
TOP SECRET

## Mission objective

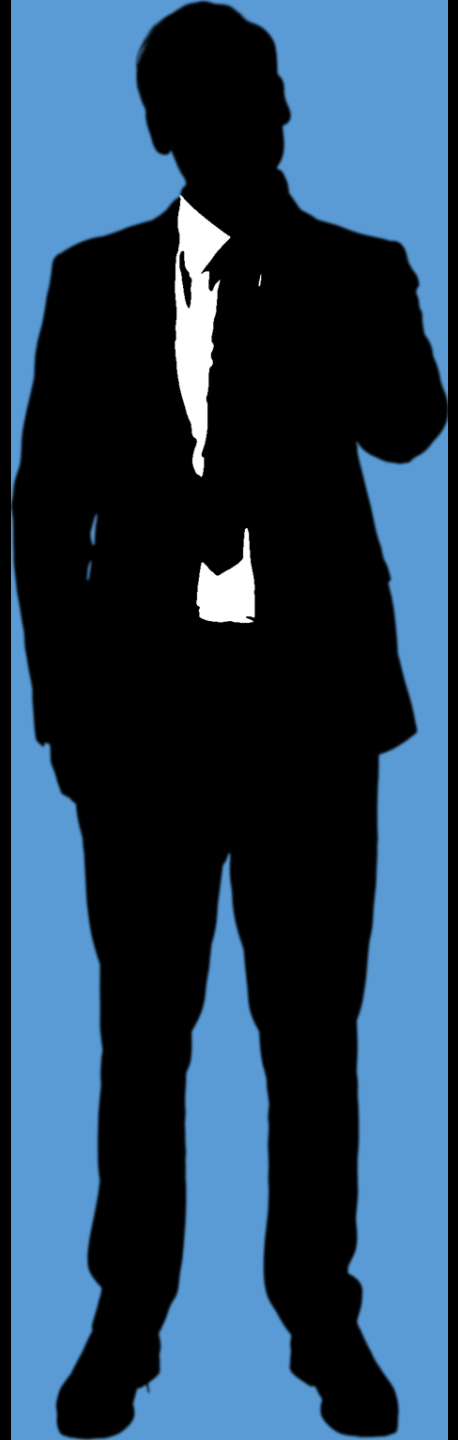
Design process of software development that minimizes risk of being hacked

## Start time

NOW

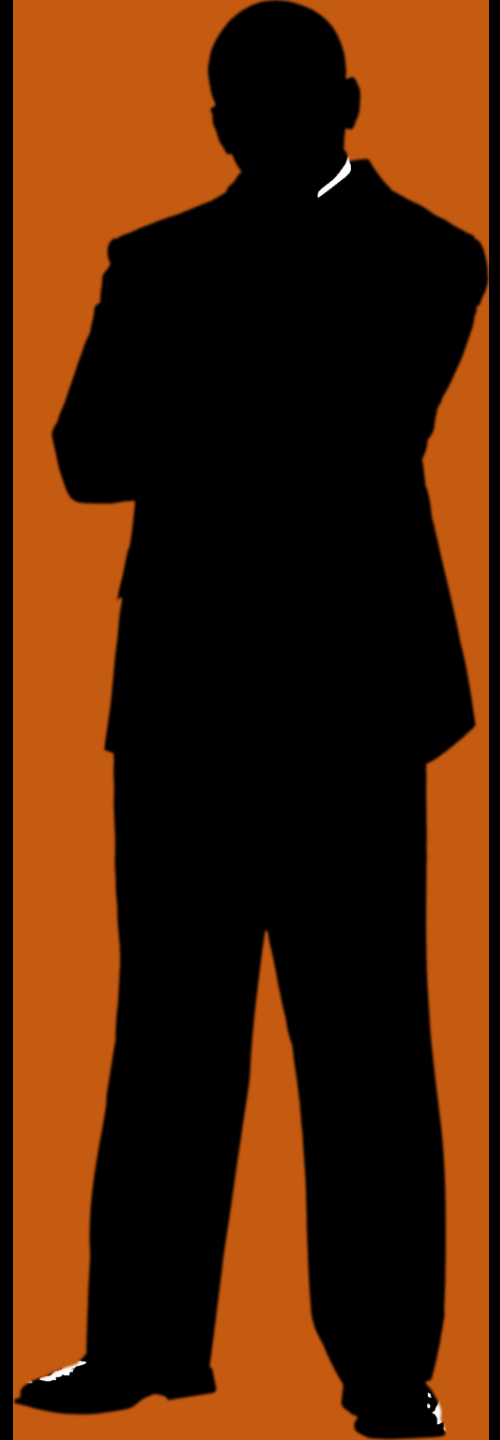
## Agents

Devel & Secar



DEVEL

SECAR



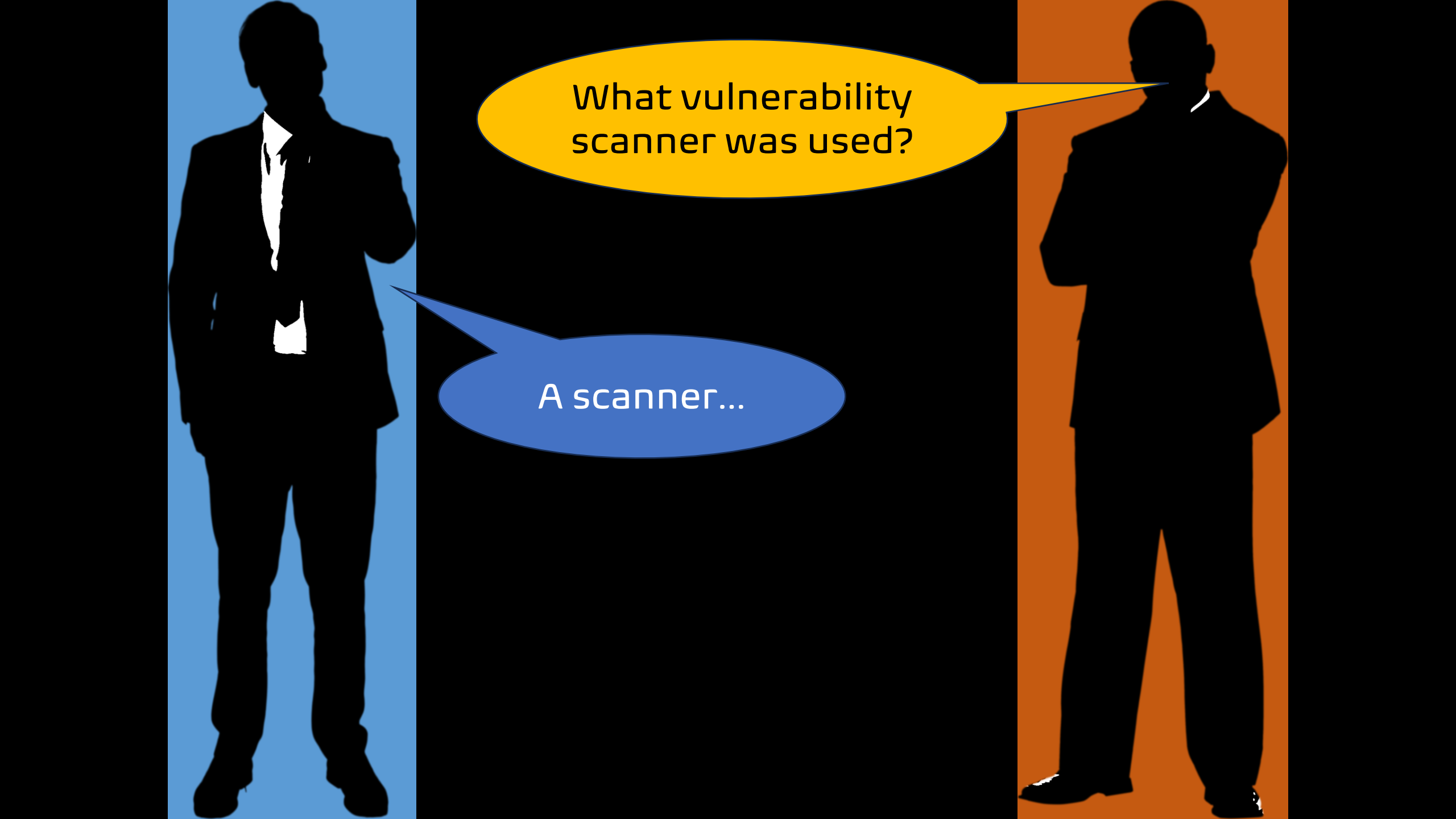

So another 0 day problem...
Yes... thankfully it's already fixed!

# 0 day problem


BRAND NEW
BRAND NEW


Nope. Don't like that.


At all


What vulnerability scanner was used?
A scanner...

# S.A.S.T.

# S.A.S.T.

t
a
t
i
c

# S.A.S.T.

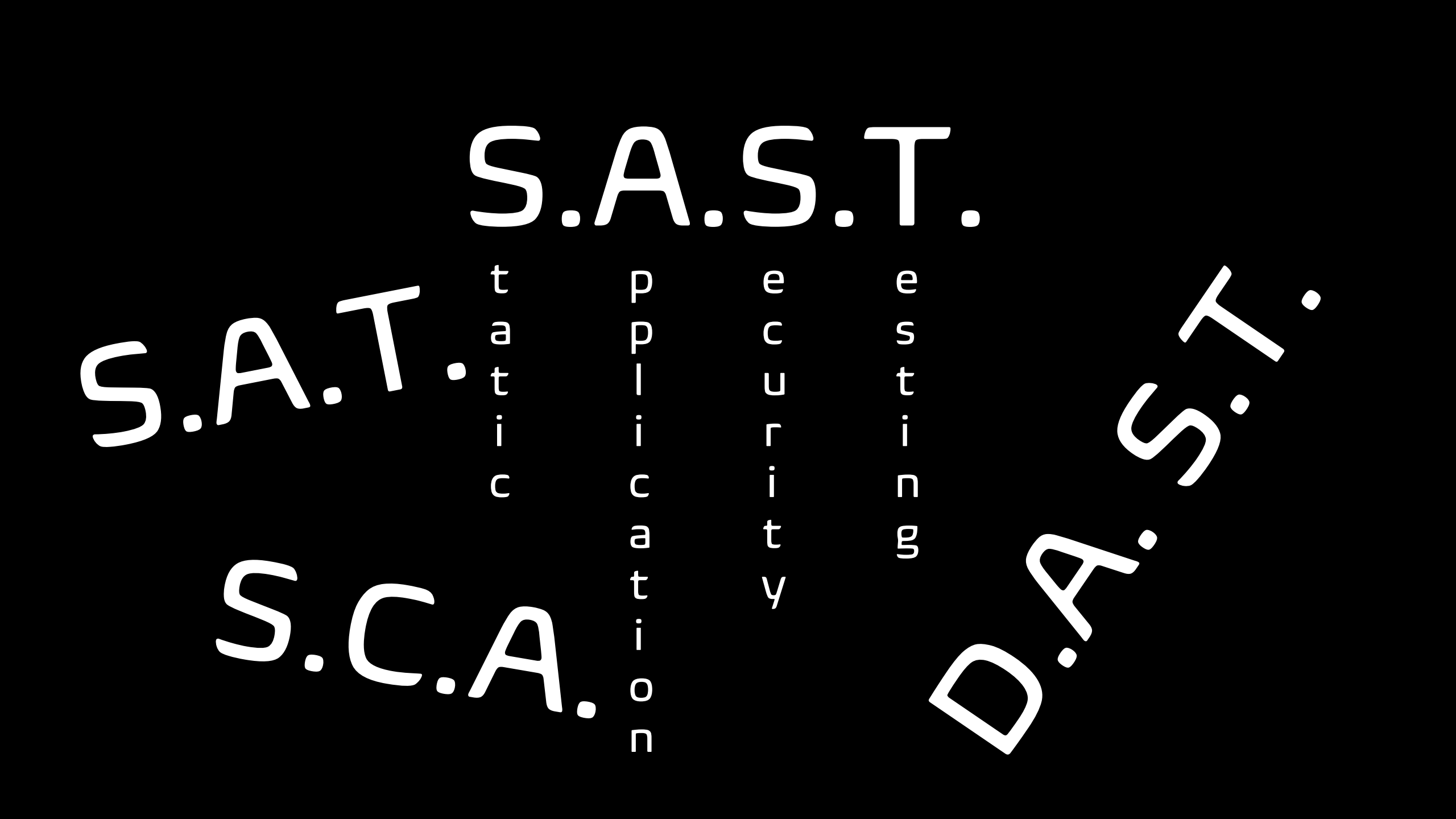
static application

# S.A.S.T.

static application security testing

# S.A.S.T.

static application security testing

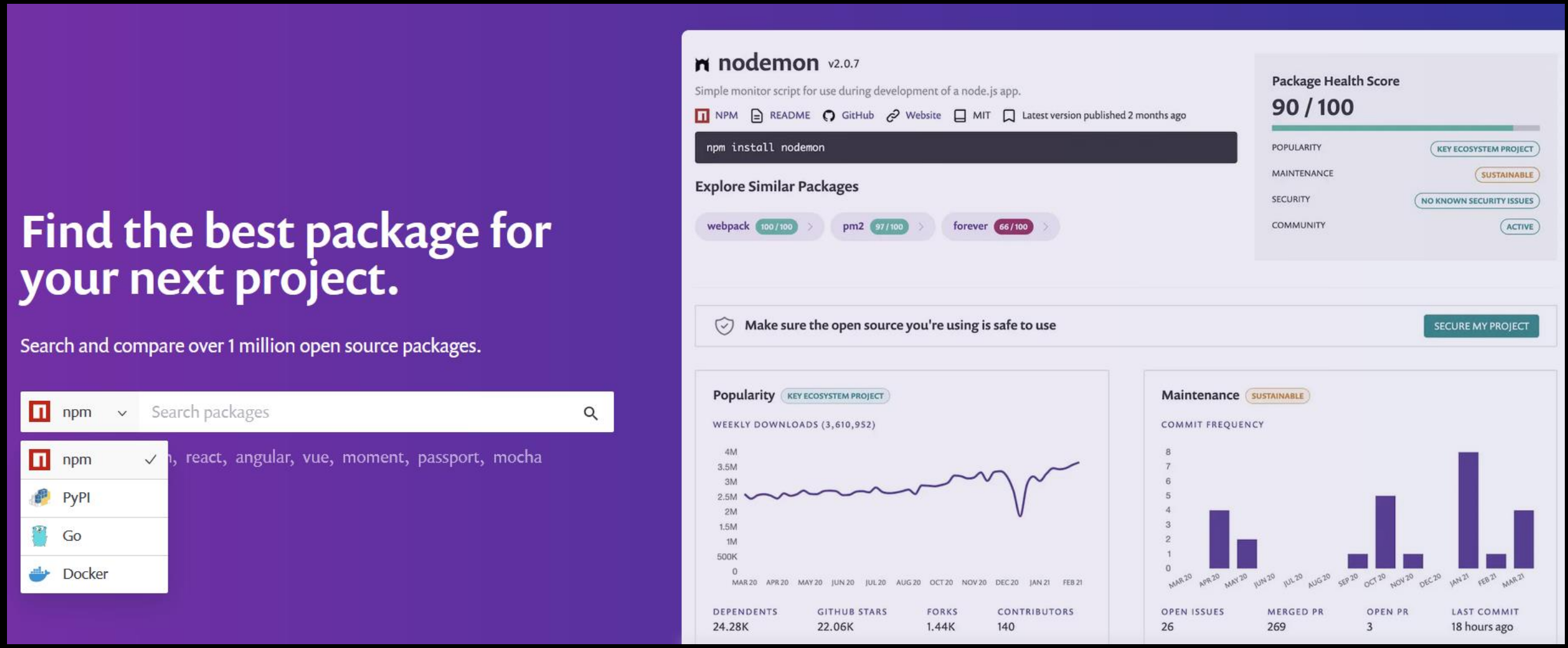

S.A.S.T.
S.A.T.
static
application
security
testing
S.C.A.
D.A.S.T.


sonarqube

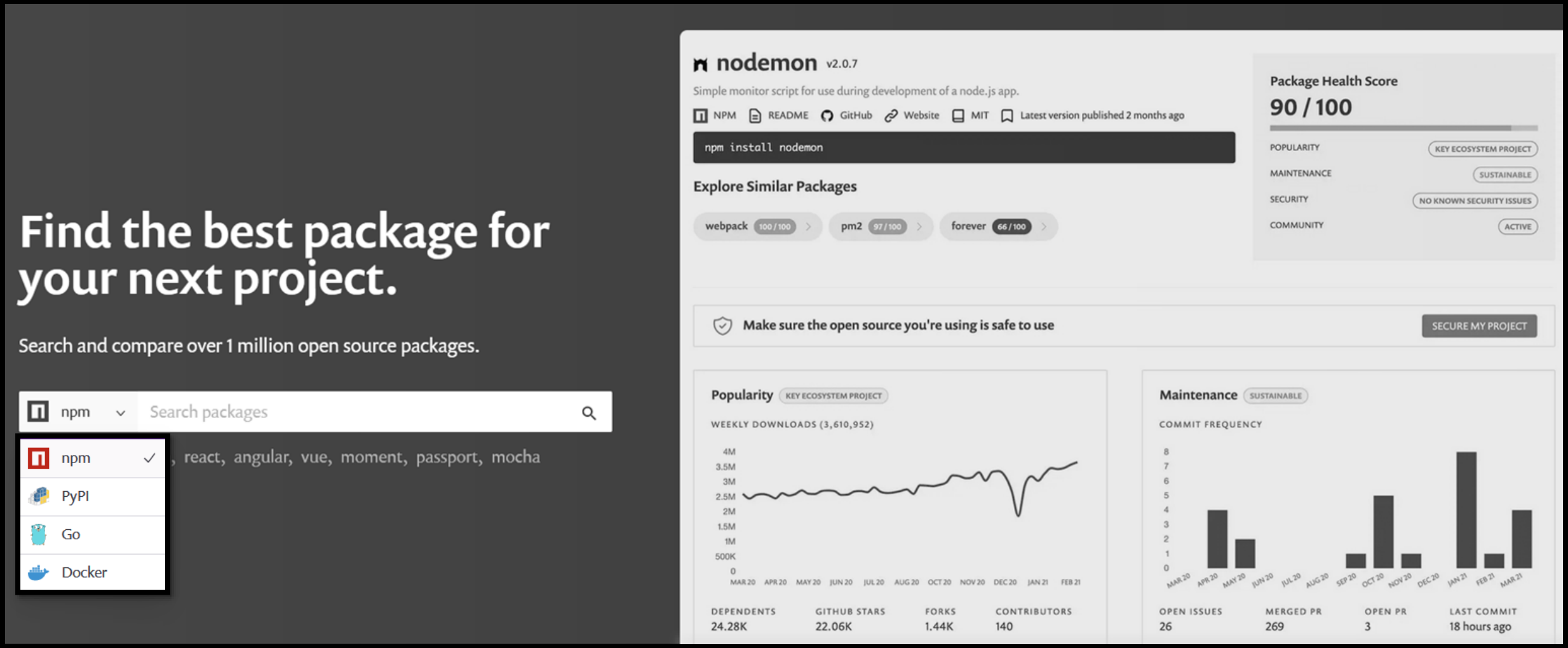

QD


snyk

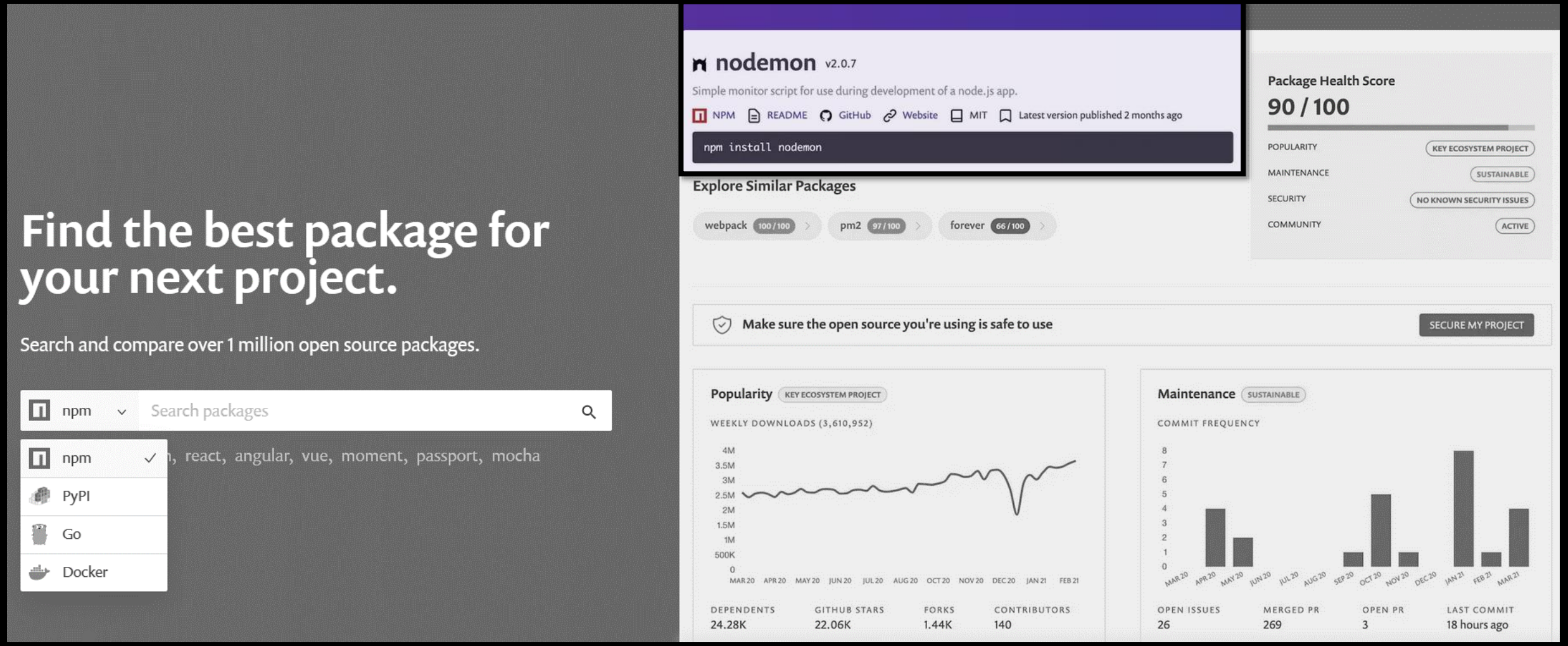
# snyk.io/advisor



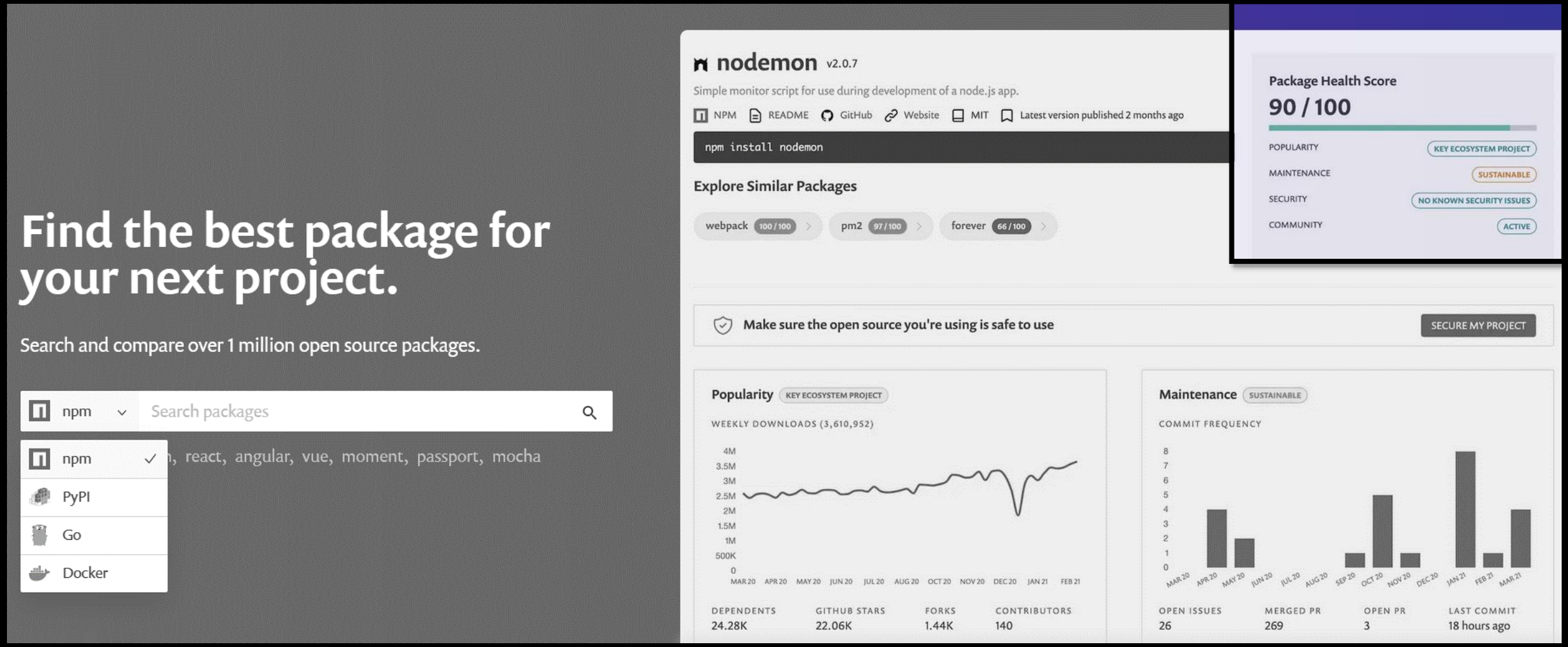
# snyk.io/advisor



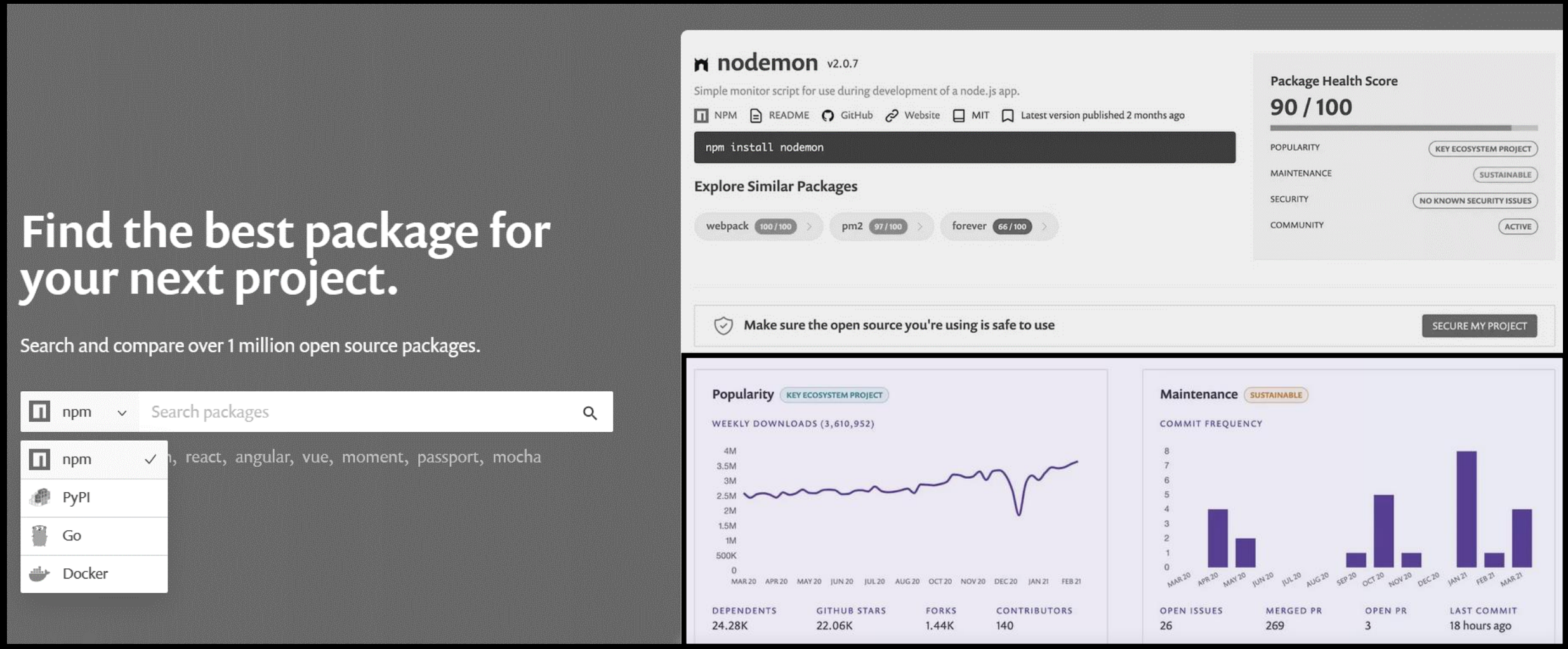
# snyk.io/advisor



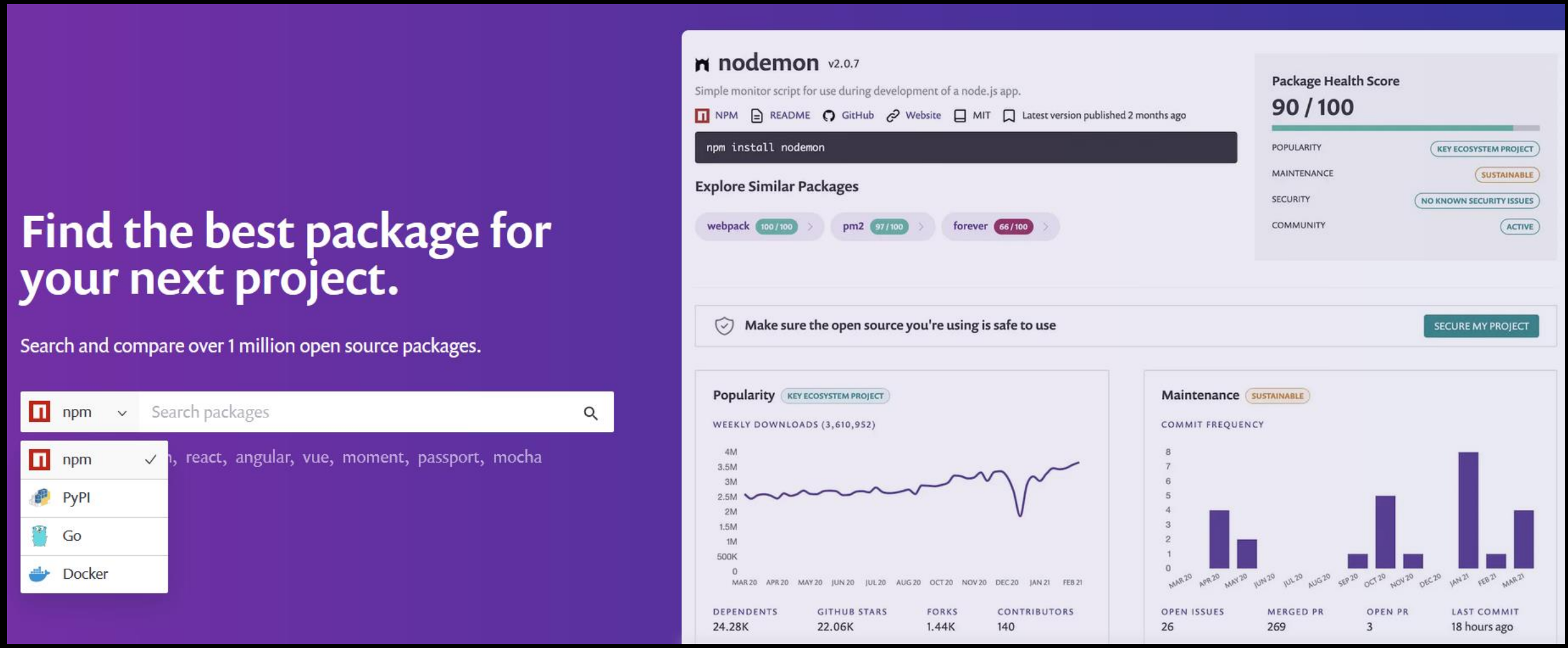
# snyk.io/advisor

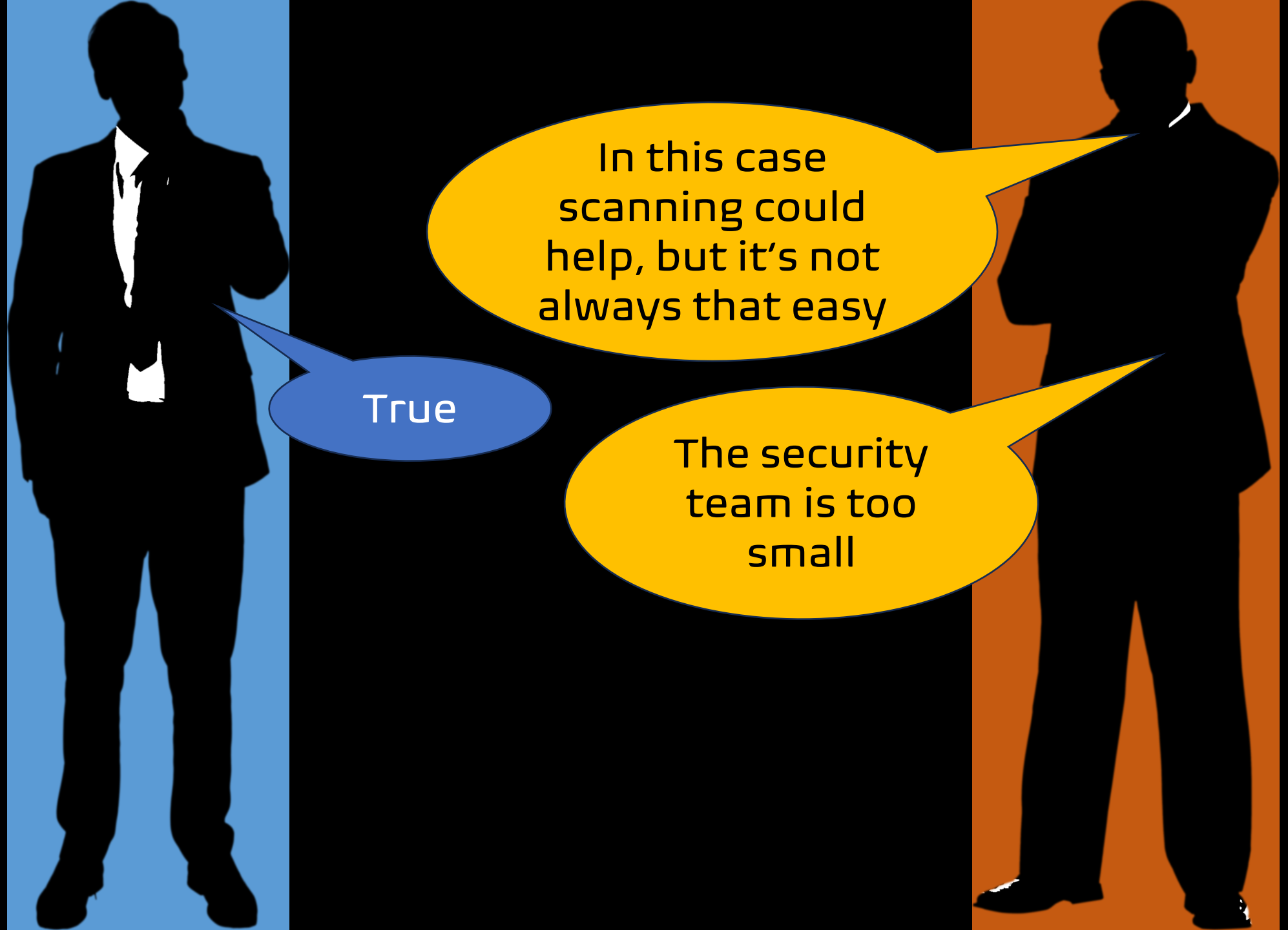


# snyk.io/advisor

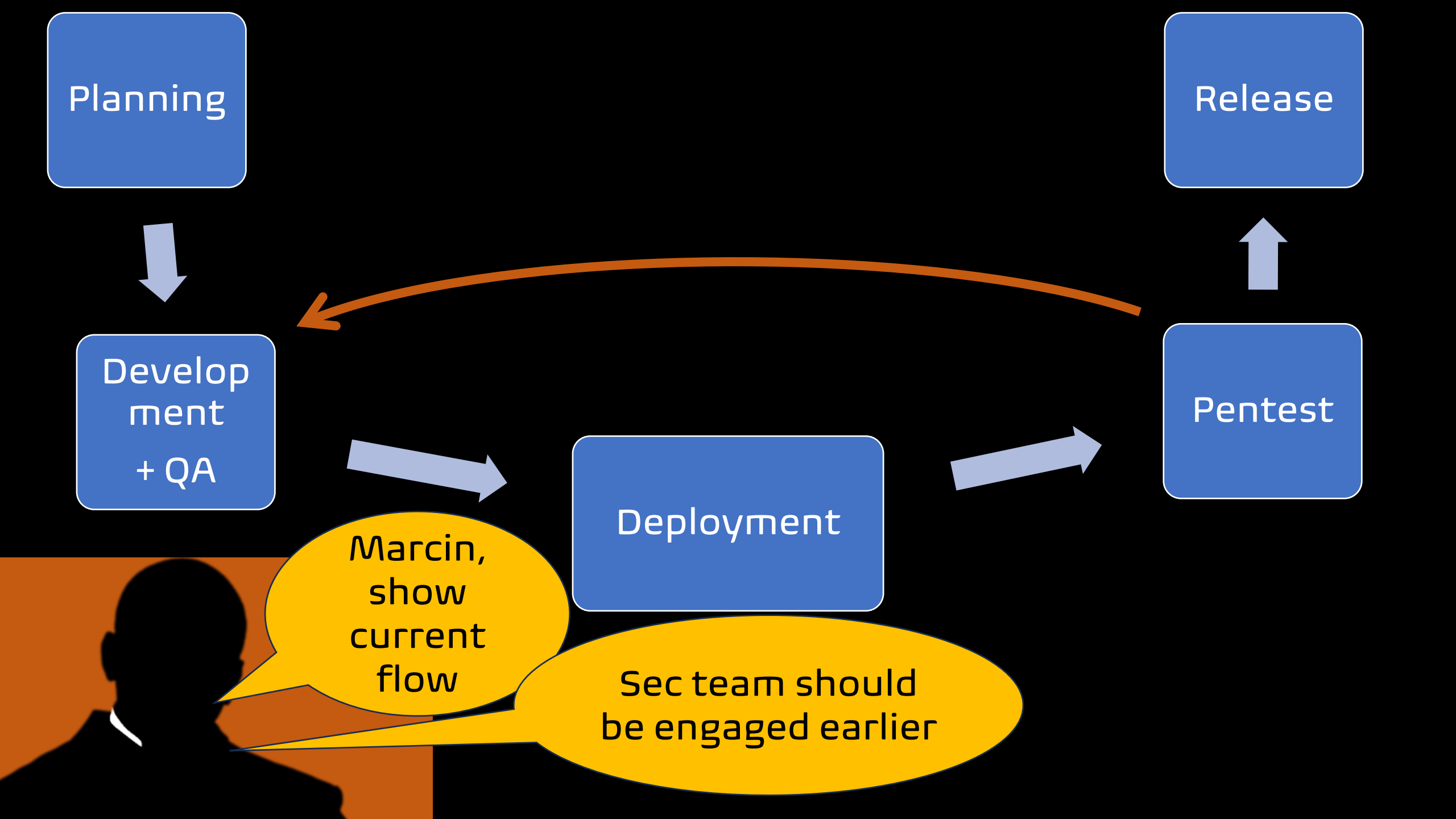


# snyk.io/advisor




In this case scanning could help, but it’s not always that easy
True
The security team is too small


They won't be happy
We should engage developers more
Still better, than being woke up by security incident


Planning
Release
Development + QA
Pentest
Deployment
Marcin, show current flow
Sec team should be engaged earlier

Planning with sec team

Release

Developme nt + QA

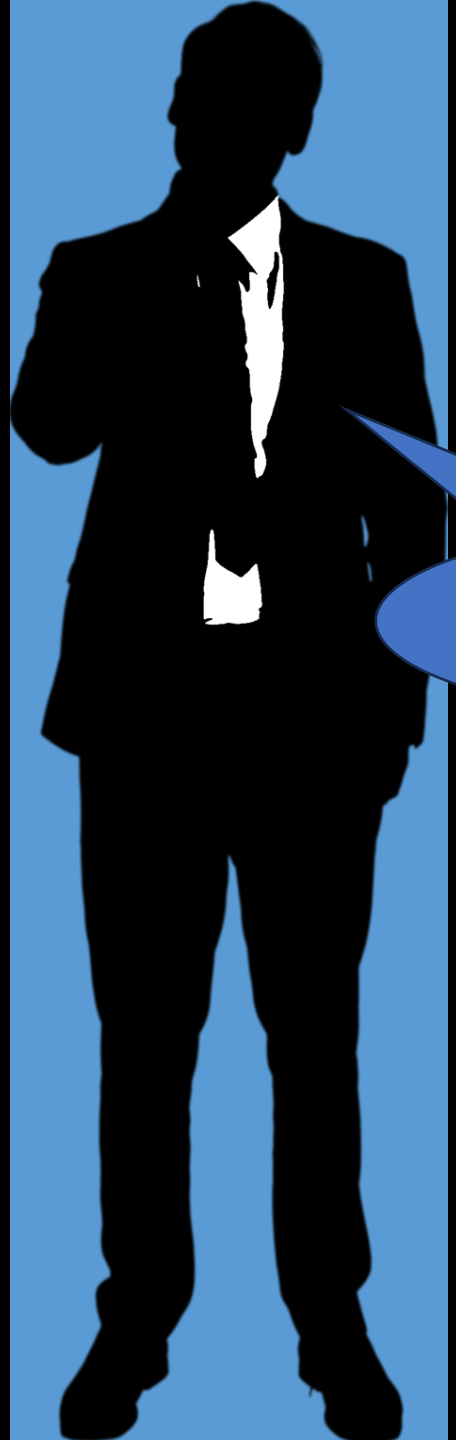
Deploymen t

Pentest

What about changing this flow to this

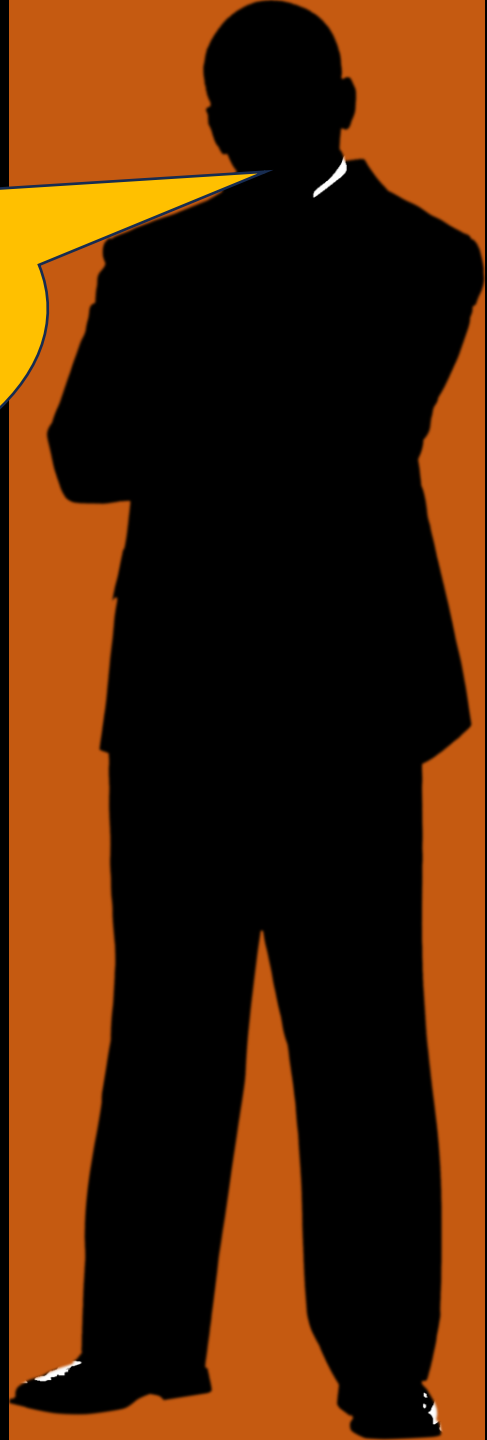
Shift to the left... Nice idea!


Dev
ting
Meeting
Meeting
Meeting
Meeting



What's that?

Actually we can start project Codename D.D.S.



# DEVELOPER DRIVEN SECURITY

# DDS

Developers take responsibility for security

# Threat modelling

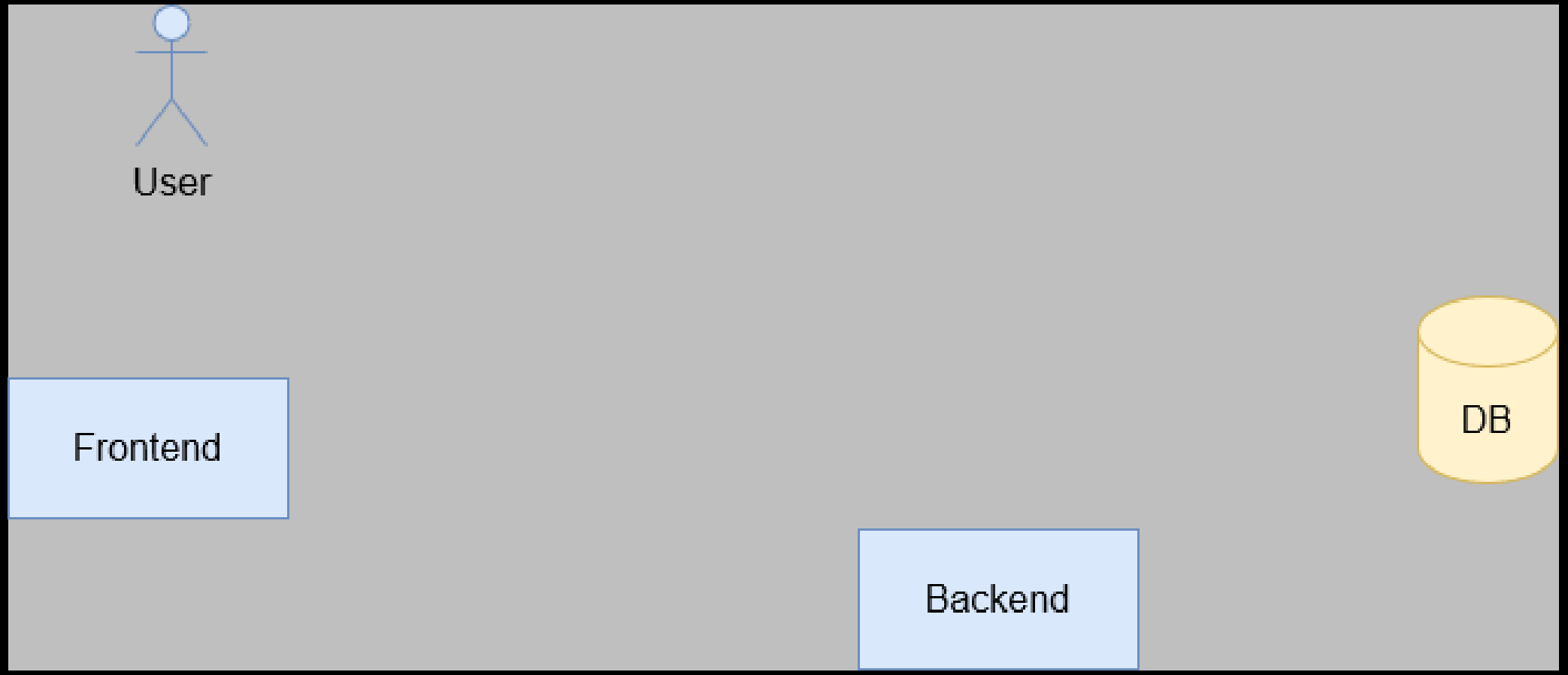
- Structurizes discussion
- Identifies security requirements
- Pinpoints security threats and potential vulnerabilities
- Quantifies threat and vulnerability criticality
- Prioritizes remediation methods

# Blog application

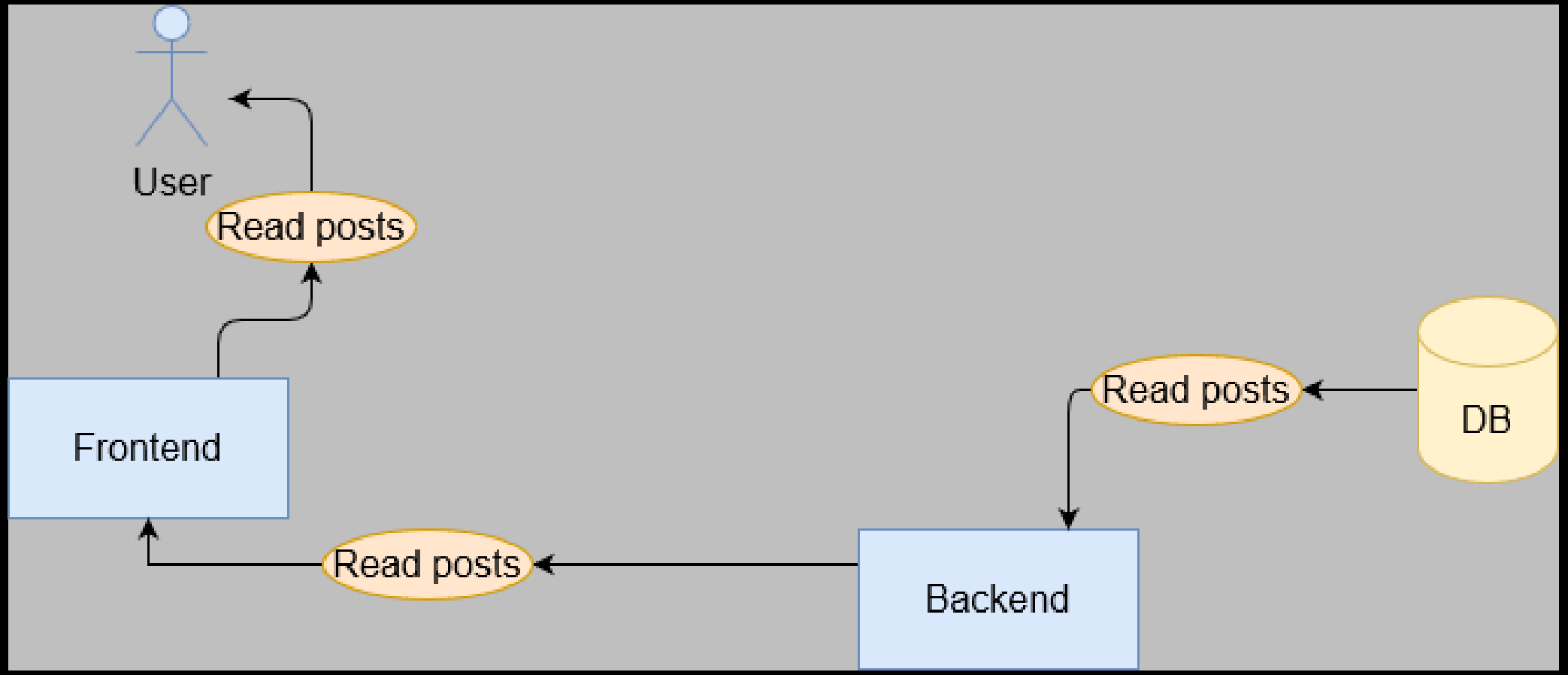
- Add post
- Read posts
- Login mechanism (for adding posts)

# Step 1: data flow diagram

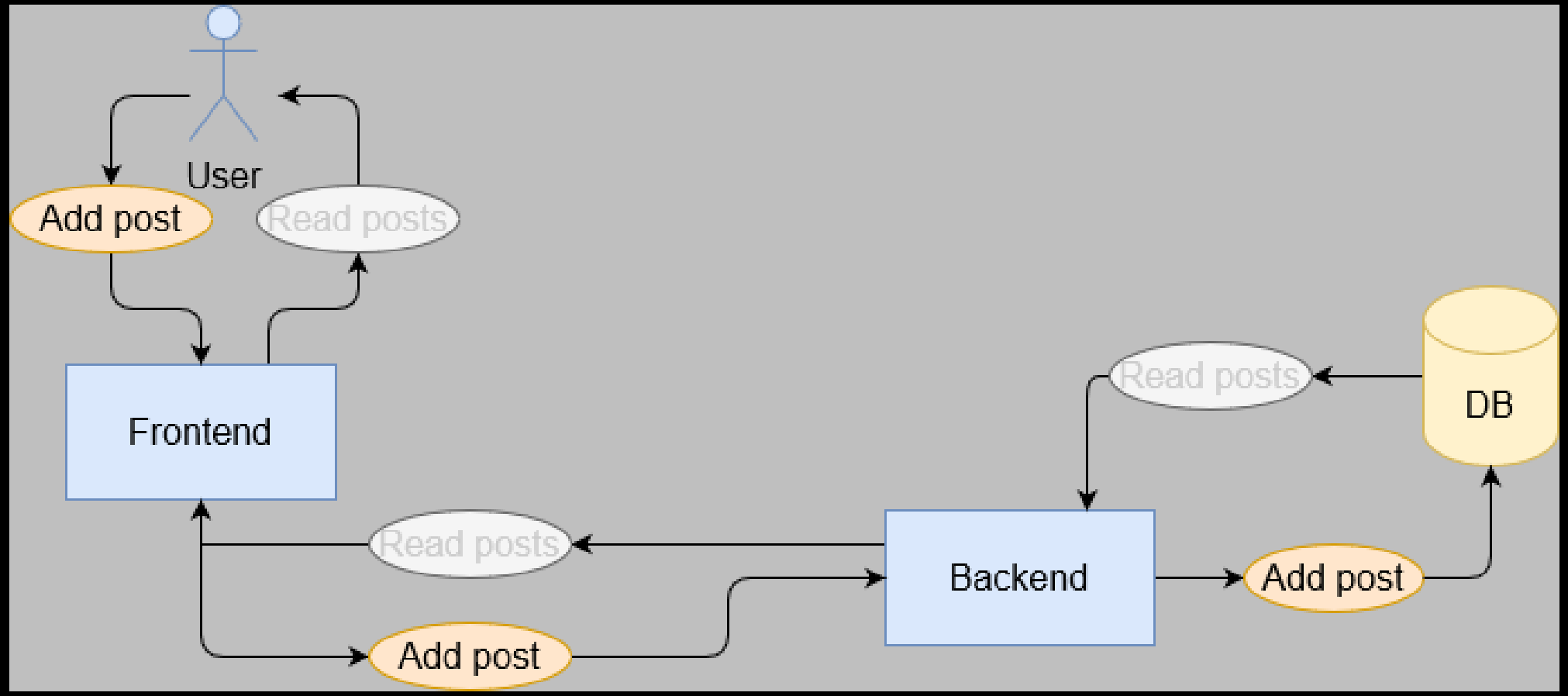
# Blog application



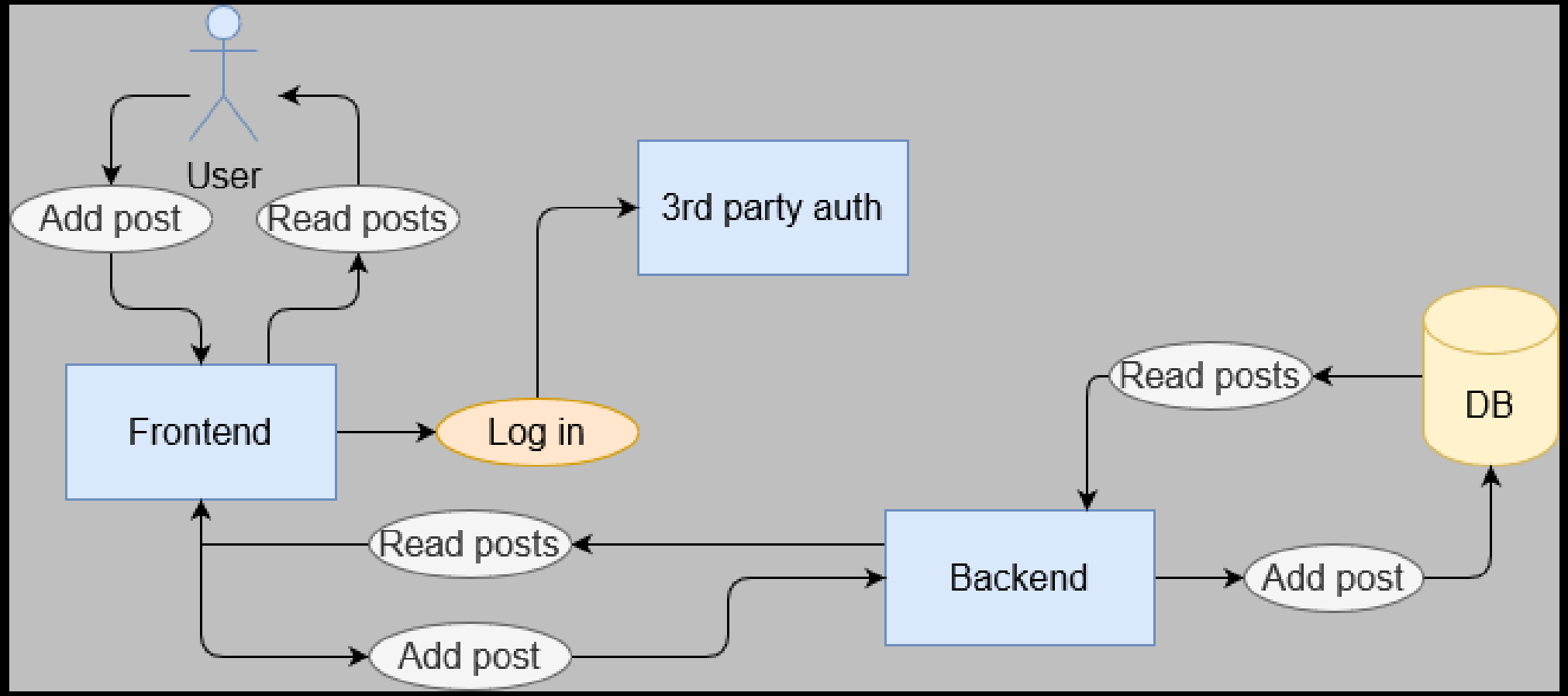
# Blog application



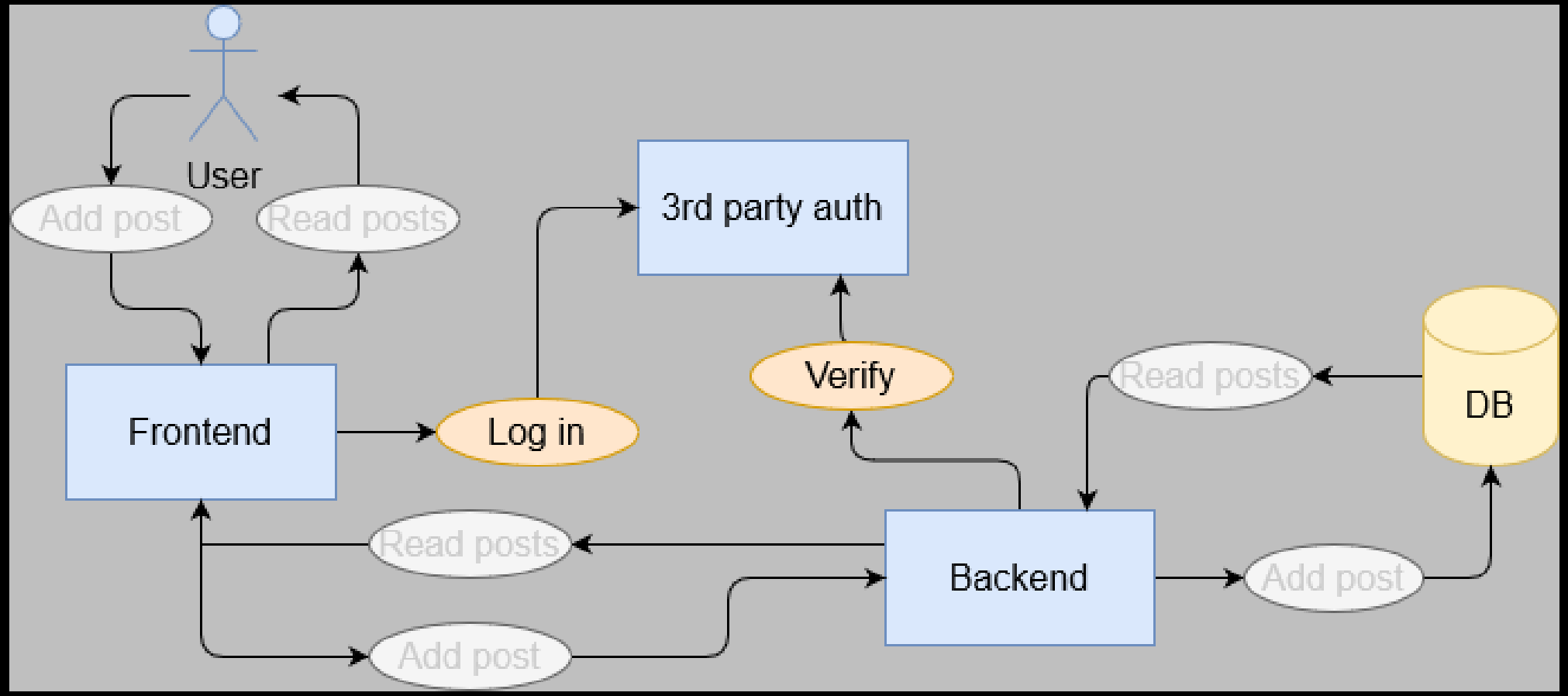
# Blog application



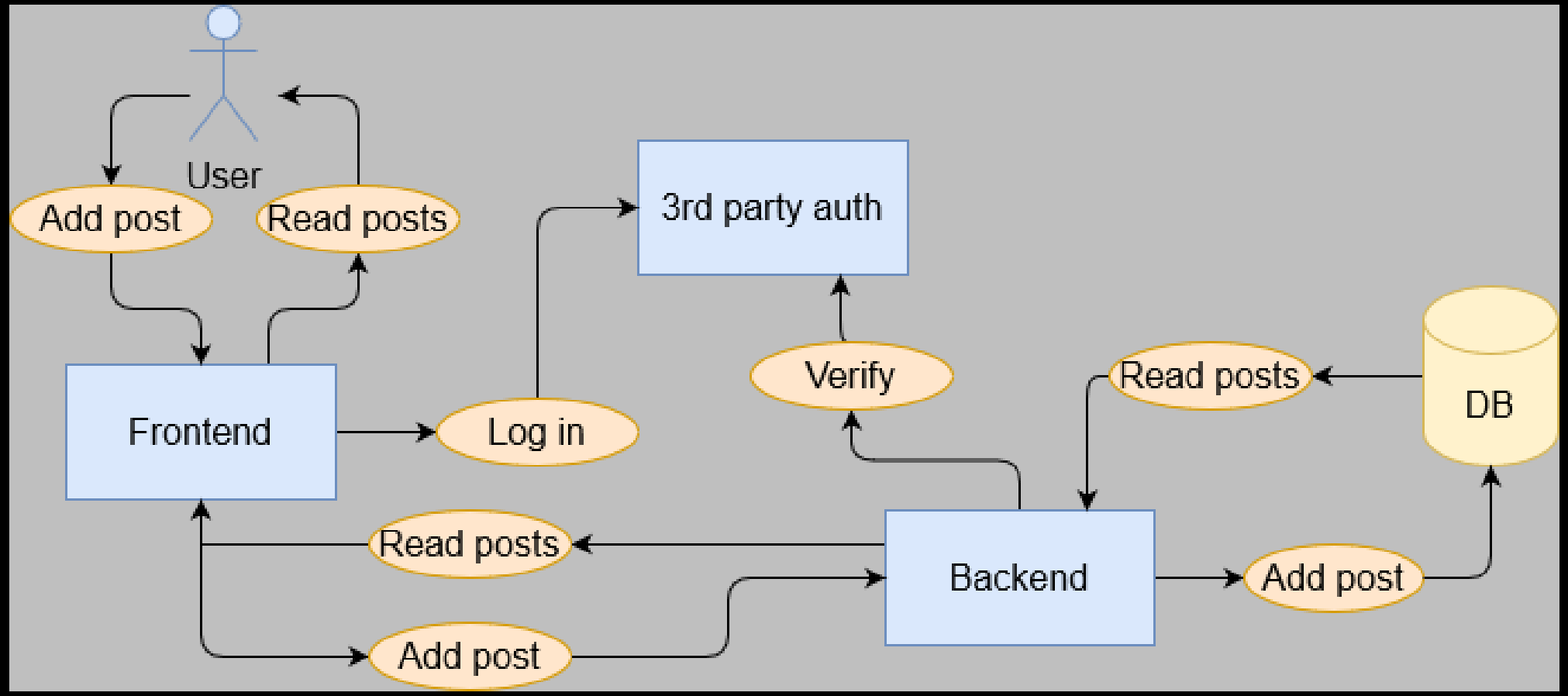
# Blog application



# Blog application



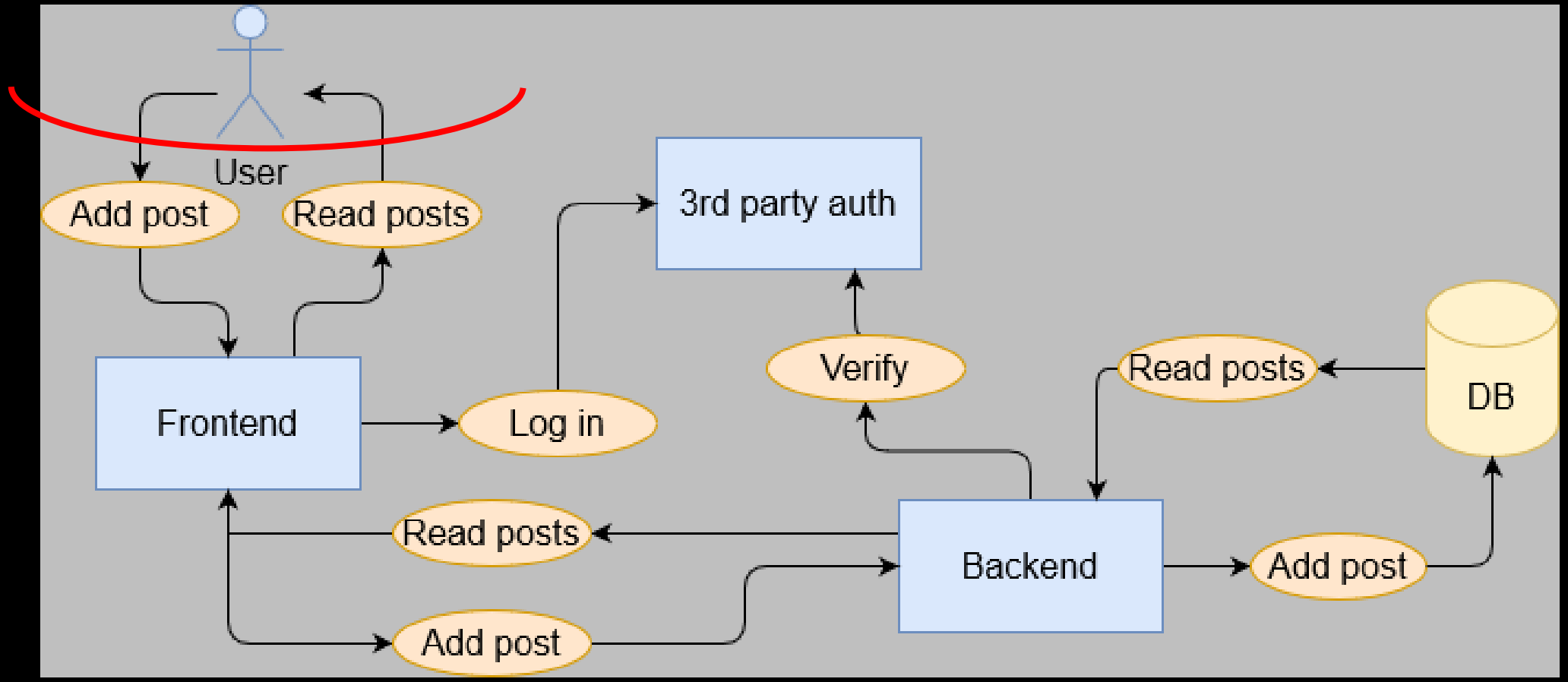
# Blog application



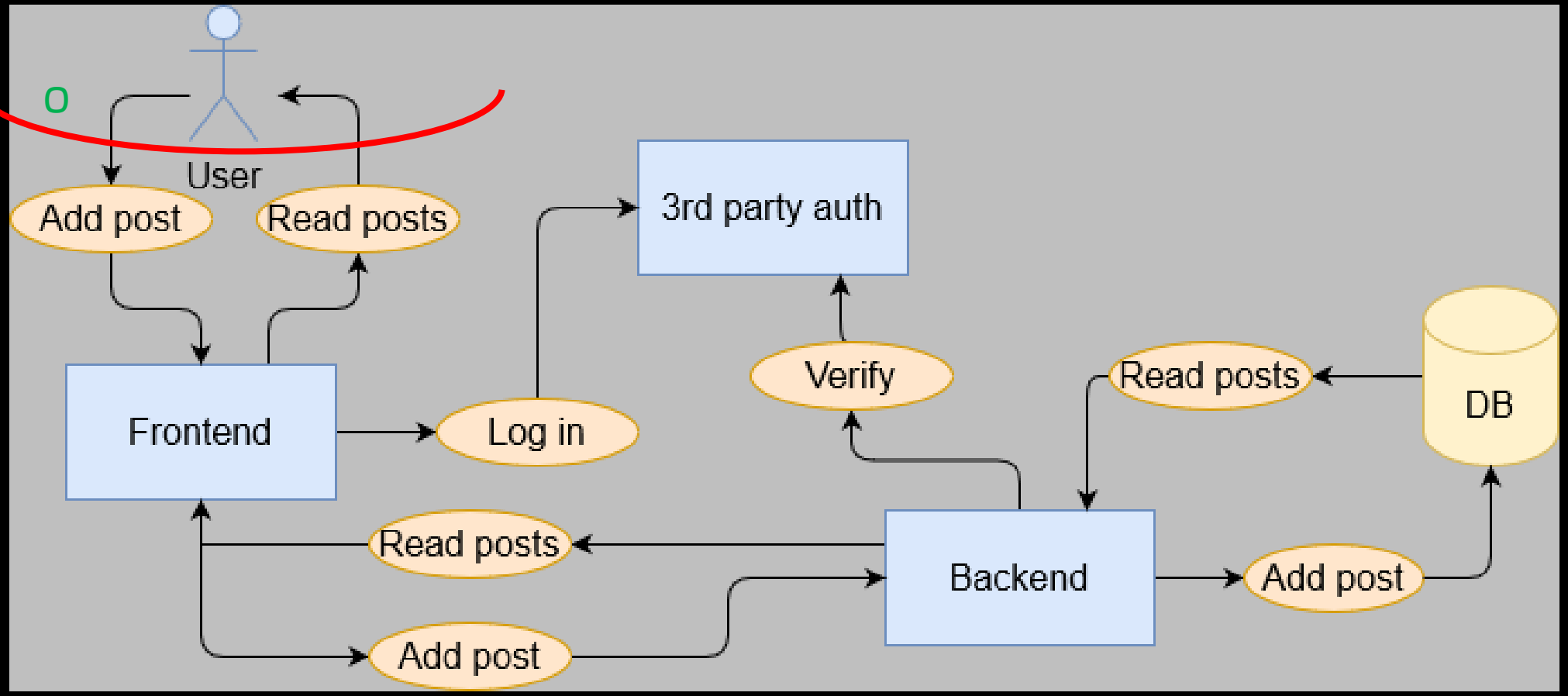
Step 1: data flow diagram

Step 2: trust zones

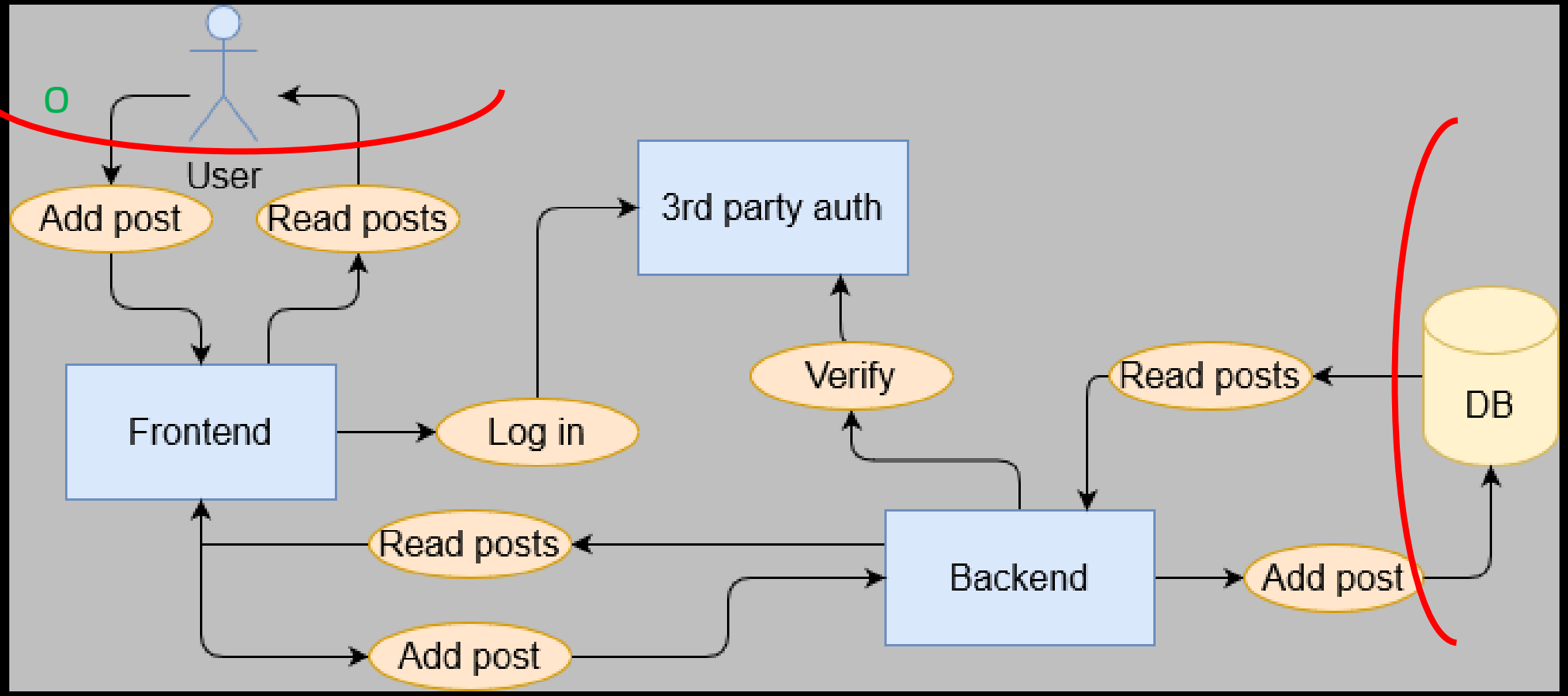
# Trust zones


User
Add post
Read posts
3rd party auth
Frontend
Log in
Verify
Read posts
DB
Read posts
Backend
Add post
Add post

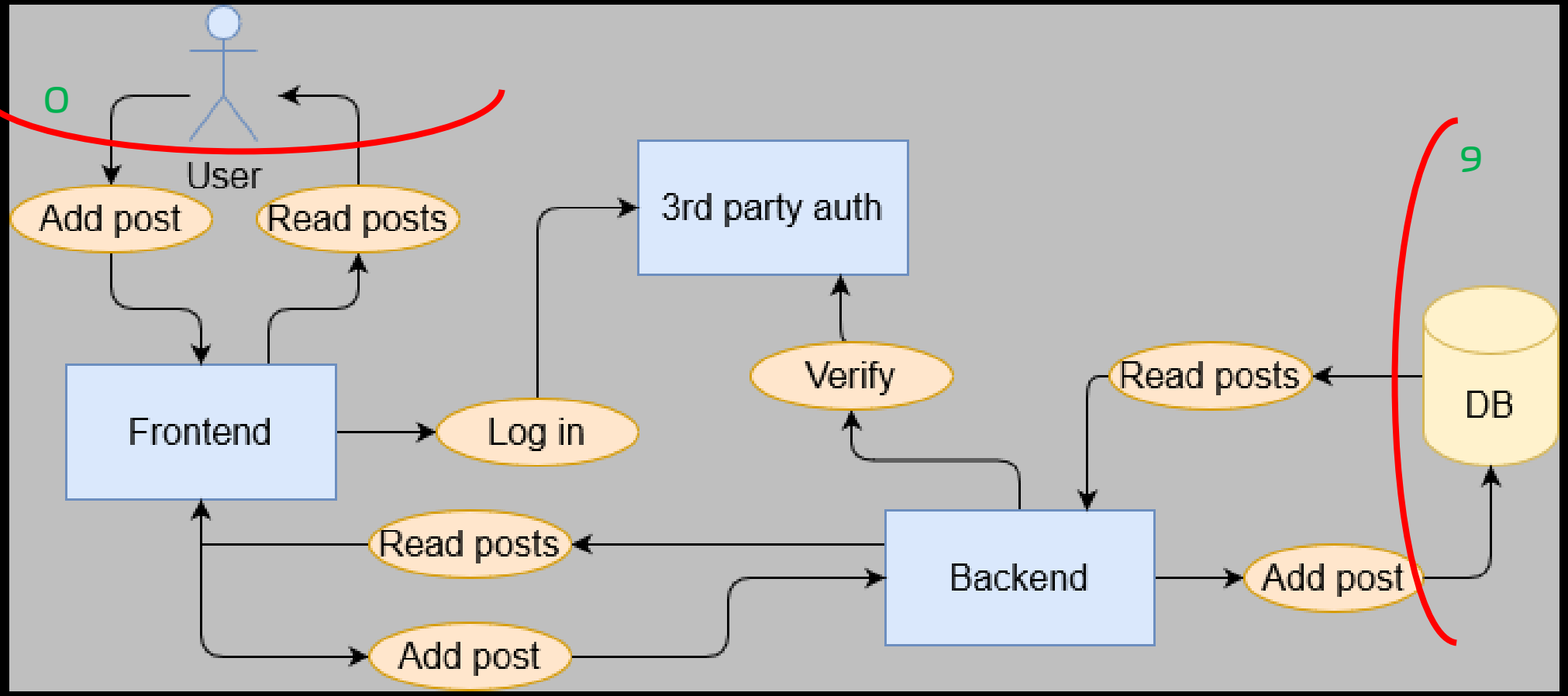
# Trust zones


User
Add post
Read posts
3rd party auth
Frontend
Log in
Verify
Read posts
DB
Read posts
Backend
Add post
Add post

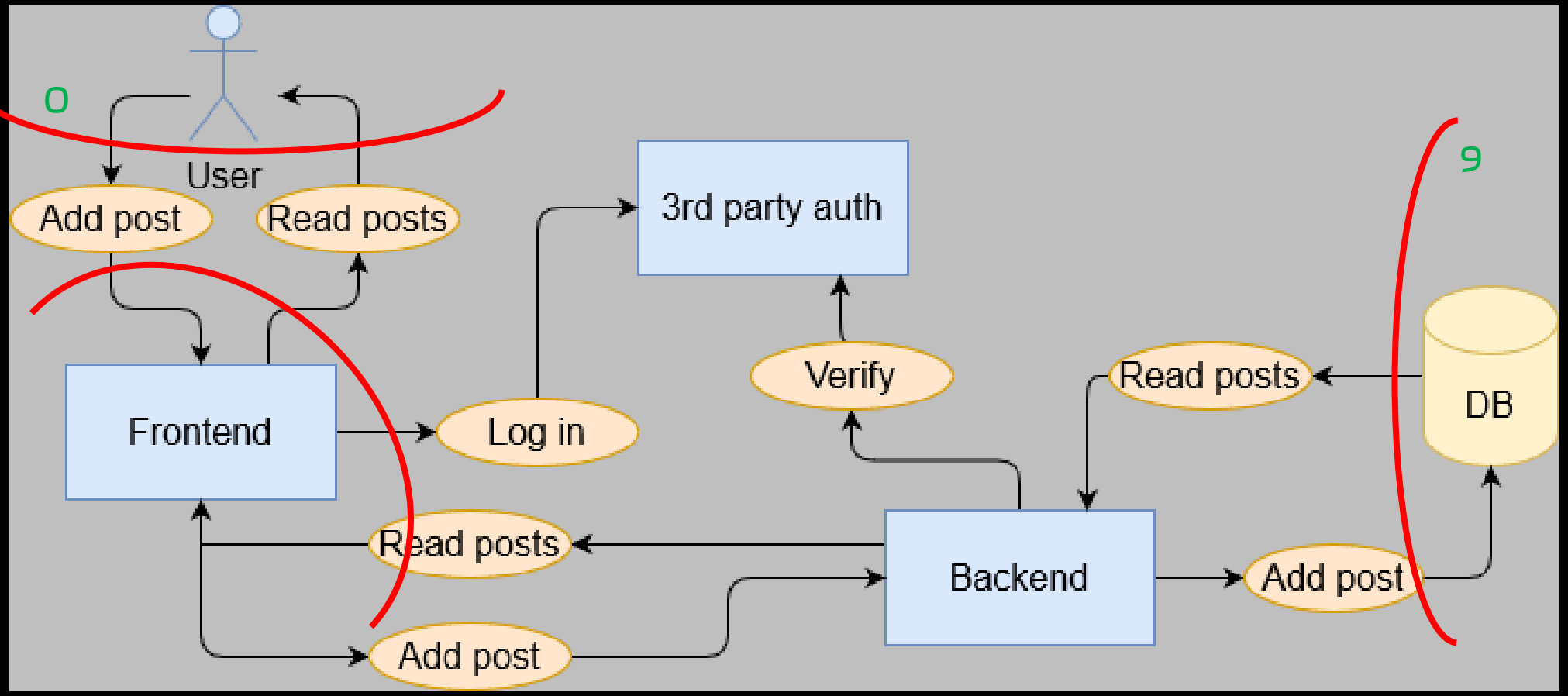
# Trust zones


User
Add post
Read posts
3rd party auth
Frontend
Log in
Verify
Read posts
DB
Read posts
Backend
Add post
Add post

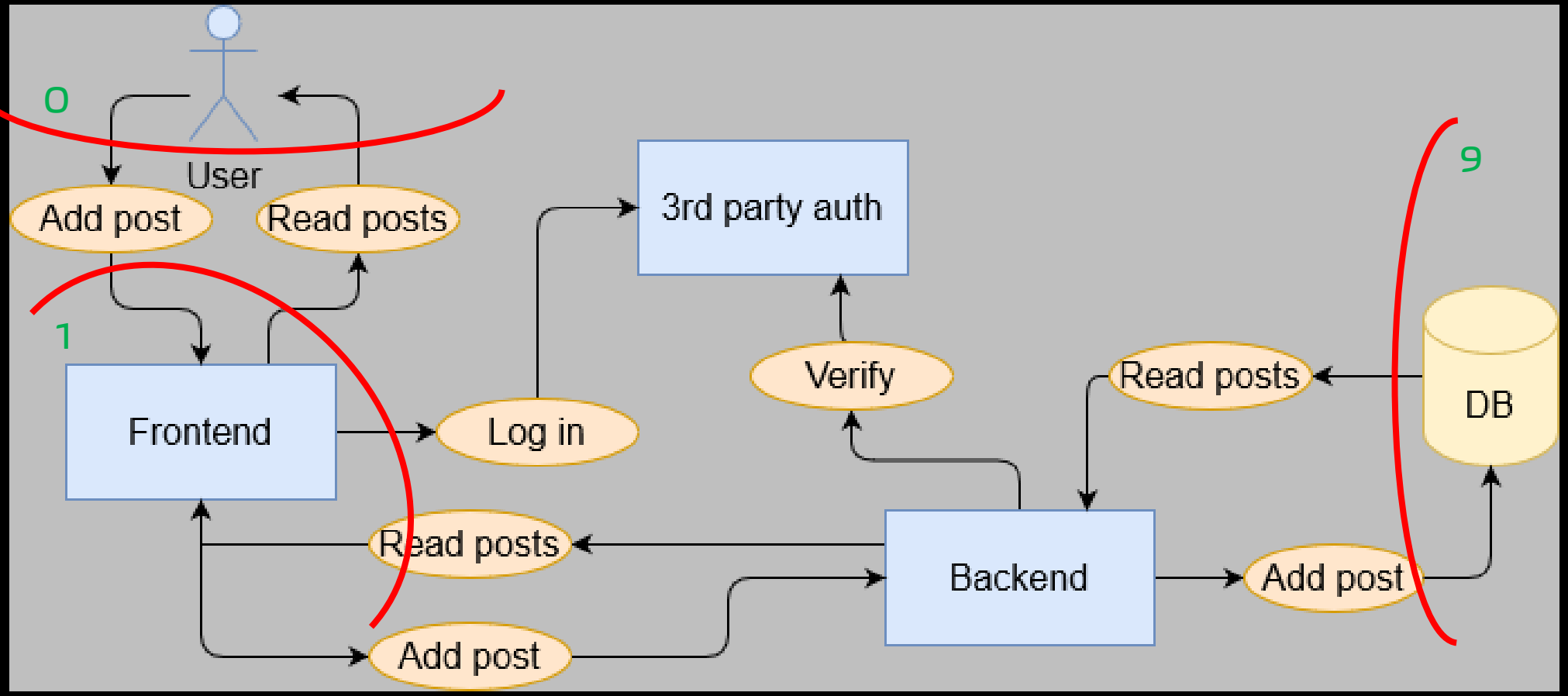
# Trust zones


0
User
Add post
Read posts
Frontend
Log in
3rd party auth
Verify
Read posts
Backend
Add post
Read posts
Add post
9
DB

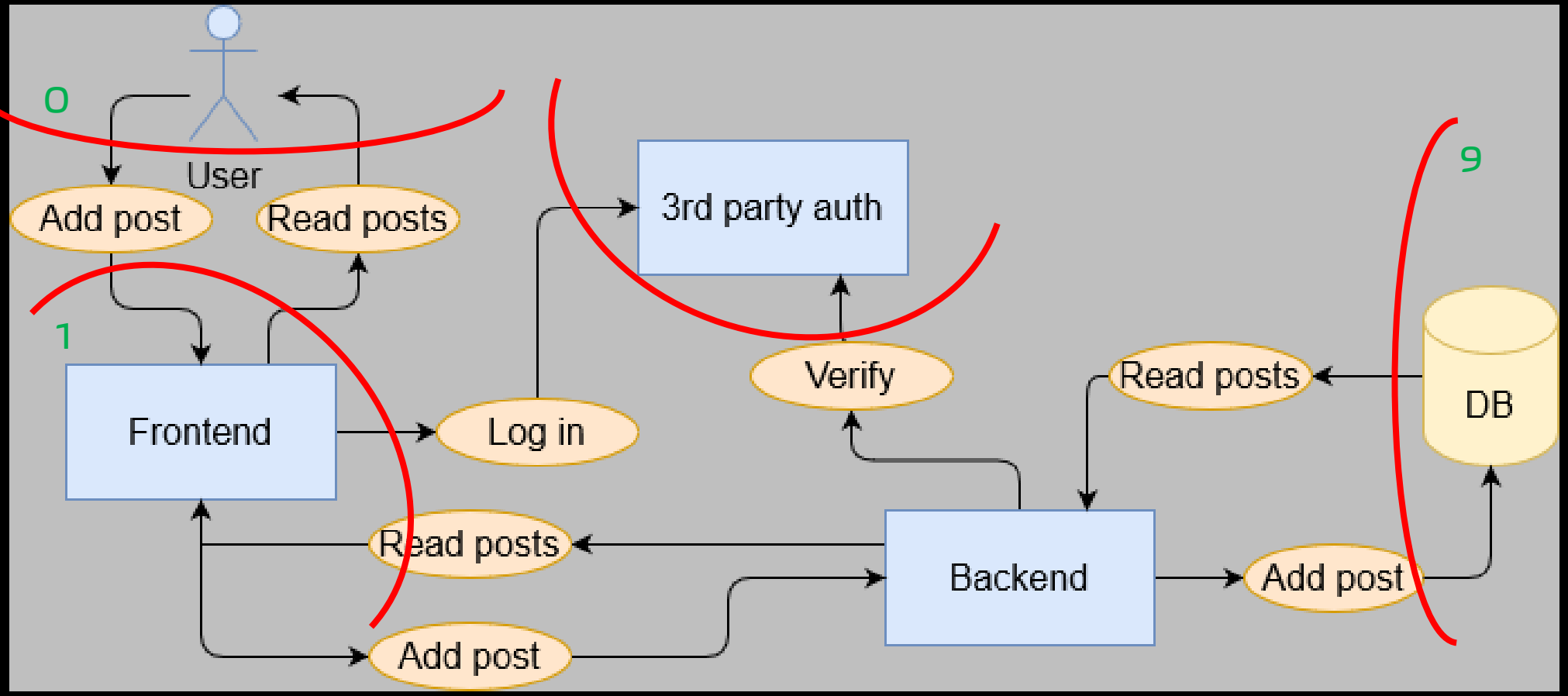
# Trust zones


0
User
Add post
Read posts
3rd party auth
9
Frontend
Log in
Verify
Read posts
DB
Read posts
Backend
Add post
Add post

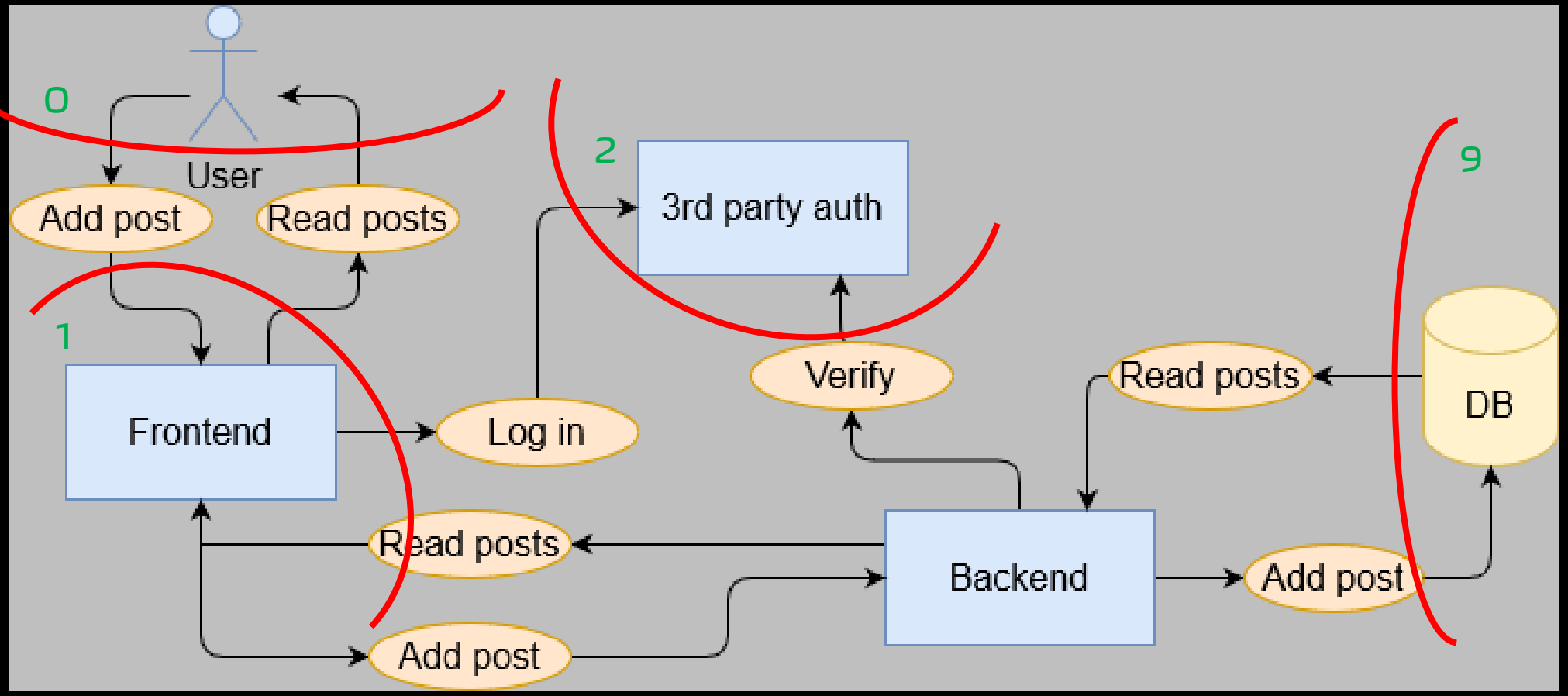
# Trust zones


0
User
Add post
Read posts
1
Frontend
Log in
3rd party auth
Verify
Read posts
Add post
Backend
Read posts
Add post
9
DB

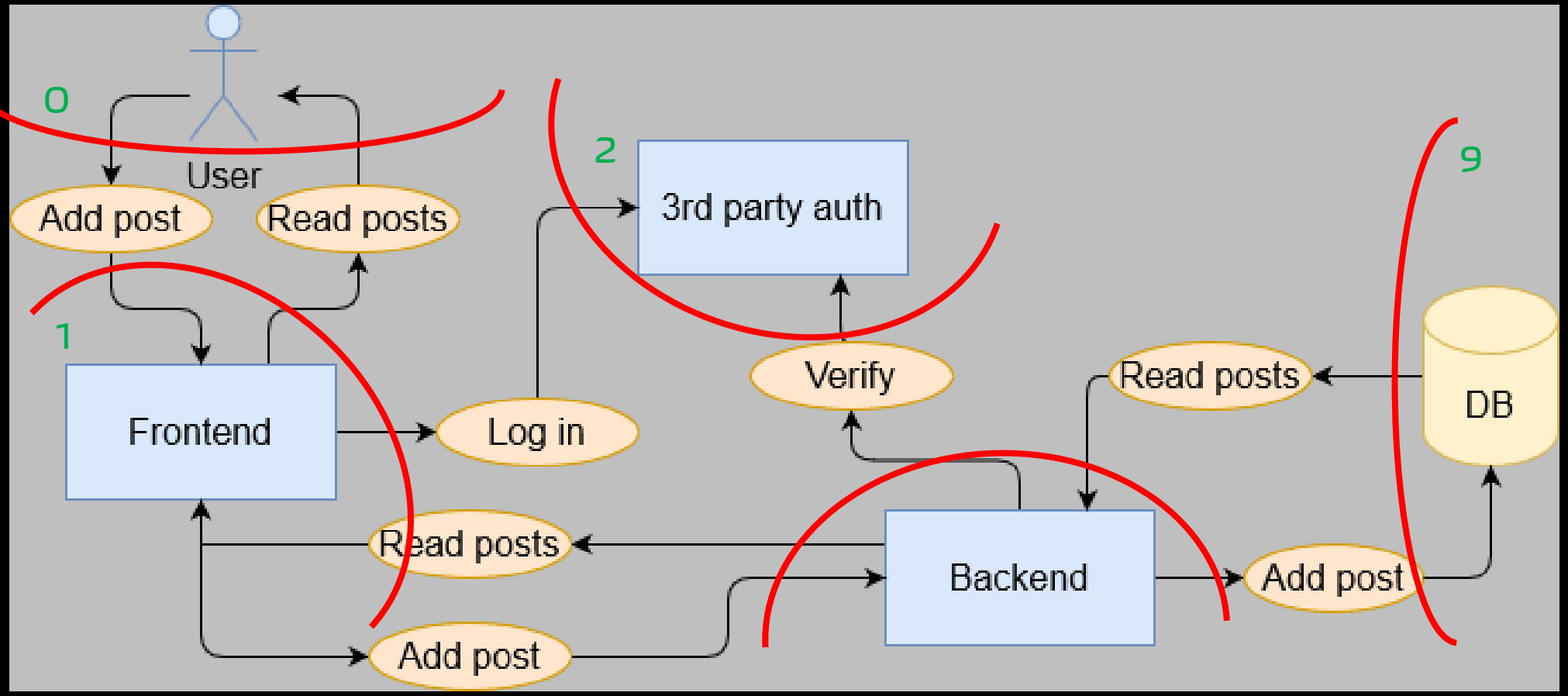
# Trust zones


0
User
Add post
Read posts
1
Frontend
Log in
3rd party auth
Verify
Read posts
Add post
Backend
9
DB
Read posts
Add post

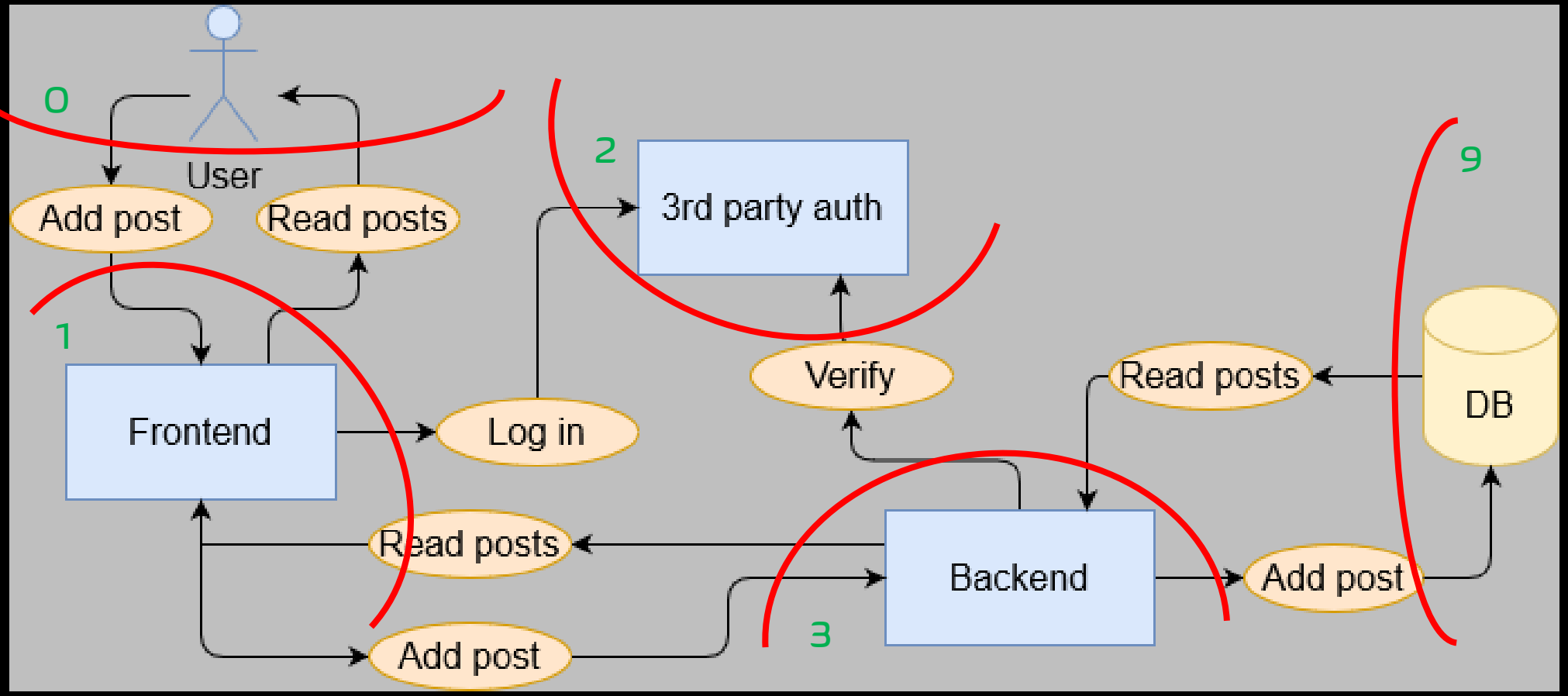
# Trust zones


0
User
Add post
Read posts
1
Frontend
Log in
2
3rd party auth
Verify
Read posts
Add post
Backend
Read posts
Add post
9
DB

# Trust zones


0
User
Add post
Read posts
1
Frontend
Log in
2
3rd party auth
Verify
9
Read posts
DB
Read posts
Backend
Add post
Add post

# Trust zones


0
User
Add post
Read posts
1
Frontend
2
3rd party auth
Log in
Verify
9
DB
Read posts
Read posts
Backend
Add post
Add post
3

Step 1: data flow diagram

Step 2: trust zones

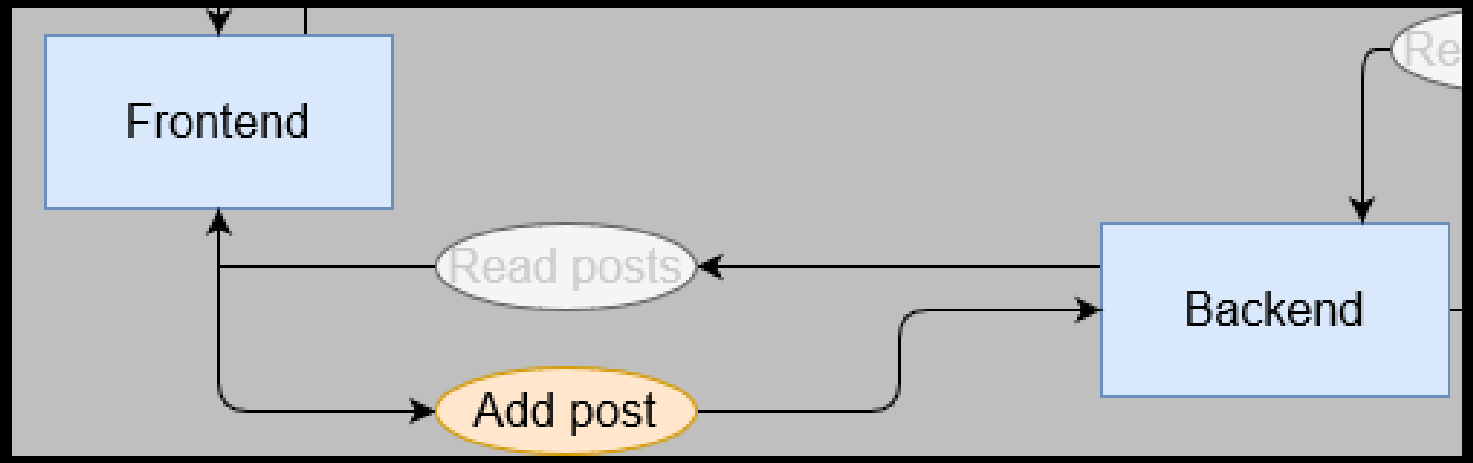
Step 3: risks

# Risks

Defined for every process that goes
between zones with different levels

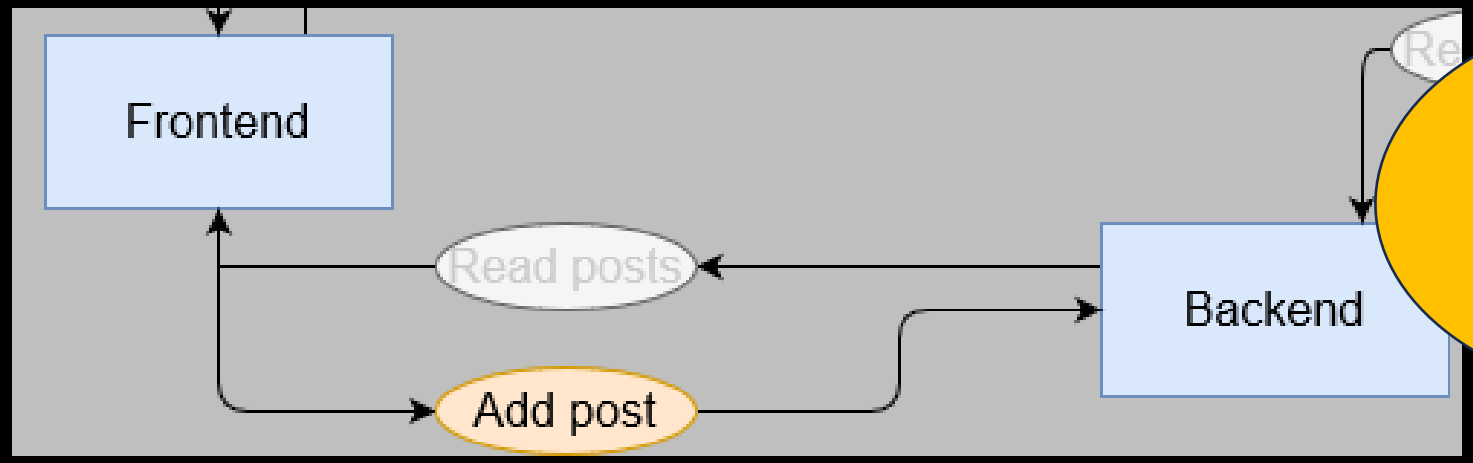
# Risks

1. XXE attack
2. Overflow
3. Unauth addition of post
4. Malicious input files
5. Man in the middle



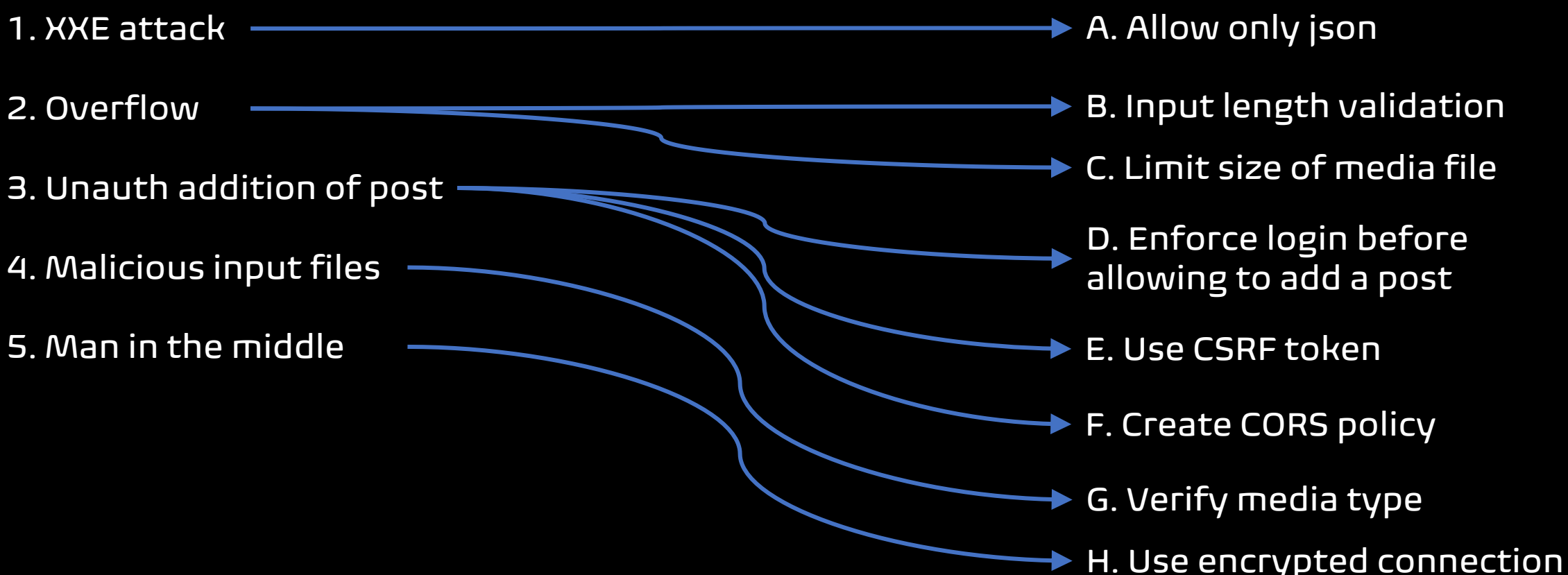
# Risks

1. XXE attack
2. Overflow
3. Unauth addition of post
4. Malicious input files
5. Man in the middle

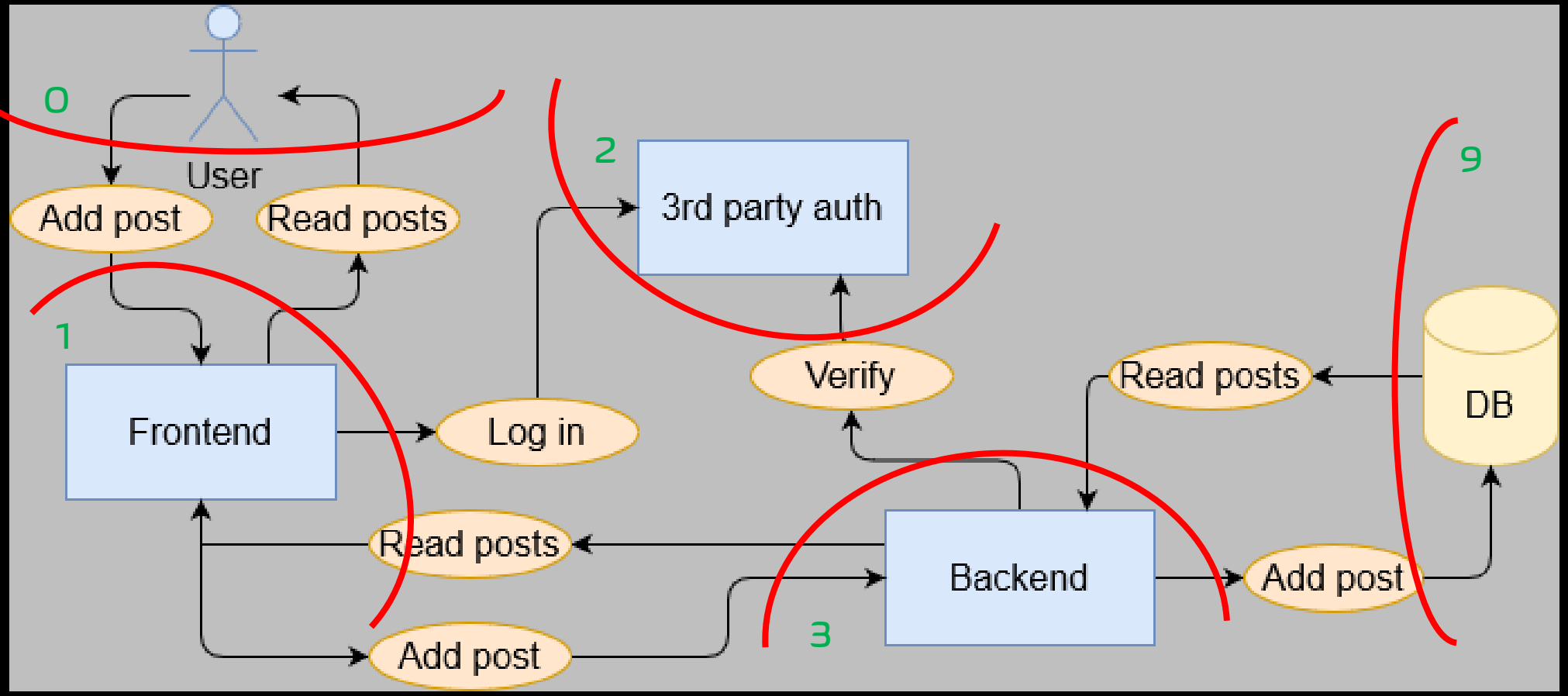



Step 1. data flow diagram
Step 2. trust zones
Step 3. risks
Step 4. remediations

# Remediations



# Repeat 8 times

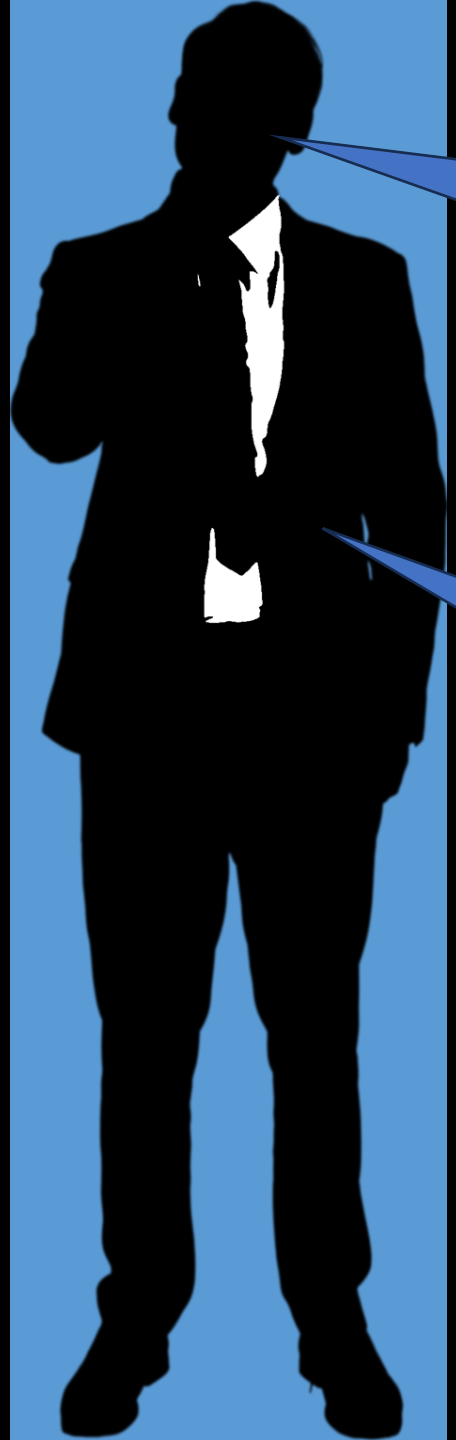


Step 1. data flow diagram
Step 2. trust zones
Step 3. risks
Step 4. remediations
Step 5. matrix

# Vulnerabilities matrix

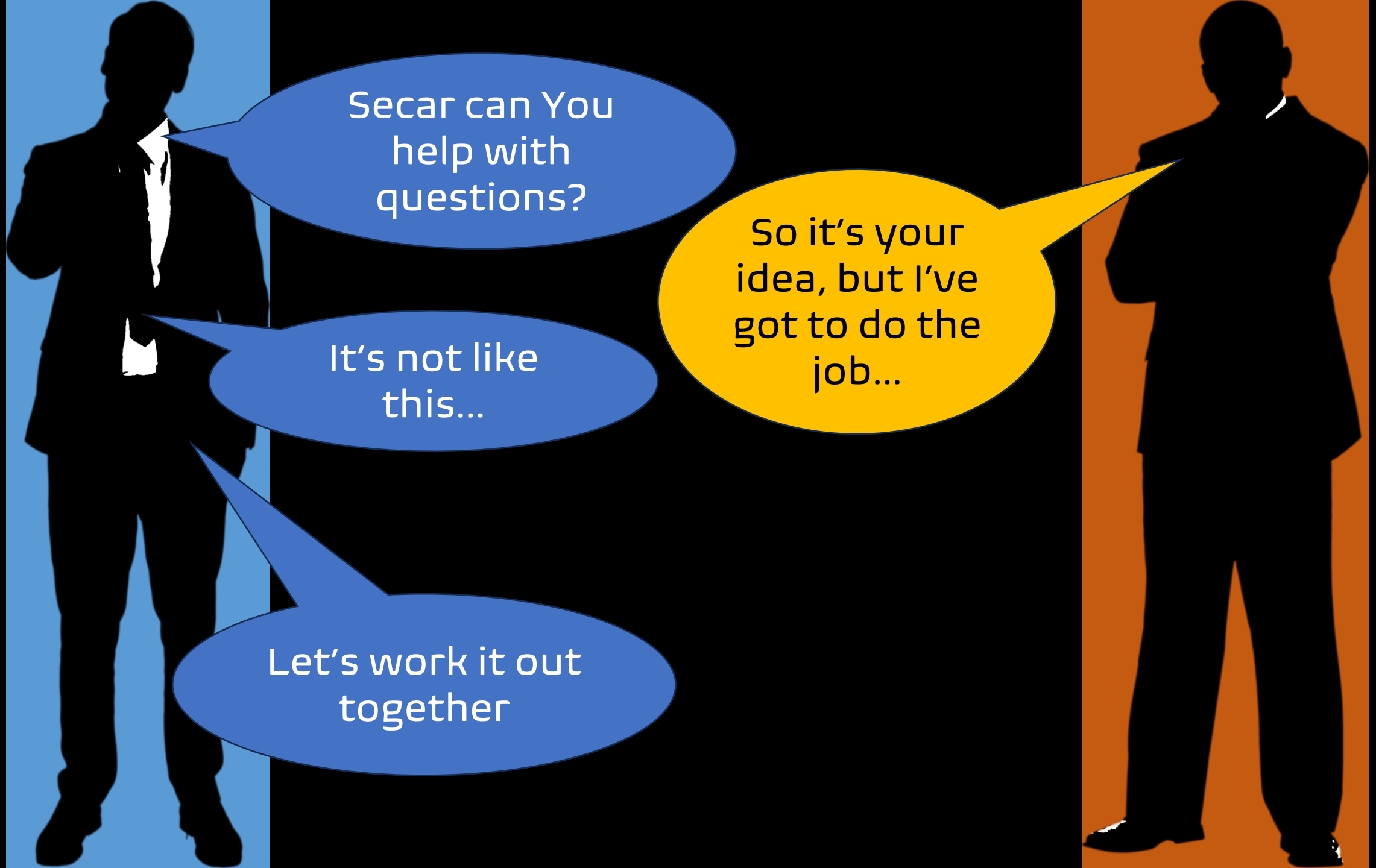
| Risk | Criticality | Probability | Score |
|---|---|---|---|
| Overflow | 3 | 7 | 10 |
| Unauth access | 9 | 6 | 15 |
| Malicious js | 5 | 9 | 14 |
| | | | |
| | | | |


Let’s introduce RA
Won’t it take too long?



Before doing TM, let's put a quesionnaire

Risk Assessment


Secar can You help with questions?
It’s not like this…
Let’s work it out together
So it’s your idea, but I’ve got to do the job…

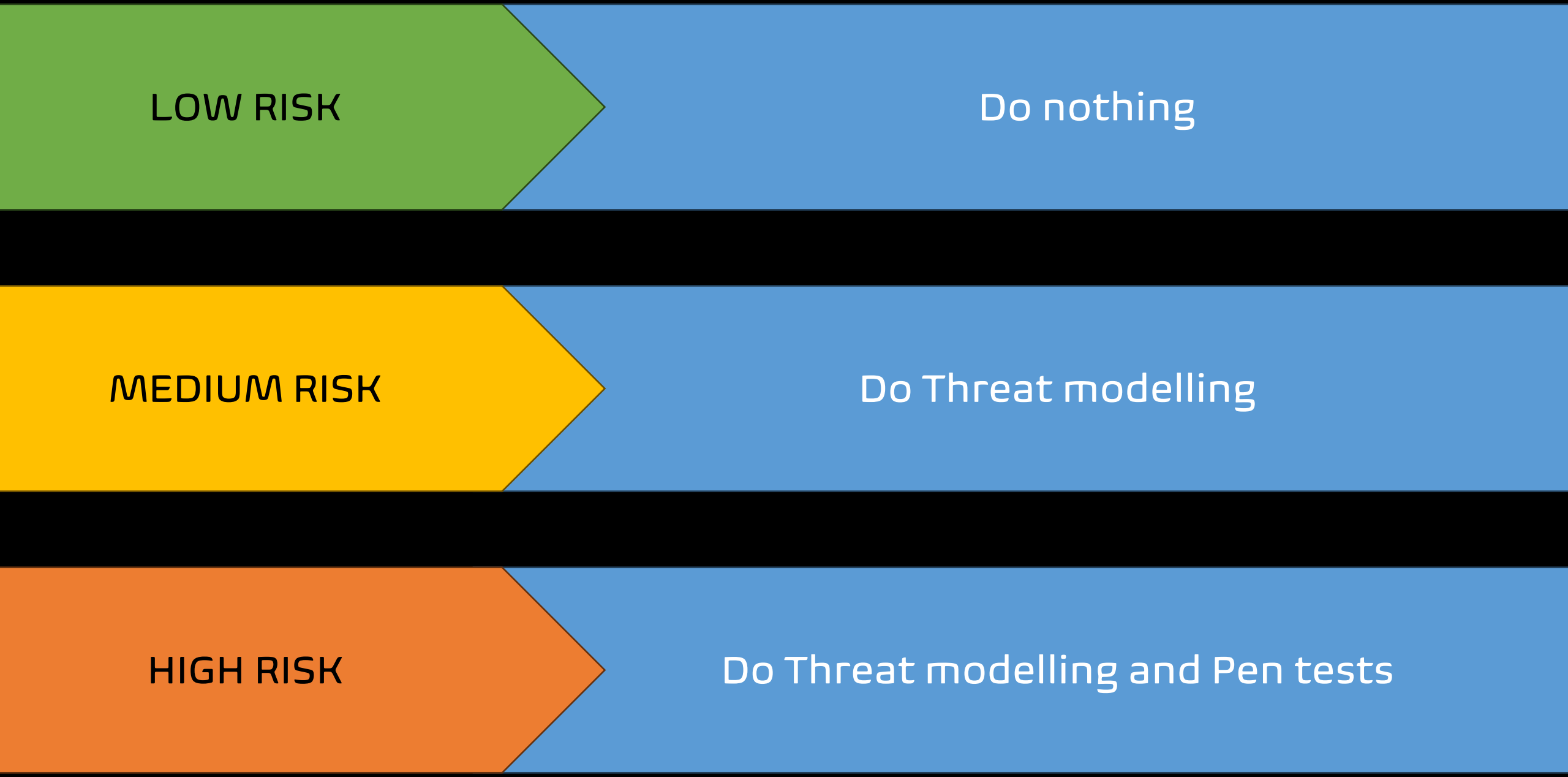
Q1: Does it touch customer data?

Q2: Does it expose new public endpoint?

Q3: Is new service introduced?

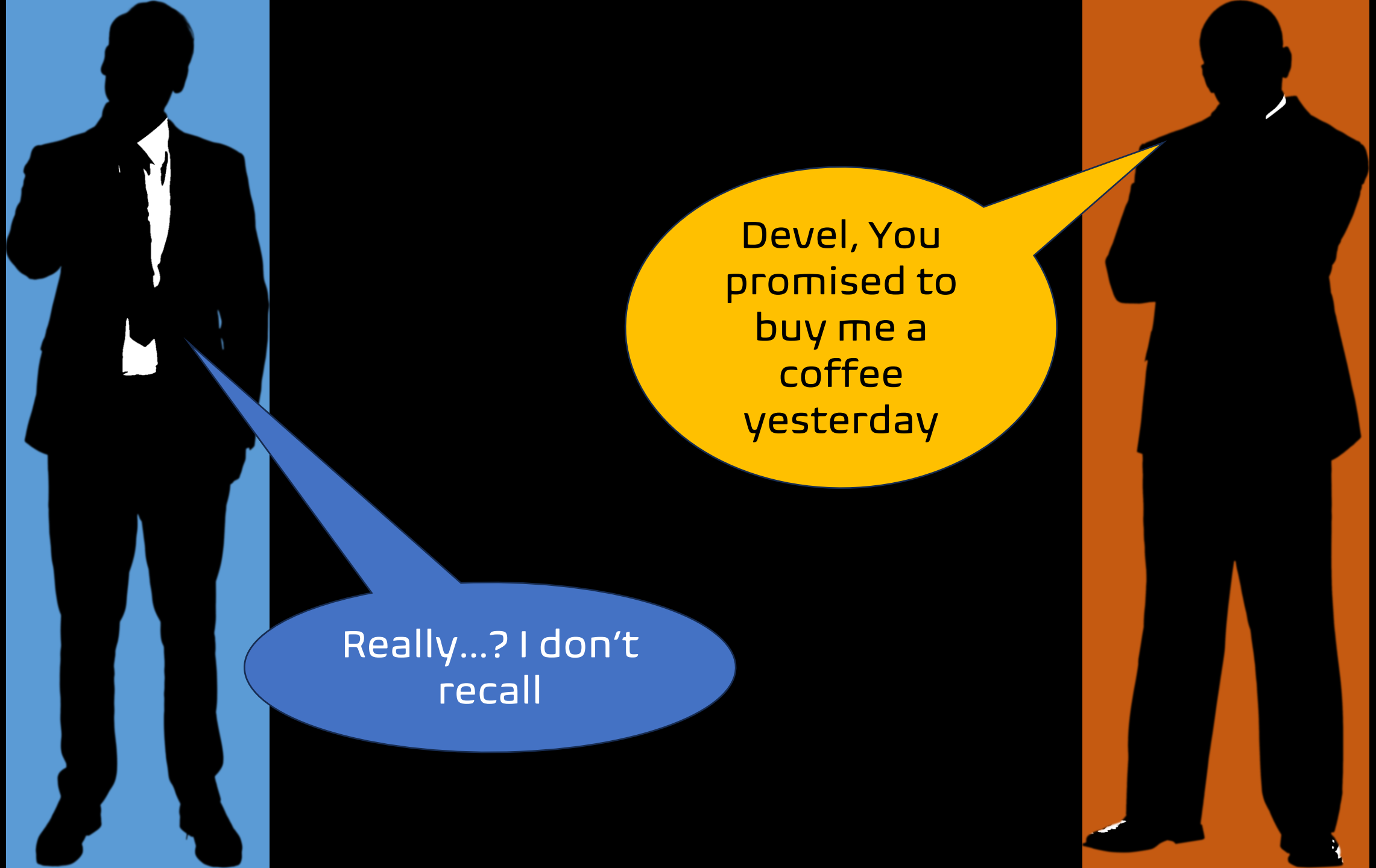
Q4: Does it change auth mechanism?

Q5: Is new 3rd party api used?


LOW RISK
Do nothing
MEDIUM RISK
Do Threat modelling
HIGH RISK
Do Threat modelling and Pen tests

Answers to RA questions and remediations in TM are

# Promises


Devel, You promised to buy me a coffee yesterday
Really...? I don't recall

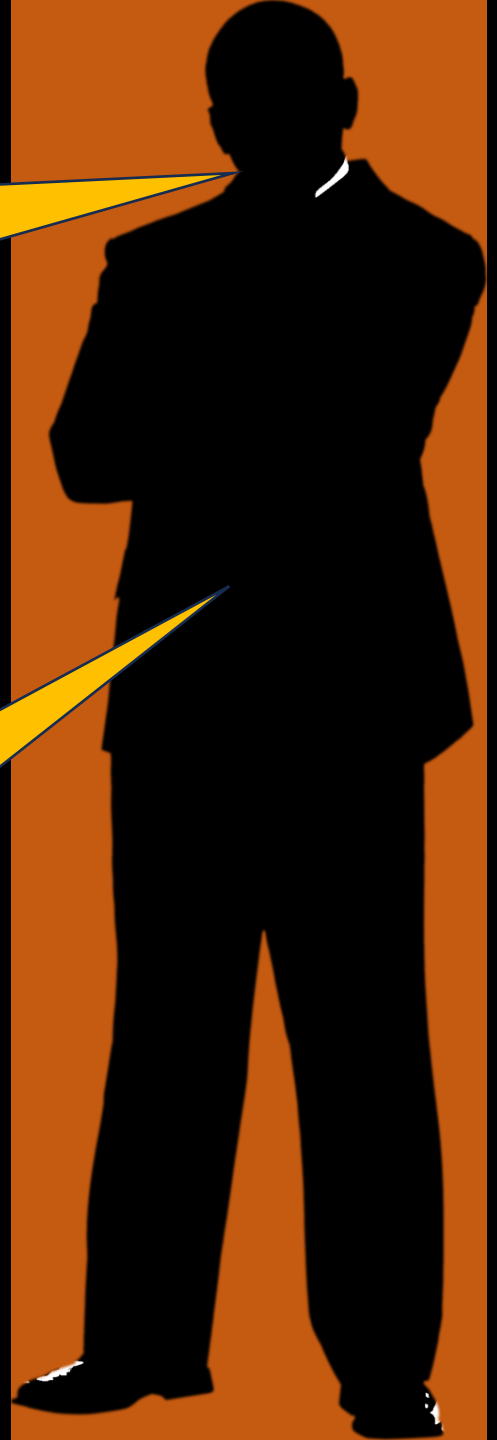
# Security Validation

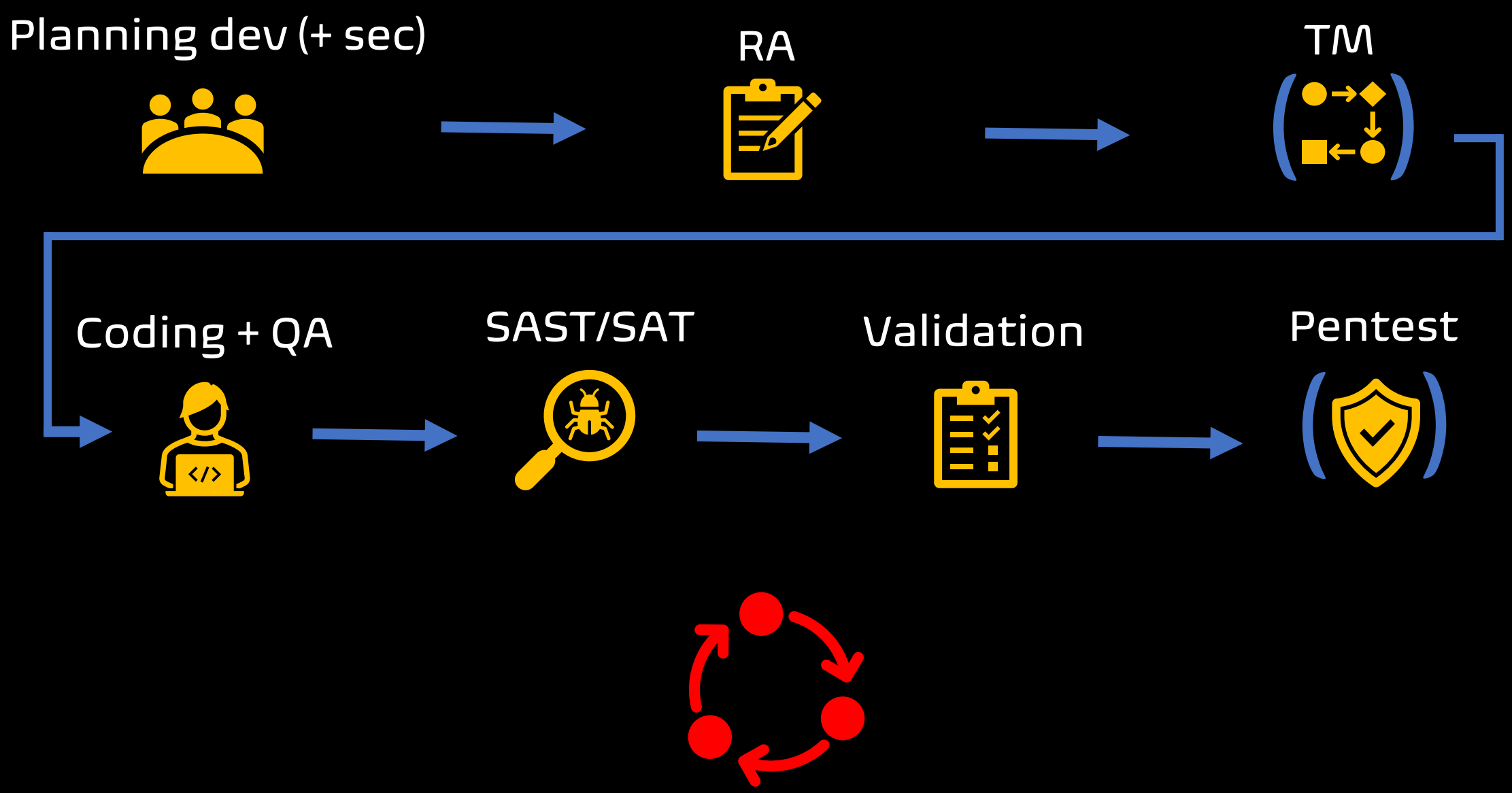
Comment to RA answers and TM remediations with proof:

- Link to code line
- Link to generated report
- Screenshot
- Logs

I think it's about time to sum up

... and for a promised coffee




Planning dev (+ sec)
RA
TM
Coding + QA
SAST/SAT
Validation
Pentest

# Cons

More work on developers side

Requires training

Delays release

Vulnerable to human errors

# Pros

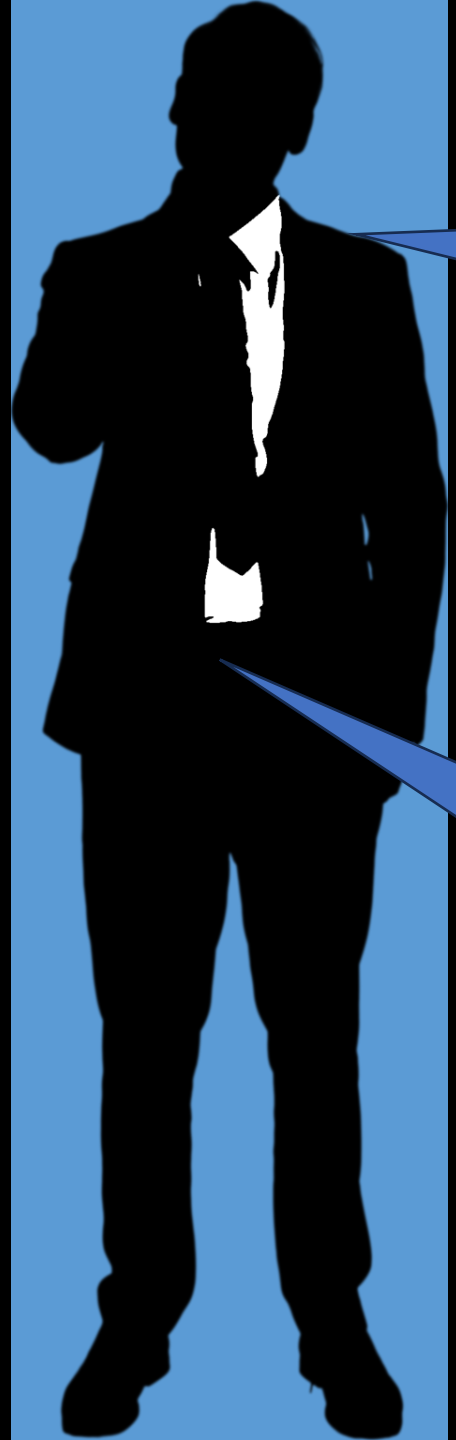
Ealier security feedback

For simple cases, doesn't delay release

Less security incidents

More secure solution

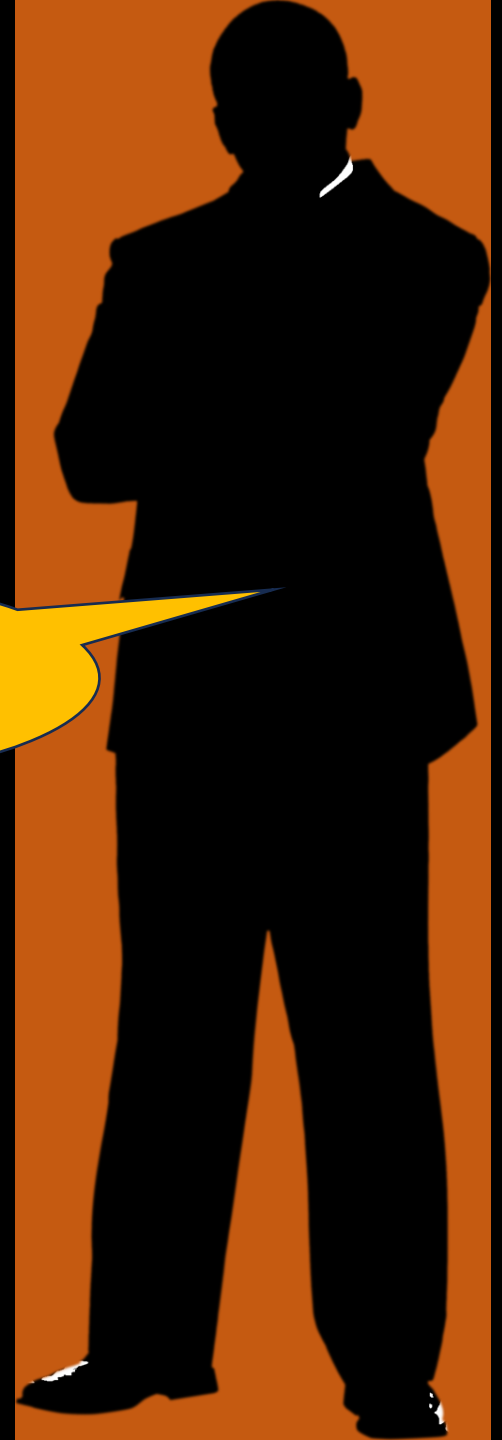
Better security awarness in dev team



I guess this is enough as 1st draft

You can go back to your talk Marcin

Definitelly




Nevertheless, have a good day, bye!
Bye!