, Fault tree analysis
Hatebur, Heisel et al.	[38, 39, 40]	Problem frames
J. McDermott et al.	[41, 42]	Abuse cases
KAOS	[43, 44, 45]	Threat graphs rooted in anti-goals, anti-models, threat trees
Karpati et al.	[46, 47]	MUC maps, MUC, attack trees
LINDDUN	[48]	Threat to (DFD) element mapping, threat tree patterns, MUC scenarios
Liu et al.	[49]	Attacker agents with goals
P.A.S.T.A.	[50]	Threat scenarios with associated risk and countermeasures
STRIDE	[9, 10]	Threat to (DFD) element mapping
Sheyner et al.	[51]	Attack graphs
Sindre and Opdahl	[52]	MUC
Tong Li et al.	[53]	Automated generation of attack trees
Tøndel et al.	[54]	MUC, attack trees
Whittle et al.	[5]	MUC

3 groups:
- Risk centric
- Software centric
- Attack centric



Risk-centric (e.g., CORAS)

See next lecure

- Focus on assets and their value to the organization
- Higher-level model of the system (top-level functionality, assets)
- Aim at estimating the financial loss for the organization in case of threat occurrence
- And finding the appropriate mitigations to minimize it
- Method: brainstorming sessions
- **Assets dictate the priority of elicited security requirements**

https://stock.adobe.com/

M. S. Lund, B. Solhaug, K. Stlen, A guided tour of the coras method, in: Model-Driven Risk Analysis, Springer, 2011, pp. 23-43.

Software-centric (e.g., STRIDE)

- Focus is the software system under analysis and its technical organization (components)
- Architectural threat analysis

- Important to include developers in the analysis for 2 reasons:
  – Accurate model of the software
  – Developers learn about assets, risk analysis

- See later in this lecture

A. Shostack, Threat modeling: Designing for security, John Wiley & Sons, 2014.

Attack-centric (e.g. Abe et al.)

- Focus the analysis around the hostility of the environment
- Put emphasis on identifying attacker profiles and attack complexity for exploiting any system vulnerability
- Main objectives
  - Achieve high threat coverage
  - identify appropriate threat migitations

https://stock.adobe.com/

T. Abe, S. Hayashi, M. Saeki, Modeling security threat patterns to derive negative scenarios, 20th Asia-Pacic Software Engineering Conference (APSEC), 2013

RISK & THREAT ASSESSMENT AND PRIORITIZATION

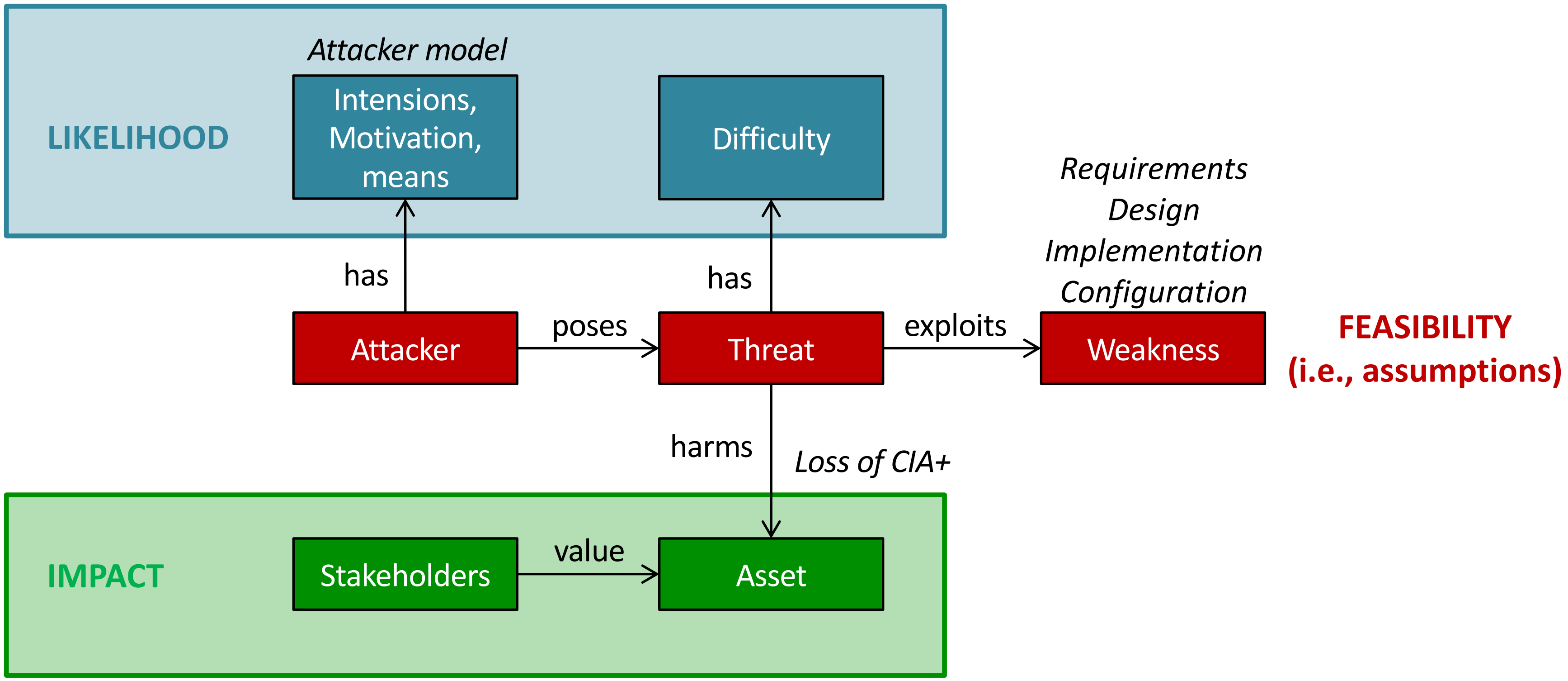
Risk value (a.k.a. rating)

- **Goal**: attach an importance value to risks
- **Why**: can be used to prioritize risks and decide on treatment

- Quantitative & qualitative methods for assessing risk, but in general:

$$\boxed{\textbf{Risk value = f(impact, likelihood)}}$$

Threats and risk analysis
ISO/IEC 15408 (Common Criteria)



Starting point: Impact and Likelihood

- Impact
  - How much value to the stakeholder is involved (e.g., loss of assets, harm of system mission, injury of humans)
  - Loss of value due to a vulnerability/weakness and the resulting successful compromise
- Likelihood
  - How hard it is to mount the attack (nature of the weakness and existence and effectiveness of controls)
  - How motivated and skillful the attacker is

Quantitative approaches (with examples)

Idea: Decompose risk into "parameters" and use a formula

Ranking → **categorizing**. E.g., "Threats with overall ratings between 100 and 50 are classified as **high risk**, between 50 and 10 are classified as **medium risk** and between 10 and 1 as **low risk**"
SP-800-30 -- Guide for Conducting Risk Assessments

Ranking with DREAD (from Microsoft)

```
risk = dp + r + e + a + di
```

- Damage potential (dp)
  - If the threat exploit is successful, how much damage will be caused?
- Reproducibility (r)
  - How easy is it to reproduce the threat exploit? What is the cost to the attacker once he has a working exploit for the problem?
- Exploitability (e)
  - What is needed to exploit the threat? What is the cost to develop an exploit for the problem?
- Affected users (a)
  - What users are actually affected if an exploit were to be widely available?
- Discoverability (di)
  - How easy is it to discover a threat?

Where is the impact in DREAD?

Ranking with OCTAVE

$$risk = w_r * reputation + w_f * financialloss + w_p * productivity + w_s * safety + w_l * legalpenalties + w_o * others$$

- Focus on impact (probability is optional)

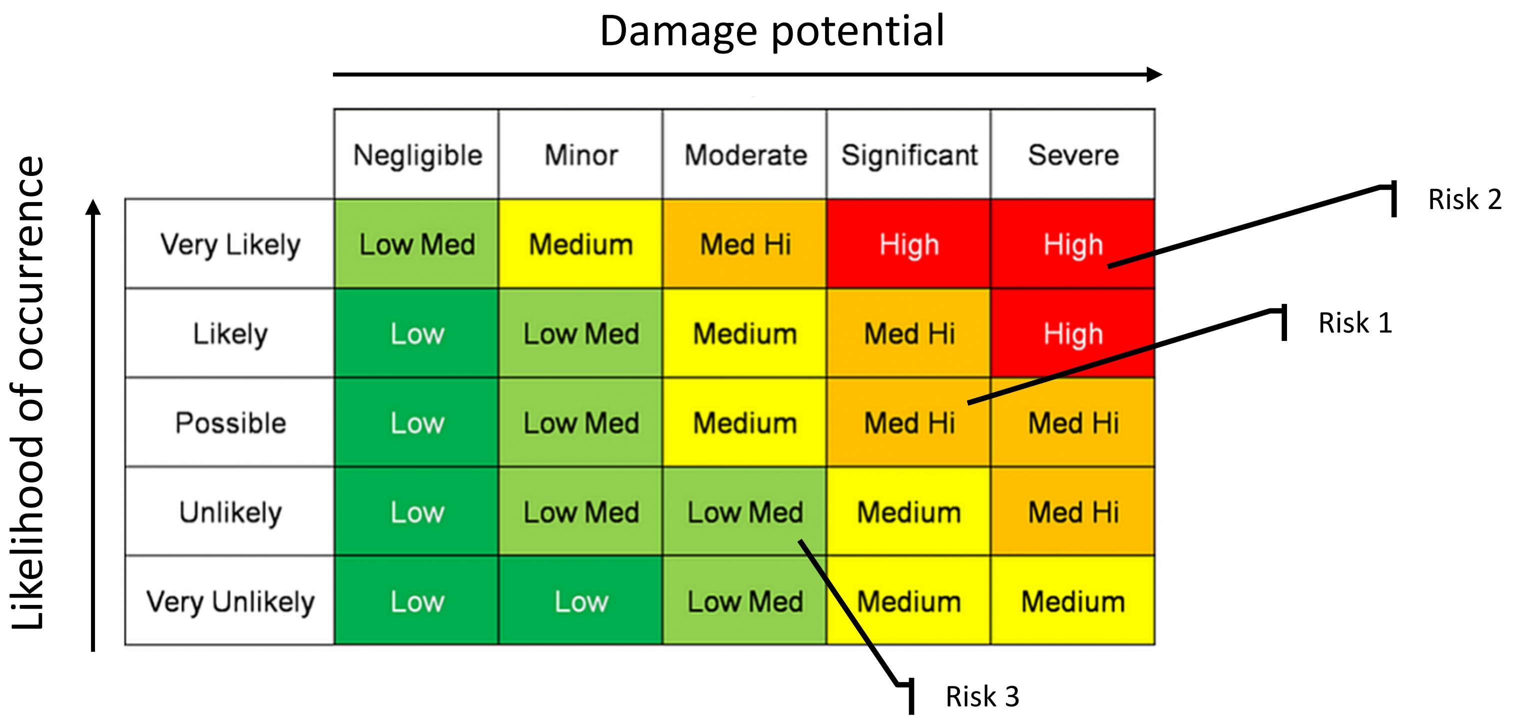
Reflection: garbage in garbage out?

- Subjectivity in the input parameters is still present

- Sensitivity of the formula to the input parameters needs to be studied

Qualitative methods based on „heat maps“

Risk Assessment Matrix - Example



	Negligible	Minor	Moderate	Significant	Severe
Very Likely	Low Med	Medium	Med Hi	High	High
Likely	Low	Low Med	Medium	Med Hi	High
Possible	Low	Low Med	Medium	Med Hi	Med Hi
Unlikely	Low	Low Med	Low Med	Medium	Med Hi
Very Unlikely	Low	Low	Low Med	Medium	Medium

Study at home

Qualitative Risk Assessment

- Risk is defined in **more subjective and general** terms such as high, medium, and low
- Qualitative assessments depend more on the expertise, experience, and judgment of those conducting the assessment
- Useful to adequately communicate the assessment to the organization's management

Study at home

Risk Assessment Matrix

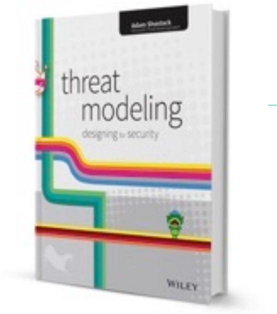
- A **risk assessment matrix** or **risk control matrix** is a tool used during the risk assessment stage. It's used to identify and document the possibility of risks, as well as to assess the possible harm or disruption that such risks could create.
- Risk assessment matrix is also a **visual depiction** of the risk analysis that categorises risks according to their likelihood, impact, and overall severity

THREAT ANALYSIS WITH STRIDE

Threat Analysis (a.k.a. Modeling)

- **Threat Model:** *Process that reviews the security of any system, identifies problem areas, and determines the risk associated with each area*

- Threat Modeling is Iterative (continuous)



STRIDE

Security threat assessment

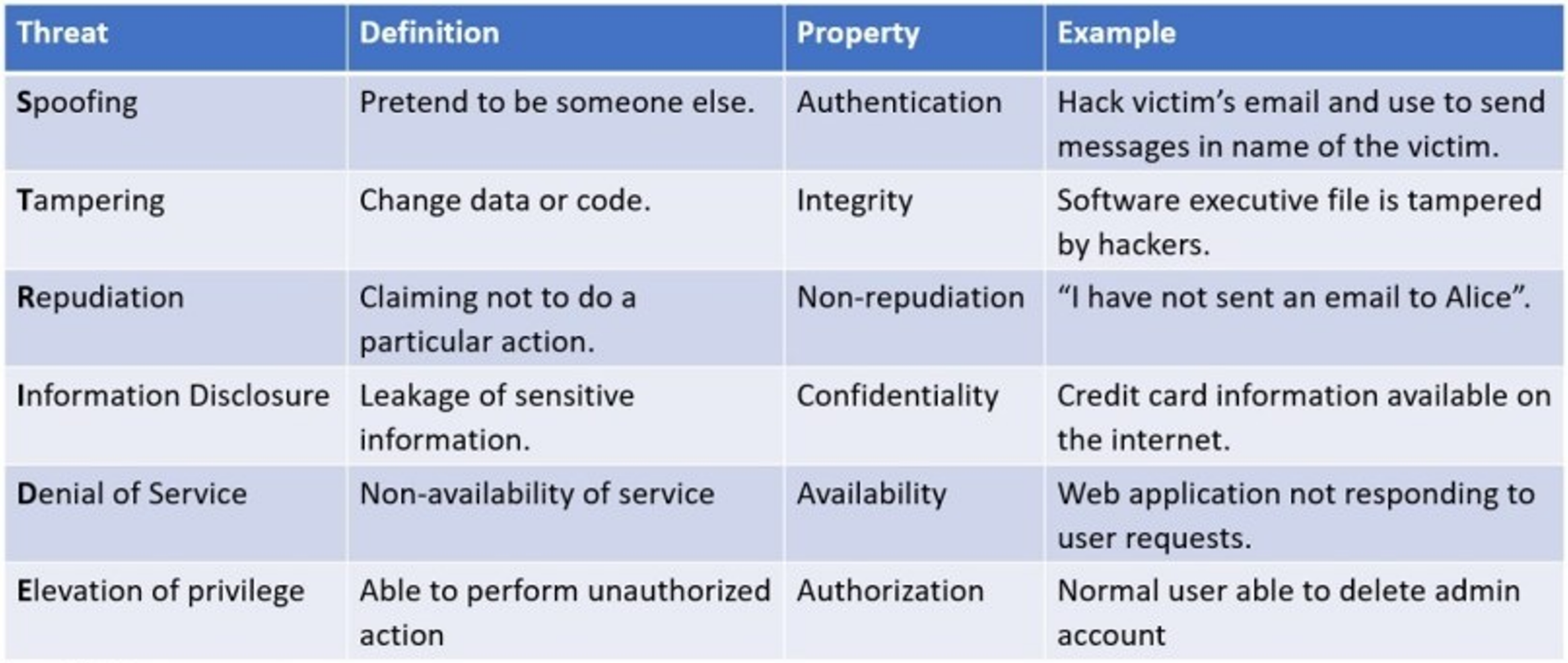
- Systematic approach for threat identification
- Threats are organized into categories, it terms of what attacker is trying to achieve
- STRIDE is a **mnemonic**

– Spoofing	(e.g., *impersonate legit user*)
– Tampering	(e.g., *defacing web site*)
– Repudiation	(e.g., *it wasn't me*)
– Information disclosure	(e.g., *Heartbleed*)
– Denial of service	(e.g., *flooding*)
– Elevation of privilege	(e.g., *running as root*)

A. Shostack, Threat Modeling : Designing for Security, Wiley 2014

STRIDE categories

Threat ➔ Property ➔ Countermeasure



Threat	Definition	Property	Example
Spoofing	Pretend to be someone else.	Authentication	Hack victim's email and use to send messages in name of the victim.
Tampering	Change data or code.	Integrity	Software executive file is tampered by hackers.
Repudiation	Claiming not to do a particular action.	Non-repudiation	"I have not sent an email to Alice".
Information Disclosure	Leakage of sensitive information.	Confidentiality	Credit card information available on the internet.
Denial of Service	Non-availability of service	Availability	Web application not responding to user requests.
Elevation of privilege	Able to perform unauthorized action	Authorization	Normal user able to delete admin account

Methodology

- Define users and realistic use scenarios
- Gather assumptions

Model-based

1. **Model** the system w/ DFD diagram (assets)
2. **Map** STRIDE to DFD element types
3. **Refine** threats via threat tree patterns
4. **Document** the threats (e.g., as MUCs)

Knowledge-based

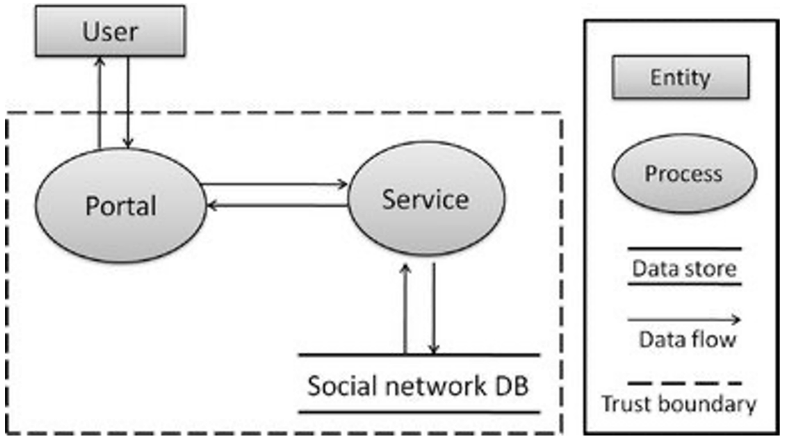
- Assign priority via risk analysis (to counter threat explosion problem)
- Draft mitigation associated to threats

Profiling the Application

- **Where will the application be deployed**
  DMZ/Internal – complete end to end scenarios
- **Who will be the Users (Actors)?**
  Customers, sales agents, public users, administrators, DBAs
- **What are the Data Elements?**
  User account data, credit card info, patient information
- **What rights will the actors have?**
  Create, Read, Update, Delete
- **What Technologies will be used?**
  OS, Web/App Servers, Databases, Architectures (SOA/EJB)
- **Programming Language?**
- **What security mechanisms applied?**

Data Flow Diagram (DFD)

- A data-flow diagram is a way of representing a flow of data through a process or a system.
- DFDs also provide information about the outputs and inputs of each entity and the process itself.
- Shows all relevant steps data goes through



DFD of a Social Network application (1)

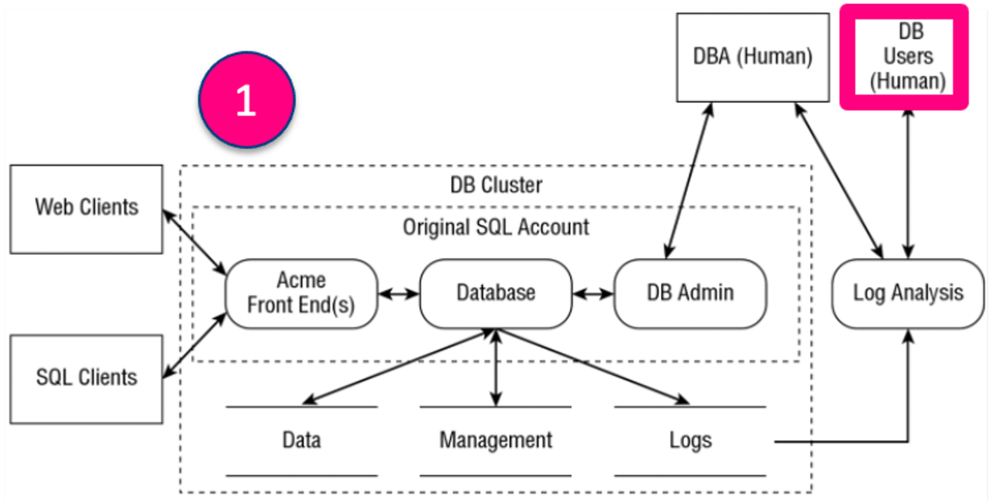
(1) Deng, M., Wuyts, K., Scandariato, R., Preneel, B., & Joosen, W. (2011). A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements. Requirements Engineering, 16(1), 3-32.

Study at home

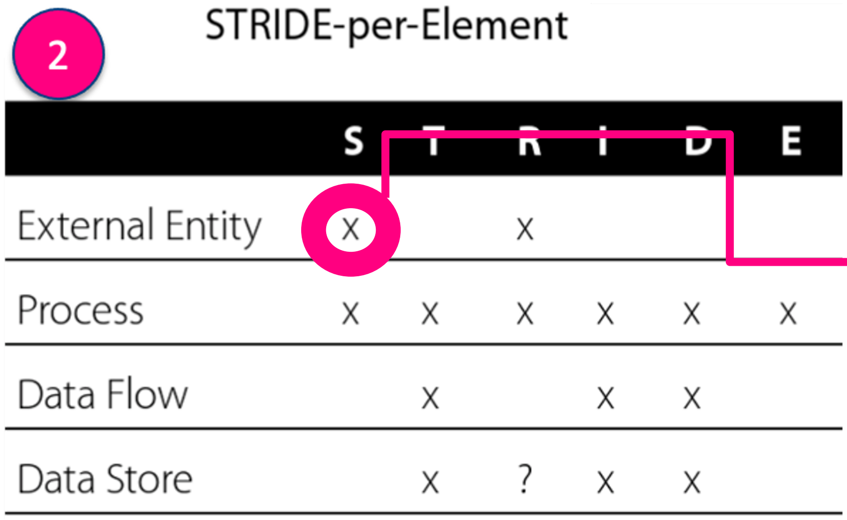
DFD Elements

- Process
  - A process is a unit of work that operates on the data
- Data flow
  - A data flow is a named flow of data through a system of processes
- Data store
  - A data store is a logical repository of data
- External entity
  - An external agent is a source or destination of data

STRIDE in Action



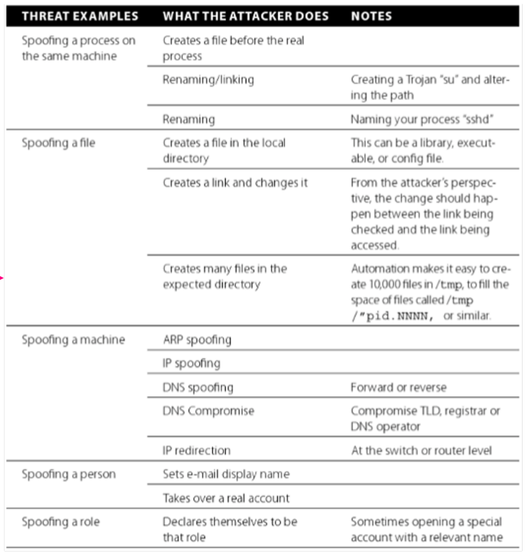
2



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	

3



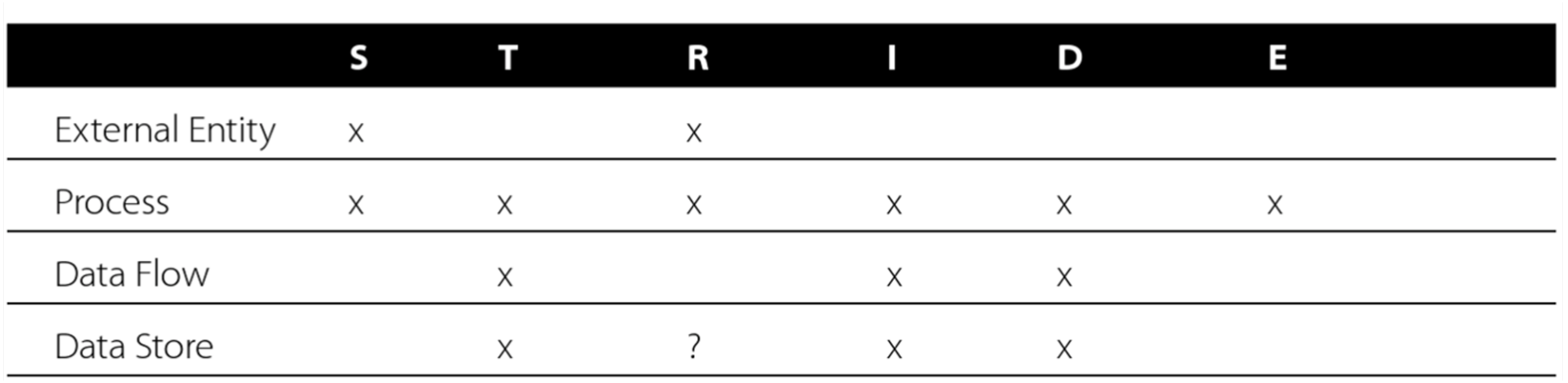
THREAT EXAMPLES	WHAT THE ATTACKER DOES	NOTES
Spoofing a process on the same machine	Creates a file before the real process	
	Renaming/linking	Creating a Trojan "su" and altering the path
	Renaming	Naming your process "sshd"
Spoofing a file	Creates a file in the local directory	This can be a library, executable, or config file.
	Creates a link and changes it	From the attacker's perspective, the change should happen between the link being checked and the link being accessed.
	Creates many files in the expected directory	Automation makes it easy to create 10,000 files in `/tmp`, to fill the space of files called `/tmp /"pid.NNNN`, or similar.
Spoofing a machine	ARP spoofing	
	IP spoofing	
	DNS spoofing	Forward or reverse
	DNS Compromise	Compromise TLD, registrar or DNS operator
	IP redirection	At the switch or router level
Spoofing a person	Sets e-mail display name	
	Takes over a real account	
Spoofing a role	Declares themselves to be that role	Sometimes opening a special account with a relevant name

Spoofing Threats

DFD Elements to Threat Categories

- Only a subset of threat categories are relevant
- Not “everything” can go wrong

  (unless it’s a process)

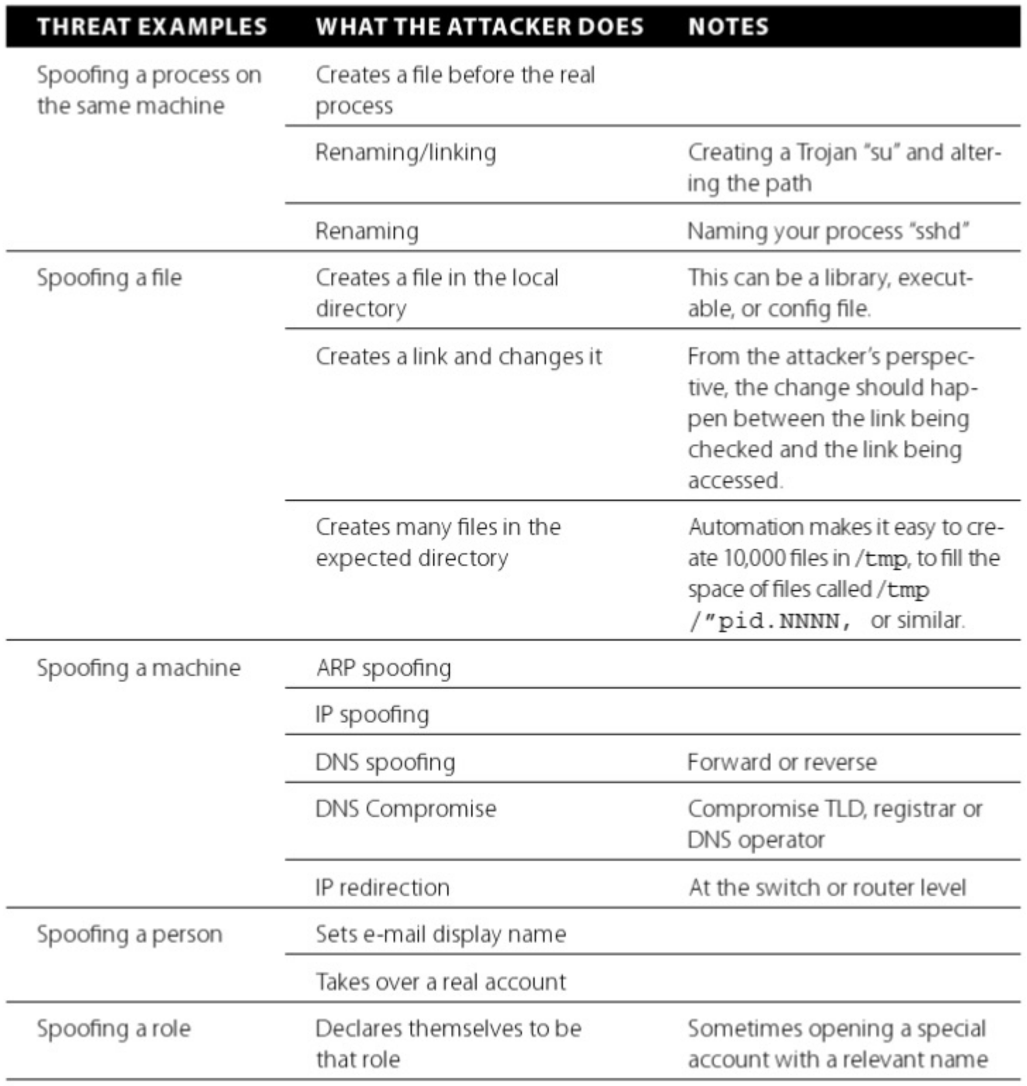


	S	T	R	I	D	E
External Entity	x		x			
Process	x	x	x	x	x	x
Data Flow		x		x	x	
Data Store		x	?	x	x	

Study at home

Spoofing threats – Examples

Table 3-2: Spoofing Threats



THREAT EXAMPLES	WHAT THE ATTACKER DOES	NOTES
Spoofing a process on the same machine	Creates a file before the real process	
	Renaming/linking	Creating a Trojan "su" and altering the path
	Renaming	Naming your process "sshd"
Spoofing a file	Creates a file in the local directory	This can be a library, executable, or config file.
	Creates a link and changes it	From the attacker's perspective, the change should happen between the link being checked and the link being accessed.
	Creates many files in the expected directory	Automation makes it easy to create 10,000 files in `/tmp`, to fill the space of files called `/tmp/"pid.NNNN,` or similar.
Spoofing a machine	ARP spoofing	
	IP spoofing	
	DNS spoofing	Forward or reverse
	DNS Compromise	Compromise TLD, registrar or DNS operator
	IP redirection	At the switch or router level
Spoofing a person	Sets e-mail display name	
	Takes over a real account	
Spoofing a role	Declares themselves to be that role	Sometimes opening a special account with a relevant name

Given by the methodology to get the novice analyst started

Tool support

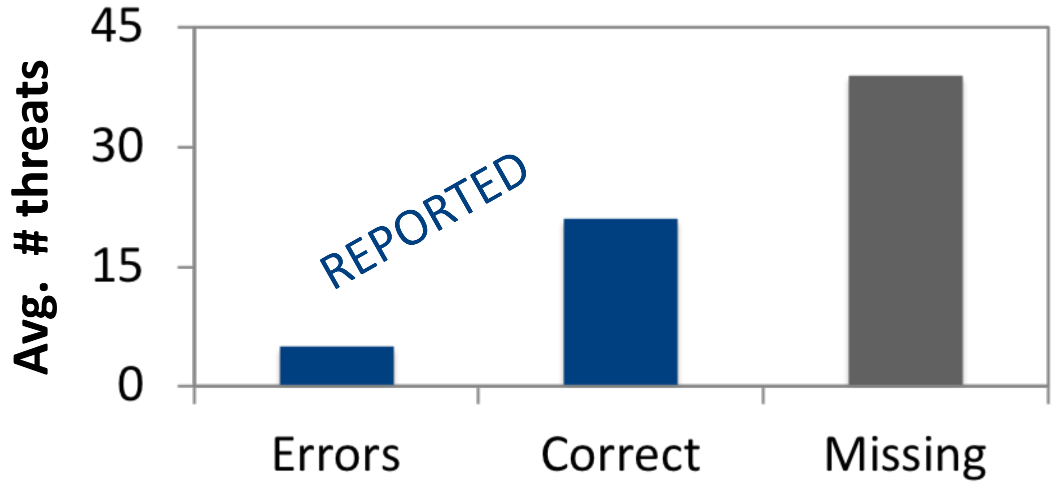
- Microsoft Threat Modeling Tool
  - Draw DFD
  - Add a lot of properties to the DFD elements
  - Tool generates threats
  - Your mileage may vary!

- OWASP Threat Dragon
  - Open source version

Is STRIDE effective?

R. Scandariato, K. Wuyts, W. Joosen, *A descriptive study of Microsoft's threat modeling technique*, Requirements Engineering), 2015

☺

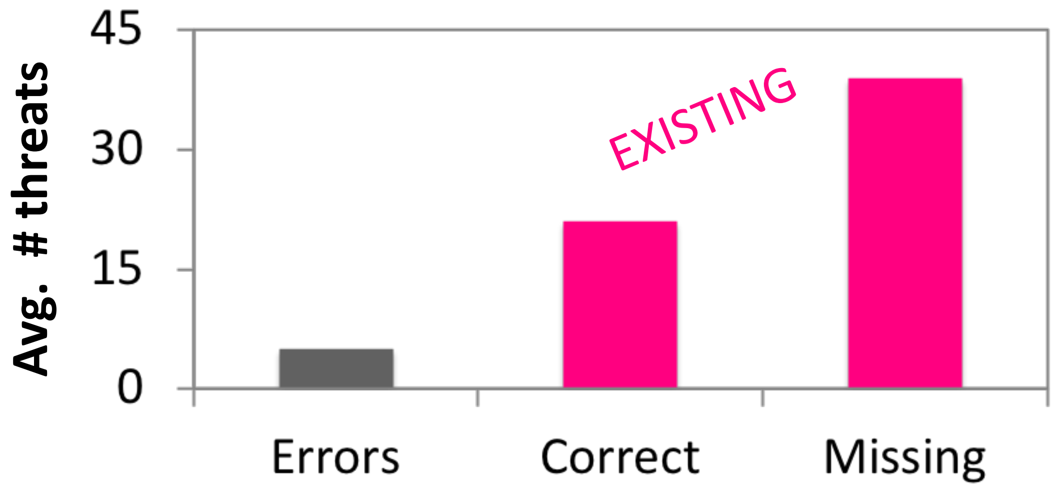


81 %

Correctness

Good guidance

☹



36 %

Completeness

Problem of threat explosion

KNOWLEDGE REPOSITORIES OF SECURITY ATTACK

Thinking to threat scenarios is hard

- Especially for novices (non security experts)
- Knowledge of "what attackers can do" is needed
  - Adversary tactics, techniques and procedures
  - Knowledge of vulnerabilities and how they can be exploited
- This helps finding threats more efficiently

- **Collections of attack types** are available
  1. To provide inspiration (mainly for novices)
  2. When performing a gap analysis (also for experts)

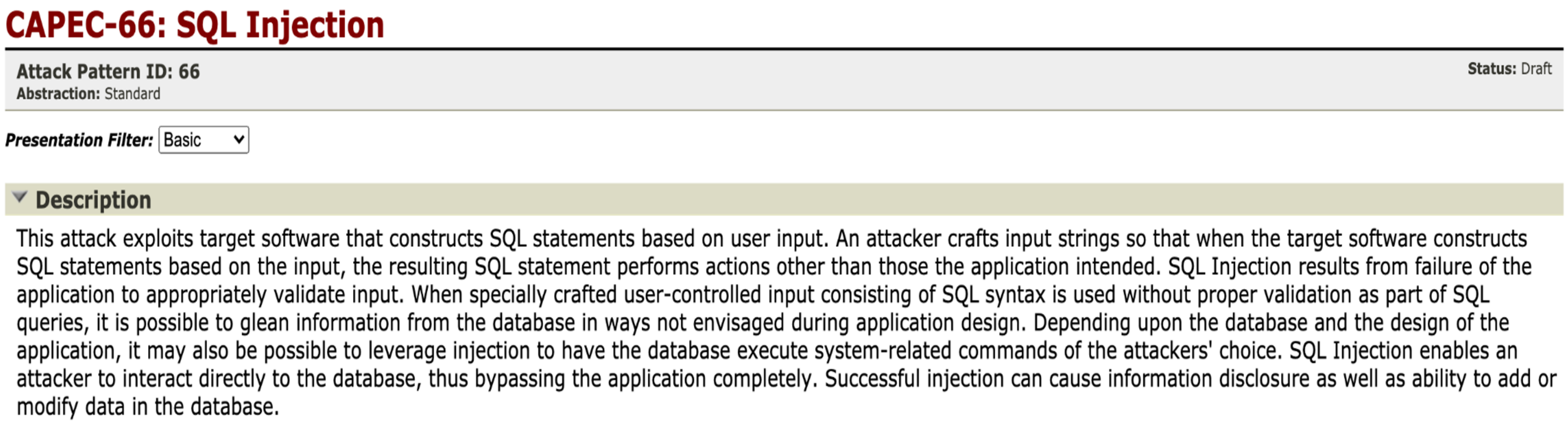
- Known ones: ATT&K, **CAPEC** (both from MITRE)

MITRE‘s CAPEC

- Common Attack Pattern Enumerations and Classifications
- CAPEC “is a comprehensive dictionary and classification taxonomy of known attacks that can be used by analysts, developers, testers, and educators to advance community understanding and enhance defenses”[1]
- CAPEC provides a publicly available catalog of common attack patterns (**with quite some focus on software security**) that helps users understand how adversaries exploit weaknesses in applications (i.e., ideal for **application threat modeling**)

CAPEC Attack Patterns

- Descriptions of the common approaches employed by adversaries to exploit known weaknesses in cyber-enabled systems
- Captures knowledge about how specific parts of an attack are designed and executed, and gives guidance on ways to mitigate the attack's effectiveness
- Contain an “execution flow” — step-by-step instructions for an adversary to explore for potential targets, experiment with their assets and defensive mechanisms, if any, and then to carry out the exploit



CAPEC-66: SQL Injection

Attack Pattern ID: 66
Abstraction: Standard
Status: Draft

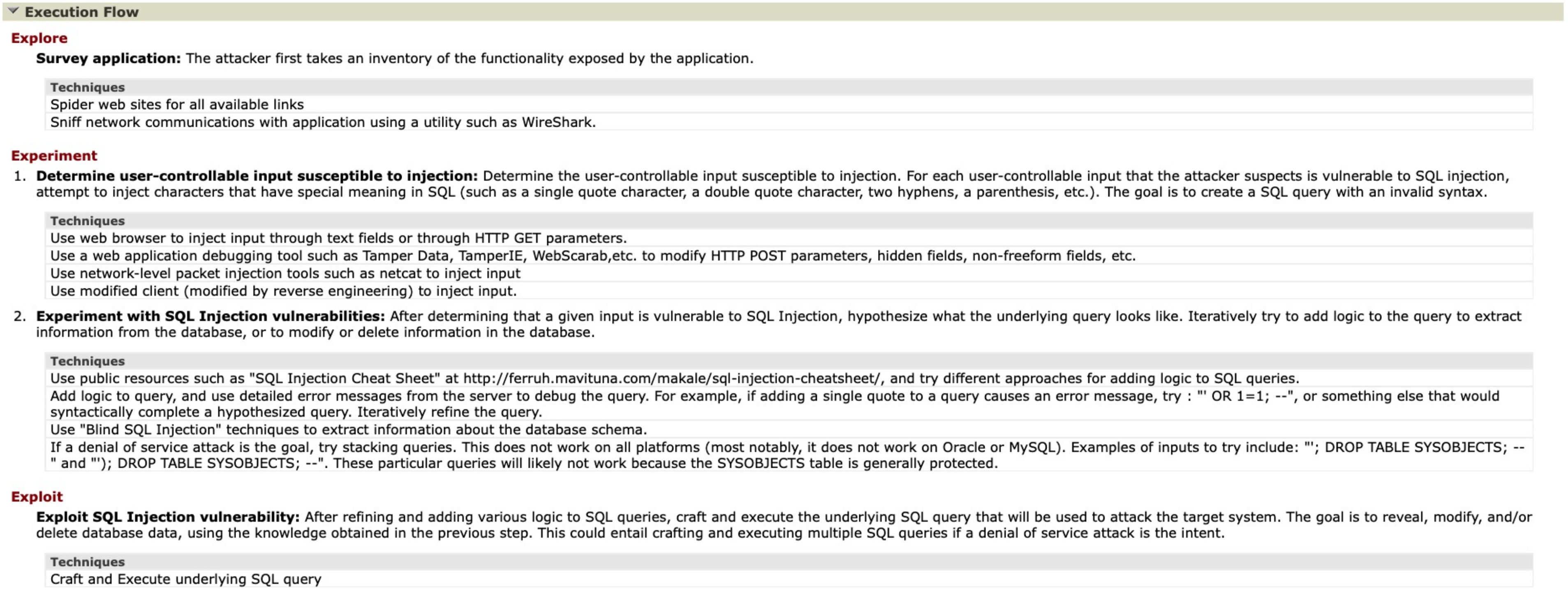
Presentation Filter: Basic

Description

This attack exploits target software that constructs SQL statements based on user input. An attacker crafts input strings so that when the target software constructs SQL statements based on the input, the resulting SQL statement performs actions other than those the application intended. SQL Injection results from failure of the application to appropriately validate input. When specially crafted user-controlled input consisting of SQL syntax is used without proper validation as part of SQL queries, it is possible to glean information from the database in ways not envisaged during application design. Depending upon the database and the design of the application, it may also be possible to leverage injection to have the database execute system-related commands of the attackers' choice. SQL Injection enables an attacker to interact directly to the database, thus bypassing the application completely. Successful injection can cause information disclosure as well as ability to add or modify data in the database.

Source: https://capec.mitre.org/data/definitions/66.html

Execution flow – Example for SQL Injection



▼ Execution Flow

Explore

Survey application: The attacker first takes an inventory of the functionality exposed by the application.

Techniques
Spider web sites for all available links
Sniff network communications with application using a utility such as WireShark.

Experiment

1. **Determine user-controllable input susceptible to injection:** Determine the user-controllable input susceptible to injection. For each user-controllable input that the attacker suspects is vulnerable to SQL injection, attempt to inject characters that have special meaning in SQL (such as a single quote character, a double quote character, two hyphens, a parenthesis, etc.). The goal is to create a SQL query with an invalid syntax.

Techniques
Use web browser to inject input through text fields or through HTTP GET parameters.
Use a web application debugging tool such as Tamper Data, TamperIE, WebScarab,etc. to modify HTTP POST parameters, hidden fields, non-freeform fields, etc.
Use network-level packet injection tools such as netcat to inject input
Use modified client (modified by reverse engineering) to inject input.

2. **Experiment with SQL Injection vulnerabilities:** After determining that a given input is vulnerable to SQL Injection, hypothesize what the underlying query looks like. Iteratively try to add logic to the query to extract information from the database, or to modify or delete information in the database.

Techniques
Use public resources such as "SQL Injection Cheat Sheet" at http://ferruh.mavituna.com/makale/sql-injection-cheatsheet/, and try different approaches for adding logic to SQL queries.
Add logic to query, and use detailed error messages from the server to debug the query. For example, if adding a single quote to a query causes an error message, try : "' OR 1=1; --", or something else that would syntactically complete a hypothesized query. Iteratively refine the query.
Use "Blind SQL Injection" techniques to extract information about the database schema.
If a denial of service attack is the goal, try stacking queries. This does not work on all platforms (most notably, it does not work on Oracle or MySQL). Examples of inputs to try include: "'; DROP TABLE SYSOBJECTS; --" and "'); DROP TABLE SYSOBJECTS; --". These particular queries will likely not work because the SYSOBJECTS table is generally protected.

Exploit

Exploit SQL Injection vulnerability: After refining and adding various logic to SQL queries, craft and execute the underlying SQL query that will be used to attack the target system. The goal is to reveal, modify, and/or delete database data, using the knowledge obtained in the previous step. This could entail crafting and executing multiple SQL queries if a denial of service attack is the intent.

Techniques
Craft and Execute underlying SQL query

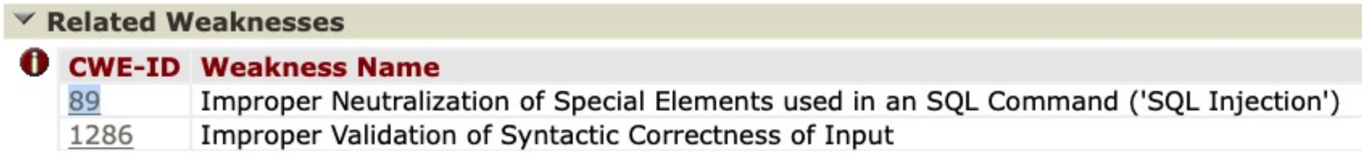
Source: https://capec.mitre.org/data/definitions/66.html

Study at home

Example question

- How is CAPEC related to CWE?
- Have a look at examples in CWE
  - CWE-89: **SQL Injection** https://cwe.mitre.org/data/definitions/89.html
- And CAPEC
  - CAPEC-66: **SQL Injection** https://capec.mitre.org/data/definitions/66.html

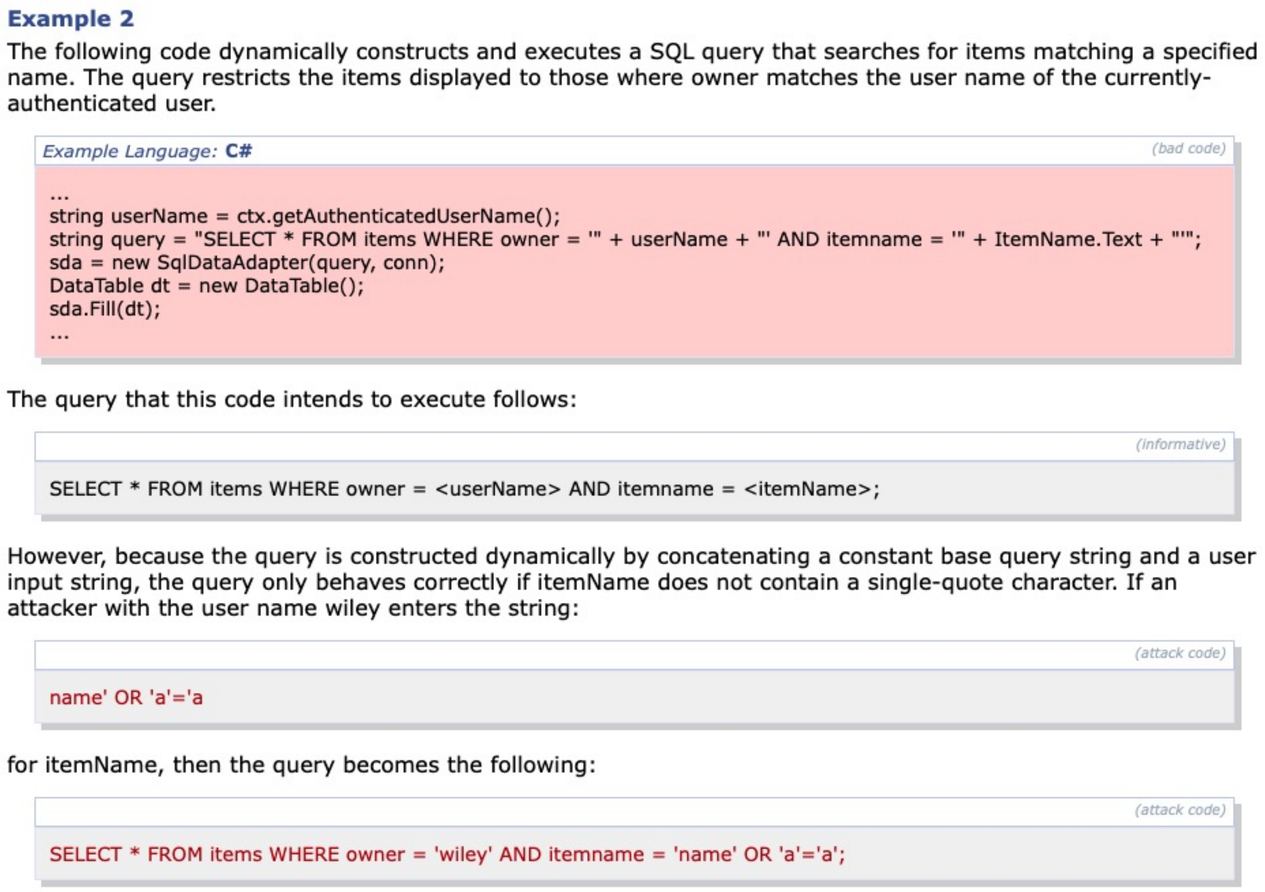
The weakness(es) that the attack pattern is exploiting (CWEs) are listed in CAPEC-66, in the “Related Weaknesses” section



Related Weaknesses

CWE-ID	Weakness Name
89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
1286	Improper Validation of Syntactic Correctness of Input

Description in CWE-89



Example 2

The following code dynamically constructs and executes a SQL query that searches for items matching a specified name. The query restricts the items displayed to those where owner matches the user name of the currently-authenticated user.

Example Language: **C#** *(bad code)*

```
...
string userName = ctx.getAuthenticatedUserName();
string query = "SELECT * FROM items WHERE owner = '" + userName + "' AND itemname = '" + ItemName.Text + "'";
sda = new SqlDataAdapter(query, conn);
DataTable dt = new DataTable();
sda.Fill(dt);
...
```

The query that this code intends to execute follows:

(informative)

```
SELECT * FROM items WHERE owner = <userName> AND itemname = <itemName>;
```

However, because the query is constructed dynamically by concatenating a constant base query string and a user input string, the query only behaves correctly if itemName does not contain a single-quote character. If an attacker with the user name wiley enters the string:

(attack code)

```
name' OR 'a'='a
```

for itemName, then the query becomes the following:

(attack code)

```
SELECT * FROM items WHERE owner = 'wiley' AND itemname = 'name' OR 'a'='a';
```

CAPEC Attack Pattern & CWE

- Common Weakness Enumeration (CWE) is a community-developed list of software (and hardware) weakness types
  - CWE serves as a common language and as a baseline for weakness identification. As well as a measuring stick for security tools
- A CAPEC attack pattern is typically a method of leveraging a CWE to execute an attack