AppSec
Threat Modeling 101

Workshop objective

Help teams to build and operate more secure systems by incorporating threat modeling into their daily work.

Please note

- Threat Modeling can be done in ♾️-ways
- In this workshop we will teach one selected approach.
- We believe this will be a good starting point for most teams.
- As teams increase their threat modeling muscles - we assume they will explore alternative ways.

❗ "There is no end state for application security, we just learn and improve."

Getting the most out of the workshop

- Learning happens when you do things you do not yet understand
- Be present (🚫📱💻🎣)!
- Assume good intentions (😇)
- Value diversity in experience and opinion (👍)
- Help creating a safe environment of trust and respect (🦺)
- Context matters a lot.
- Your perspectives matters!

Who are we?

- Equinor's AppSec Team
- #AppSec @ Slack
- https://equinor.github.io/appsec/
- Introducing the instructors 👨🏻‍🏫🧑🏼‍🏫👩🏽‍🏫

Workshop Outline

- Threat modeling introduction
- What are we working on?
- What can go wrong?
- What are we going to do about it?
- Did we do a good job?
- Threat modeling *the* SDLC
- Getting started with Threat modeling in your team
- Wrapping up

Practicalities

- Workshop channel on Equinor Slack workspace is #appsec-threatmodeling-workshop
- Schedule
  - 09:00 - 16:00
  - 12:00- 12:30 Lunch
  - 5-10 minutes break every hour

    - or as an part of the exercises.

    (🎗️Please remind us if we forget the breaks)
- All content will be available after the workshop.

Threat modeling Introduction

🖖

What is threat modeling?

From the Threat Modeling Manifesto:

Threat modeling is analyzing representations of a system to highlight concerns about security and privacy characteristics.

Ideally, we want to identify weaknesses as early as possible.

❗ Threat modeling "must" be a cyclic/continuous effort, not a one off activity.

Why Threat modeling?

- Develop more secure systems (what we develop)
- Develop more securely (how we develop)
- Operate more securely (how we operate)
- Recognize what can go wrong in a system
- Pinpoint design and implementation issues
- Enabled informed decisions on threats and mitigations.
- Educate developers and teams, share knowledge
- Have a systematic and consistent approach to security

❗ Threat modeling will guide our designs
and help us make decisions with our eyes open.

❓ What are your <u>current</u> reasons for wanting to Threat Model?

The 4 Key Questions of Threat modeling

- What are we working on?
- What can go wrong?
- What are we going to do about it?
- Did we do a good job?

🕵🏻‍♂️ The Threat modeling Manifesto

On models

"All models are wrong, some models are useful"
- *George Box.*

We acknowledge that; models always fall short of the complexities of reality - but can still be useful nonetheless.

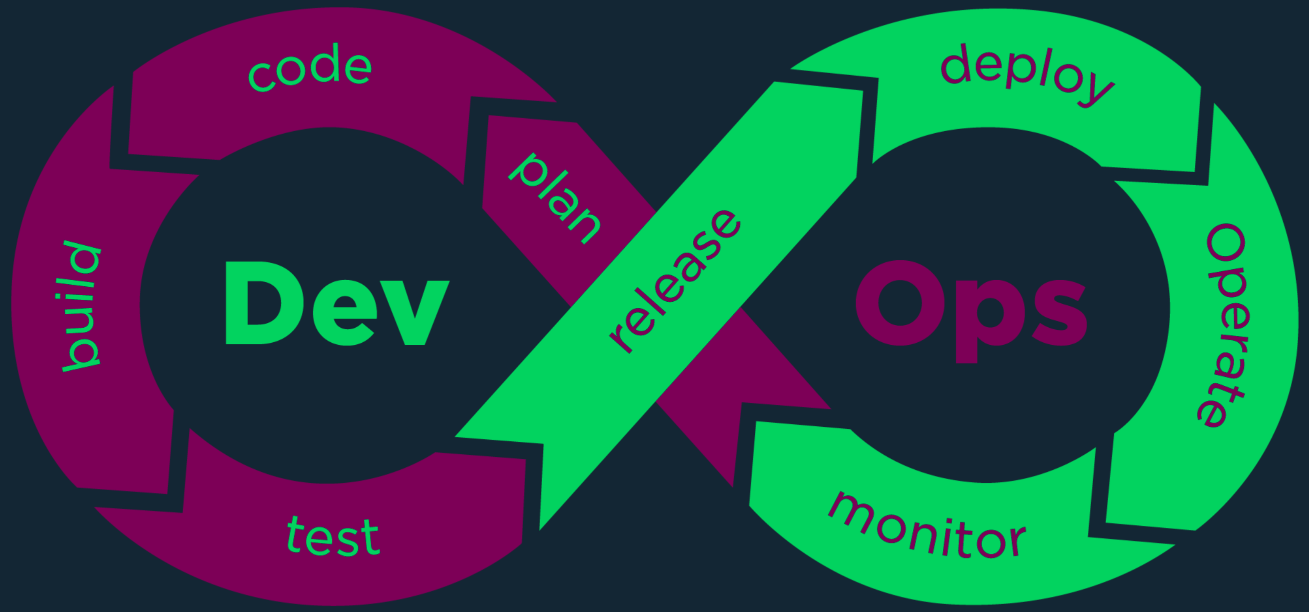
It starts with security requirements

- All systems must have documented security requirements (SR)!
- SR will guide and inform our threat modeling.
- In all/most organisations the governance will be a good starting point for identifying SR
- Legal and governmental requirements will also provide SR
- Threat modeling will often trigger update of SR requirements
- External sources for "inspiration"
  - OWASP SAAM
  - OWASP ASVS

❓ What are typical security requirements in <u>your context</u> ?

When to threat model?

- Threat Modeling of a system usually happens in the early phases of the SDLC (DevOps)
- Threat Modeling must include the SDLC (The DevOps Cycle)
- Threat Modeling must be a continuous effort



Threat modeling is much like brushing your teeth 🪥;
- daily short sessions -

Basic terminology

- Weakness; is an underlying defect that modifies behaviour or functionality or allows unverified or incorrect access to data (Common Weakness Enumeration)
- Exploitability; is a measure of how easily an attacker can make use of a weakness to cause harm.
- Vulnerability; when a weakness is exploitable it is known as a vulnerability (Common Vulnerability Enumeration)
- Severity; The potential damage and "blast radius" of a weakness to a system (Common Vulnerability Scoring System)
- Threat; A future problem, the possibility that something unwanted will happen

Misconceptions

- "Think like an attacker" =>
  Who knows how an attacker think?
  Focus on serious work, structure and consistency
- "You never done threat modeling" =>
  You do it all the time!
- "Threat modeling is easy" =>
  No, it requires continuos planned effort to build a muscle.
- "Threat modeling is for specialists" =>
  No, every team role plays it's part, everyone should participate
- "Threat modeling as ONE skill ..." =>
  Threat modeling is experience, techniques (DFD, STRIDE,..),
  repertoire, (tools, books, blogs) ... just like software development

Rabbit holes

- "Too much focus on "how-to" TM (tools, frameworks,)..." =>
  Just do it! Focus on people, skills, perspectives, development, operations ...
- "Admiring the problem, going too deep" =>
  Maintain the bigger picture, avoid exaggerating attention on problem, adversaries, assets or techniques.
- "Searching for the perfect model" =>
  It does not exist! The better approach is multiple smaller models representing multiple views?

❗ Threat modeling can be the most effective way to drive security through a product, service or system.

What are we working on?

🏗️

Exercise 0

❓ What pops into your head about "windows"?

- Write it down silently (1 minute)
- What did you write down?

Purpose of ™Models / Diagrams

- Diagrams expose thinking and triggers discussions
- Diagrams are scoping tools
  (what's in, out, boundaries)
- Scope may be this sprint, this story,
  this project, this feature, ...

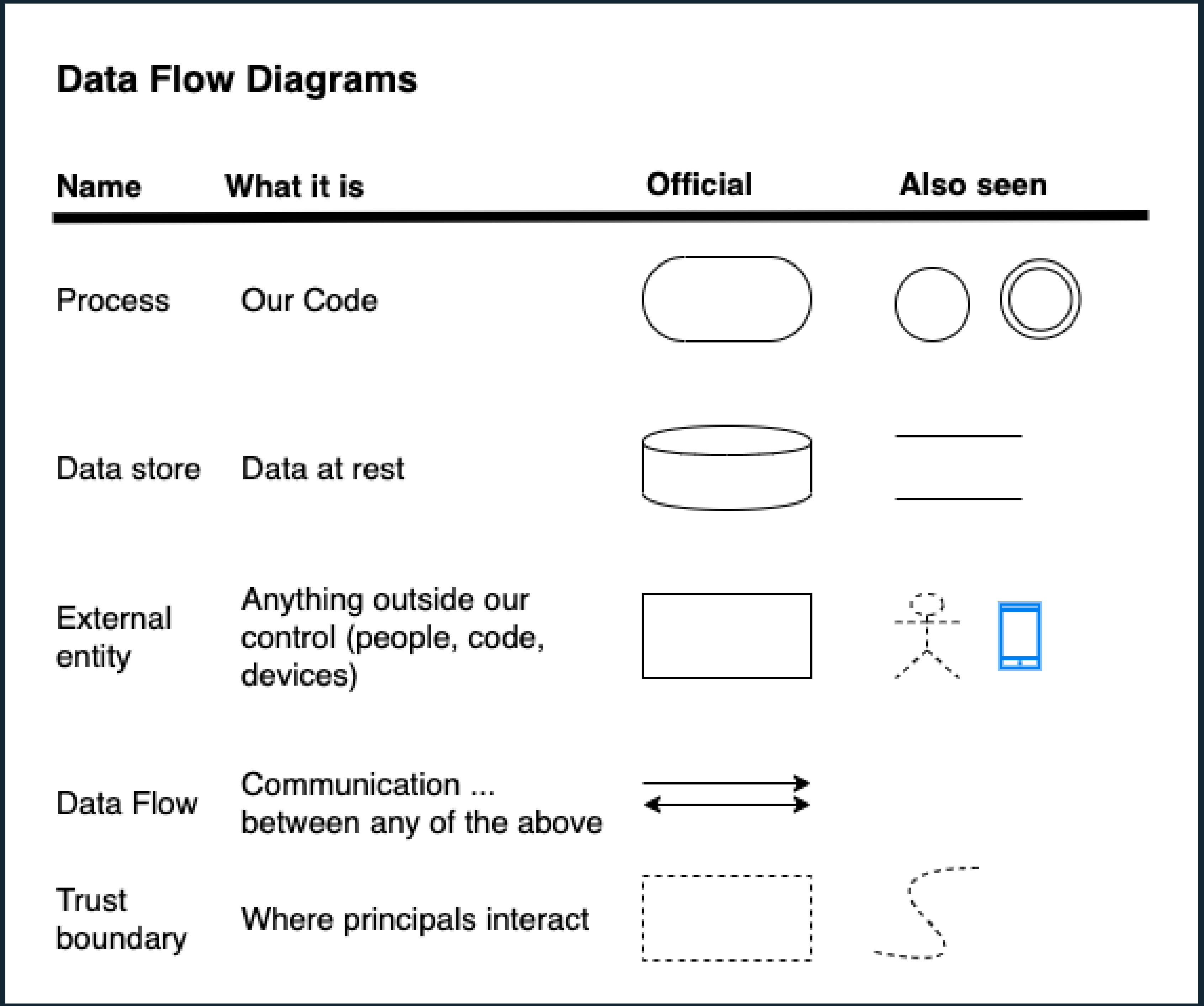
🏁 Pick up a pen and start drawing,
tell a story, show data flow and boundaries

DFD (Data Flow Diagrams)

- Threats often follow data => Data Flow Diagrams
- DFD are simple, easy to learn and sketch
- Whiteboards are excellent tools

❗ We use data flow diagrams because they provide us with a simple representation of how data, and thus threats, flow through a system.

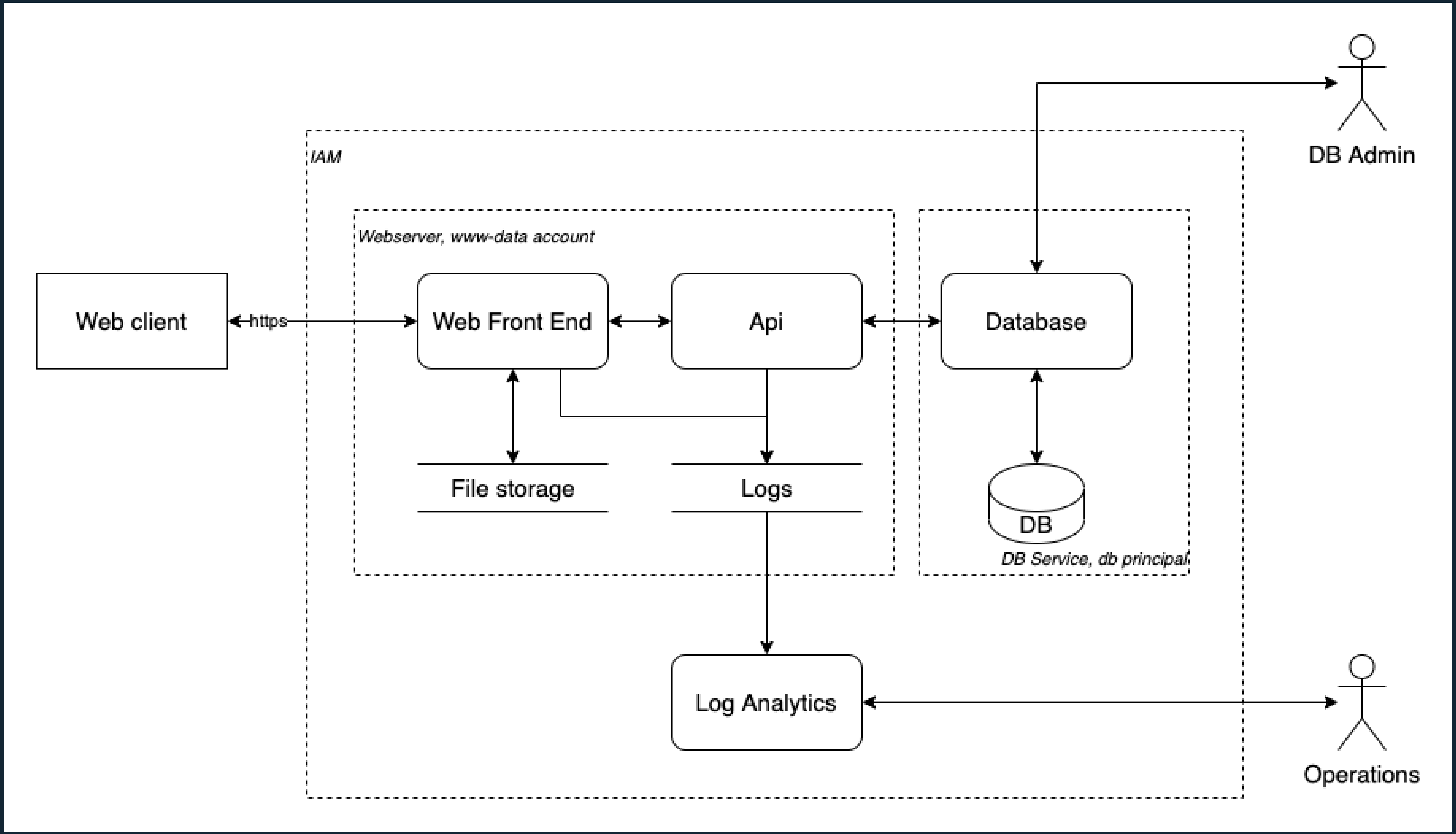
DFD components



Data Flow Diagrams

Name	**What it is**	**Official**	**Also seen**
Process	Our Code	⬭	○ ◎
Data store	Data at rest	⛁	═
External entity	Anything outside our control (people, code, devices)	▭	🧍 📱
Data Flow	Communication ... between any of the above	→ ↔	
Trust boundary	Where principals interact	⬚	⌇

DFD - An Example



Trust boundaries

- Trust boundaries are where different users interact
  - Where principals interact
    - Users, Apps, Identities
    - A principal is often the smallest unit you can name in a policy
  - The mechanism that isolates them
  - Policies are enforced at boundaries
- Trust boundaries needs to be explicit about what, where, how.
- Trust boundaries needs agreements
  (how to enforce, configure, test)
- Trust boundaries should be labeled

💡 Examples: File permission - enforced by kernel, network hosts - enforced by firewall, app permissions - enforced by IAM and app

What do we record/store?

- Think lifecycle and maintenance for models,
  store some for the records - some for keeping up-to-date?
- Effort grows with formality,
  legal requirements, regulations, GDPR, ++
- Store with code, in separate repos, in sharepoint, ...?
  Explore options. Some data could be sensitive

Examples:

- Whiteboard => Pictures or Drawings (draw.io) stored with project
- Collaborative tools like Miro could be ok
  - Don't underestimate the learning curve and the "tool trap"!
  - Provide some Miro training before starting to TM
  - Remember information sensitivity!
- External systems like Irus Risk, OWASP Threat Dragon, Draw.io

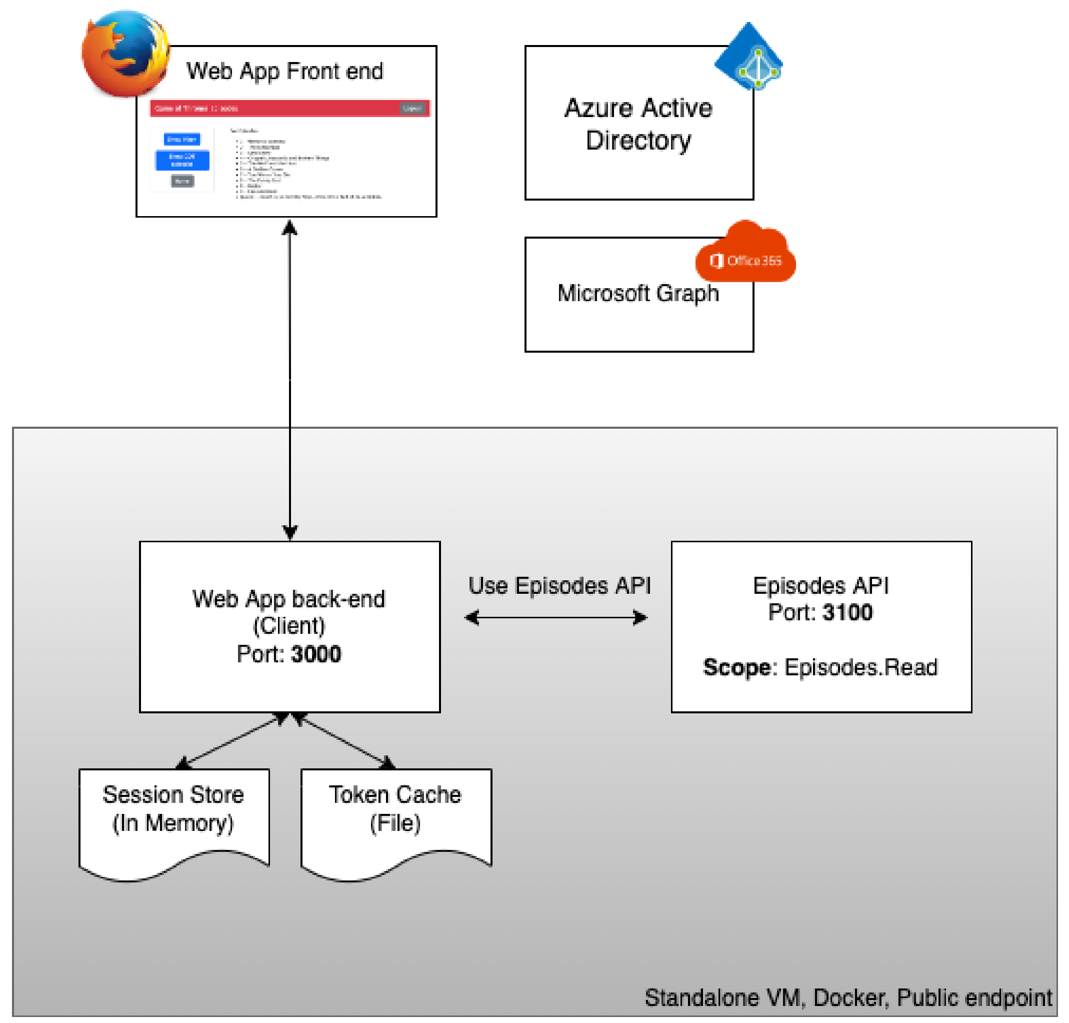
Exercise 1 - Creating a model

Organizing groups and work

- We form groups (4-5 persons)
- The group will stay together for all exercises
- The exercises build on each other
- We draw on A3 sheets (boards, flip-overs or similar)
- We use thick-✏️ when drawing (it's easier to read)
- We make notes on paper
- Groups take a picture of drawings and notes and share in Slack channel #appsec-threatmodeling-workshop before presenting
  - We will delete most of these posts from the Slack channel after the workshop
- Make sure to take a short round of introduction in the groups 🤝

🚶 Reshuffle groups now if needed 🚶

Our example system



Functionality

Web app
- Log in
- List context of O365 Inbox
- List episodes from "Episodes API"
- Log out

Episodes Api
- Receive request to get list of episodes
- Validate request and serve list if ok

Assumptions

- Standalone Linux VM in Azure
- Linux is standard Ubuntu 18.10 out of the box
- Running in dedicated resource group
- Public endpoint for web app at port 80/http
- Public endpoint for ssh and admin at port 22
- Components are running in containers
- Docker engine is running as root
- Internal comm is http
- All admin is manual using ssh
- The Web App is using a backend-for-frontend (BFF) pattern
- The Web App is using oAuth2/OICD Code Grant for authn/z

Office365 Inbox is classified as confidential. Episodes api is classified as "internal". The business is considering making the episodes information open - but if so the business would like to know a lot about those who access the episodes data.

Security Requirements

1. Information shall be encrypted at rest
2. Information shall be encrypted at transport
3. Information security classification must be known
4. RBAC shall be used to access information
5. Systems shall use company provided IAM
6. Access to all internal, restricted or confidential information shall be authorized
7. App to app permission shall be recertified at least once per year
8. Access to system credentials, tokens etc. shall be restricted and controlled
9. System shall implement logging and monitoring to detect unauthorized access
10. Logs shall be protected from unauthorized access after recording
11. All system shall be scanned for vulnerabilities regularly
12. Application shall generate new session tokens on user authentication
13. Session stores shall be secured
14. Session cookies shall only utilise secure transport
15. Session tokens shall be generated using approved cryptographic algorithms
16. Caching of sensitive data shall be protected
17. A development shall use a Secure SDLC.
18. Secure Development Policies shall minimum include:
    1. Protection of source code
    2. Code review
    3. Code scanning
    4. Security by design
    5. Security in depth
19. Development environments shall be secured.

EX-1: Drawing a data flow diagram

Tasks:

For our example system

- Examine the system, the assumptions and security requirements
- Draw a DFD for the system
- Document any assumptions you make
- Take a picture of your DFD, share on Slack
  and prepare to present the model to the class

The person in the group which has the next birthday becomes group lead for this exercise.

⏰ Time boxed schedule (20m):

- 3 minutes to discuss and clarify system
- 15 minutes to create DFD
- 2 minutes to take picture and share on Slack

Remember: "All models are wrong. Some models are useful"

EX1: Presenting models

Each group present their DFD along with assumptions

❓ Reflections, Observations, Learning

What can go wrong? 💥

Introducing STRIDE

STRIDE is a model for identifying computer security threats. It provides the following mnemonic:

Threat	Violates	Definition	Example
Spoofing	Authentication	Impersonating someone or something	Pretending to be you manager, a CFO, bankid.no, ntdll.dll, expressJs
Tampering	Integrity	Modify data and code	Modifying a file, changing code in a NPM repo, change a packet as it traverses the network, change a binary
Repudiation	Non-repudiation Accountability	Claiming to not have performed an action	"I did not send that email", "I did not make that change"
Information disclosure	Confidentiality	Exposing information to someone not authorised to see it	Publish a list of customers to a web site, allowing someone to read source code
Denial of service	Availability	Deny or degrade service to users	Crashing a web site, eat all memory for a program, route packages to void
Elevation of privilege (Expansion	Authorization	Gain capabilities without proper authorization	Allowing a remote internet user to run commands, XSS, SQL Injection, RCE, going from limited to admin user

How to use STRIDE?

- Use S-T-R-I-D-E mnemonic to identify threats to system
- Ways to use:
  - Follow a user story through the diagram,
    look for STRIDE threats, iterate
  - Focus on an element, component, parts of a system
    and apply STRIDE
  - Looping the diagram:
    - For all elements in the diagram -> for each (STRIDE) threat ->
      specify how threat to element works
    - For each S-T-R-I-D-E threat -> for each element in diagram ->
      specify how threat to element works
- As you learn -> iterate

❗ There are quite a few alternatives to STRIDE. We choose to focus on STRIDE when introducing Threat Modeling to teams. It has proven to be useful in many relevant scenarios.
Some alternatives: MITRE ATT&CK, Attack Tree, PASTA, DREAD, VAST, STRIKE ..

STRIDE per element

Part	Spoof	Tamper	Repudiation	Info disclosure	Deny Service	EoP
External entity	X		X			
Process	X	X	X	X	X	X
Data store		X	?	X	X	
Dataflow		X		X	X	

🕵🏻 The Elevation of Privileges game (EoP) is helpful and fun.
The AppSec Team can provide an introduction to EoP.

Tracking threats and assumptions

- Track issues as they are found
- Track assumptions as they are discoverd
- Issues and assumptions are input to "What are we going to do about it?"
- Recommended practice:
  - Appoint a note taker
  - Record meetings / sessions
  - Create a team strategy for how to document, what to document, where to store ++. Experiment and iterate

---

❗ The best tools are the one that works for the people involved, for the project you are working on!

What could go wrong - brainstorming

(As an alternative/add-on to using STRIDE)

Some useful questions/statements:

- "What is The one thing you are worried about?"
- "How would you attack your system?"
- "Remember we have technical/security debt?"
- "Last iteration we made this temporary solution ...."
- "I read through ASVS and was wondering how we ...."
- 🚫 Think like an attacker

❗ There are many guides and books available discussing the mechanics of threats. There are threats libraries available. It's usually a good idea to augment our threat modeling with these aids after we have reached a bit of maturity

Exercise 2 - Identifying threats

EX-2: Applying STRIDE

Tasks:

For our example system, using your DFD for whats currently implemented:

- Examine the system, the assumptions and security requirements
- Apply STRIDE, document issues, assumptions and threats
- **Team decide scope**, iterating as you are learning is smart!
- Document format suggested (👇)
- Take a picture of your documents, share on Slack
  and prepare to present the model to the class

The person in the group which last bought a car becomes group lead for this exercise.

⏰ Time boxed schedule (30m):

- 30 minutes to apply STRIDE

💡 Try to avoid rabbit holes. Stop early. Iterate

EX-2: Document example

Part	STRIDE	Threat action	Impact	#
Web app	Spoof	Attacker steals session cookie	Information disclosure, access to all emails	1
Token Cache	Information disclosure	Attacker is stealing O365 access tokens	Information disclosure, system integrity, GDPR …	2
..	..	..	..	3
..	..	..	..	4

Assumptions:

- There are no key stores used
- Storage - VM discs are not encrypted

Issues:

- Should internal traffic be https as well?
- What about PKI infrastructure for http certs?

EX-2: Presenting threats

Each group present identified threats

- STRIDE strategy
- Key assumptions
- How were the Security Requirements used?

❓ Reflections, Observations, Learning

What are we going to do about it?

Tactics, Strategies, Alignment, Prioritisation, Documentation

Addressing threats

- Address each threat - decide on strategy/tactics, document
- Strategies are: eliminate, accept, transfer or mitigate
  - Eliminate; tactic:remove -> example: remove code, component...
  - Accept; tactic: document acceptance, transfer risk to customer?
  - Transfer; tactic: "UI,license,insurance,..", document and track
  - Mitigate; tactic: add controls
- Check/verify assumptions

💡 Threat vs. Risk: A threat is a future problem. A risk is a threat with probability and impact. We do not need probability and impact to manage threats!

Mitigation

- To mitigate means to add controls that address the threat
- Controls are features or technologies
    - Use technology before people or process!
    - Protect, detect or respond to threats
- Mitigation tactics could be
    - Code by developers
    - Signals to operations
    - Products / services
    - Complete or partial
    - Strong or weak
- The right tactics will always be situational/context dependent
- Custom controls = New Security Tech - avoid?
- Explore standard as the first choice
- 💡 OWASP Proactive Controls

Managing "What are we going to do about it"

- Document
  - Write down and track threats (use teams tool chain)
  - Track as bugs, features, security stories, security debt
- Prioritize
  - Align with reality, team, product owner
  - Prioritization always includes effort to fix - and operate.
  - Agree on severity categories:
    example sev1 means fix ASAP, sev2 means within 7 days ....
- Implement
  - Add to team backlog
  - Don't re-invent the wheel
  - Assessing implementation will be a lot easier if
    a larger part of team participate in threat modeling
  - Code tests to identified threats - that provoke threats -
    to trigger controls, signals to operations ...?

Exercise 3 - Managing Threats

EX-3: Threat management

Tasks:

For **some** of the threats identified in the previous exercise:

- Select a strategy; at least a few should be "mitigation"
- Discuss realistic fix'es and document them (controls)
- Document format suggested (👇)
- Take a picture of your documents, share on Slack and prepare to present the model to the class

The person in the group which last went to the movies becomes group lead for this exercise.

⏰ Time boxed schedule (20m):

- 20 minutes to discuss what to do about threats

EX-3: Document example

ID	Threat #	Threat summary	Strategy	Fix idea	Pros/Cons
1	Stolen session cookie from front end	An attacker is able to steal the session cookie of a valid user session	**Mitigate**	Assess terminating session if sudden change in access ip	Pro: Could fix the problem Con:May introduce errors if user is access from multiple clients
2	Token cache exposure	An attacker is able to access token cache on the back-end and thus access service impersonation a user/service	**Mitigate**	Move token cache to external service, encrypt, add logging of all access, lock down permissions	Pro: Would reduce the risk Con: Adds complexity, cost and

EX-3: Presentations

Each group present their threats

- Threats
  - Strategies
  - Fix ideas
  - Pros/Cons
- How were the Security Requirements used?

❓ Reflections, Observations, Learning

How to prioritise threats?

Threat vs. Risk:

- A threat is a future problem.
- A risk as a threat with probability and impact.
- We do not need probability and impact to manage threats!

- Some threats should be prioritized by the team (a team decision)
  - Find guidance in the security requirements
- Some threats MUST involve the business/product owner in prioritising
- Business/Product Owner, Team and potentially other support should be involved with "identifying" threats
- Important to know "who owns the risk and can accept it on behalf of the company?"

❓ Reflections

Did we do a good job?

👀

Threat Modeling Retrospectives

- Do Threat Modeling retrospectives!
  - As part of teams retrospectives?
  - As separate sessions?
  - Where do you naturally evaluate topics like quality,standards,security code review++?
- Did we find any threats?
  - Example: Have goals like "2 STRIDE threats per element" - and how did it go?
- How many of the identified threats have we been able to handle? (closed vs open)
- Did our mitigations work?
  - Do we have any incidents?
  - Any new bugs related to our mitigations?
- Are we (the team) happy with our way of doing threat modeling?
  - Any experiments we should try out?
- 🤔 Some have suggested -
  "spend as much time on retro's as you do on threat modelling"

❓ Reflections

Threat Modeling the SDLC

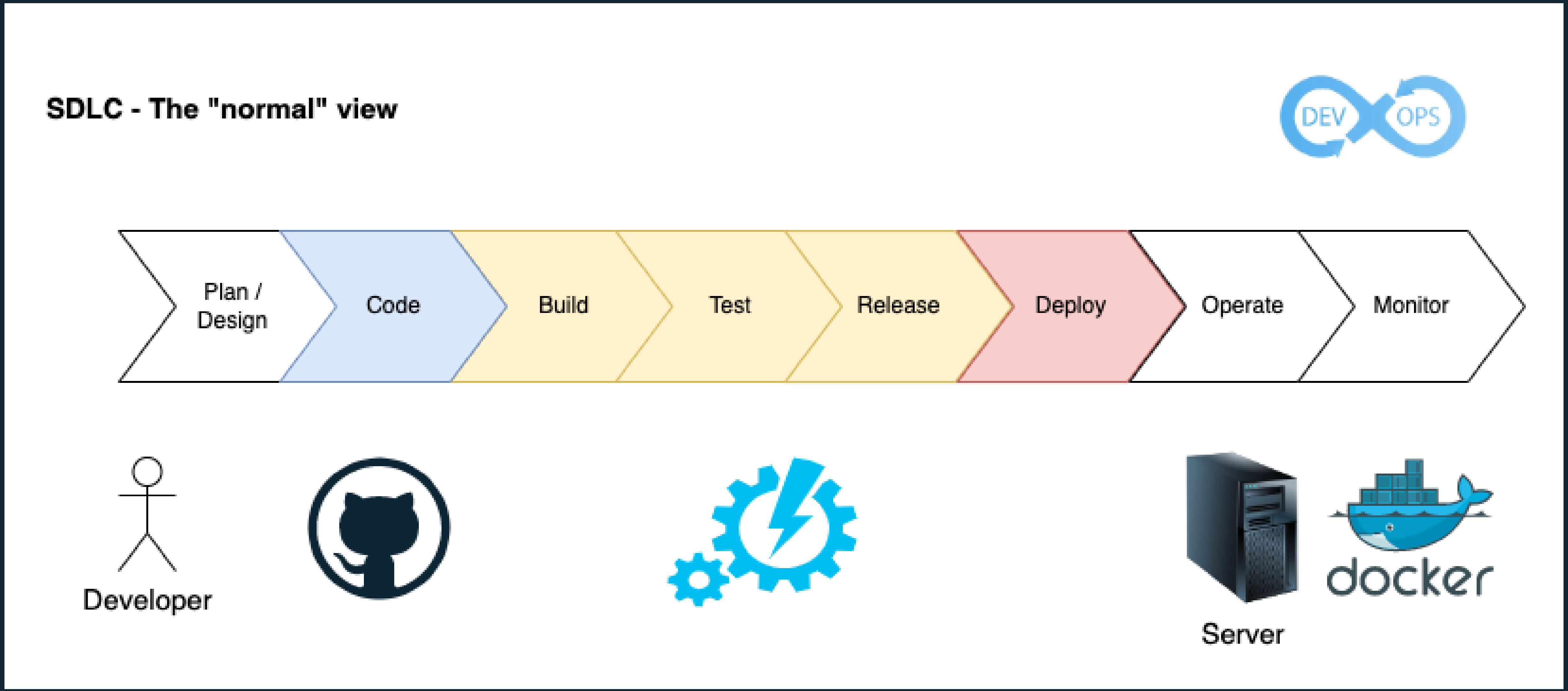
∞

SDLC = Software/System Development Life Cycle

Current Threat Modeling effort

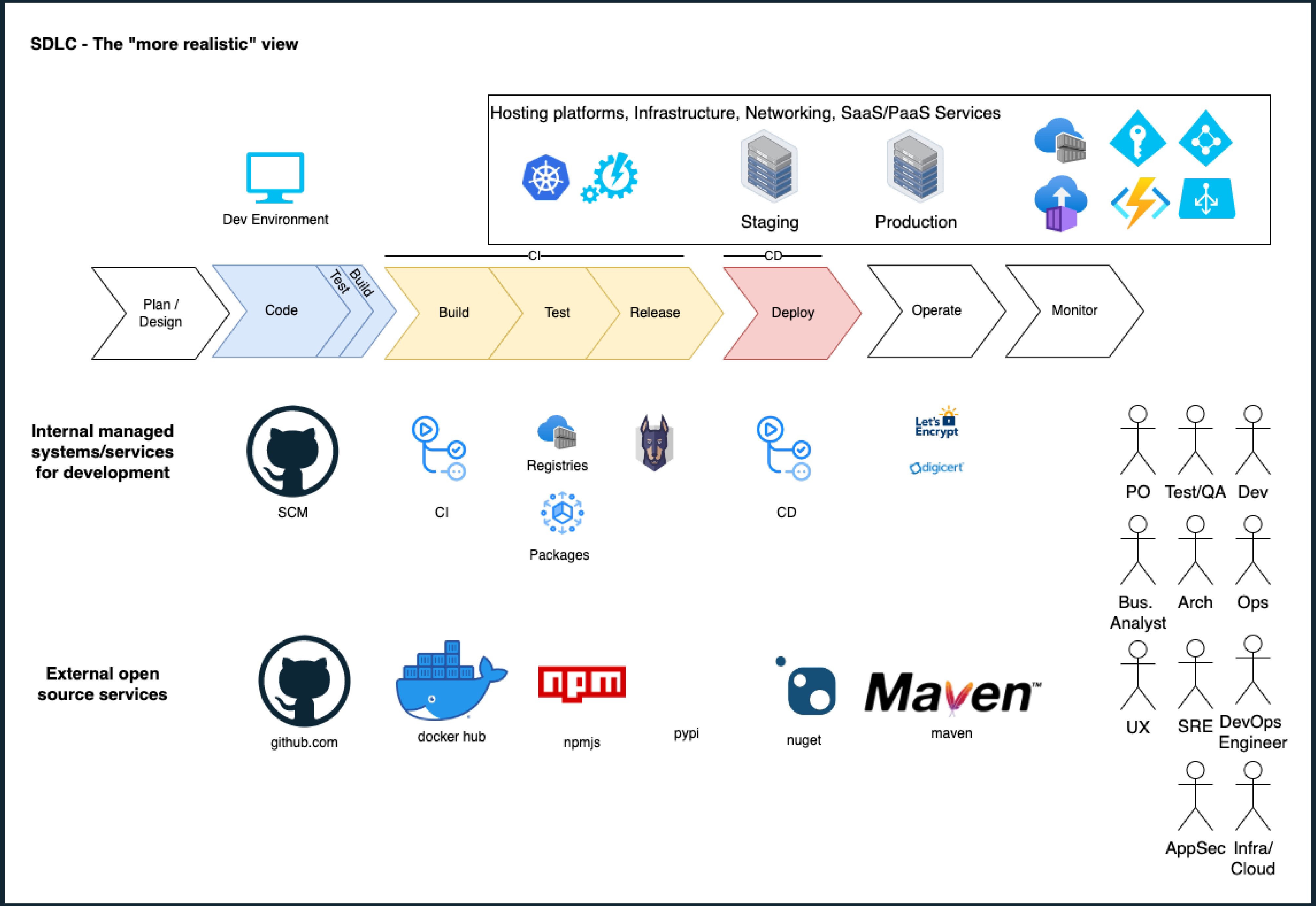
- Most security effort is going into security for the systems that we have deployed, the applications we develop
- We spend a lot less time on threat modeling how we work - our SDLC
- Supply chain attacks are in the wind
- The SDLC contains many attack vectors

A traditional view of a SDLC



"This looks easy enough"

A more realistic view of a SDLC

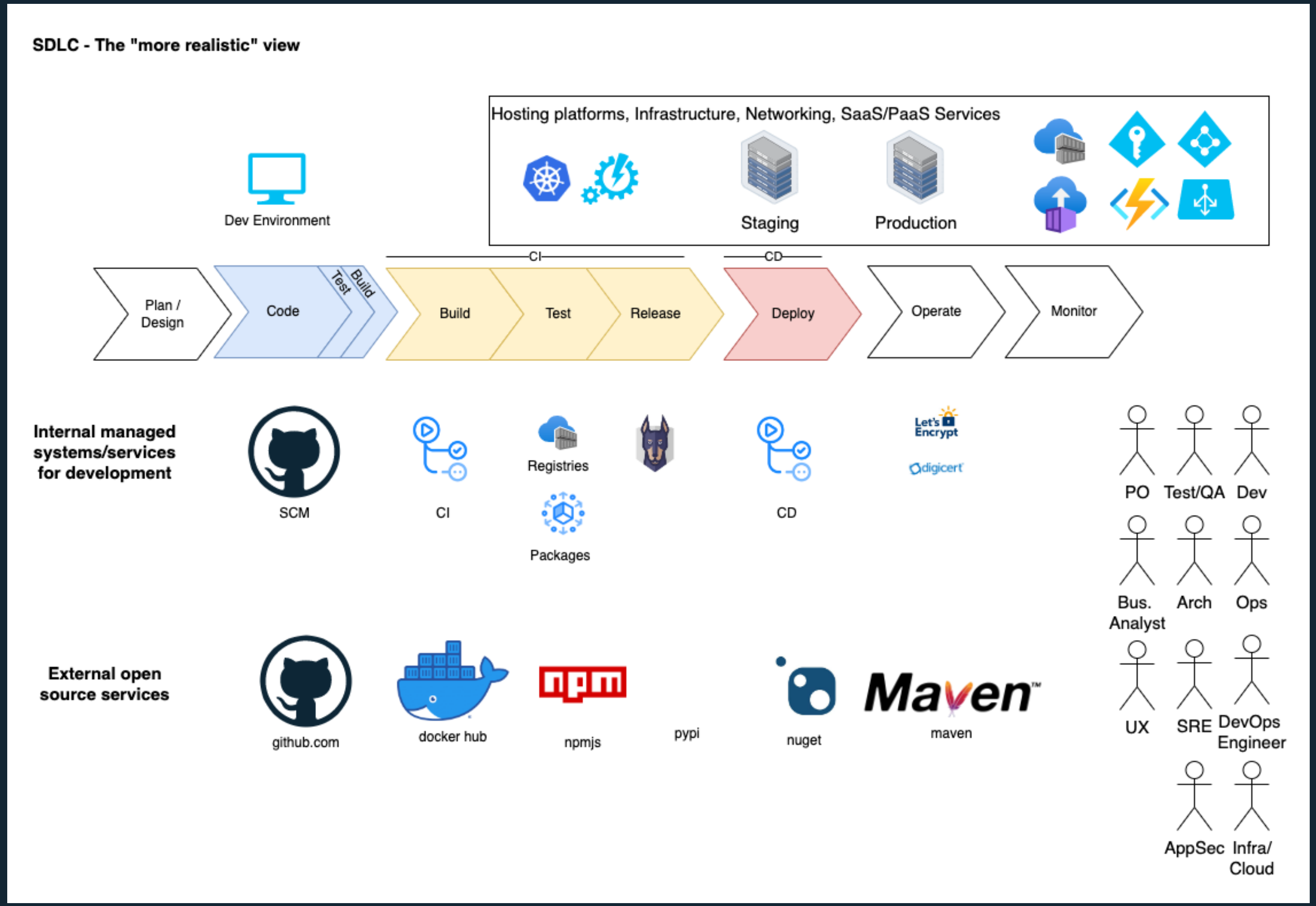


This example is by no means exhaustive, reality is more complex.
"All models are wrong. Some models are useful"

Threat Modeling our SDLC

What could possibly go wrong?

1. Scope - which part/information flow of the SDLC are we focusing on?
2. Follow the code?
3. What can go wrong?
4. What are we going to do about it?
5. Did we do a good job?



Getting started

- Identify and document security requirements.
- Document SDLC (How we work, develop)
- Document runbook (How we operate)
- Select threat model strategy
    - Part(s)/component(s) of the SDLC
    - Information flow - follow the code
- Apply STRIDE (what could go wrong?)
- Discuss & prioritise mitigations (what are we going to do about it?)
- Assess approach/results from SDLC threat modeling in team retrospective

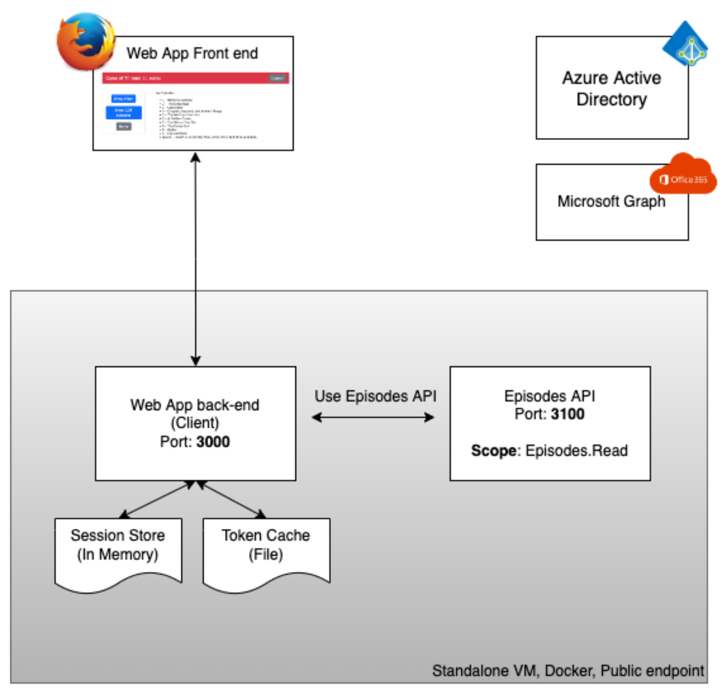
---

❗ Avoid the "perfect" document/model syndrome. Start small, iterate. This goes for security requirements, the SDLC documentation, the runbook etc. Think life cycle - what documentation do we intend to maintain and keep up to date?

❓ Reflections

Exercise 4 - Threat modeling a SDLC

EX-4: Example SDLC



Functionality

Web app
- Log in
- List context of O365 Inbox
- List episodes from "Episodes API"
- Log out

Episodes Api
- Receive request to get list of episodes
- Validate request and serve list if ok

System Assumptions

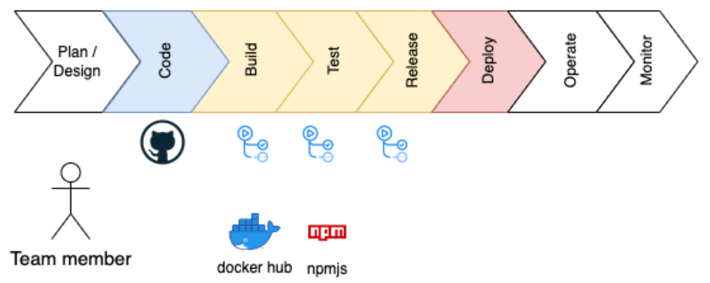
- Standalone Linux VM in Azure
- Linux is standard Ubuntu 18.10 out of the box
- Running in dedicated resource group
- Public endpoint for web app port 80/http
- Public endpoint for ssh and admin at port 22
- Components are running in containers
- Docker engine is running as root
- Internal comm is http
- All admin is manual using ssh
- The Web App is using a backend-for-frontend (BFF) pattern
- The Web App is using oAuth2/OICD Code Grant for authn/z

Office365 Inbox is classified as confidential. Episodes api is classified as "internal". The business is considering making the episodes information open - but if so the business would like to know a lot about those who access the episodes data.

Security Requirements

1. Information shall be encrypted at rest
2. Information shall be encrypted at transport
3. Information security classification must be known
4. RBAC shall be used to access information
5. Systems shall use company provided IAM
6. Access to all internal, restricted or confidential information shall be authorized
7. App to app permission shall be recertified at least once per year
8. Access to system credentials, tokens etc. shall be restricted and controlled
9. System shall implement logging and monitoring to detect unauthorized access
10. Logs shall be protected from unauthorized access after recording
11. All system shall be scanned for vulnerabilities regularly
12. Application shall generate new session tokens on user authentication
13. Session stores shall be secured
14. Session cookies shall only utilise secure transport
15. Session tokens shall be generated using approved cryptographic algorithms
16. Caching of sensitive data shall be protected
17. A development shall use a Secure SDLC.
18. Secure Development Policies shall minimum include:
    1. Protection of source code
    2. Code review
    3. Code scanning
    4. Security by design
    5. Security in depth
19. Development environments shall be secured.

SDLC



SDLC Assumptions

- We use managed computer as dev environment
- Code is stored on github.com
- Developer authenticate using SSH keys
- Code is not signed
- We use Github Action for CI
- We pull dependencies from NPMJS and DockerHub
- We build container images and push to github package registry
- We have installed docker on the host VM
- We ssh into VM, pull docker images and re-deploy
- To avoid interrupting the business we rarely update VM
- All infrastructure / Azure set-up is manual
- We have no dedicated test environments
- We have little and out-of-date system documentation
- We have little and out-of-date operational documentation
- We have basic monitoring on VM (memory, storage, traffic, cpu)
- We have no alerting

EX-4: Doing an end-to-end Threat Model

In this exercise we will apply the skills we have acquired in the previous exercises to a fictive SDLC.

Group tasks:

- Examine the system, the SLDC, the assumptions and security requirements
- Select a flow/part of the example SDLC 👇
- Create a DFD diagram
- Identify threats using STRIDE
- For threats, select strategy, identify mitigations
- Document assumptions and security requirements
- Take a picture of your DFD and documents, share on Slack and prepare to present the model to the class

The person in the group which last joined an Software Development Conference becomes group lead for this exercise.

⏰ Time boxed schedule (40m):

- 10 + 10 + 10 + 10= 40 to threat model

EX-4: Presentations

Each group present their results

❓ Reflections, Observations, Learning

Getting started with threat modeling in your team

🏡

Re-iterating on our objective

Help teams to build and operate more secure systems by incorporating threat modeling into their daily work.

Exercise 5 - Gather insight, make suggestions

Group task:

Given our objective, consider your actual team/context. Explore some of questions below:

- For TM, "what is the first thing we will suggest for our team when we get back home"?
- What do we need to do to make TM a consistent part of "what we do"?
- What external help do we need, and when?

Take notes. Anonymize if needed. Post result to workshop slack channel #appsec-threatmodeling-workshop.

The taller person in the group becomes group lead for this exercise.

⏰ Time boxed schedule (15m):

- 15 Group work

EX-5: Presentations

Each group present their results

❓ Reflections, Observations, Learning

🎬 Suggestions for getting started

1. W1: Create list of security requirements, max 10.
   50/50 from governance and OWASP ASVS L1
2. W2: End-2-End TM of a key system/app (⏲️-box to 1,5 hours)
   1. Create a high level DFD
   2. Use brainstorming to identify threats
   3. Manage threats, prioritize, select 2-3 items for teams backlog
3. W3: End-2-End TM of SDLC (⏲️-box to 1,5 hours)
   1. Create a high level DFD of your SDLC, follow the code.
   2. Use brainstorming to identify threats
   3. Manage threats, prioritize, select 2-3 items for teams backlog
4. W4: Select "a more narrow" scope of a system/app
   - Create a DFD and do a full end-2-end TM using STRIDE
5. W5: Do TM retrospective, align on next steps, continue
6. W5: Share experiences with the #AppSec community!
7. W6: Experiment with EoP

Wrapping up 🎁

Engage the AppSec community in your Threat Modeling Journey!

🥰 The AppSec Team can engage/help (The EqN AppSec Site)

🧐 Post a **real** threat model to the course Slack channel, ask for friendly advice/critique

🎪 Share from your Threat Modeling stories in #AppSec

Key Resources

- 🕵️ Threat Modeling - design for Security
  - Adam Shostack
- 🕵️ Threat Modeling - A Practical Guide for Development Teams
  - Izar Tarandach
  - Matthew J. Coles
- 🕵️ Threat Modeling Manifesto
- 🕵️ A Guide to Threat Modelling for Developers
  - Jim Gumbley, Martin Fowler

Workshop Retrospective

Gathering data - the 4L's

For this workshop - discuss/reflect in groups on the following topics:

- Liked
- Learned
- Lacked
- Longed for

⏰ We timebox for 5 minutes and then share verbally.

Thank You

We are the @appsecteam -
serving the developer community -
living in #appsec on Slack.