

DevSecOps - Threat modelling



Copyright, license, etc.

- This presentation is intellectual property of Unixerius.
 - Copyright and all rights remain with Unixerius.
- This presentation is offered to Unixerius' customer:
 - **For free**, to use for in-house training.
 - With the stipulation that it remains **unaltered**, as-is.
 - And that it is **not distributed** further.

Threat modelling

Happy-flow vs realism

- Often, the SDLC is focused on getting the job done.
 - Features and functionality.
 - Testing for "known good", not for "known bad".
- It's better to be prepared, for "bad things" too.

TLDR: threat modelling

- A repeatable process which helps you find:
 - The crown jewels and treasures,
 - The weak points in your defenses,
 - The threats against them and
 - What you need to do about them.

Why?!

- It helps you define:
 - Improvements to your design.
 - Required protections and mitigations.
 - Security logging events.
 - Security testing steps.

Who can play?

- It's not just developers! Invite your:
 - Ops engineers
 - Product owner
 - Business analyst
 - Architect
 - SOC analyst, risk manager, cloud engineer

"How do I threat model?"

- Microsoft Threat Modelling
- PASTA
- OCTAVE
- Trike
- LINDDUN
- Attack Trees
- STRIDE
- DREAD
- OWASP Threat Modelling Process

Art: [LittleClyde](#)



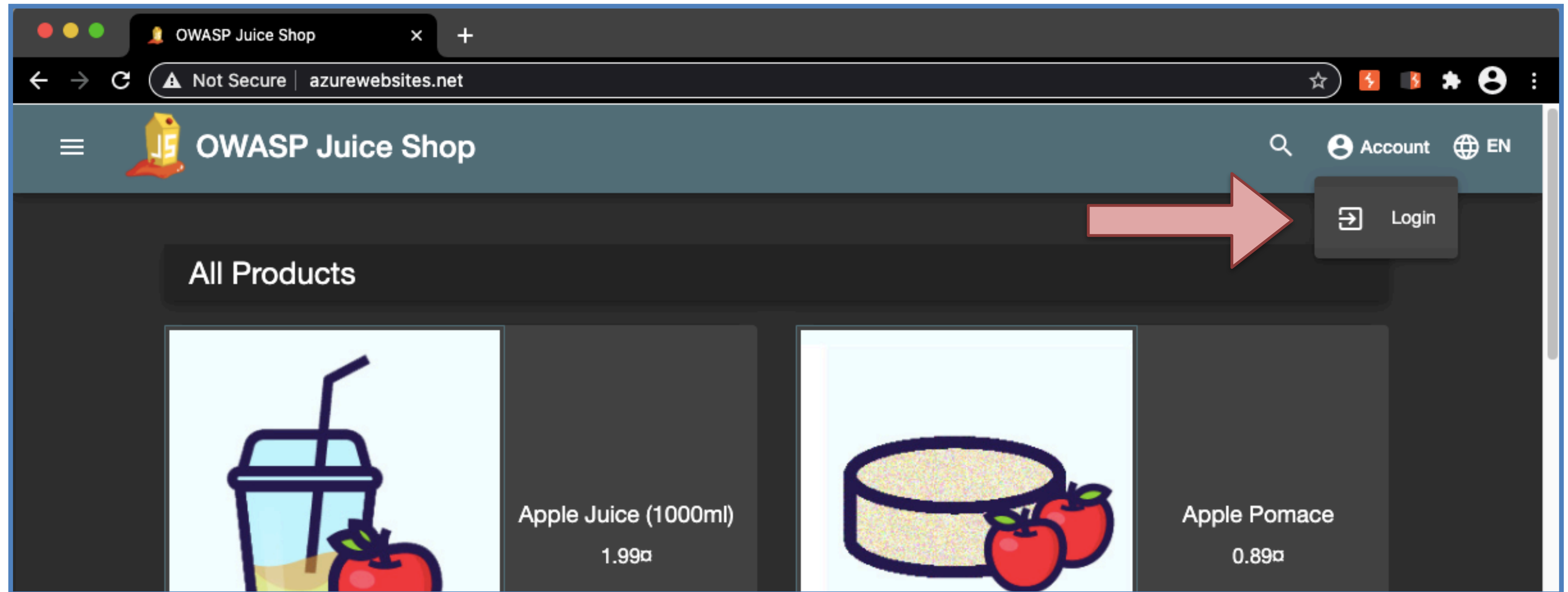
Let's keep it simple

- We will try two methods, plus one.
 - The absolute simplest one,
 - Followed by STRIDE and PNG.
- While keeping things realistic.
 - *Mr Robot* and *Dade 'Zero Cool' Murphy* do not apply.

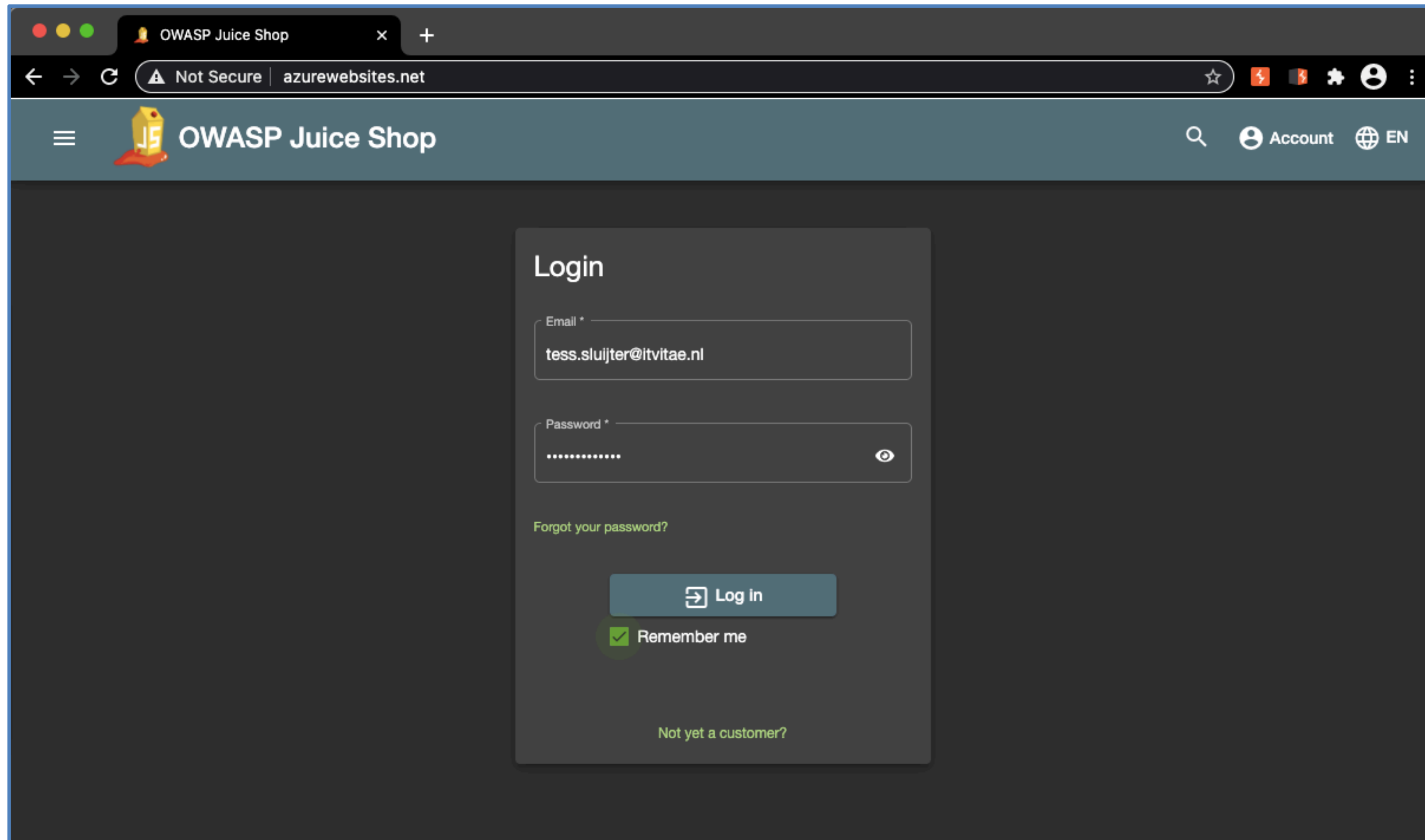
Our "victim": JuiceShop login

- Threat modelling is applied to features,
 - To data-flows, to specific use-cases.
- You don't threat model a whole website or app.
 - At least not in one go.
- So let's look at the JuiceShop login!

Our "victim": JuiceShop login



Our "victim": JuiceShop login



The screenshot shows a web browser window with the title "OWASP Juice Shop" and the URL "azurewebsites.net". The page features a dark blue header with the "OWASP Juice Shop" logo and navigation links for "Account" and "EN". The main content area is dark gray and contains a "Login" form. The form has two input fields: "Email *" with the value "tess.sluijter@itvitae.nl" and "Password *" with masked characters. Below the password field is a "Forgot your password?" link. A "Log in" button is positioned below the "Remember me" checkbox, which is checked. At the bottom of the form is a "Not yet a customer?" link.

OWASP Juice Shop

Account EN

Login

Email *
tess.sluijter@itvitae.nl

Password *
.....

[Forgot your password?](#)

☒ Remember me

[Log in](#)

[Not yet a customer?](#)

Our "victim": JuiceShop login

Burp Suite Community Edition v2020.12 - Temporary Project

Dashboard

Target

Proxy

Intruder

Repeater

Sequencer

Decoder

Comparer

Extender

Project options

User options

Intercept

HTTP history

WebSockets history

Options

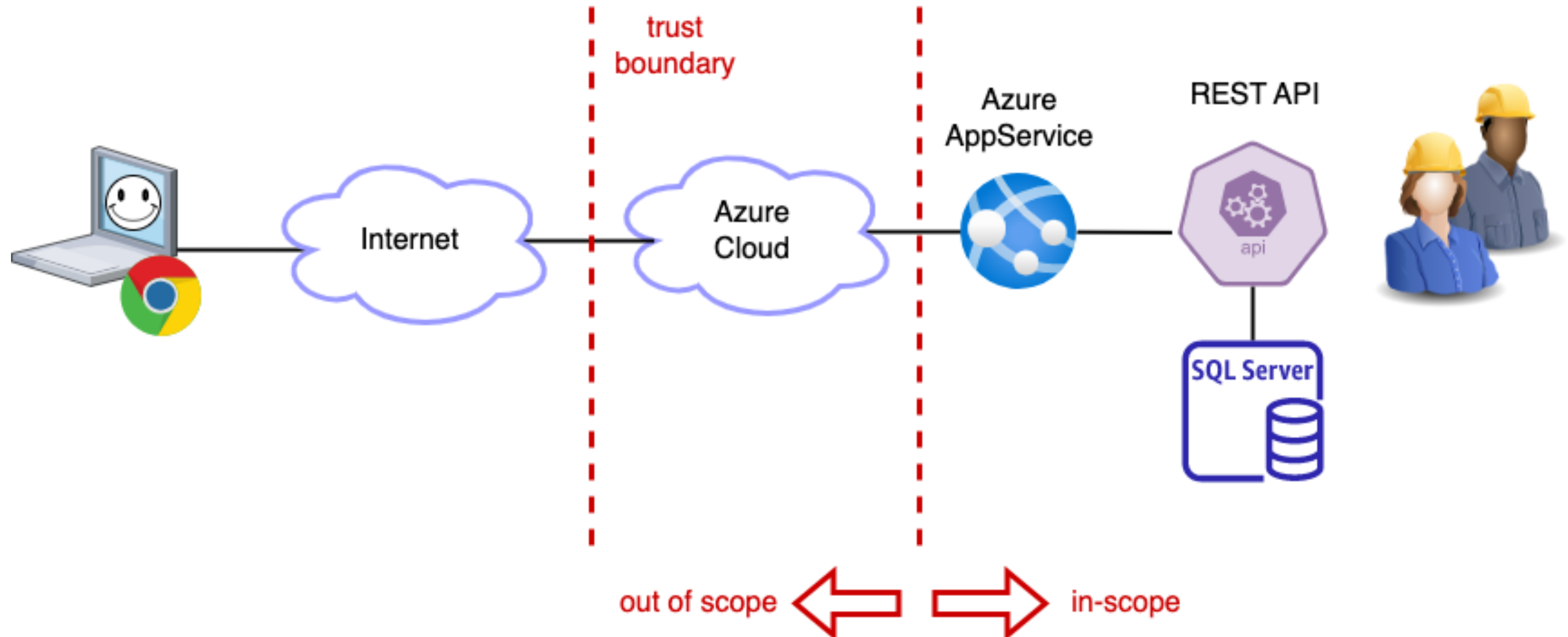
Filter: Hiding out of scope items; hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status	Length	MIME type
163	https://unixeriUSDso-team1.azurewebsites.net	GET	/rest/user/whoami			200	831	JSON
161	https://unixeriUSDso-team1.azurewebsites.net	GET	/rest/user/whoami			200	831	JSON
160	https://unixeriUSDso-team1.azurewebsites.net	GET	/rest/products/search?q=	✓		200	13577	JSON
159	https://unixeriUSDso-team1.azurewebsites.net	GET	/api/Quantitys/			200	6717	JSON
158	https://unixeriUSDso-team1.azurewebsites.net	GET	/rest/basket/6			200	847	JSON
157	https://unixeriUSDso-team1.azurewebsites.net	GET	/rest/user/whoami			200	702	JSON
156	https://unixeriUSDso-team1.azurewebsites.net	GET	/rest/user/whoami			200	702	JSON
155	https://unixeriUSDso-team1.azurewebsites.net	POST	/rest/user/login	✓		200	1544	JSON
153	https://unixeriUSDso-team1.azurewebsites.net	GET	/rest/admin/application-configuration			200	19488	JSON
144	https://unixeriUSDso-team1.azurewebsites.net	GET	/rest/admin/application-configuration			200	19488	JSON
139	https://unixeriUSDso-team1.azurewebsites.net	GET	/api/Challenges/?name=Score%20Board	✓		200	1346	JSON

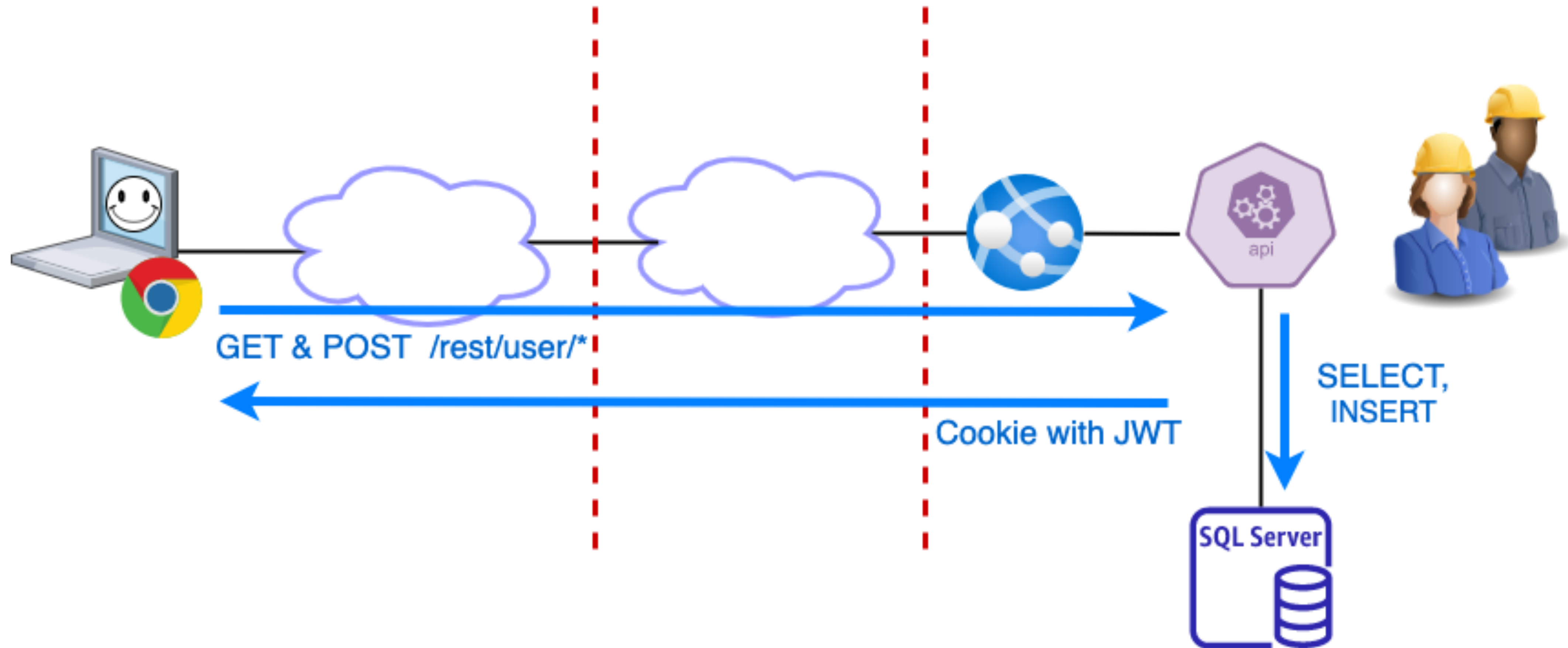
Data flow diagram

- A clear overview of all parties in a use-case.
 - Actors
 - Software, infrastructure, etc.
 - Artefacts, resources, protection boundaries

Data flow diagram



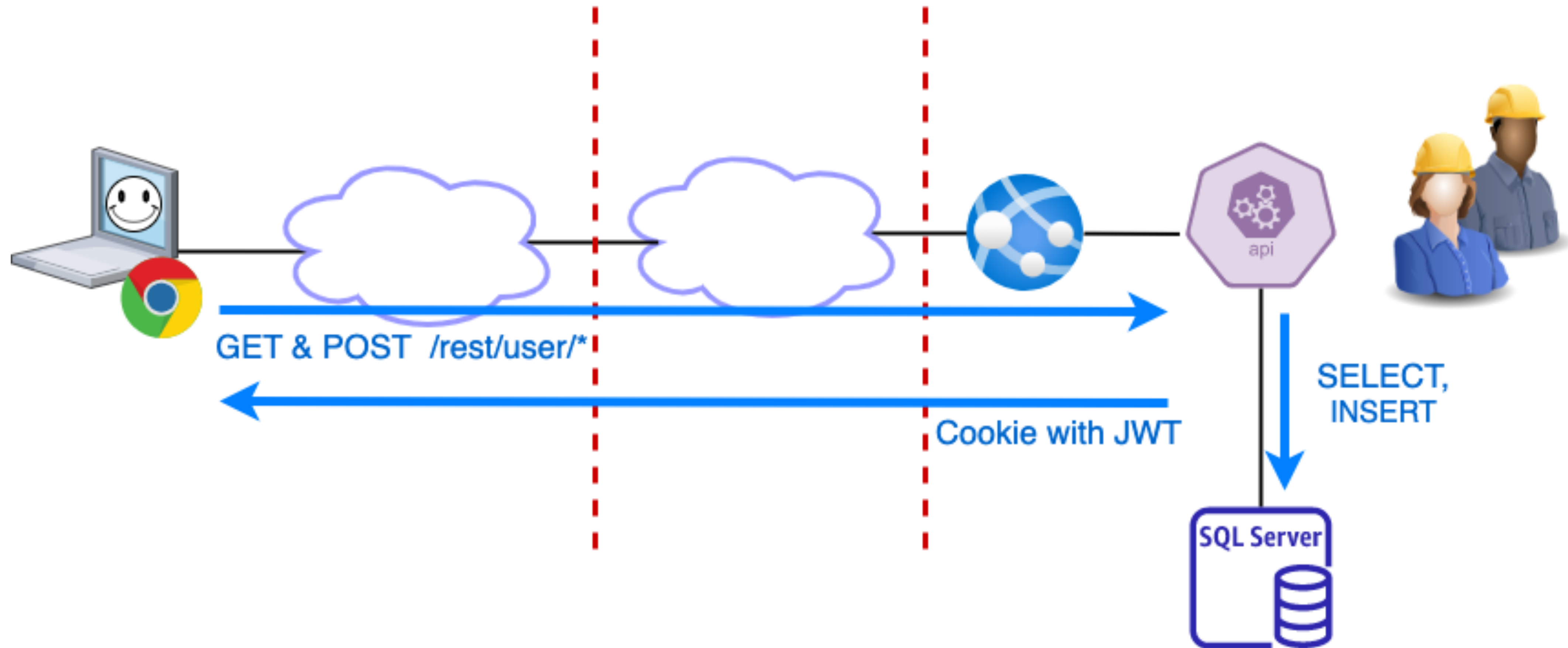
Data flow diagram



The absolute simplest way

- I've made this beautiful, cool thing!
 - How can someone set fire to it?
 - What's the very worst thing that can happen?
 - How do I prevent that?

Data flow diagram



Let's document that!

Resource	Damage	Prevent	Mitigate
User database, leak of user info	GDPR fines Loss of customers Reputation loss	DB content ACLs Prevent SQLi in code	Azure WAF Cyber-insurance
...

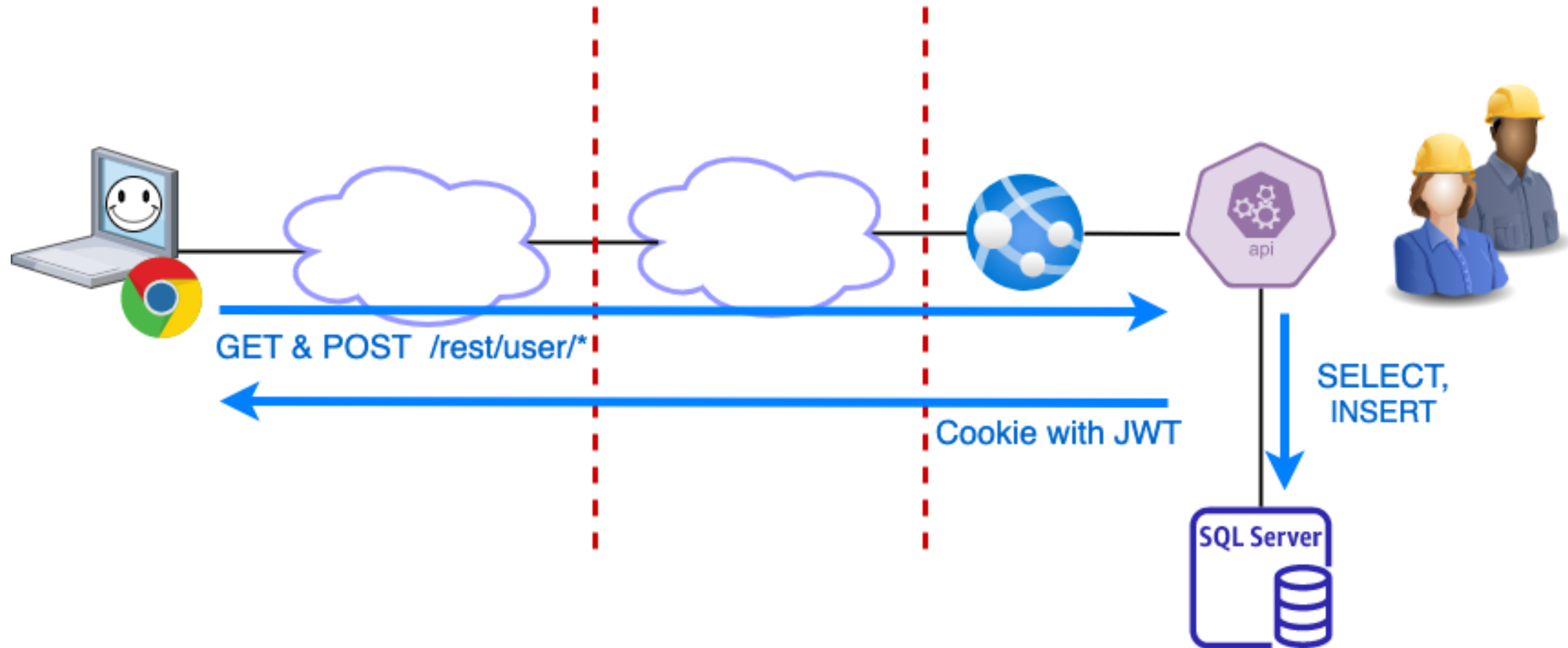
STRIDE

- One of many methods for analysis.
 - Looks at the technical aspects of your app.
 - Suggests categories of abuse to consider.

STRIDE

Spoofing	Identity forgery
Tampering	Integrity of data
Repudiation	Plausible deniability
Information disclosure	Breach confidentiality
Denial of service	Damage availability
Elevation of privileges	Get extra authorizations

Data flow diagram



Let's document that!

STRIDE	Effect	Prevent	Mitigate
SE	Cookie tampering	Signed JWT / cookie	n.a.
S	Brute-forcing passwords	Rate-limiting	Azure WAF
TI	SQL Injection -> steal data SQL Injection -> tamper users	Prevent SQLi in code	Azure WAF
D	SQL Injection -> deletion of users	Prevent SQLi in code	Azure WAF
D	Login flooding	Rate-limiting	Azure WAF

Persona non-grata

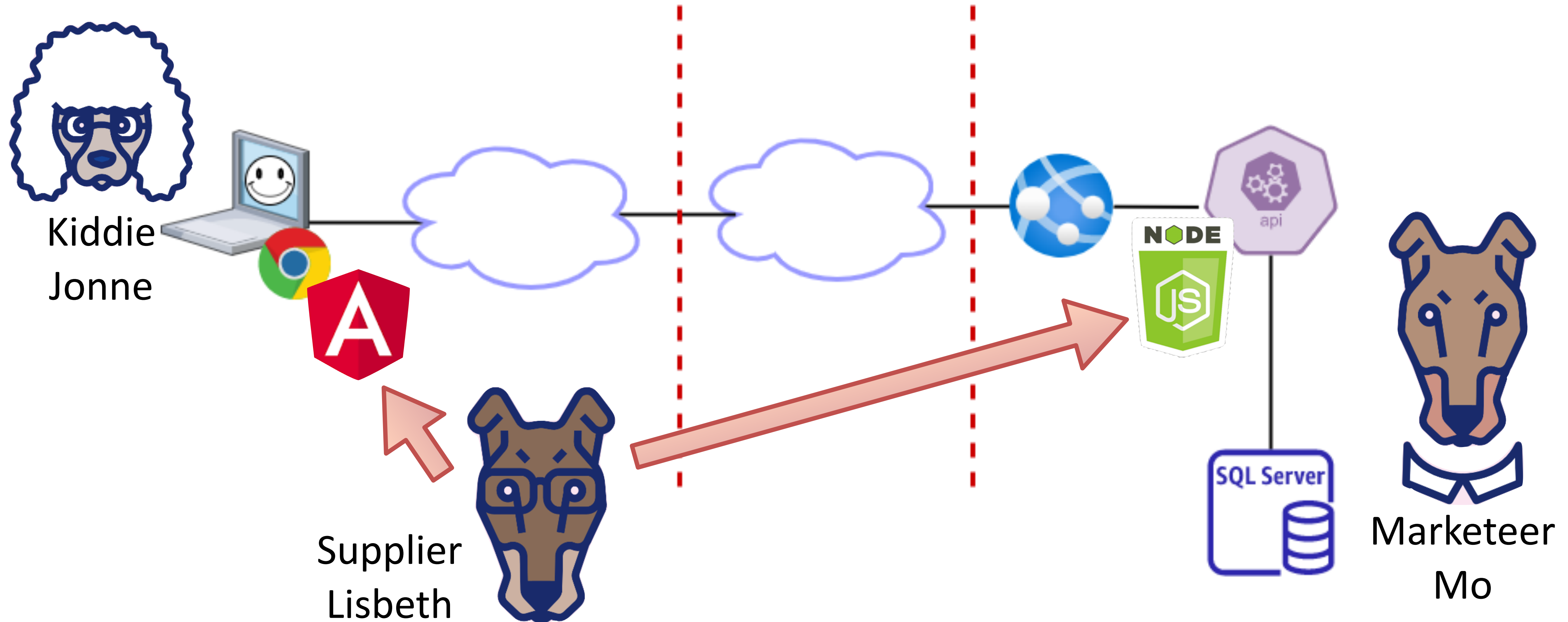
- Literally: "*unwanted persons*"
- Instead of a technical point of view,
 - It considers human nature and motivations.
- This may give new insights!

Persona non-grata

- You can come up with your own,
 - Or use online samples like [Nixu CyberBogies](#).

#22, Marketeer Mo	Employee, max profits
#15, Supplier Lisbeth	0-days & vulns in libraries
#02, Script Kiddie Jonne	Just trying "fun stuff"

Persona non-grata



Let's document that!

PnG	Effect	Prevent	Mitigate
Jonne	Brute-forcing passwords	Rate-limiting	Azure WAF
Jonne	SQL Injection	Prevent SQLi in code	Azure WAF
Mo	Data leaks, user-data misuse	Access Controls 4-Eyes principle	Training
Lisbeth	Breach of application server	Library LCM SCA, threat intel	Network segregation
Lisbeth	Breach of user's PC	Library LCM SCA, threat intel	NONE, so a risk!

Outcome

- Each method leads to improvement points:
 - Better design, secure coding
 - Training, personnel, mitigations to buy.
 - Identify unwanted situations to test and log.
- A list of items for the backlog, to work on.

Integrating into the SDLC

- Threat modelling fits into your DoD.
 - A new feature is only "done",
 - A new feature is only "built right",
 - If it was assessed for threats.

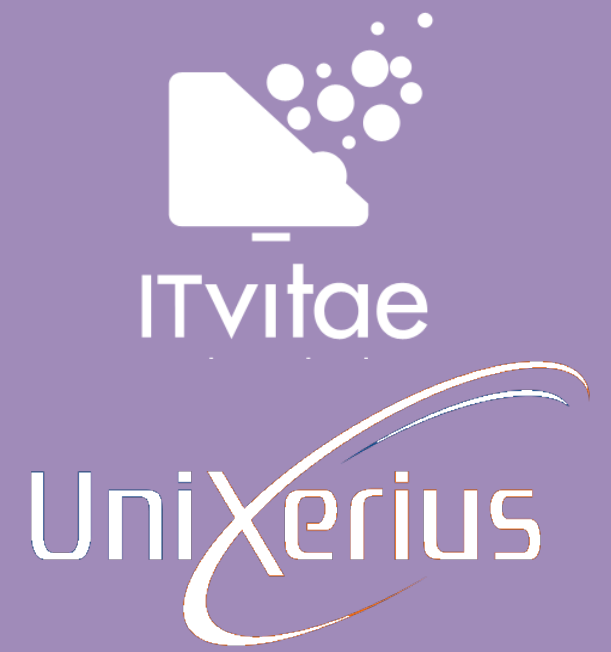
Integrating into the SDLC

- Threat model, if the user story:
 - Makes changes to **authorization**
 - Makes changes to **authentication**
 - Involves **authorised or anonymous** callers
 - Involves changes to **input validation**
 - Interacts with '**critical**' data in the 'critical storage'

Where to from here?

- Your squad can start threat modeling!
 - It's fun, take your time.
- If it's too daunting,
 - Invite FLoD, Security Officers or an architect.

Reference materials



Resources

- [The threat modelling field guide](#)
- [The threat modelling manifesto](#)
- [PluralSight learning path: threat modelling](#)
- [PDSO certified threat modelling professional](#)
- [Crowd sourcing the creation of persona non-grata](#)
- [Nixu CyberBogies](#) (PnG cards)