<DevSum>

Building Practical Zero Trust APIs with .NET9 and Azure

Roy Cornelissen

active SOLUTION Cornerstone

Agria Djurförsäkring

VONAGE Part of Ericsson

RaySearch Laboratories

Duende

bulbul

SOFTRONIC

Polisen

Photo by the blowup on Unsplash

Hi, I'm Roy

Cloud Solution Architect at Xebia

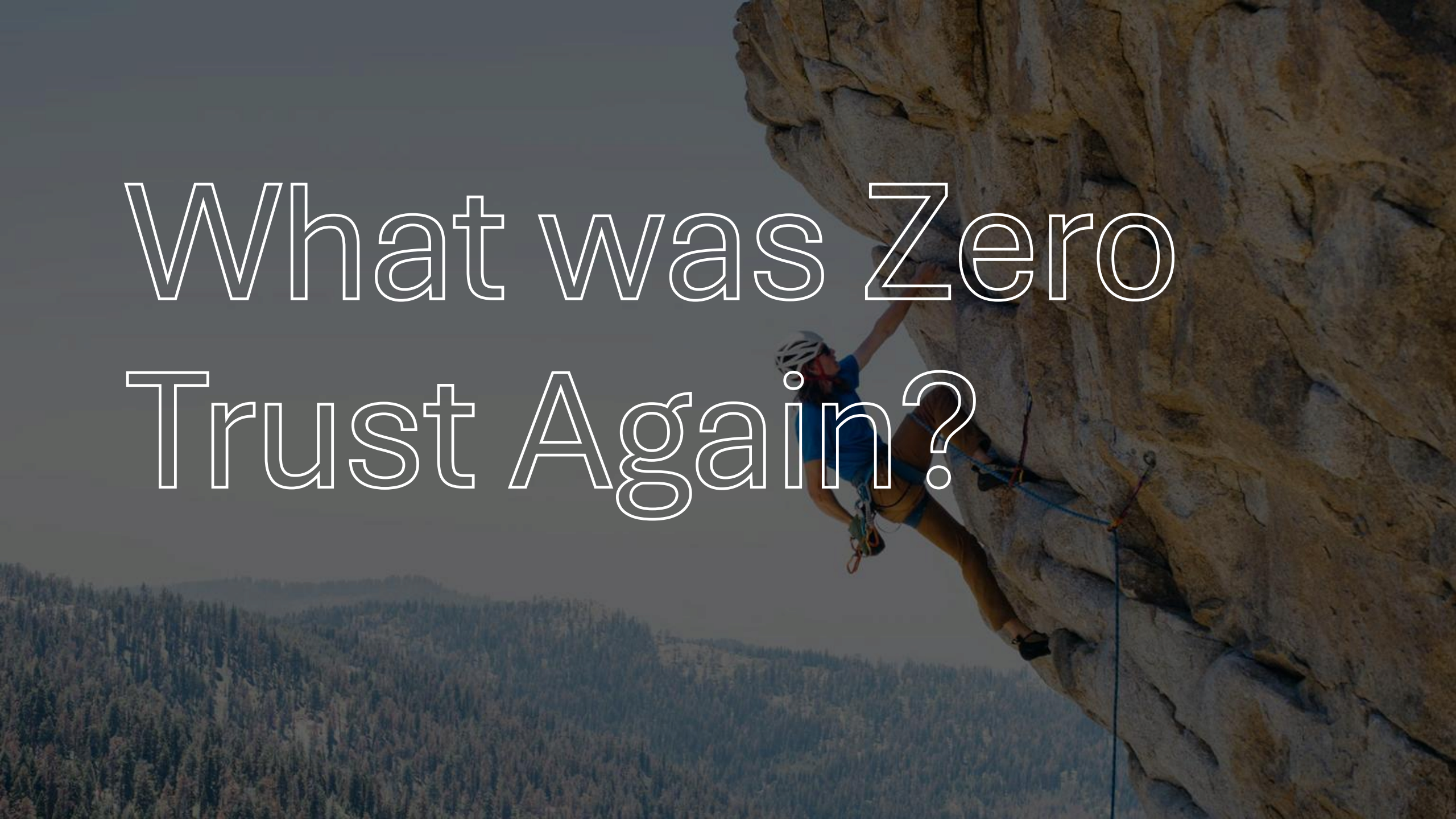
roy.cornelissen@xebia.com
@roycornelissen
linkedin.com/in/roycornelissennl

Photo by Andras Vas on Unsplash

14,985,121,175

Have I Been Pwned

What was Zero Trust Again?



The concept

Zero Trust is a **security concept** based on the principle of "**never trust, always verify**." It assumes that no component - whether hardware or software - should be implicitly trusted, regardless of whether it is inside or outside the network perimeter.

Every access request must be continuously authenticated, authorized, and validated to ensure security.

Stephen Marsh

April 1994

Zero Trust, and the fact that there is always a Handsome Prince...

John Kindervag - Forrester Research 2010

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
U.S. DEPARTMENT OF COMMERCE

Zero Trust Architecture - NIST SP 800-207

2018

https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf

Perimeterless Security

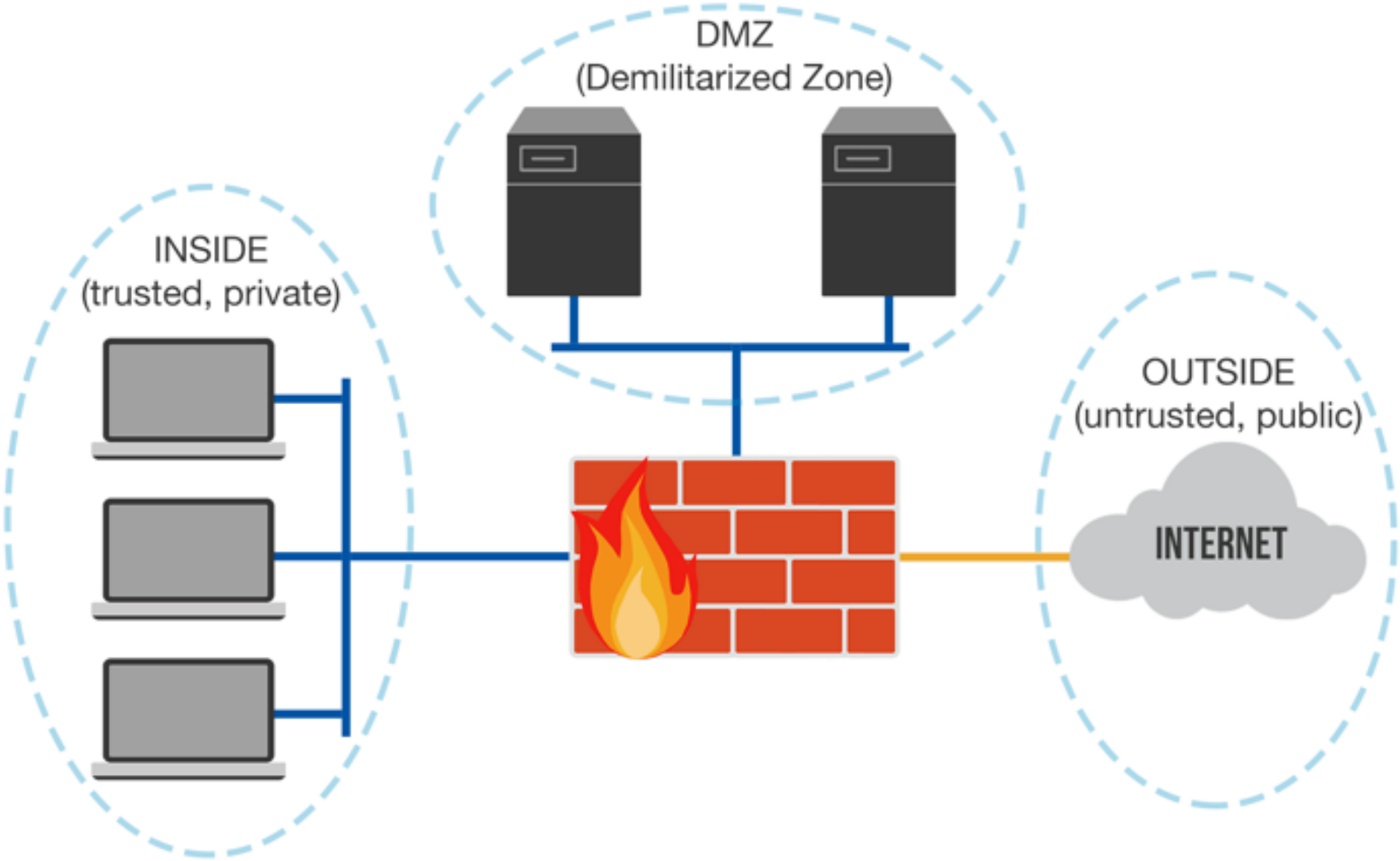

DMZ
(Demilitarized Zone)
INSIDE
(trusted, private)
OUTSIDE
(untrusted, public)
INTERNET

Photo by BRUNO CERVERA on Unsplash

Zero Trust

Explicitly Verify

Zero Trust teaches us to never trust, and always verify

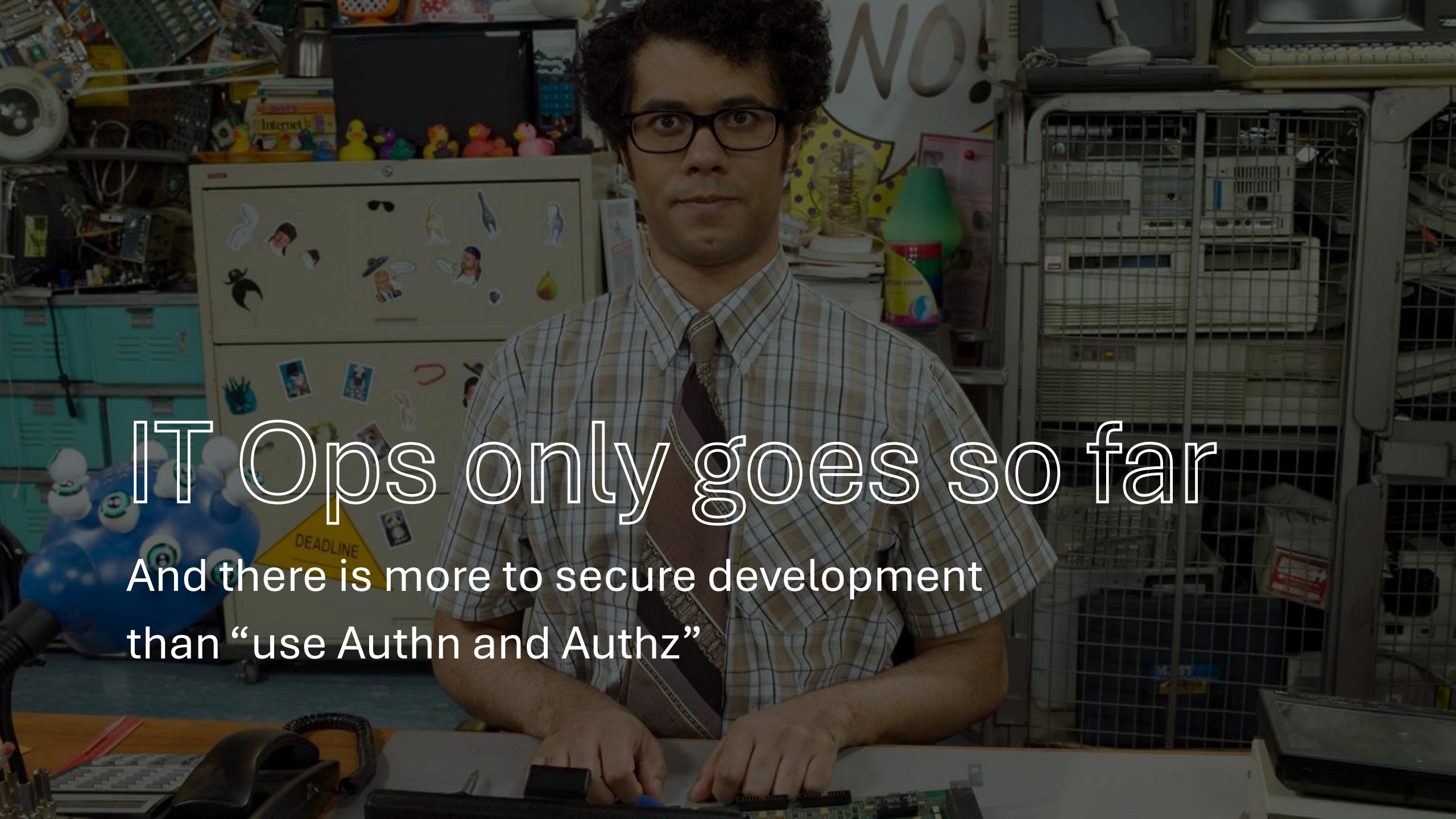
Limit User Access

Zero Trust uses the principle of least privilege access and limits users

Assume Breach

Zero Trust always assumes breach and verifies each request

Why this talk?



IT Ops only goes so far

And there is more to secure development than “use Authn and Authz”

Practical Use Case

Mini Bank: Payment handling... overly simplified

Few notes from our CTO

"I'm gonna need you to build this"

- Payments & Accounts API should be available
- Use .NET 9 with C# and host it in containers
- Azure Subscription is setup by Platform team
- There's an agreed drawing somewhere on a paper napkin

Agreed Drawing

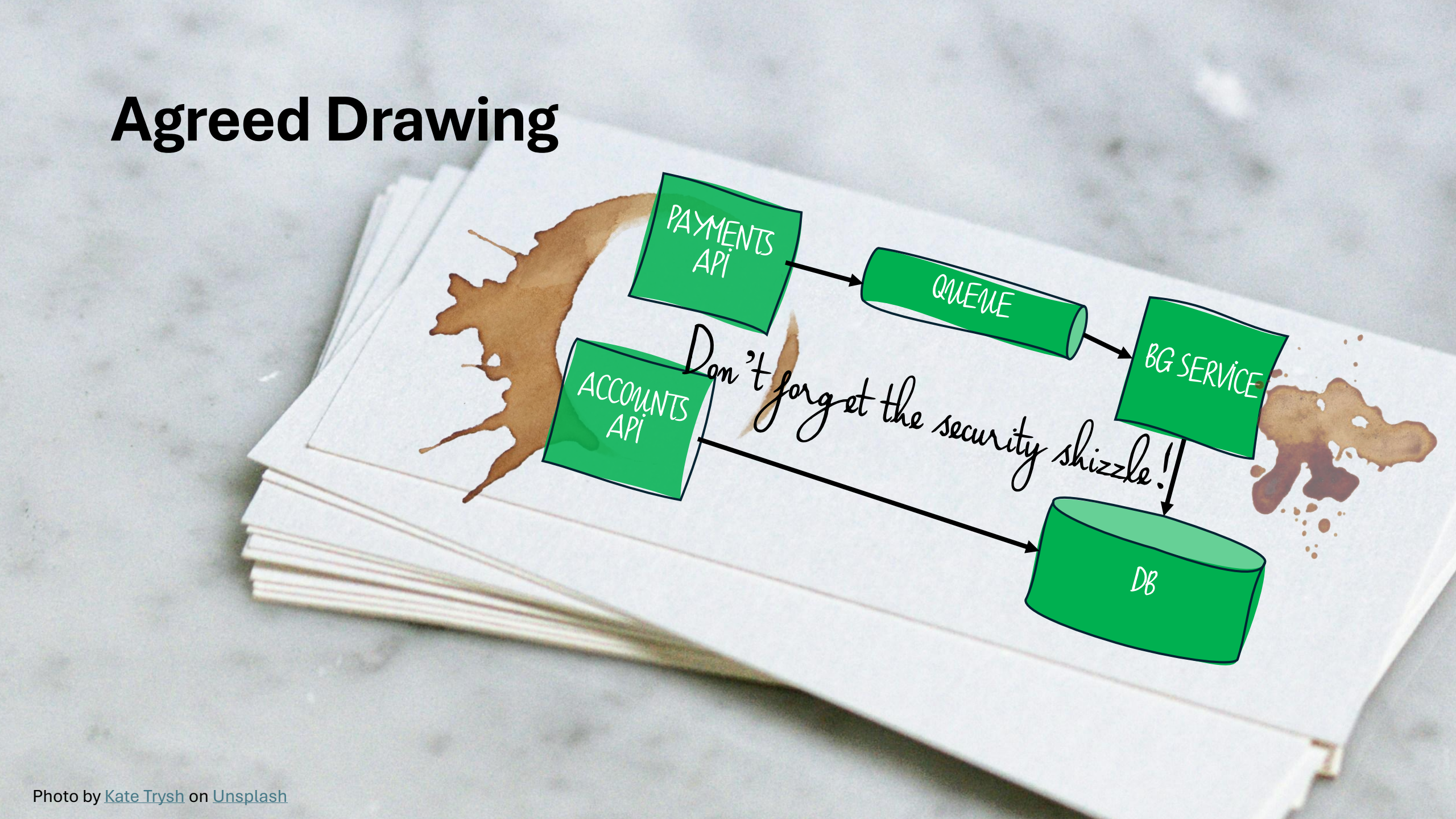


Photo by Kate Trysh on Unsplash

Let’s look at the code for a second

Now let’s make it secure

Never Trust, Always Verify
never trust, always verify

Identity: hook up to Corporate OAuth

Data Classification: Ensure both input and output validation

Anomalies: Set bounds to our input data

Anomalies: Signal strange behavior

Use least privilege access
limit impact and secure data

Just enough access: Only HTTP requests to the endpoints

Just enough access: Restrict the access to the queue and database

Just enough access: Just read specific keys from Vault

Just enough access: Avoid excessive requests

Out of band: Minimize your hosting surface

Assume breach
prevent ‘lateral movement’

Segmenting access: Isolate service in its own segment

Encrypt: Ensure TLS connections everywhere

Avoid: Unwanted outbound access

Avoid: Access to the host

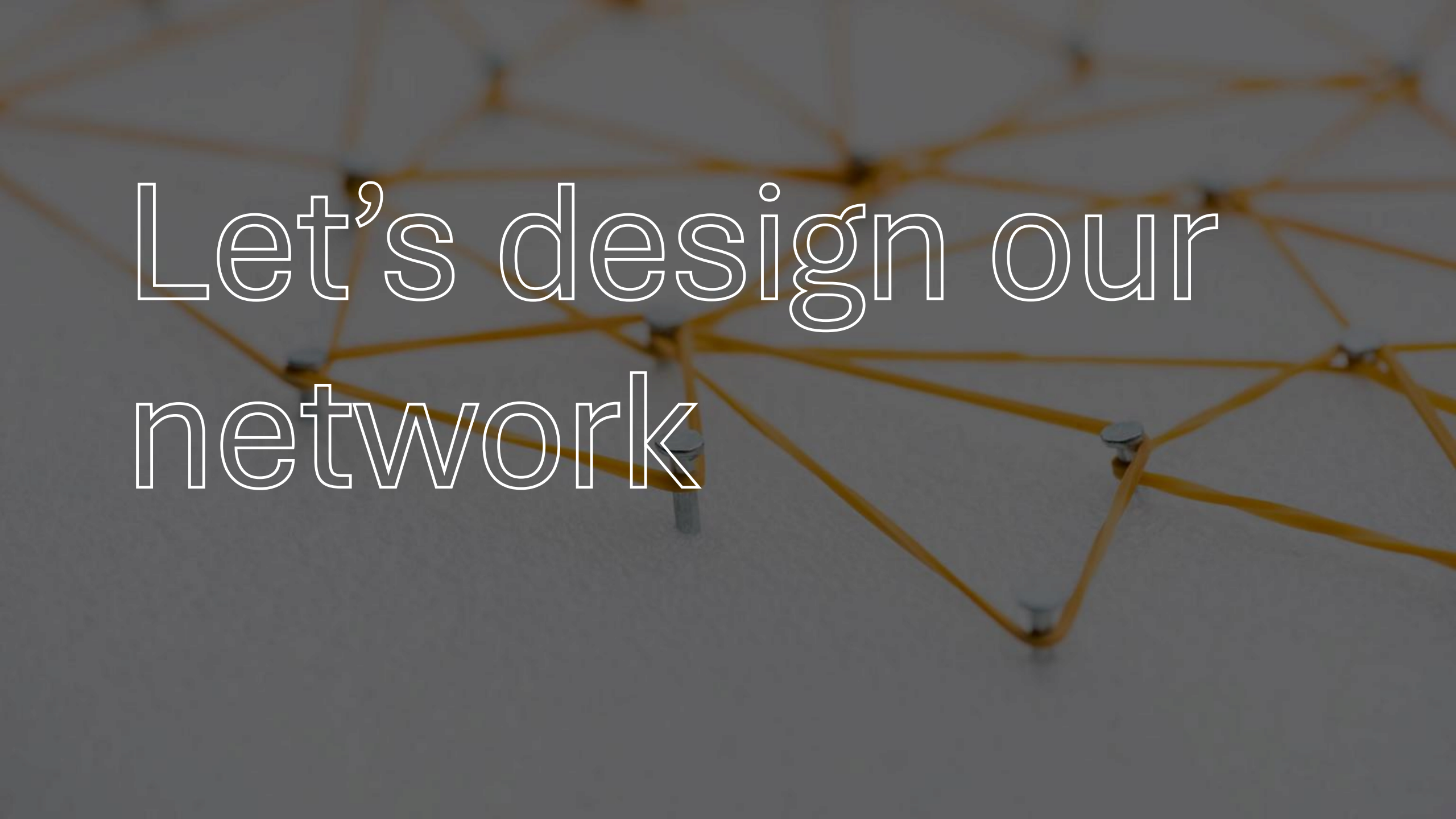
Use Analytics to spot anomalies: Setup threat detection

How big is your container?

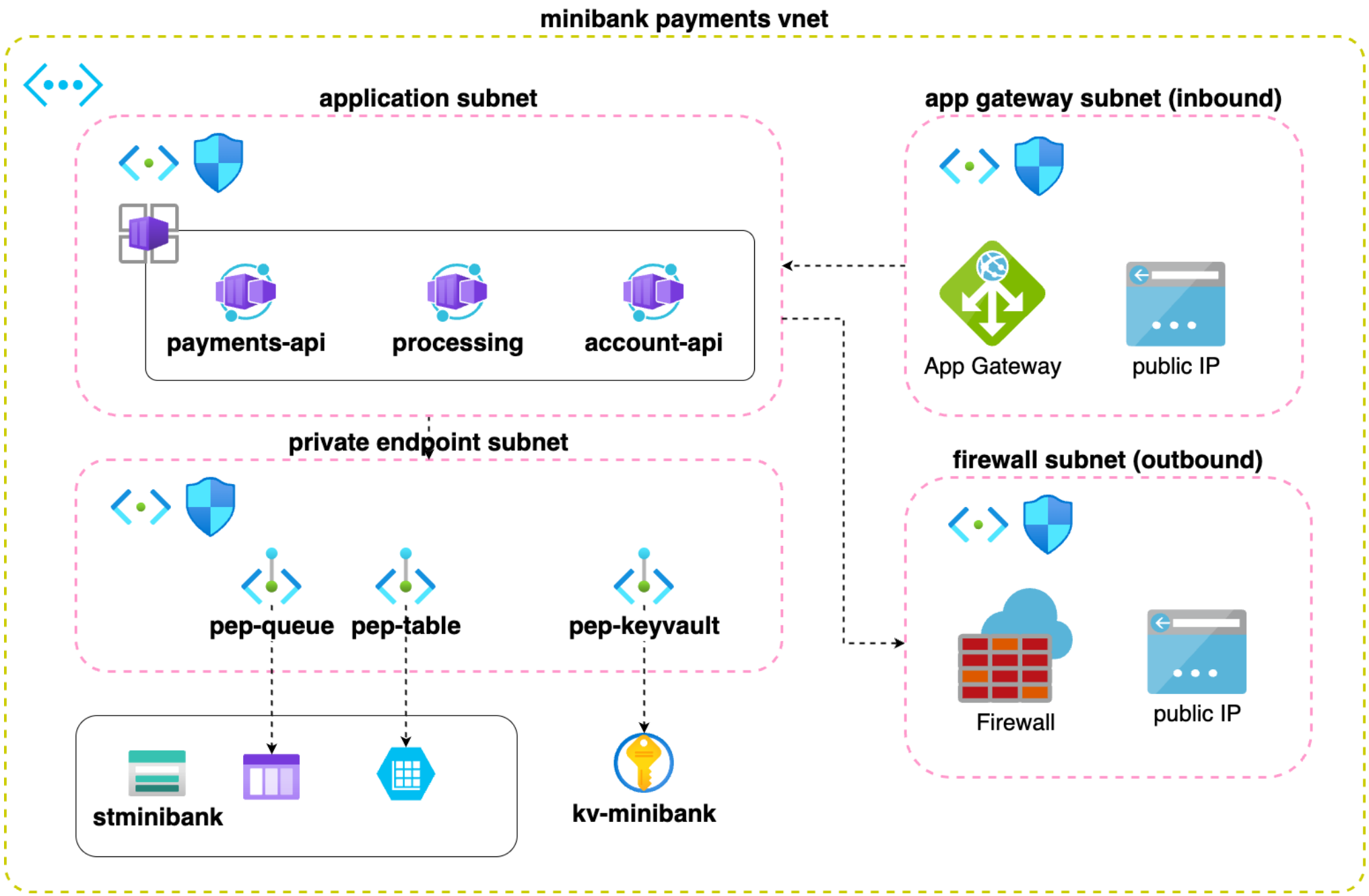
Photo by Teng Yuhong on Unsplash

SBOM

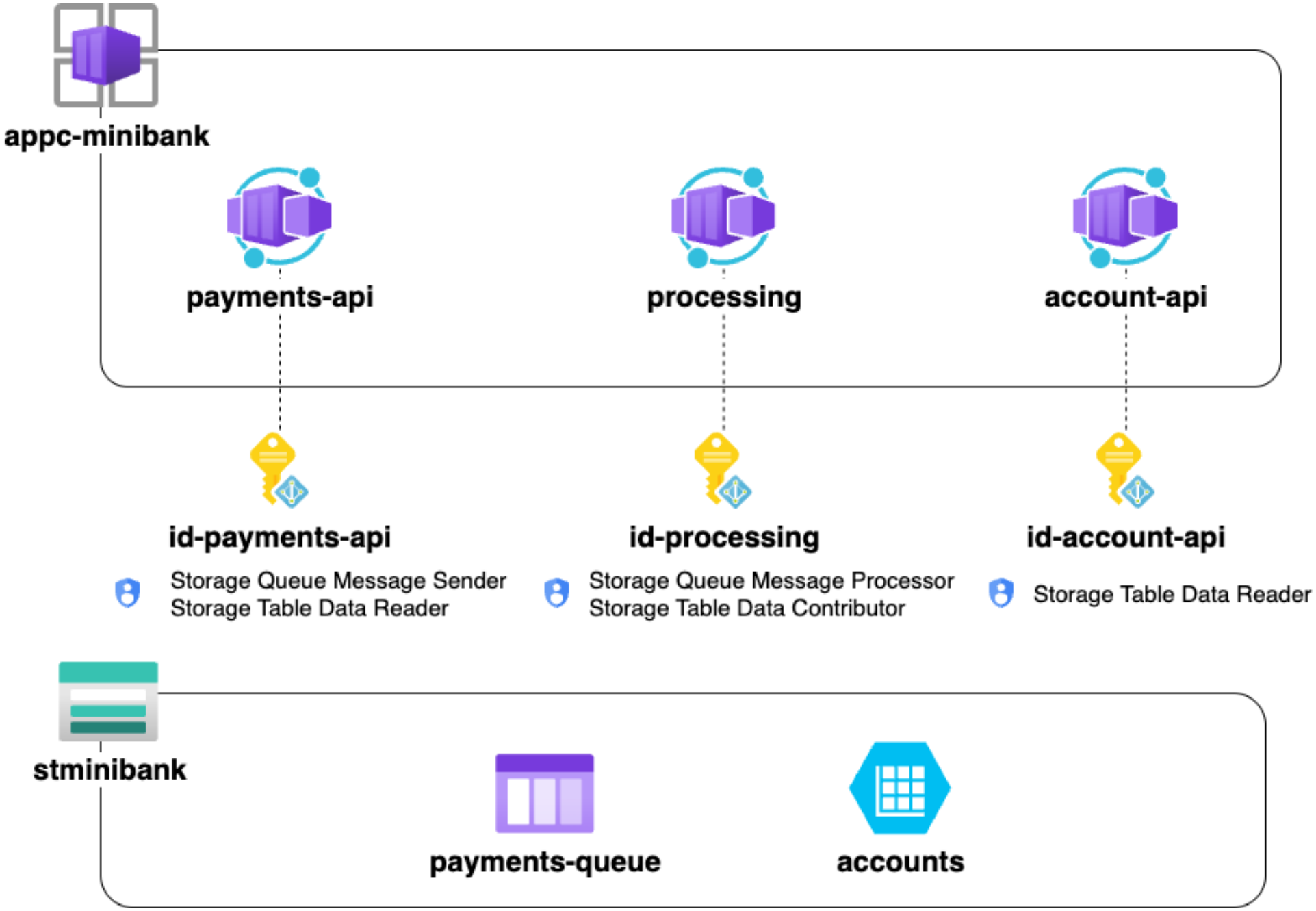
CycloneDX



Let’s design our network


minibank payments vnet
application subnet
payments-api
processing
account-api
app gateway subnet (inbound)
App Gateway
public IP
private endpoint subnet
pep-queue
pep-table
pep-keyvault
firewall subnet (outbound)
Firewall
public IP
stminibank
kv-minibank


IT Ops approves

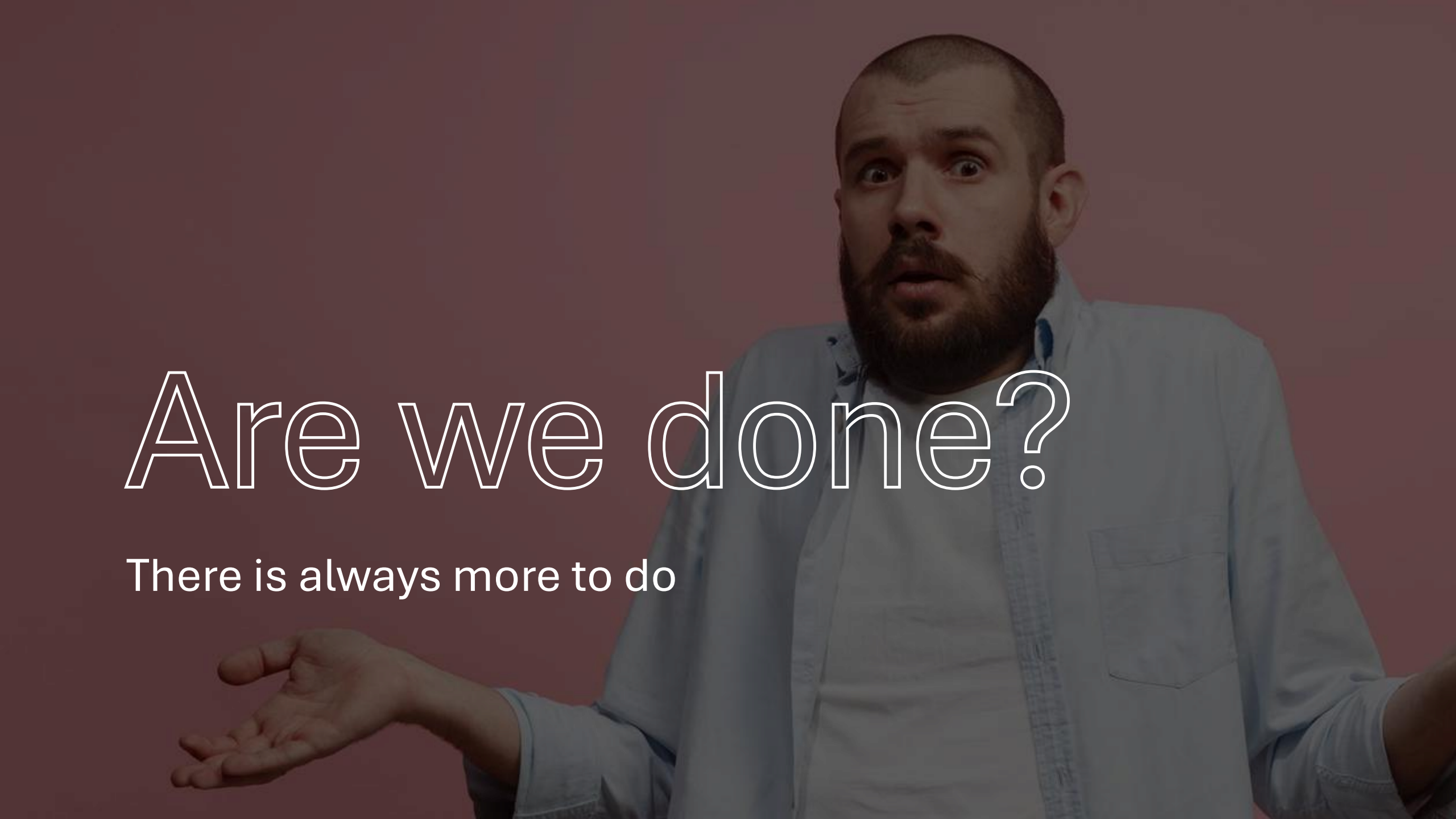

appc-minibank
payments-api
processing
account-api
id-payments-api
Storage Queue Message Sender
Storage Table Data Reader
id-processing
Storage Queue Message Processor
Storage Table Data Contributor
id-account-api
Storage Table Data Reader
stminibank
payments-queue
accounts


How does that
look in the portal?

See it in action

Wrap Up: Zero Trust as a Developer

- **Verify explicitly**: always authenticate and authorize based on all available data points, including user identity, location, device health, service or workload, data classification, and anomalies. Verify both input and output.
- **Use least-privilege access**: limit user access with just-in-time and just-enough-access, risk-based adaptive polices, and data protection to help secure data and improve productivity.
- **Assume breach**: minimize attack surface in containers, verify end-to-end encryption and use analytics to gain visibility, detect threats, and improve defences.


Are we done?
There is always more to do

Thank you!

Roy Cornelissen
roy.cornelissen@xebia.com
@roycornelissen
linkedin.com/in/roycornelissennl

Demo: github.com/roycornelissen/ztdemo

Photo by Morvanic Lee on Unsplash