THREAT MODELING CONNECT

HACKATHON

APRIL 1-21, 2024

Brought to you by: IriusRisk SHOSTACK + ASSOCIATES

WORKSHOP

Threat Modeling Kata

April 2, noon-1pm

Speaker

Luis Servín

Platform Security Lead @ Hapag-Lloyd AG

HACKATHON

APRIL 1-21, 2024

Agenda (ET)

12:00 - 12:10	Presentation
12:10 - 12:20	Exercise (breakout room)
12:20 - 12:40	Presentation
12:40 - 12:50	Exercise (breakout room)
12:50 - 01:00	Readout

Brought to you by: THREAT MODELING CONNECT SHOSTACK + ASSOCIATES

Kata 形

Kata is a Japanese word (型 or 形) meaning "form". It refers to a detailed choreographed pattern of martial arts movements made to be practised alone. It can also be reviewed within groups and in unison when training. It is practised in Japanese martial arts as a way to memorize and perfect the movements being executed.

Source: wikipedia

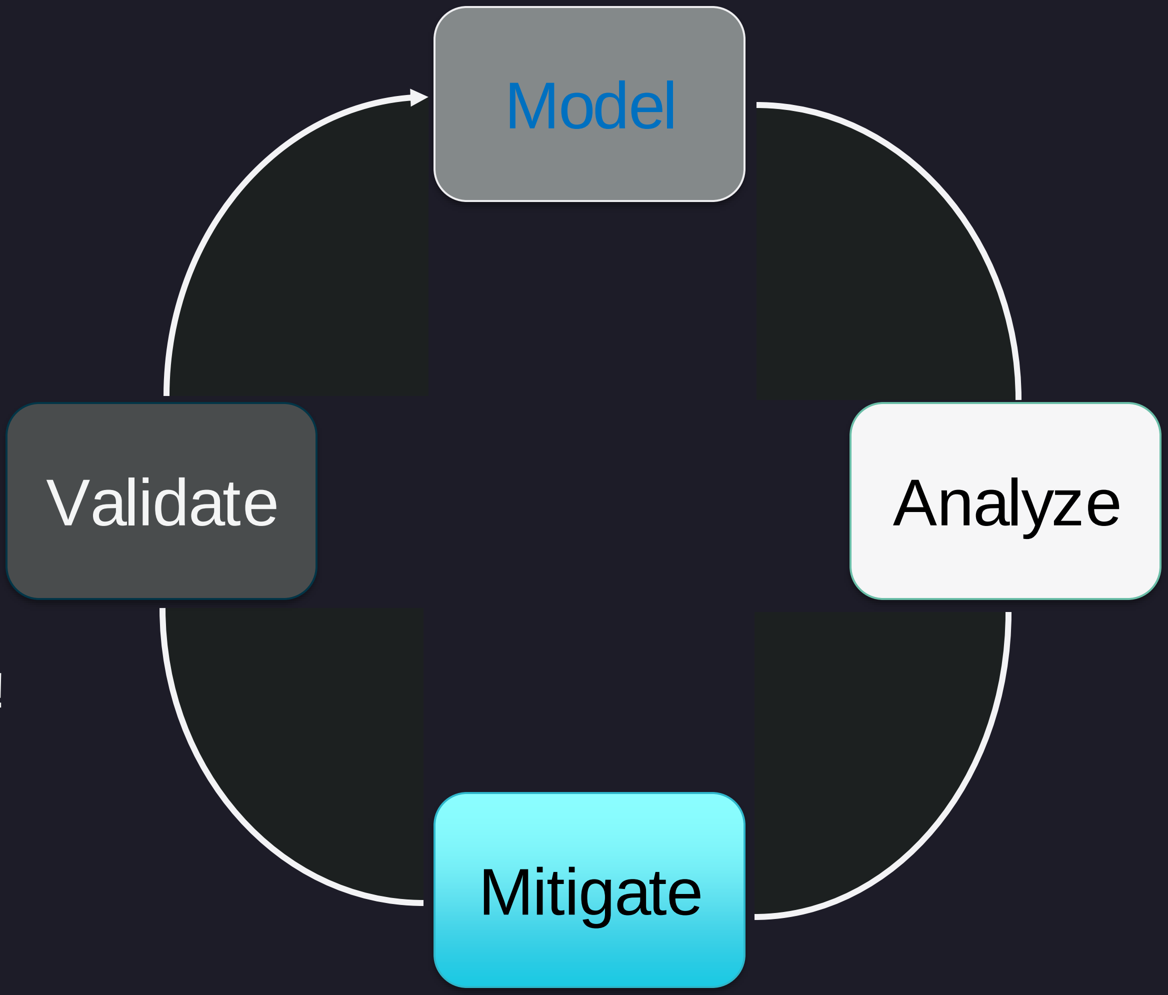
Threat Modeling Methodology

1. **Model** the system
   - Understand, Scope, and **Model** the system
   - **Validate** the model

2. **Analyze** the threats
   - **E.g. apply STRIDE, LINDDUN, etc**
   - **Validate** them

3. Do some **Mitigation** analysis
   - Accept, avoid, reduce, or transfer the threats!

4. **Validate** the outcome!



Getting Started with TM

Model

- Approach the target of evaluation systematically
  - Different backgrounds = different perspectives = different starting points
  - Beware: Don’t lose sight of the forest because of the trees!

Analysis

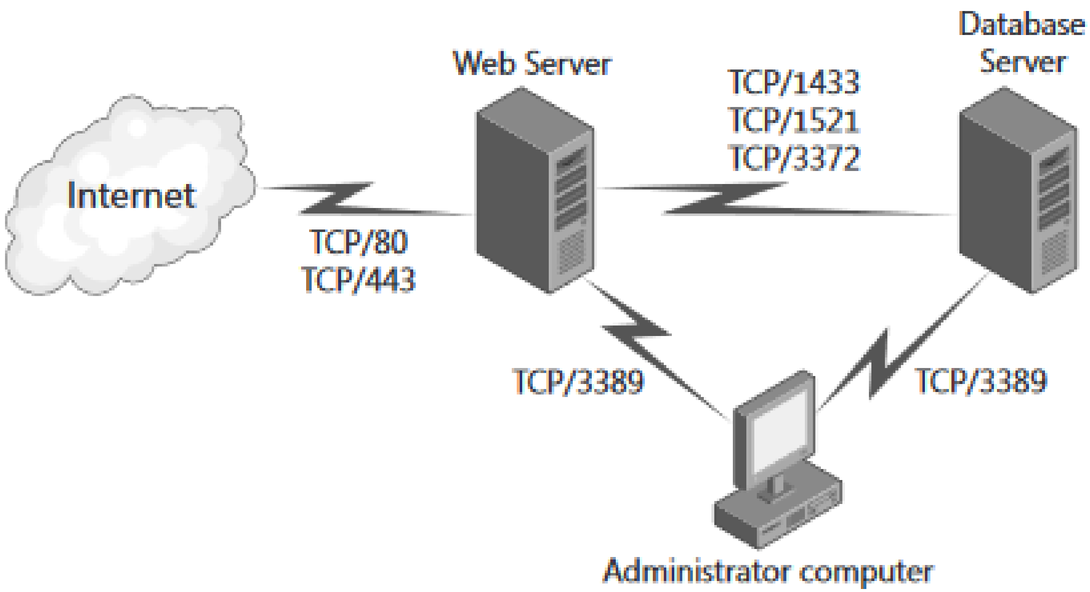
- Framework of methods (aka tools in the belt) to fall back upon
  - Use different threat elicitation methods (STRIDE != Threat Modeling)
  - Look for vulnerabilities to validate threats
  - “Sell” your findings using CAVs!

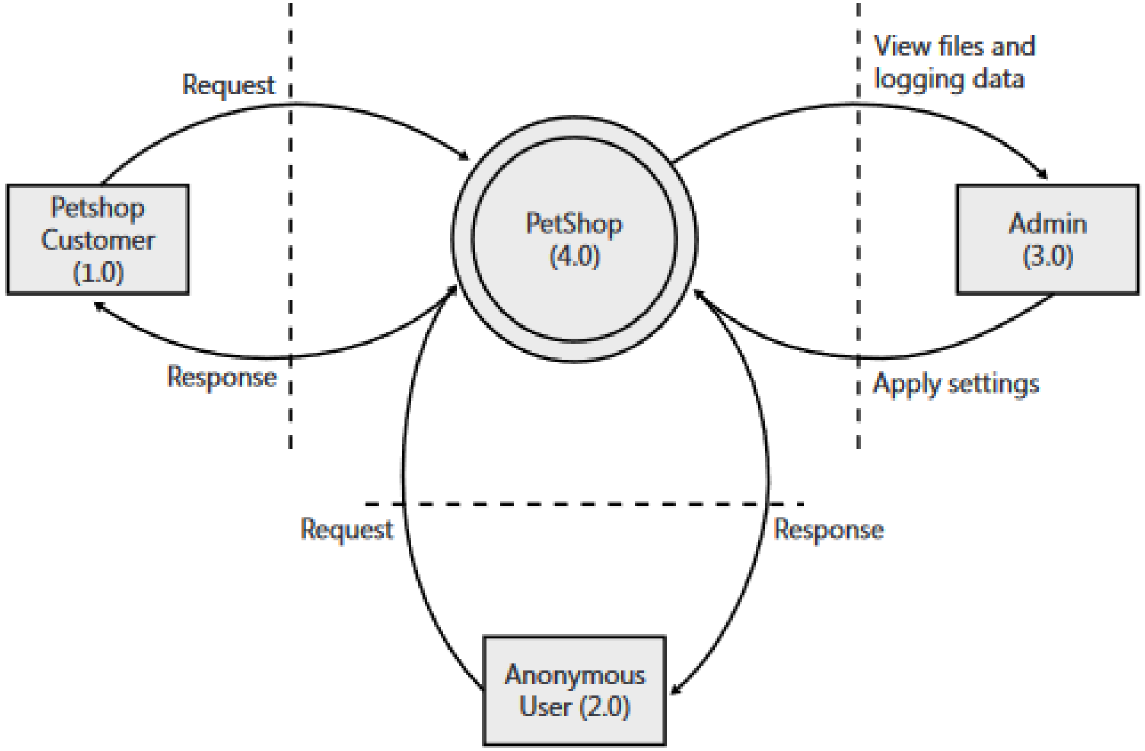
Mitigate

- Address the vulnerability (identify, protect) or the threat scenario (detect, respond, recover)

I. Model: Diagrams

Detail-level : Firewall placement Diagram (nodes, ports, protocols), logical layer

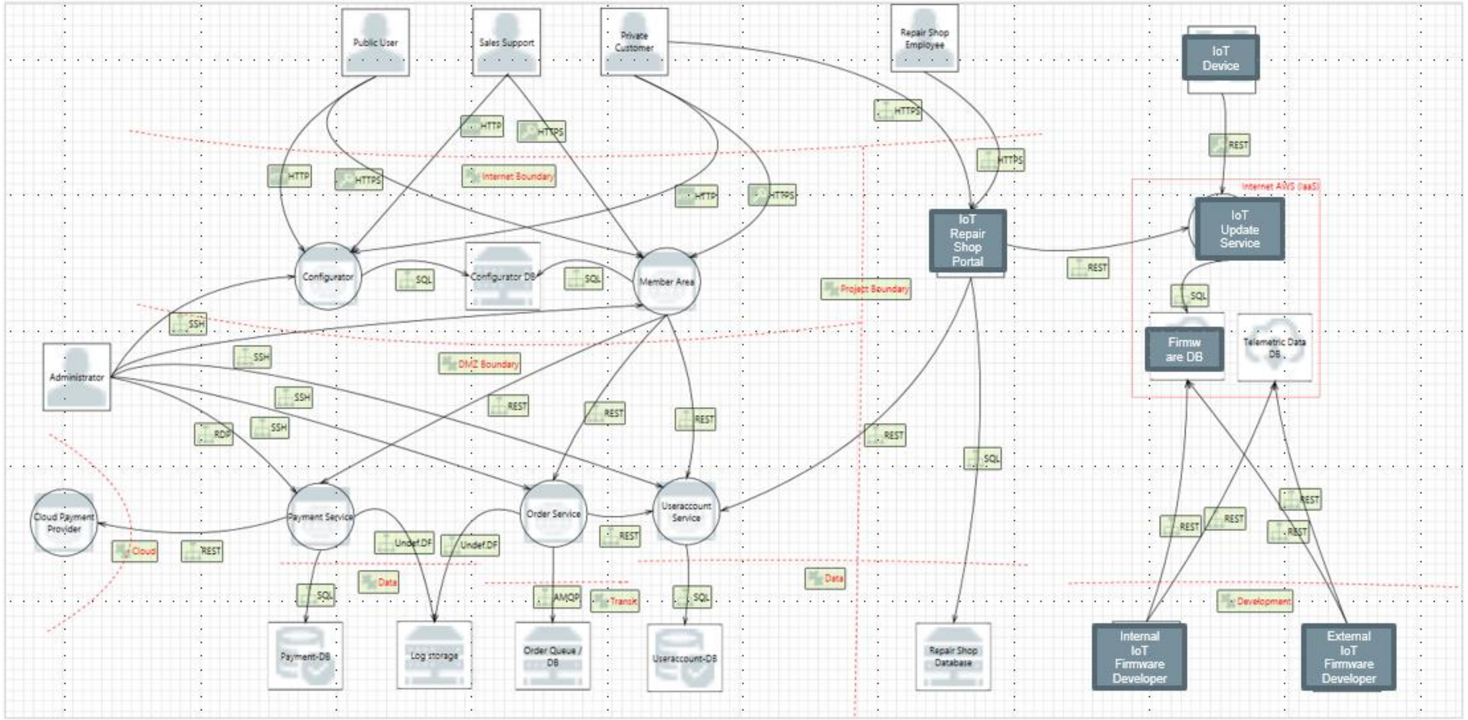




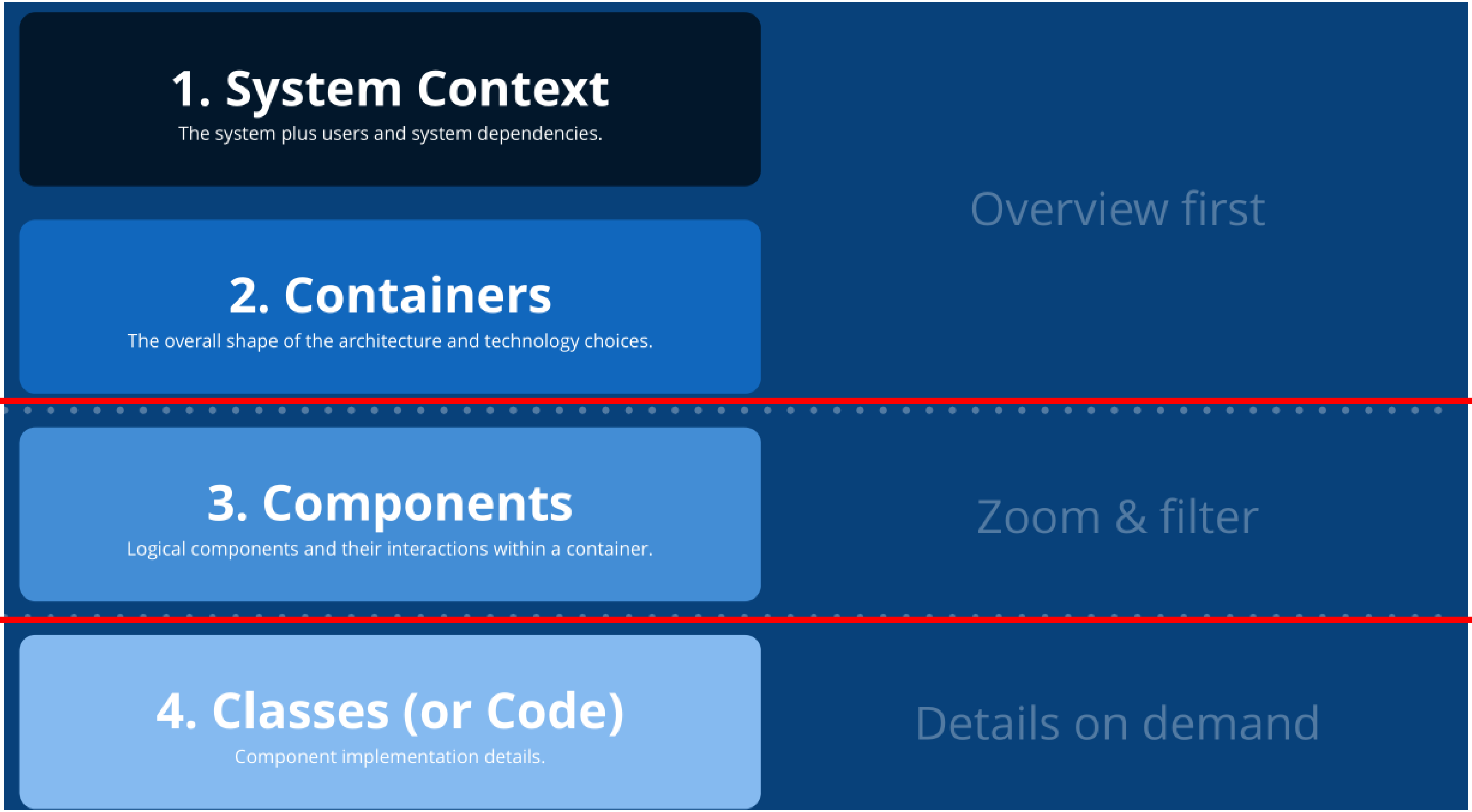
Source: Howard, M. Lipner, S. *The security development Lifecycle.* Ch.9, 2006. Microsoft Press

I. Model: Diagrams II

Correctness, re-usability
Up-to-date (how?, who?)
Duplication of work (different tools in the security team?)
“Messy” Diagrams (“everything but the sink” diagrams)
Abstraction vs Notation



I. Model: C4 model



1. System Context
The system plus users and system dependencies.

2. Containers
The overall shape of the architecture and technology choices.

3. Components
Logical components and their interactions within a container.

4. Classes (or Code)
Component implementation details.

Overview first

Zoom & filter

Details on demand

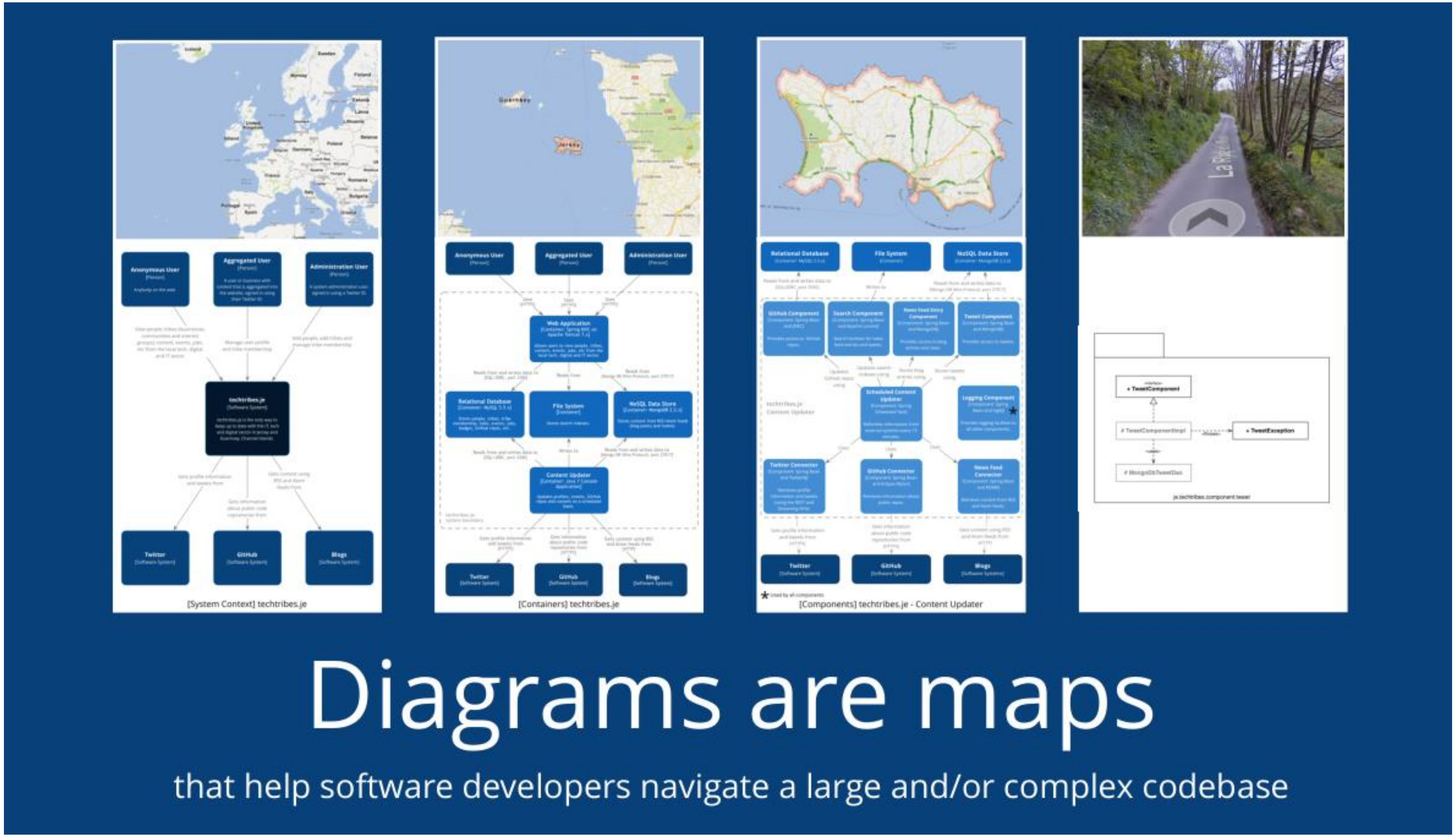
Always get here!

Reserve for sensitive elements/ functions

Hardly ever.

Source: Simon Brown C4model.com

I. Model: C4 model II

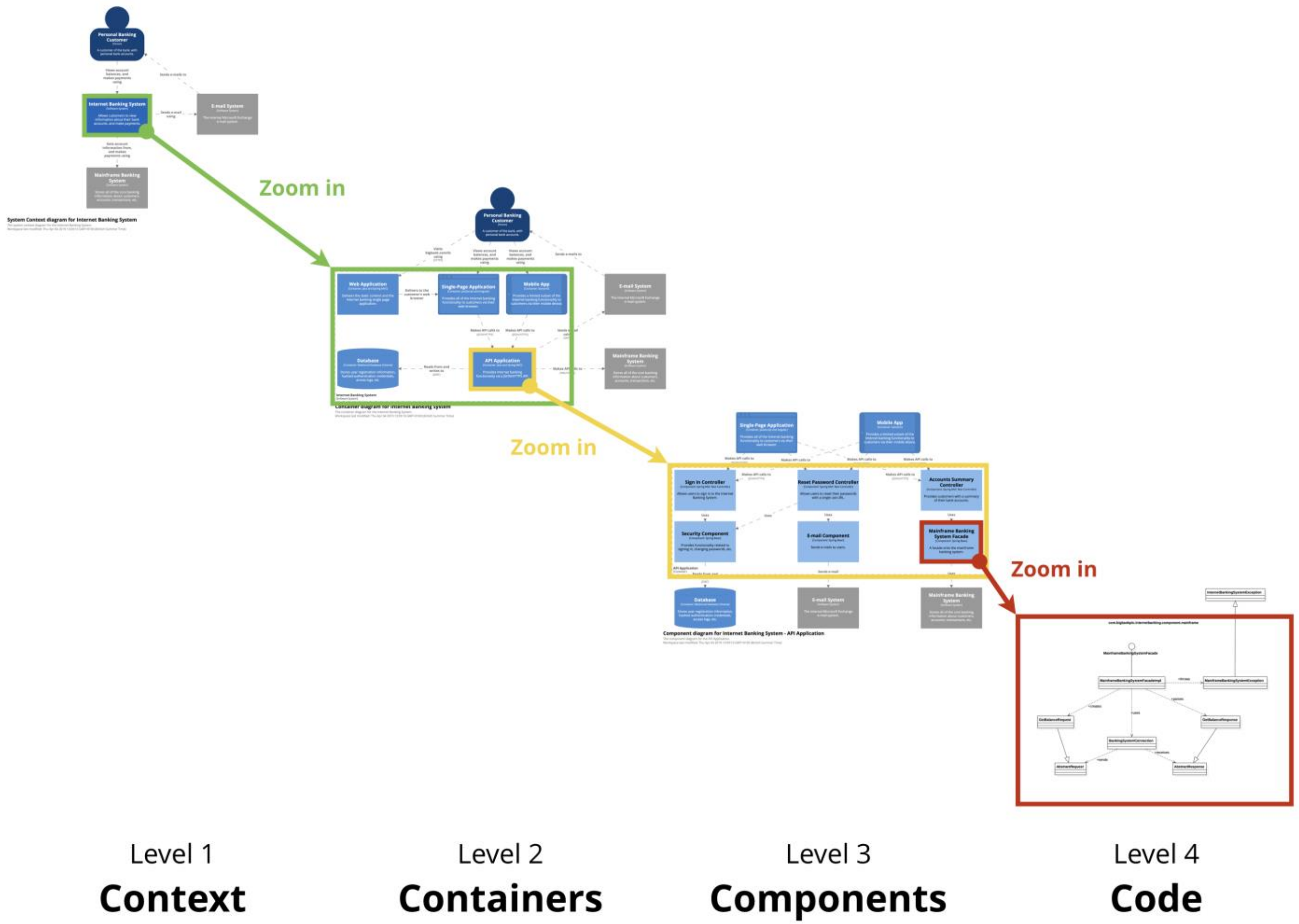


Diagrams are maps

that help software developers navigate a large and/or complex codebase

Source: Simon Brown C4model.com

I Model: C4 model III



Source: Simon Brown C4model.com

I. Model: Principles of the C4model

Start with simple boxes containing the
    + element name   + type
    + technology   + description / responsibilities

Most important thing in the middle

Favor uni-directional lines showing most important dependencies or dataflow

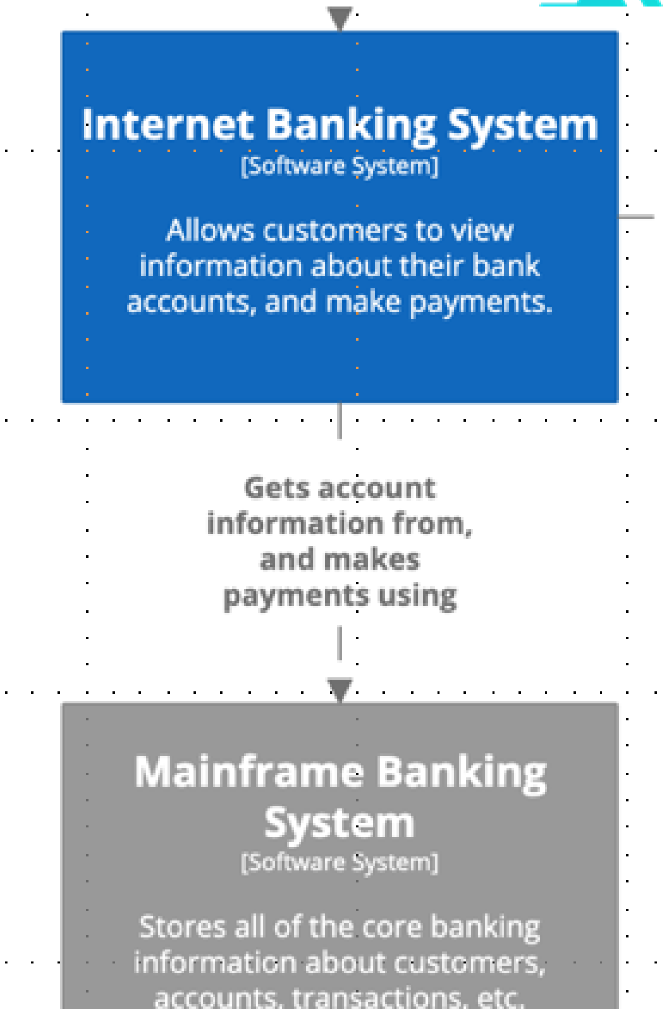
Use two uni-directional lines to describe two different use cases or asynchronous communication

Use an annotation to be explicit about the purpose of the line and the direction

Use the spoken description method. Arrow shows how sentence is built

Color: In-scope (e.g. blue) different as out-of-scope (e.g. gray)

Label intention, rather than (only) port/protocol


Internet Banking System
[Software System]
Allows customers to view information about their bank accounts, and make payments.
Gets account information from, and makes payments using
Mainframe Banking System
[Software System]
Stores all of the core banking information about customers, accounts, transactions, etc.

Source: Simon Brown C4model.com

Exercise 1

1. Create a context Diagram of the study case.
   - Set your TOE in the center
   - Identify Actors
   - Identify 3rd party systems
   - Connect them and label the connection
2. Create a container diagram of an MVP which would satisfy the requirements
   - Identify technology choices: cloud/on-prem, DB, services, etc.
   - Identify relationships between tech choices
   - Label the relationships (connections)

Case Study: Pet Store

Startup selling SaaS solution to Pet Stores to optimize their processes and have an online presence

“The etsy of Pet Stores”

Requirements

Pet Shop owners

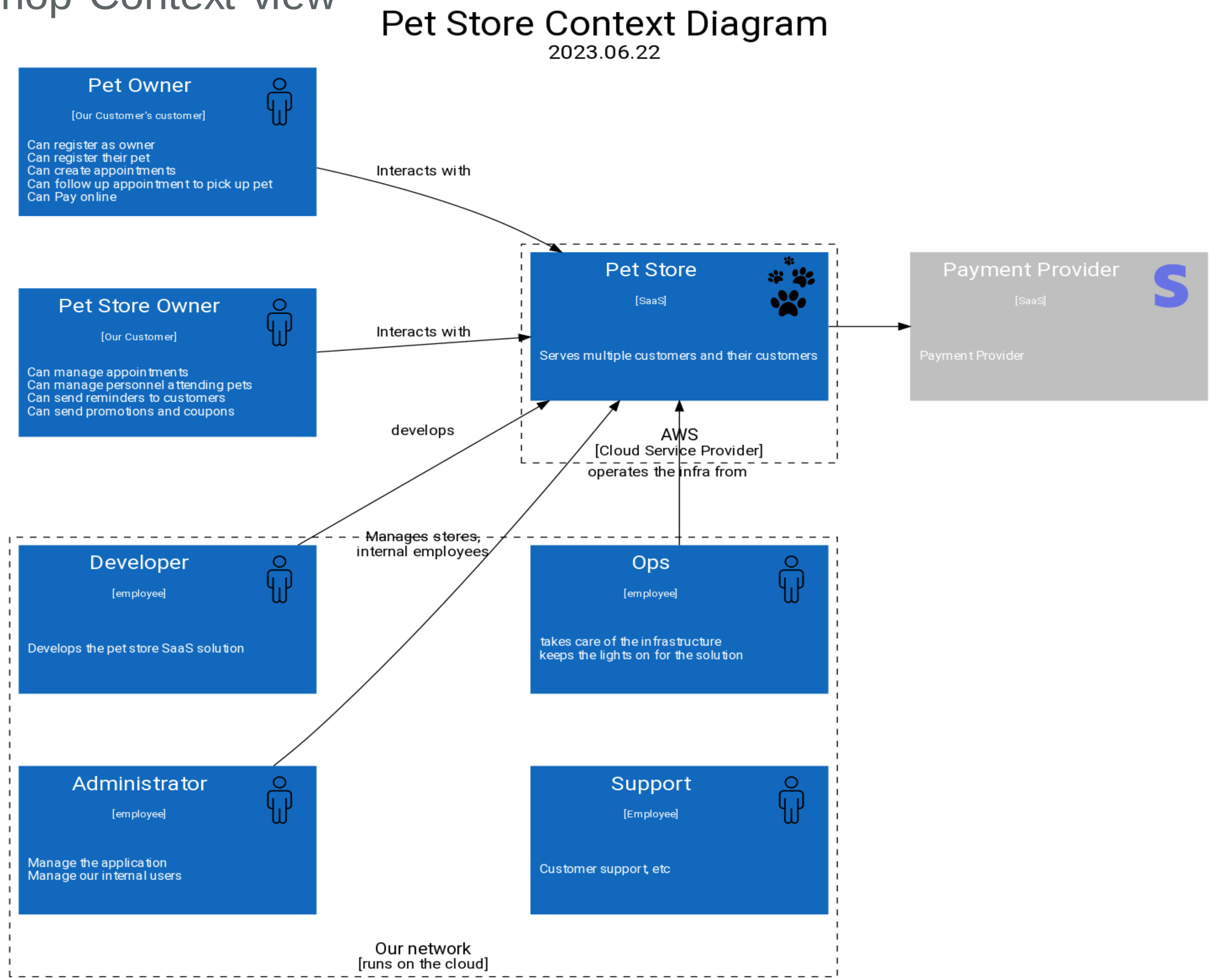
- Register their shop at superPets.com
- Get a subdomain as: *theirShop*.superPets.com
- Customize “their” shop
- Announce their services and specialties
- Can use the platform’s 3rd party payment provider
- Can manage their employees and appointments
- Can send coupons, promotions, and reminders to their customers
- Can respond to reviews

Requirements

Pet owners

- Can sign up with superpets.com
- Can select a pet store from their area
- Can register a pet (type, name, breed, age)
- Can upload an image of the pet
- Can manage appointments
- Can leave reviews for a pet shop
- Can pay online

Exercise 1: PetShop Context view



Pet Store Context Diagram

2023.06.22

Pet Owner
[Our Customer's customer]
Can register as owner
Can register their pet
Can create appointments
Can follow up appointment to pick up pet
Can Pay online

Pet Store Owner
[Our Customer]
Can manage appointments
Can manage personnel attending pets
Can send reminders to customers
Can send promotions and coupons

Pet Store
[SaaS]
Serves multiple customers and their customers

AWS
[Cloud Service Provider]

Payment Provider
[SaaS]
Payment Provider

Developer
[employee]
Develops the pet store SaaS solution

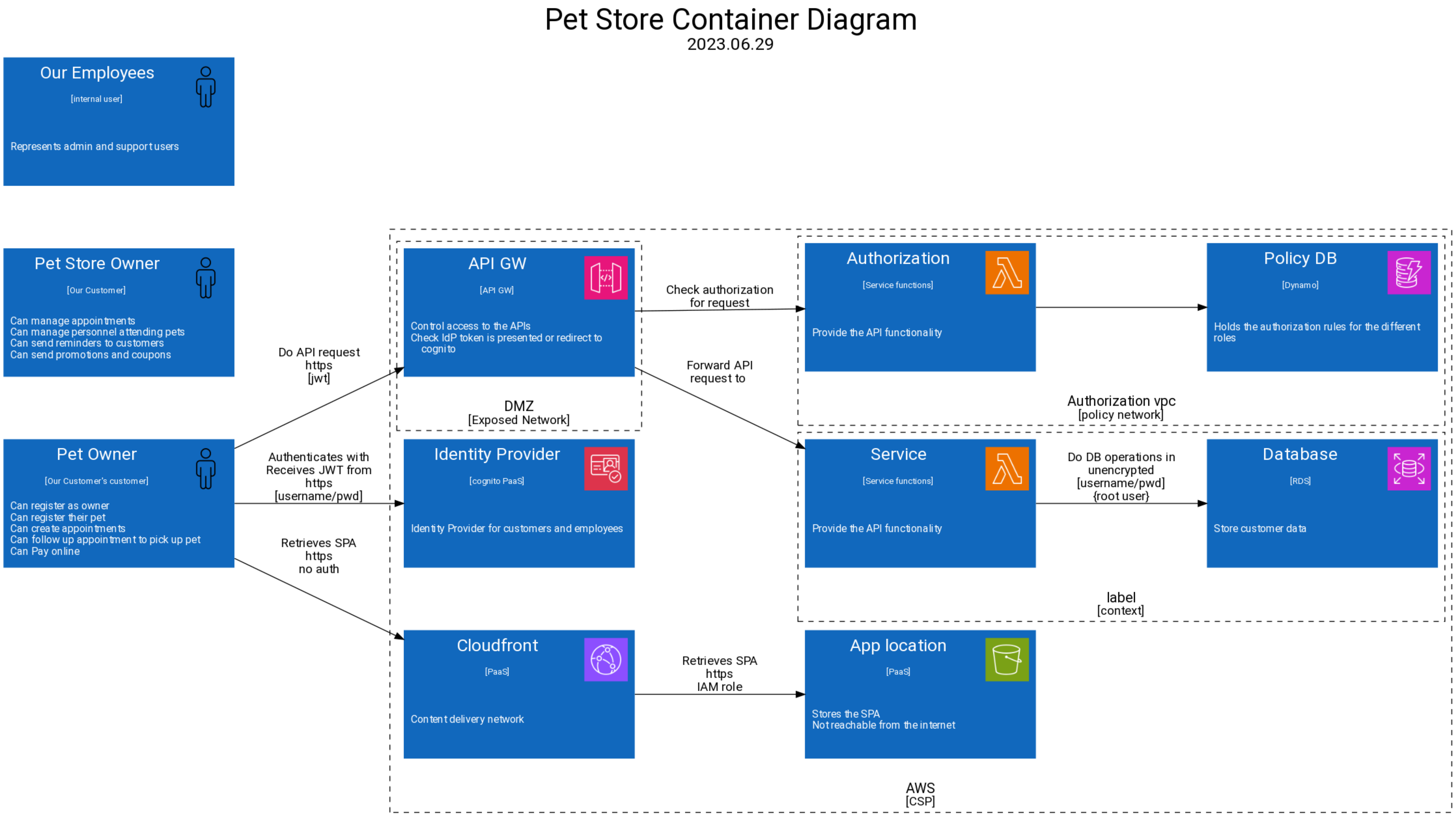
Ops
[employee]
takes care of the infrastructure
keeps the lights on for the solution

Administrator
[employee]
Manage the application
Manage our internal users

Support
[Employee]
Customer support, etc

Our network
[runs on the cloud]

Exercise1: Container View



Pet Store Container Diagram

2023.06.29

Our Employees
[internal user]
Represents admin and support users

Pet Store Owner
[Our Customer]
Can manage appointments
Can manage personnel attending pets
Can send reminders to customers
Can send promotions and coupons

Pet Owner
[Our Customer's customer]
Can register as owner
Can register their pet
Can create appointments
Can follow up appointment to pick up pet
Can Pay online

Do API request
https
[jwt]

Authenticates with
Receives JWT from
https
[username/pwd]

Retrieves SPA
https
no auth

API GW
[API GW]
Control access to the APIs
Check IdP token is presented or redirect to cognito

DMZ
[Exposed Network]

Identity Provider
[cognito PaaS]
Identity Provider for customers and employees

Cloudfront
[PaaS]
Content delivery network

Check authorization
for request

Forward API
request to

Retrieves SPA
https
IAM role

Authorization
[Service functions]
Provide the API functionality

Policy DB
[Dynamo]
Holds the authorization rules for the different roles

Authorization vpc
[policy network]

Service
[Service functions]
Provide the API functionality

Do DB operations in
unencrypted
[username/pwd]
{root user}

Database
[RDS]
Store customer data

label
[context]

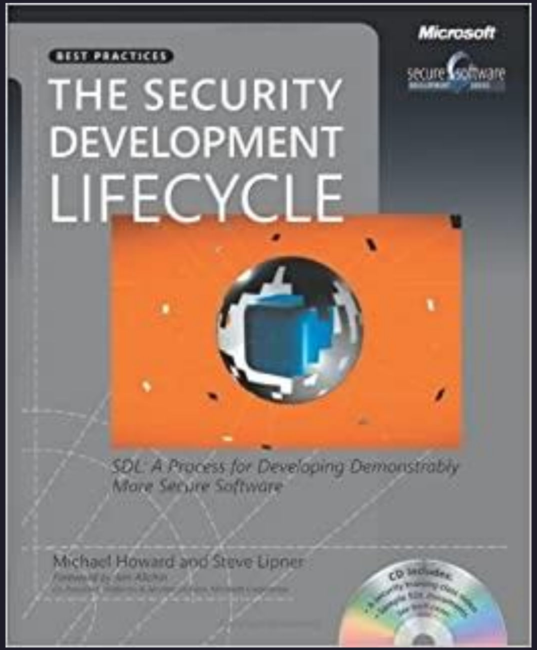
App location
[PaaS]
Stores the SPA
Not reachable from the internet

AWS
[CSP]

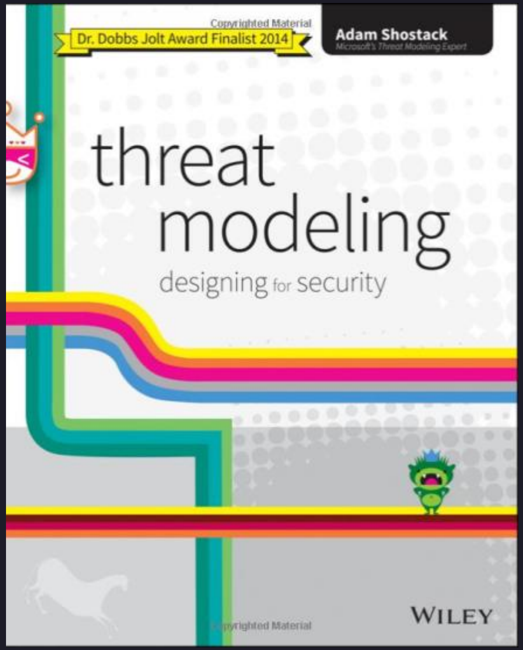
Thoughts on Threat Modeling or STRIDE != TM

A rose by any other name would smell as sweet – William Shakespeare

STRIDE has become almost synonymous with Threat Modeling.


BEST PRACTICES
THE SECURITY DEVELOPMENT LIFECYCLE
Microsoft
secure software
SDL: A Process for Developing Demonstrably More Secure Software
Michael Howard and Steve Lipner
CD includes:

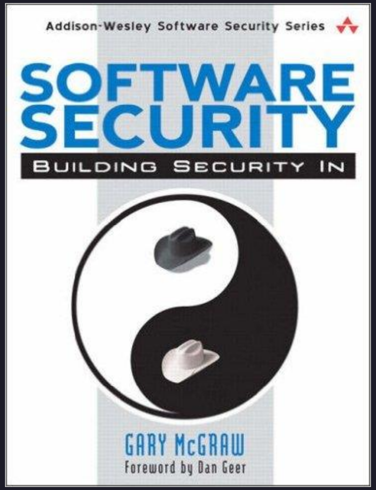
Howard, M. Lipner, S. *The security development Lifecycle.* Ch.9, 2006. Microsoft Press

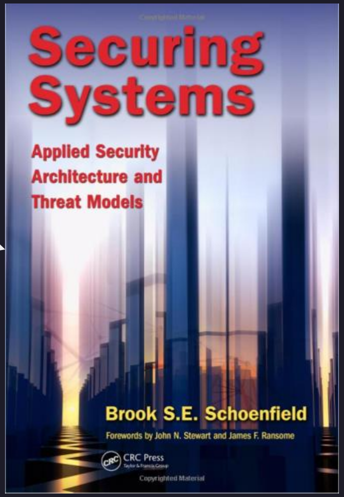


Shostack, Adam. *Threat Modeling: Designing for Security.* Wiley, 2014

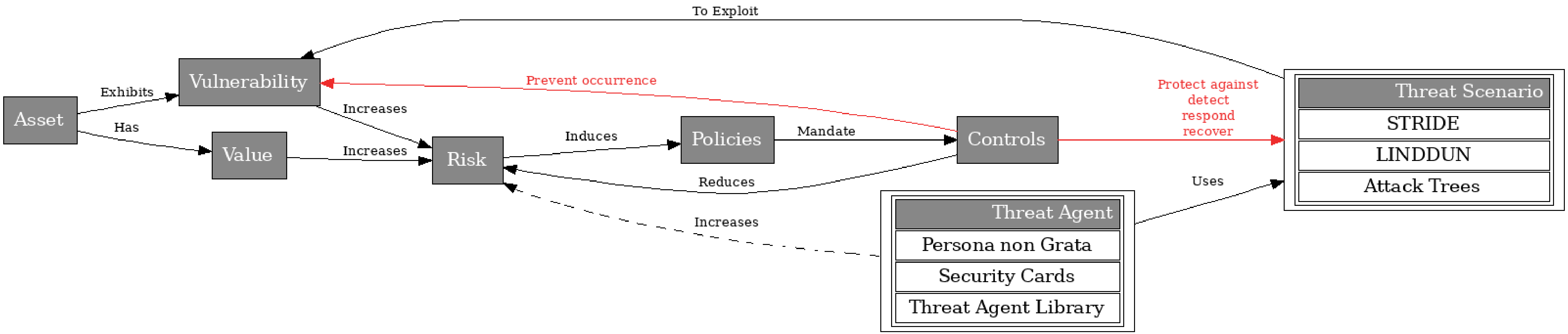
How many methods are there for Threat Modeling?

STRIDE
PASTA
LINDDUN
Attack Trees
Persona non grata
ARA – Architectural Risk Analysis
ATASM – Architecture, Threats, Attack Surface, Mitigations
Trike
OCTAVE




Securing Systems
Applied Security Architecture and Threat Models
Brook S.E. Schoenfield
Forewords by John N. Stewart and James F. Ransome
CRC Press

Threat Modeling in the context of Risk Mgmt.



Identify Vulnerabilities

Threat libraries help identify threat scenarios

- Examples: Elevation of privilege, elevation of MLSec, LINDDUN

What helps you identify vulnerabilities?

Technologies change, principles are perennial

IEEE – Top 10 Security Design Principles (2013)

1. Earn or give, but never assume, trust
2. Use an authentication mechanism that cannot be bypassed or tampered with
3. Authorize after you authenticate
4. Strictly separate data and control instructions, and never process control instructions received from untrusted sources.
5. Define an approach that ensures all data is explicitly validated
6. Use cryptography correctly
7. Identify Sensitive data and how they should be handled
8. Always consider the users
9. Understand how integrating external components changes your attack surface
10. Be flexible when considering future changes to objects and actors

Software Security Principles (1975)

I. **Economy of mechanism**
II. **Fail-safe defaults**
III. **Complete mediation**
IV. **Open design**
V. **Separation of privilege**
VI. **Least privilege**
VII. **Least common mechanism**
VIII. **Psychological acceptability**

Violations to security principles are indicators of vulnerabilities!

Credible Attack Vector (CAV)

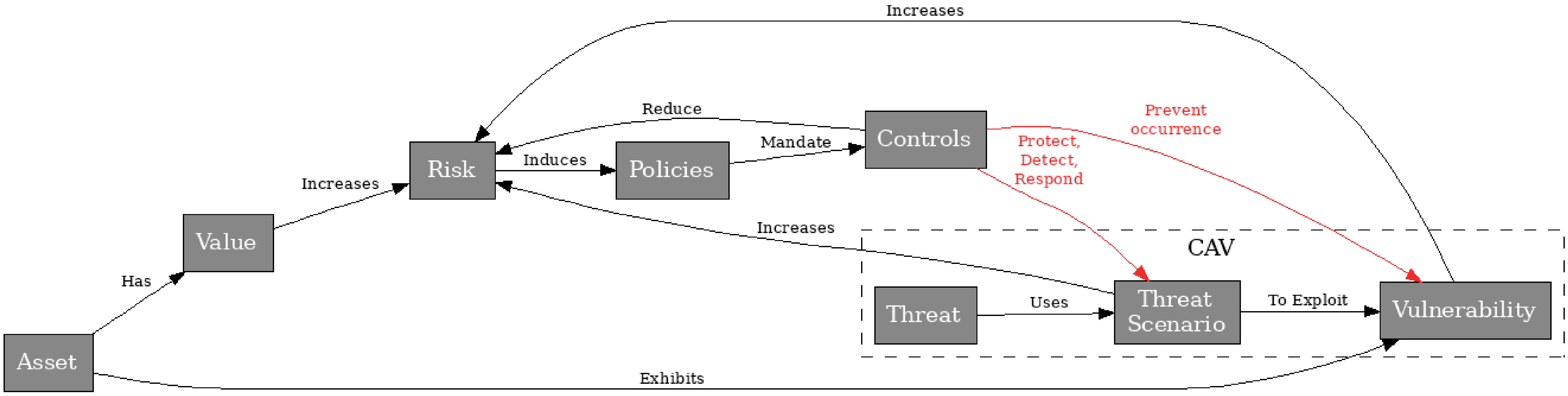
A credible threat exercising an exploit on an exposed vulnerability.

CAV = active threat agent & exploit & exposure & vulnerability & Damage

Who/Why	How	Where	What	Outcome
Capability Motivation Risk Appetite	Action to harm the Asset	Contact surface with the vulnerabilty	System Weakness	Mission Impact

* Term coined by Brook Schoenfield

CAV in the context of risk management



Credible Attack Vector

Describe CAVs using Gherkin:

- extremely logical,
- will probably be familiar to developers

Given *PRECONDITION*
When *THREAT SCENARIO*
Then *CONSEQUENCES*

Exercise 2: CAVs

- **Create 1-2 CAVs from the high-level requirements (attack the business ideas, not the implementation)**
- **Translate some of the Elevation of Privilege threat scenarios into CAVs that could apply to the petshop.**
- **For every entry define one or more countermeasures**
- **Fill out the table.**

- **Example**:

Tampering

2: An attacker can modify your build system and produce signed builds of your software

Tampering example

2: An attacker can modify your build system and produce signed builds of your software

ID	Vulnerability	Threat scenario	Countermeasures
1	Internet-facing build system with default admin credentials	GIVEN the build system is internet-facing and has default admin credentials WHEN an adversary abuses these credentials THEN changes can be introduced to the build pipelines without anyone noticing.	1. Admin access should force Mfa 2. Admin access must rotate keys/passwords upon setup and on a regular basis. 3. If possible, restrict admin access to require a VPN or jump host
2	Unsigned commits are allowed	Given developers are not signing their commits, WHEN anyone with access to the system pretends to be another user THEN it is possible spoof the victim and repudiate the commit	1. Force signed commits in all repos
3	Missing merge request requirement	GIVEN the CI pipeline is defined in code, WHEN an adversary with access to the repository makes a change to the pipeline THEN there is no peer review happening AND then change makes it to production	1. Force protected main branch, peer reviews, and merge requests into the main branch
4	Lack of MFA for dev login	GIVEN developers are accessing the code repository without MFA, WHEN a user password is reused among sites and leaked OR easily guessed, THEN an adversary can log in and make changes to the code base.	1. Force MFA for all access to SCM 2. Force SSO for all access to SCM

Need more inspiration?

Here you can find other katas:

https://github.com/lfservin/oss-threatmodeling

And here there are the accompanying videos (Threat modeling katas 1-4):

https://open-security-summit.org/participant/organizers/luis-servin/

My writeup of last year's hackathon:

https://github.com/lfservin/threatmodel-hackathon/blob/main/writeup/writeup.md

Thank you for your participation!