Experiment E4: Training

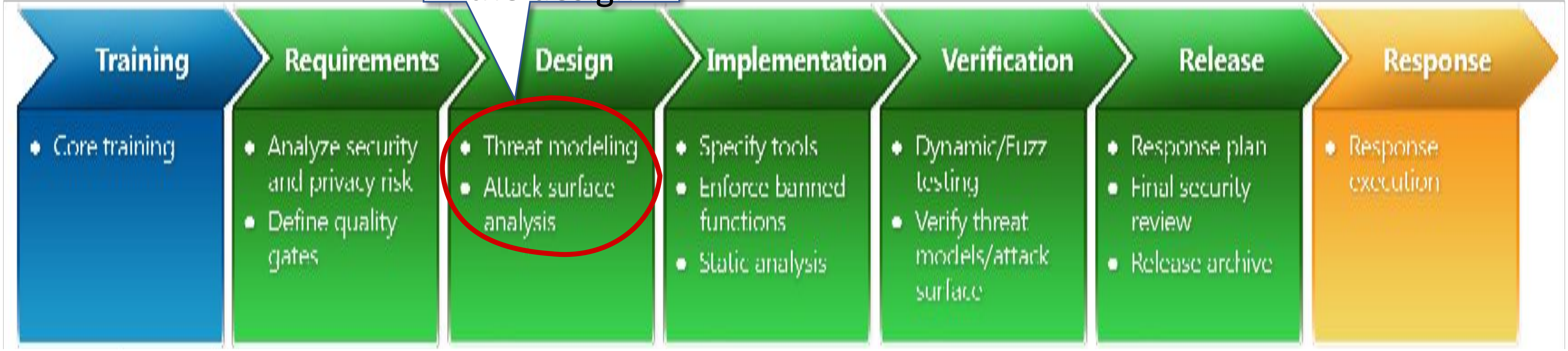
Threat Analysis

Katja Tuma

katjatuma.github.io

Secure Development Lifecycle

Experts meet and scrutinize the design



Training
- Core training

Requirements
- Analyze security and privacy risk
- Define quality gates

Design
- Threat modeling
- Attack surface analysis

Implementation
- Specify tools
- Enforce banned functions
- Static analysis

Verification
- Dynamic/Fuzz testing
- Verify threat models/attack surface

Release
- Response plan
- Final security review
- Release archive

Response
- Response execution

Figure: The Secure Development Life-cycle (@Microsoft).

Agenda

01

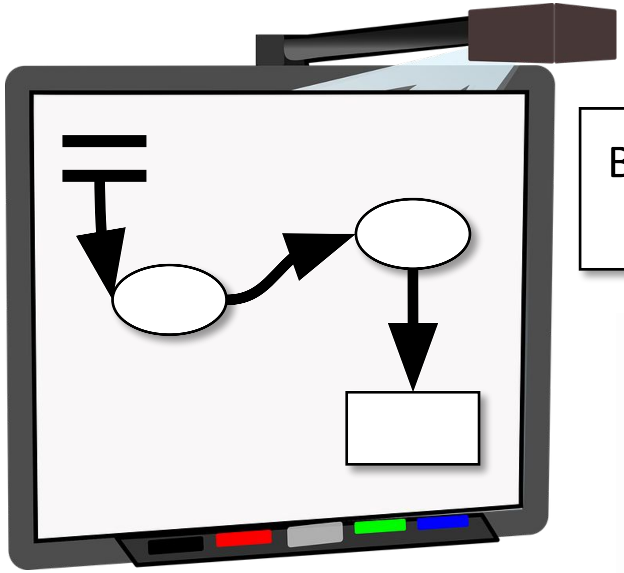
STRIDE Threat Categories

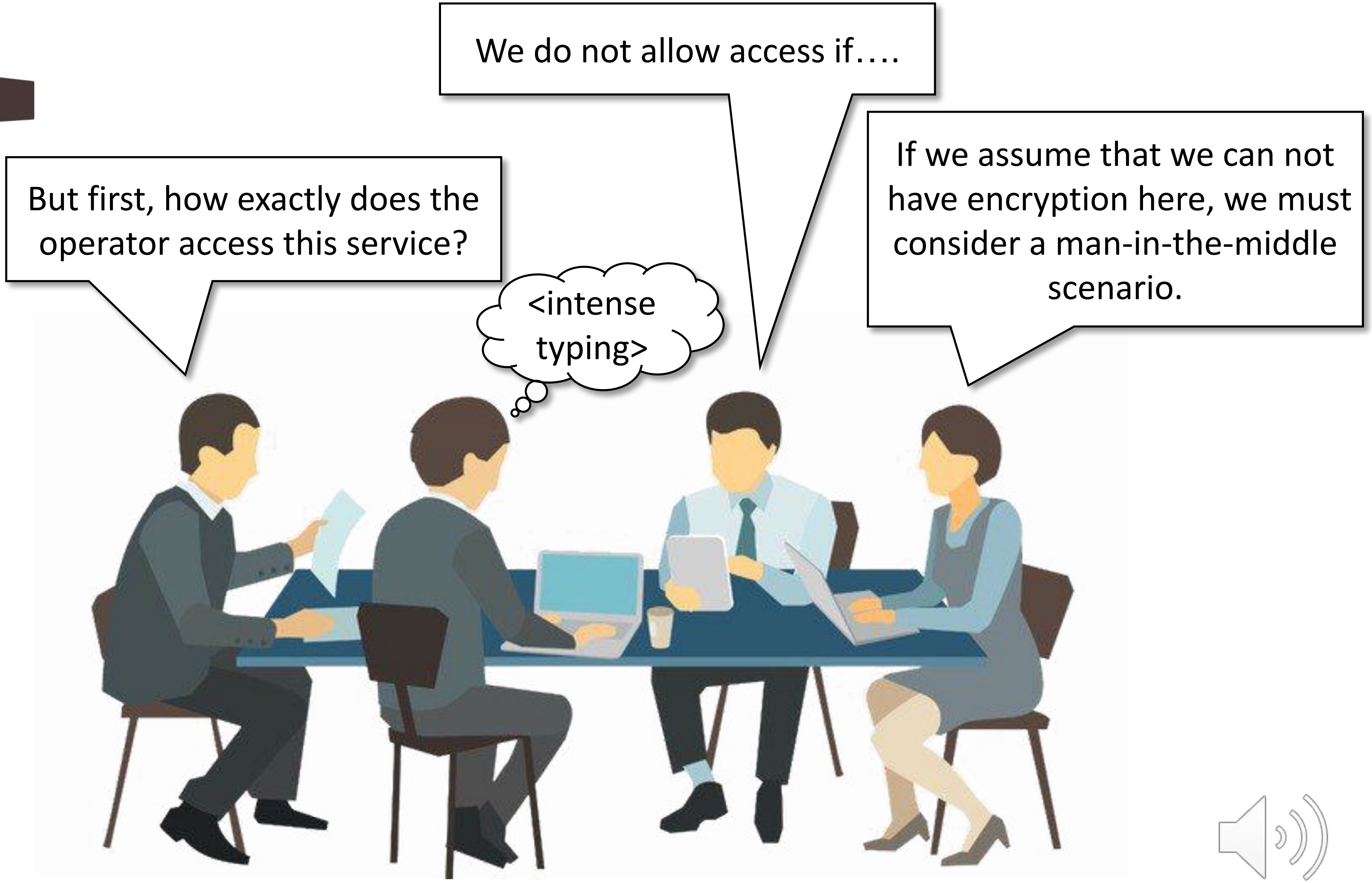
02

Data Flow Diagram (DFD)

+ Example

STRIDE




But first, how exactly does the operator access this service?
<intense typing>
We do not allow access if….
If we assume that we can not have encryption here, we must consider a man-in-the-middle scenario.

STRIDE

Methodology

- Define users and realistic use scenarios
- Gather assumptions

1. **Model** the system with **DFD diagram** (assets)
2. **Map** STRIDE to DFD (visit the elements)
3. **Refine** threats
4. **Document** the threats

Depending on the technique
(per-el,per-int,end-to-end,..)

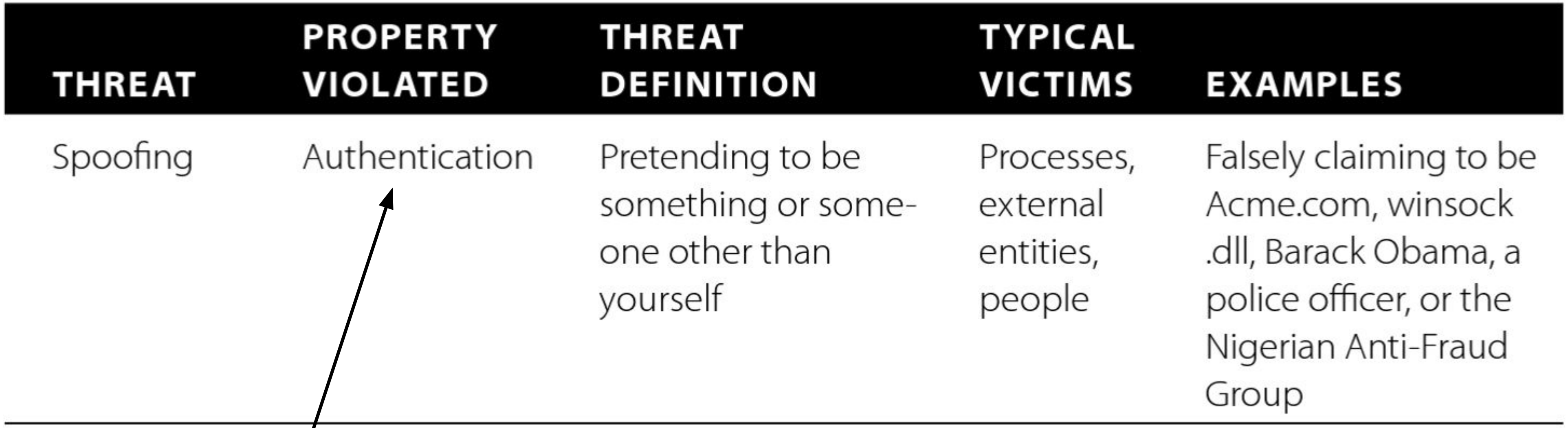
- Assign priority via risk analysis
- Draft mitigations associated to threats

STRIDE categories (definitions)

- **Spoofing** is pretending to be something or someone you're not.
- **Tampering** is modifying something you're not supposed to modify. It can include packets on the wire (or wireless), bits on disk, or the bits in memory
- **Repudiation** means claiming you didn't do something (regardless of whether you did or not)
- **Information Disclosure** is about exposing information to people who are not authorized to see it
- **Denial of Service** are attacks designed to prevent a system from providing service, including by crashing it, making it unusably slow, or filling all its storage
- **Elevation of Privilege** is when a program or user is technically able to do things that they're not supposed to do

A. SHOSTACK, THREAT MODELING : DESIGNING FOR SECURITY, WILEY 2014

Threat categories: Spoofing



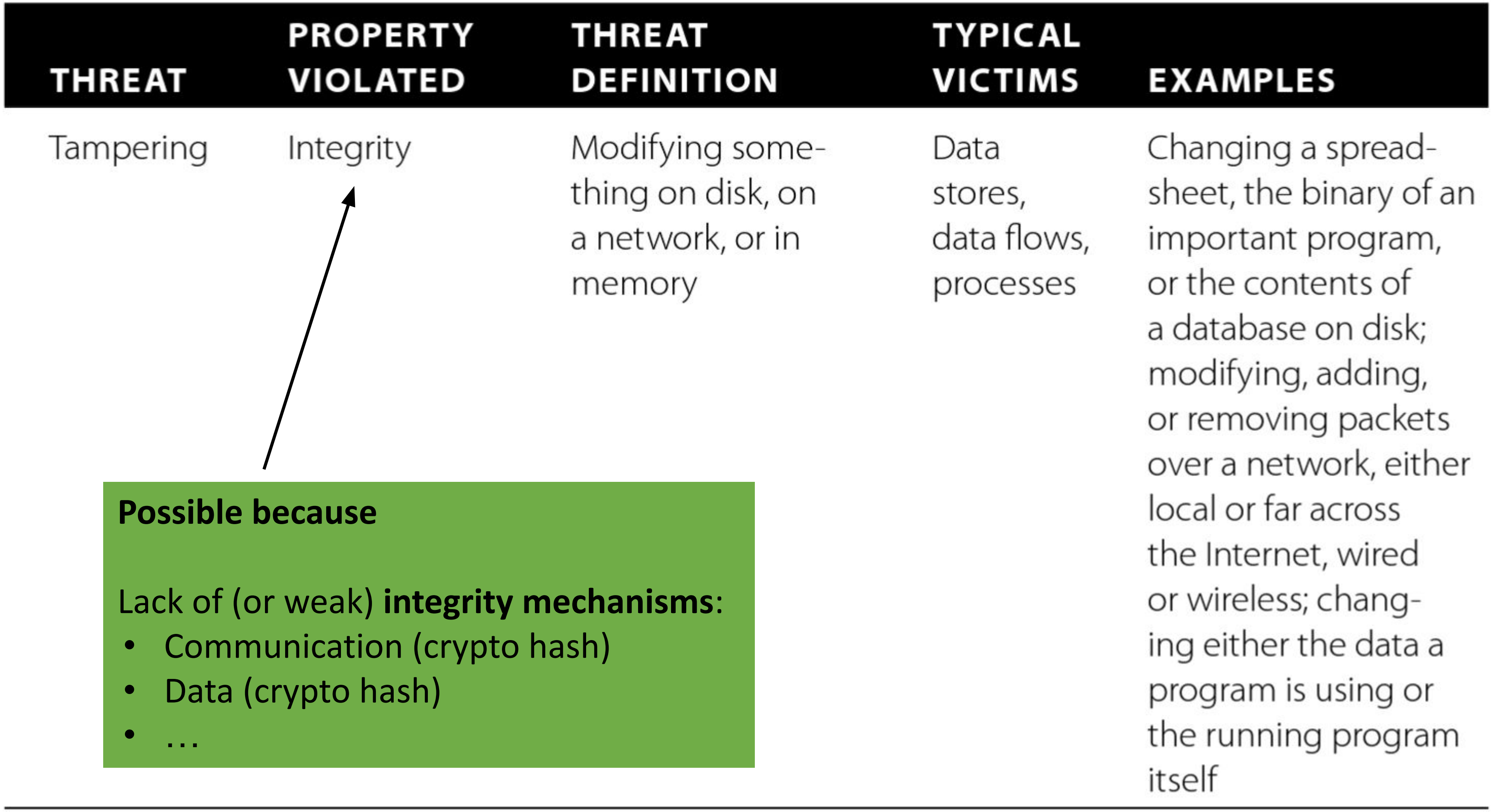
THREAT	PROPERTY VIOLATED	THREAT DEFINITION	TYPICAL VICTIMS	EXAMPLES
Spoofing	Authentication	Pretending to be something or someone other than yourself	Processes, external entities, people	Falsely claiming to be Acme.com, winsock .dll, Barack Obama, a police officer, or the Nigerian Anti-Fraud Group

Possible because

Lack of (or weak) **authentication**:

- Sender of a message (signature)
- Identity of other party (certificate)
- Identity of user (credentials)
- …

Threat categories: Tampering



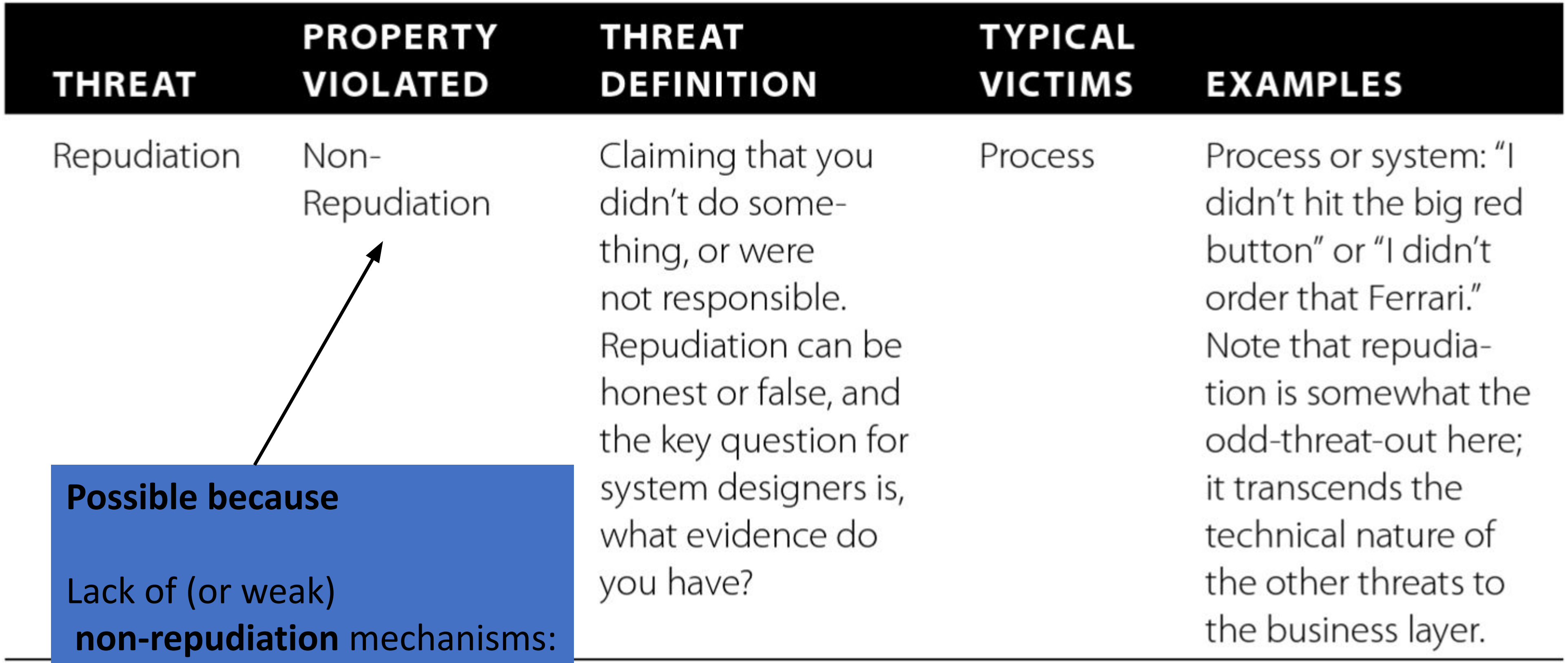
THREAT	PROPERTY VIOLATED	THREAT DEFINITION	TYPICAL VICTIMS	EXAMPLES
Tampering	Integrity	Modifying something on disk, on a network, or in memory	Data stores, data flows, processes	Changing a spreadsheet, the binary of an important program, or the contents of a database on disk; modifying, adding, or removing packets over a network, either local or far across the Internet, wired or wireless; changing either the data a program is using or the running program itself

Possible because

Lack of (or weak) **integrity mechanisms**:

- Communication (crypto hash)
- Data (crypto hash)
- …

Threat categories: Repudiation



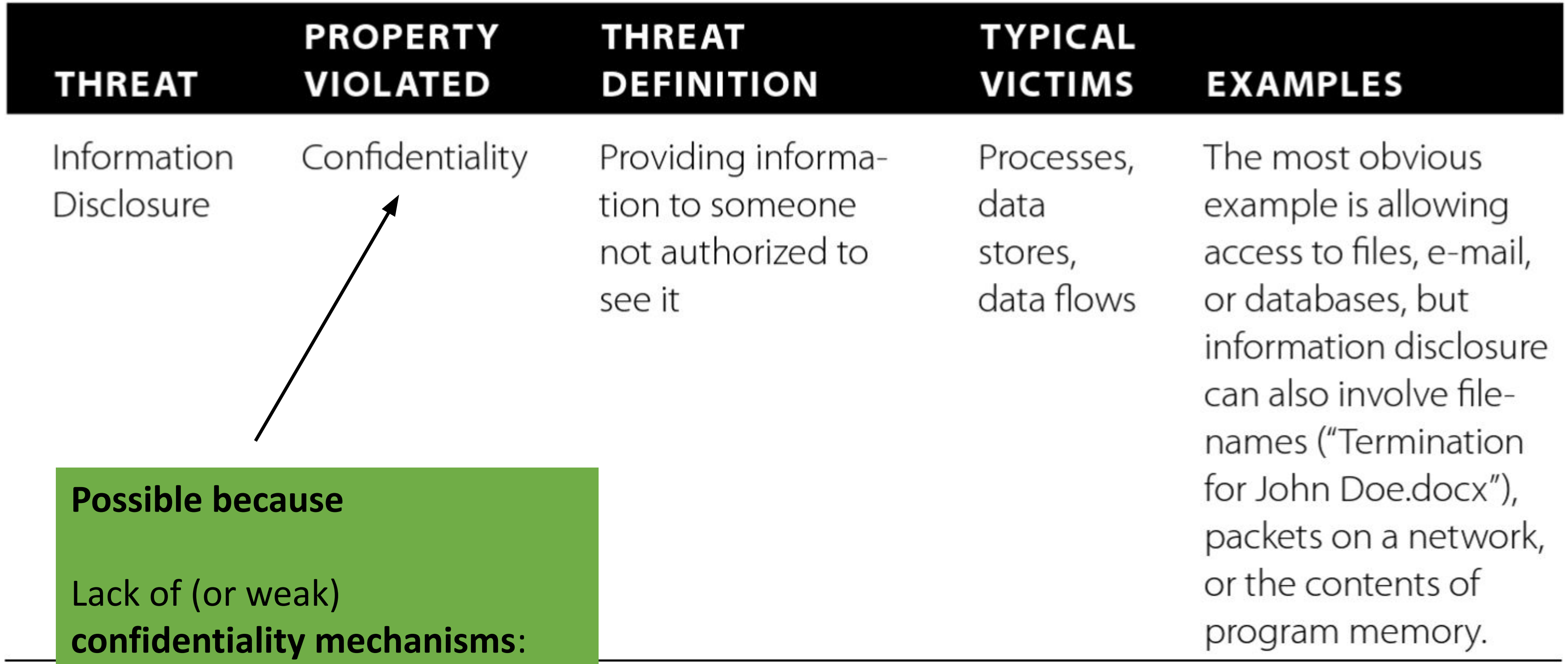
THREAT	PROPERTY VIOLATED	THREAT DEFINITION	TYPICAL VICTIMS	EXAMPLES
Repudiation	Non-Repudiation	Claiming that you didn't do something, or were not responsible. Repudiation can be honest or false, and the key question for system designers is, what evidence do you have?	Process	Process or system: "I didn't hit the big red button" or "I didn't order that Ferrari." Note that repudiation is somewhat the odd-threat-out here; it transcends the technical nature of the other threats to the business layer.

Possible because

Lack of (or weak) **non-repudiation** mechanisms:

- Audit trails
- Signed requests
- Trusted third party
- …

Threat categories: Info disclosure



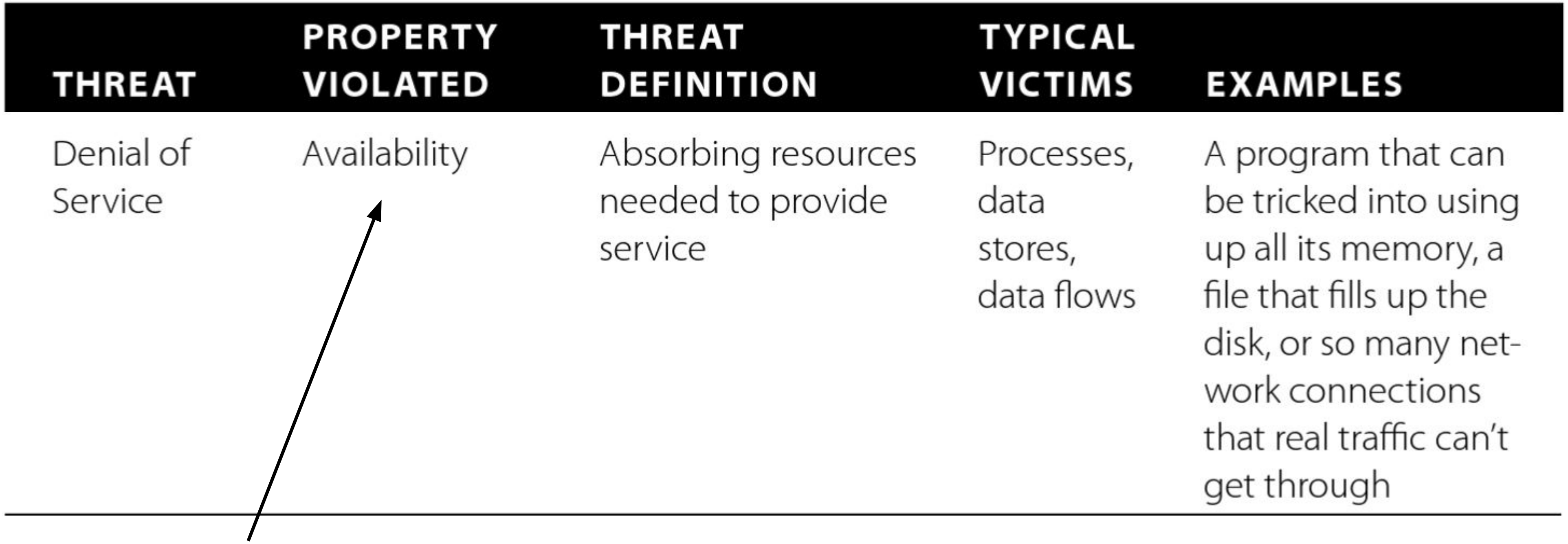
THREAT	PROPERTY VIOLATED	THREAT DEFINITION	TYPICAL VICTIMS	EXAMPLES
Information Disclosure	Confidentiality	Providing information to someone not authorized to see it	Processes, data stores, data flows	The most obvious example is allowing access to files, e-mail, or databases, but information disclosure can also involve filenames (“Termination for John Doe.docx”), packets on a network, or the contents of program memory.

Possible because

Lack of (or weak) **confidentiality mechanisms**:

- Communication (encryption)
- Storage (encryption)
- Access control
- …

Threat categories: Denial of service



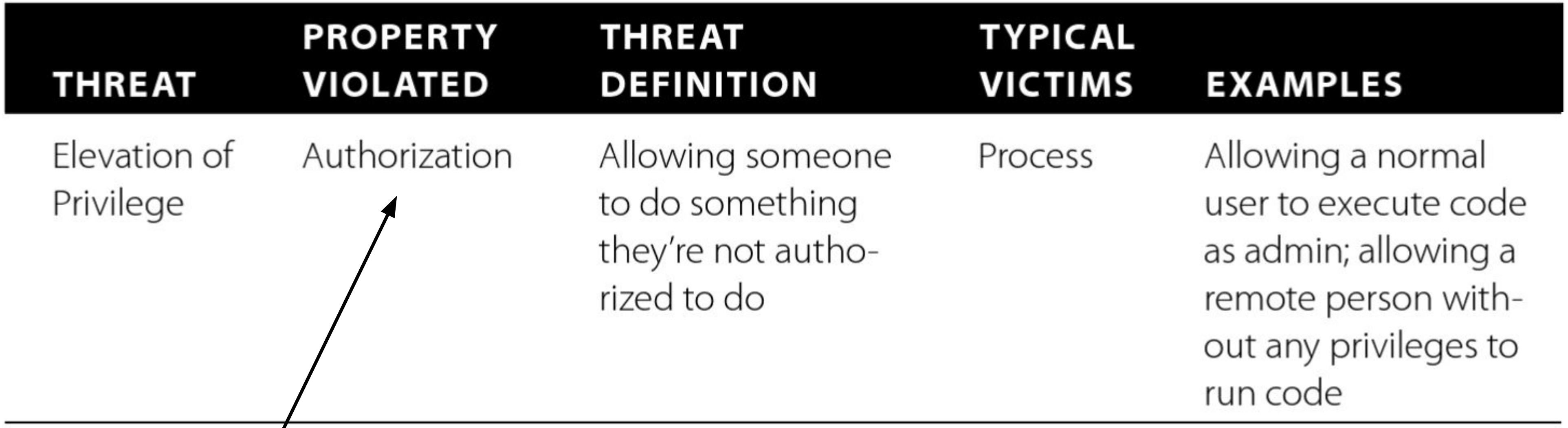
THREAT	PROPERTY VIOLATED	THREAT DEFINITION	TYPICAL VICTIMS	EXAMPLES
Denial of Service	Availability	Absorbing resources needed to provide service	Processes, data stores, data flows	A program that can be tricked into using up all its memory, a file that fills up the disk, or so many network connections that real traffic can't get through

Possible because

Lack of **availability mechanisms**:

- Load balancing/replication
- Lock-down
- …

Threat categories: Elevation of privilege



THREAT	PROPERTY VIOLATED	THREAT DEFINITION	TYPICAL VICTIMS	EXAMPLES
Elevation of Privilege	Authorization	Allowing someone to do something they're not authorized to do	Process	Allowing a normal user to execute code as admin; allowing a remote person without any privileges to run code

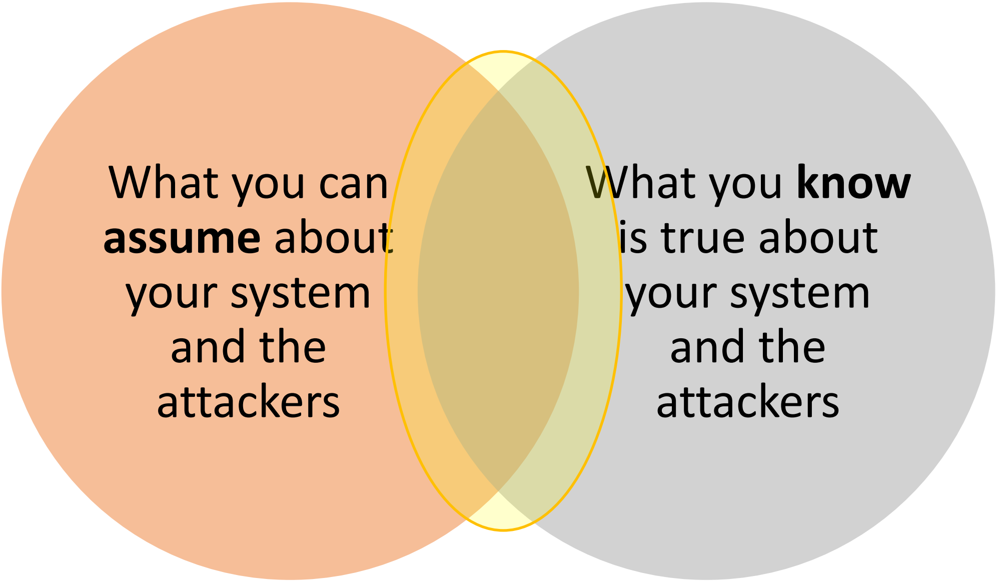
Possible because

Lack of **authorization** and input validation

Assumptions

- Assumptions are **choices to trust an element of the system** to behave as expected
    - e.g., **human** follows procedures (user chooses hard-to-guess passwords)
    - E.g., **piece of software** works as advertised (firewall blocks network intrusions)

- Used to **reason about threats**
    - Possible/feasible ?

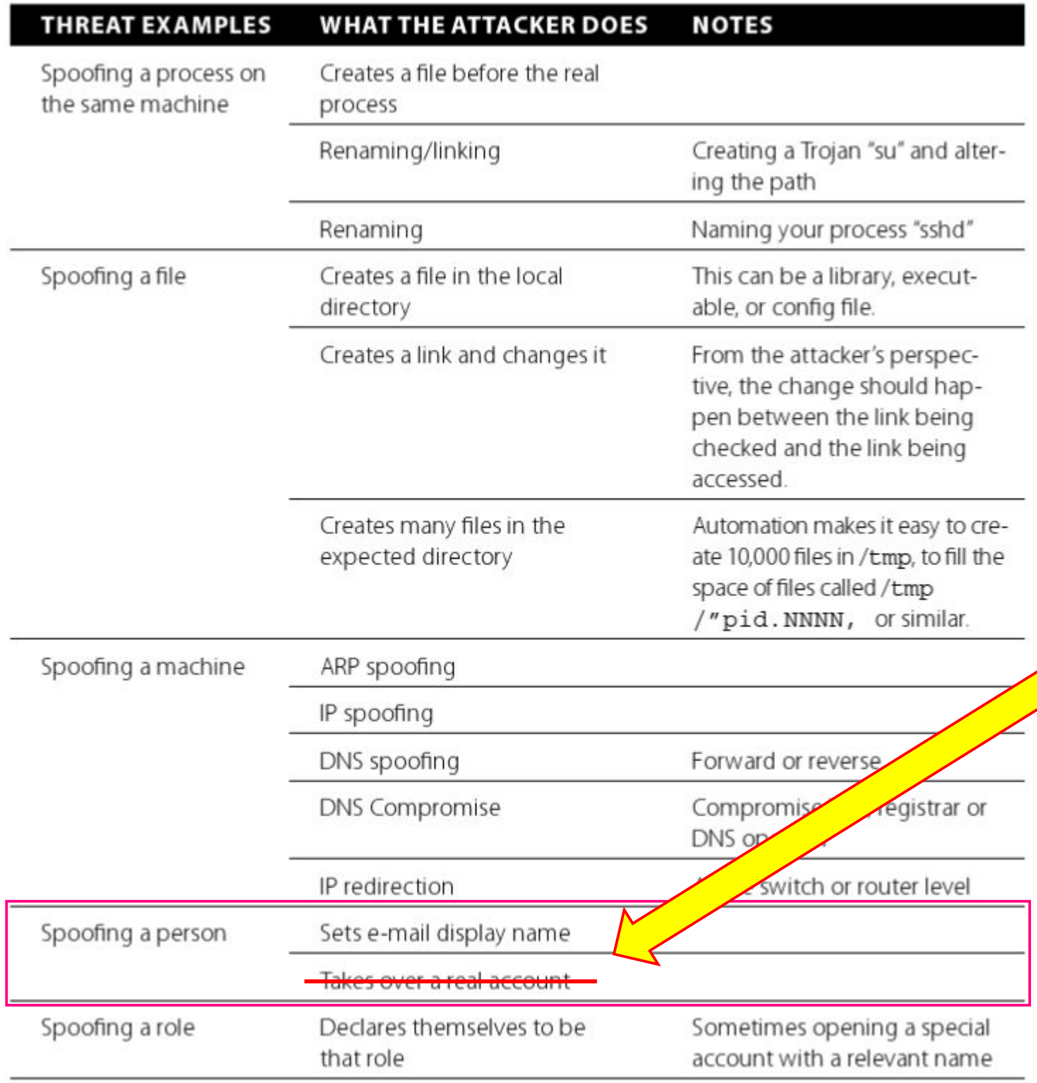


What you can **assume** about your system and the attackers

What you **know** is true about your system and the attackers

Assumptions: example

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 [illegible] registrar or DNS op[illegible]
	IP redirection	[illegible] 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 possible because

Lack of (or weak) **authentication**:

- Identity of user (credentials)
- …

E.g., if assumption is

- The attacker **cannot take over a real account** of another user because we use two-factor authentication (code sent via SMS)
- … and smarphones cannot be stolen by attacker in Russia

3647 2286

Up next

01

STRIDE Threat Categories

02

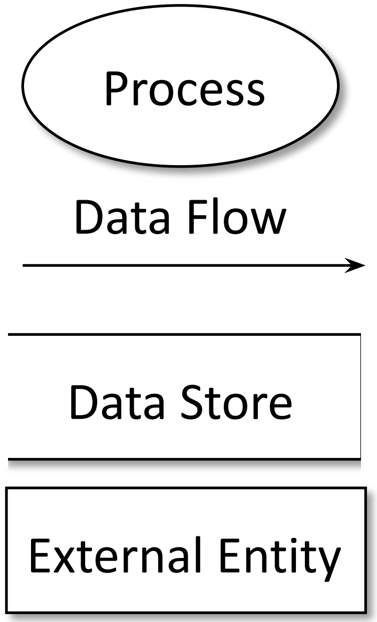
Data Flow Diagram (DFD)

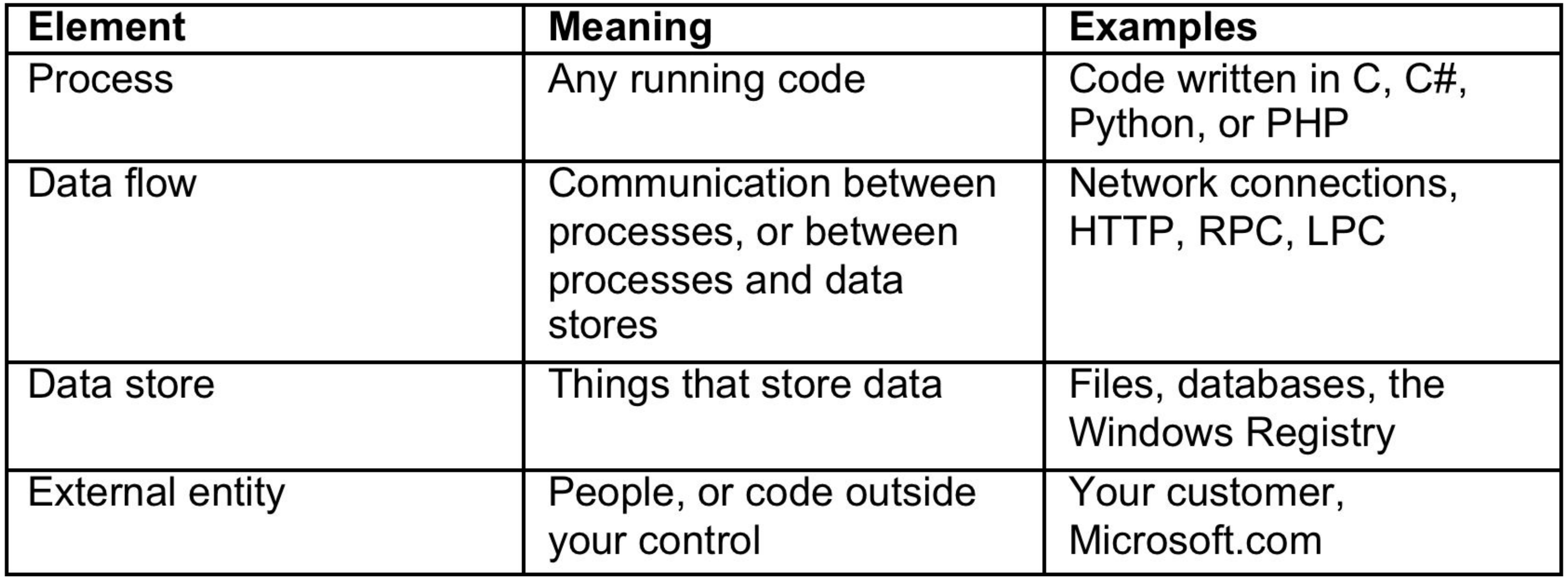
+ Example

Data Flow Diagrams (DFDs)

- A DFD is a graphical representation of how **data enters, leaves, and traverses your system**
- Shows all data **sources** and **destinations**
- Shows all relevant **steps** that data goes through

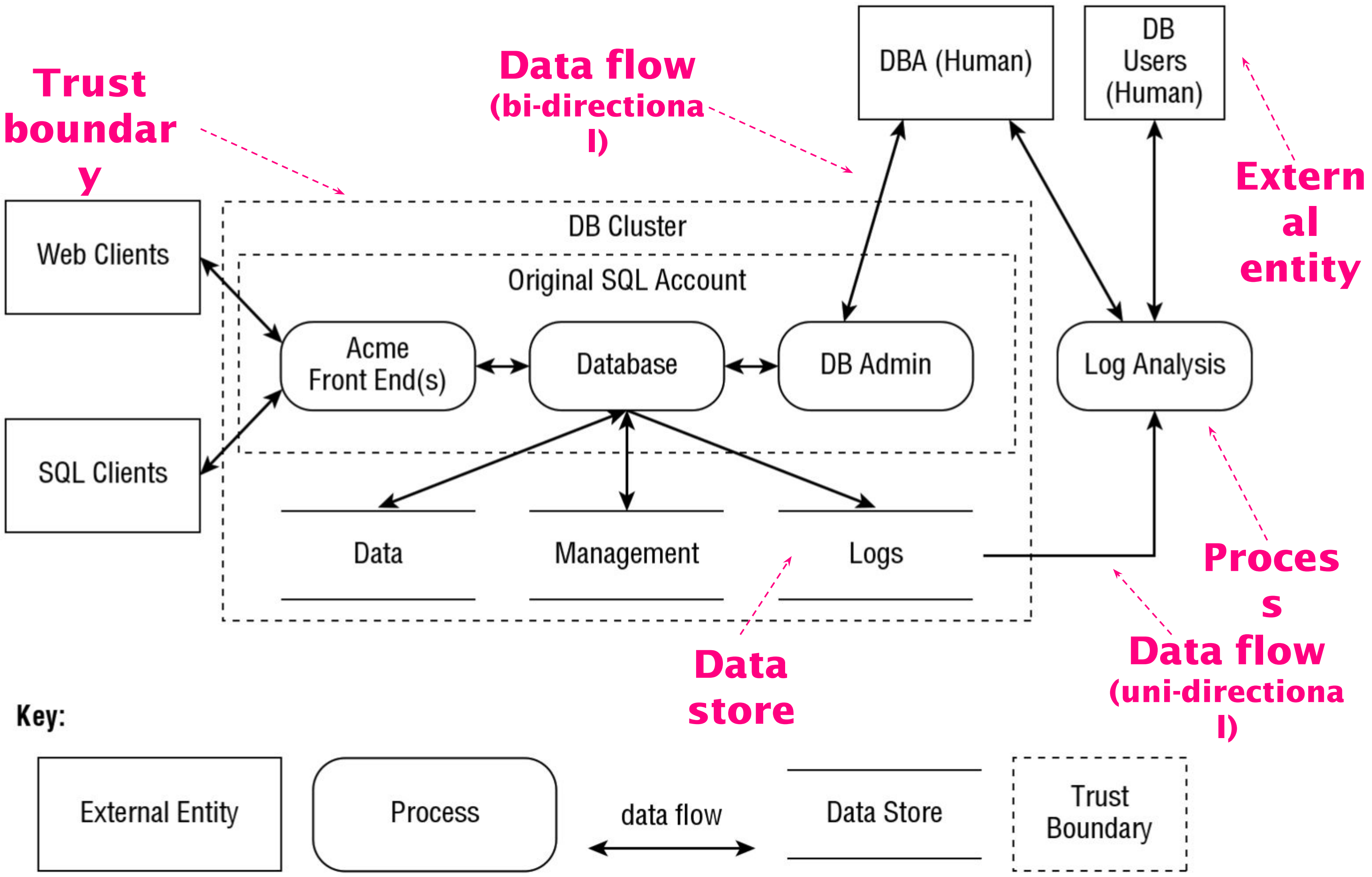
DFD Elements





Element	Meaning	Examples
Process	Any running code	Code written in C, C#, Python, or PHP
Data flow	Communication between processes, or between processes and data stores	Network connections, HTTP, RPC, LPC
Data store	Things that store data	Files, databases, the Windows Registry
External entity	People, or code outside your control	Your customer, Microsoft.com

DFD example



Trust boundary

Data flow (bi-directional)

External entity

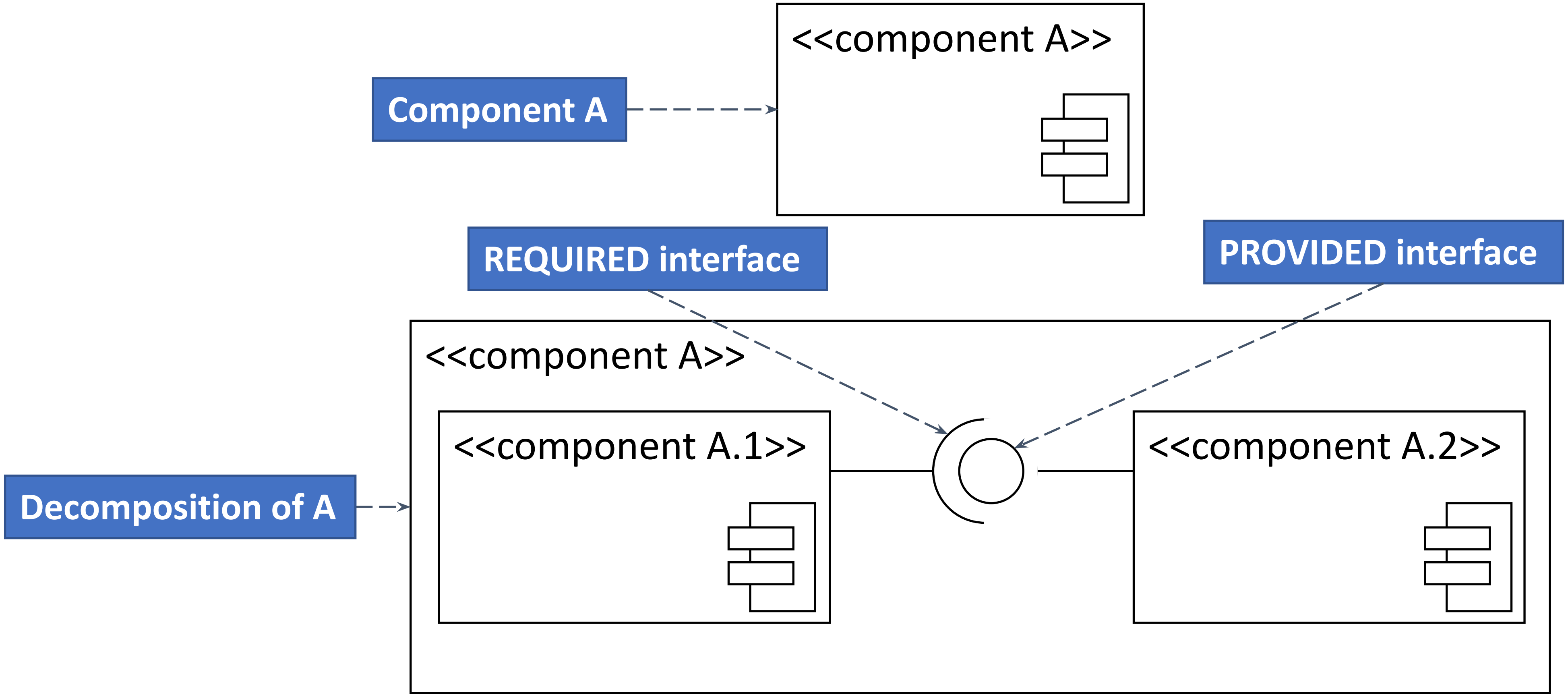
Data store

Process Data flow (uni-directional)

Figure 2-4: A modern DFD model

Component diagrams (quick refresher)

Purpose = provide structural relationships between system components



Component A

<<component A>>

REQUIRED interface

PROVIDED interface

<<component A>>

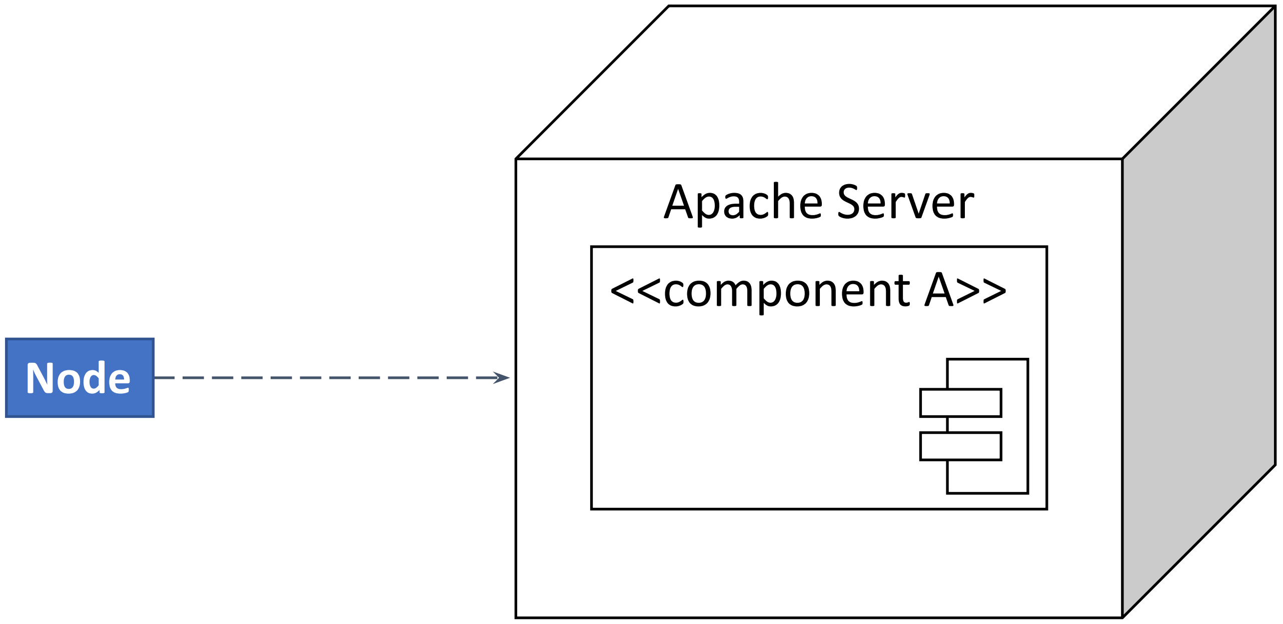
<<component A.1>>

<<component A.2>>

Decomposition of A

Deployment diagram

Purpose = depiction of a physical deployment of system



How to build DFD (heuristics)

- Start drawing the external entities
  - E.g., from context diagram

- Map nodes in the deployment diagram to processes or data stores
  - If node contains both data and logic:
    split into process(es) + data store(s)
  - If node contains multiple databases:
    consider splitting into multiple data stores

How to build DFD (heuristics)

- Derive data flows from *interfaces, links, connectors*

- Use *main component diagram* and *decompositions* to refine the DFD (if necessary!)
  - Ignore the inner workings
  - Security-relevant processes need to be shown

- Add trust boundaries (each boundary box should have a label inside it)

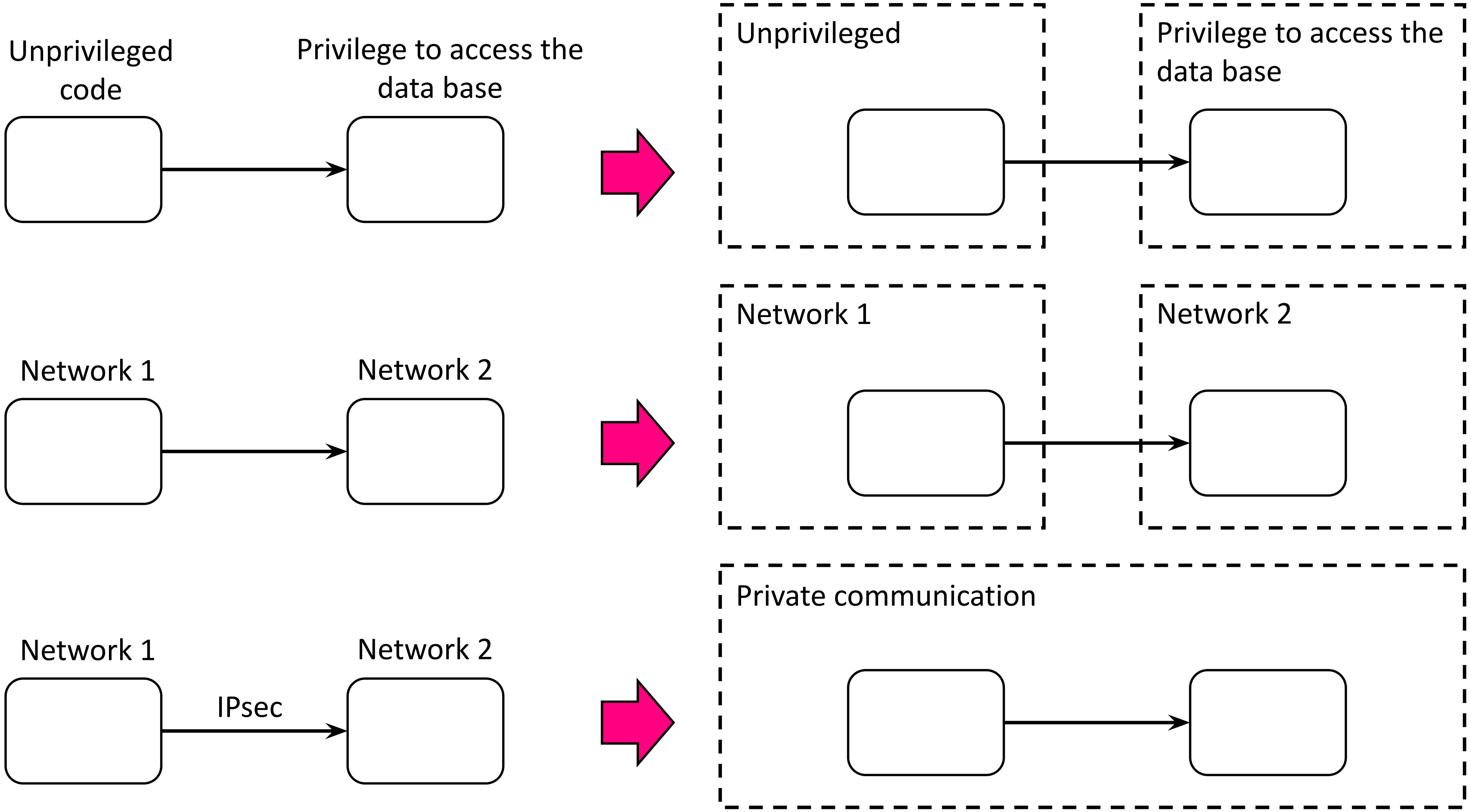
Trust boundaries

Ask yourself two questions

- Does everything in the system have the same level of privilege and access to everything else on the system?
  - Credential to access the DB?
  - Privileges to access the file system?
- Is everything your software communicates with inside that same boundary?
  - And do you trust all the potential "observers" of that communication?
  - Same network segment, machine, etc?

Trust boundaries

Examples



DFD validation

- Diagrams should be visible on a printable page
- Tell “the story” of the main use cases using the DFD without referring to things not in the diagram
- Show the security mechanisms for controlling data flows (such as firewalls, encrypted channels, identity checks, enforcement of permissions)

DFD validation

- Data flows are **NOT ALLOWED** between
  - External entity □□ Data store
  - Data store □□ Data store

  Show the process that moves the data

- All processes must have at least one entry data flow and one exit data flow

25 70 65 4

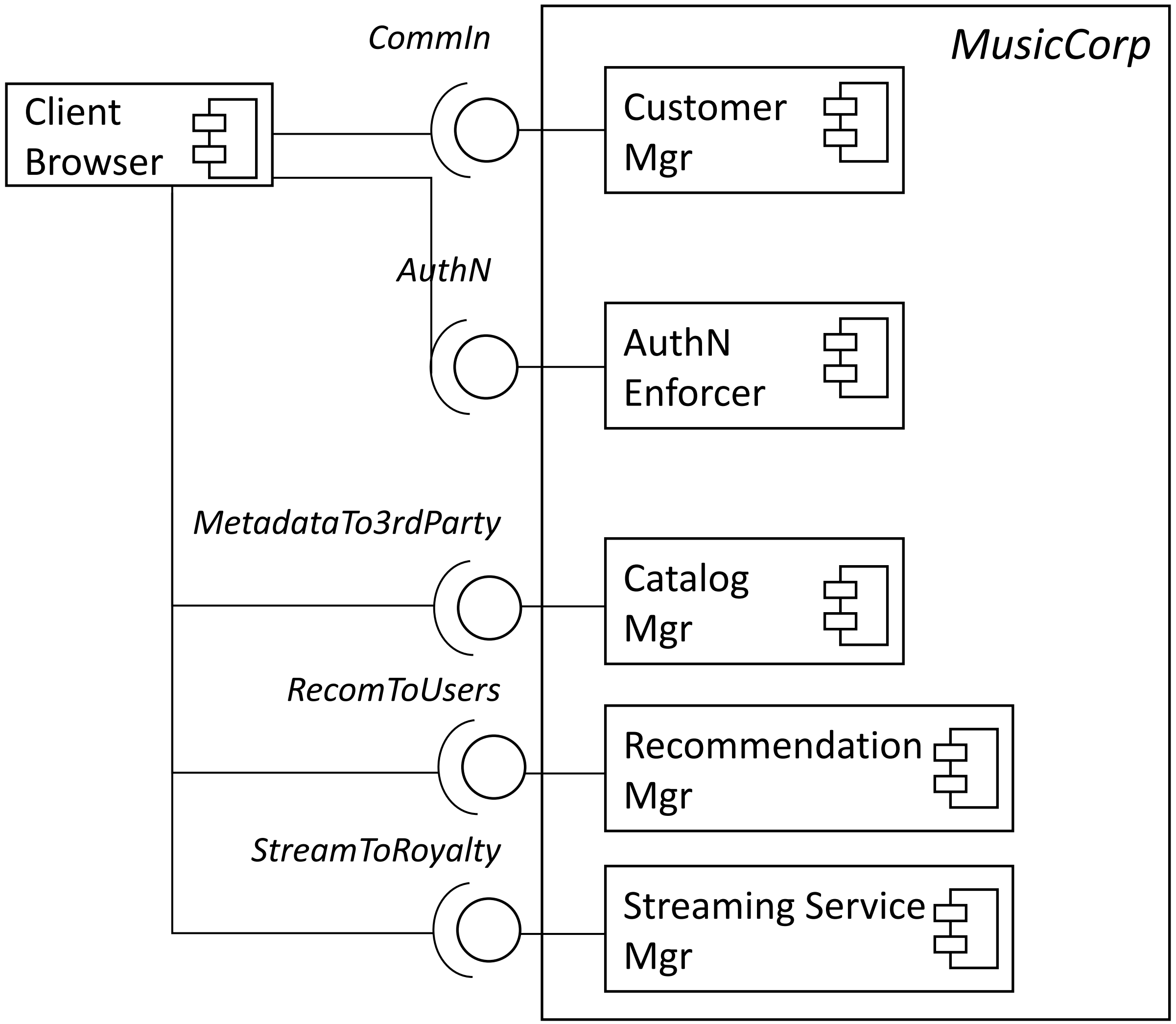
Example

MusicCorp

SAM NEWMAN, ”BUILDING MICROSERVICES: DESIGNING FINE-GRAINED SYSTEMS” , O’REILLY 2015

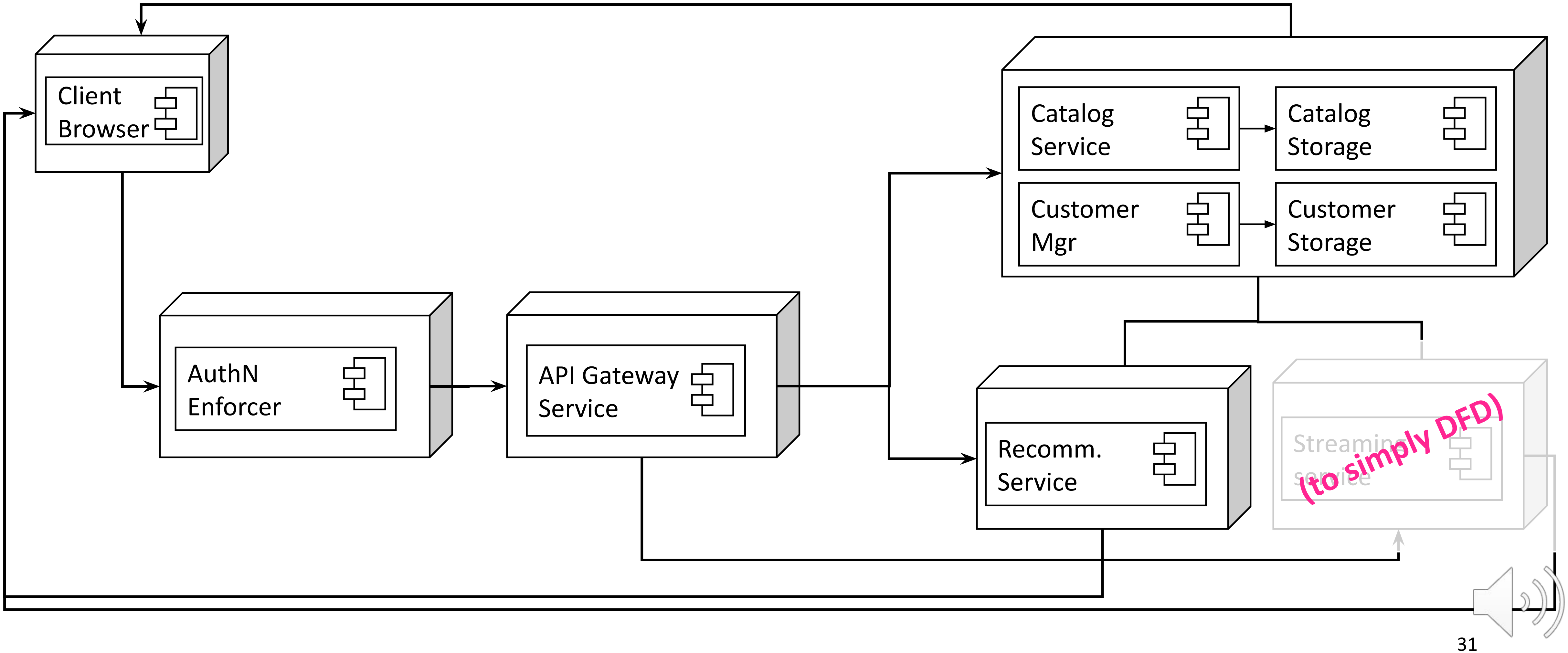
High-level component diagram

Example



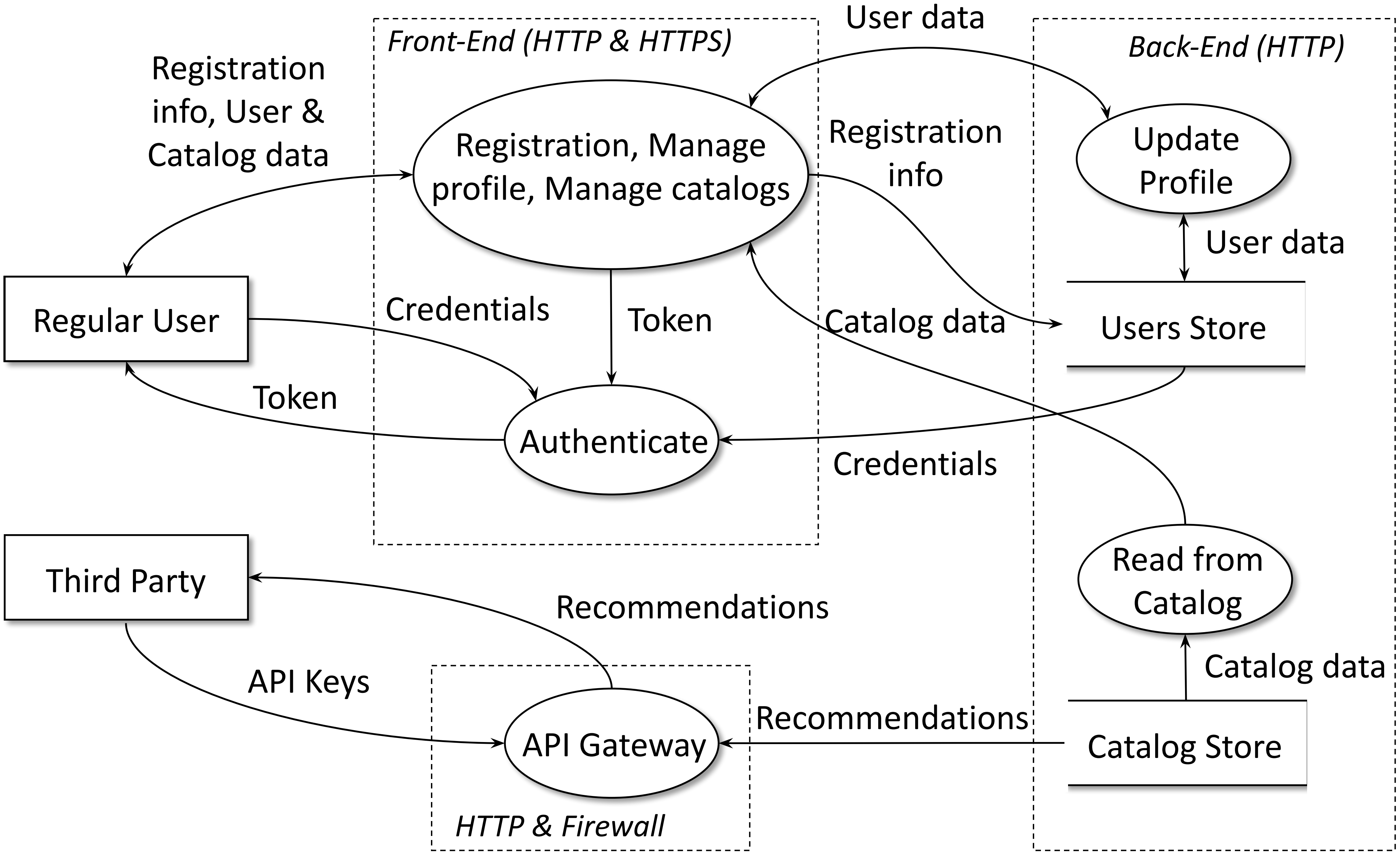
Example

Deployment diagram



(to simply DFD)

Solution



Front-End (HTTP & HTTPS)
Back-End (HTTP)
User data
Registration info, User & Catalog data
Registration, Manage profile, Manage catalogs
Registration info
Update Profile
User data
Regular User
Credentials
Token
Catalog data
Users Store
Token
Authenticate
Credentials
Read from Catalog
Third Party
Recommendations
Catalog data
API Keys
Recommendations
API Gateway
Catalog Store
HTTP & Firewall