Application Security (apsi)

Lecture at FHNW

Lecture 10, 2020

Arno Wagner, Michael Schläpfer, Rolf Wagner

<arno@wagner.name>, <{michael.schlaepfer, rolf.wagner}@fort-it.ch>

Agenda

First lesson

- Zero Trust Security Model

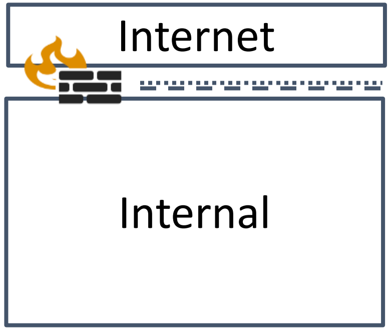
Second lesson

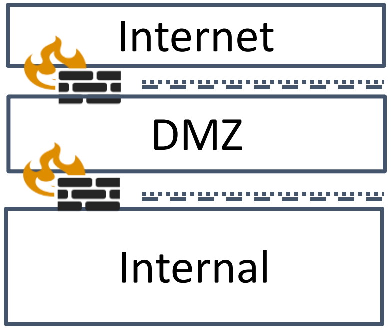
- Secrets Management
- Hyperscaler Cloud Security

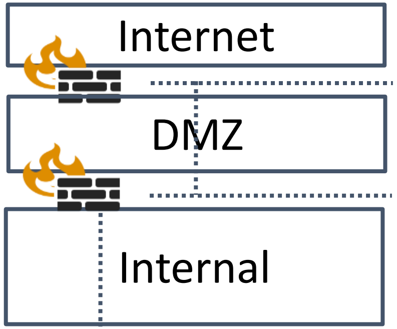
How does a enterprise network look like?

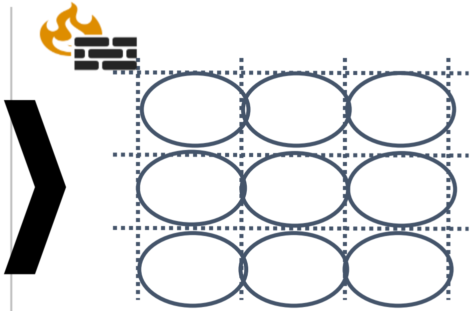
- Network?
- Segmentation?
- Access Control?
- Monitoring?

How does a enterprise network look like?









Internet Firewall

- ● Perimeter protection against Internet ··········
- ● Minimal filtered traffic from inside to outside – – – –
- ▬ Internal zone can be attacked without restriction via the weakest "link

DMZ-Zone, North-South Protection

- ● (Web-) Services which should be available from the Internet are operated in the separate Trust Zone DMZ
- ● Segmentation into trust zones
- ▬ Access from Internal to DMZ often not very restricted
- ▬ Static segmentation

East-West Protection

- ● Static segmentation within a trust zone
- ● Traffic from inside to outside is filtered restrictively
- ▬ Segmentation often only by "type" (all web servers, all clients, ...) and not by data or service criticality
- ▬ Still rather static segmentation

Microsegmentation & Zero Trust

- ● No more perimeter security
- ● Internal and external (Internet) access to applications and services are no longer distinguished
- ● Zero Trust: Never Trust, Alway Verify
- ▬ Challenges regarding complexity and performance

ZT – Why is something «new» needed?

- Traditional
  - Perimeter security with huge defense lines
  - Inside: Good & trusted
  - Outside: Bad & untrusted

Pixabay.com

- Challenges
  - Devices as workloads are more and more on the move
    - Cloud, SaaS
    - Homeoffice
    - BYOD
  - Where is the perimeter?

ZT - Where is it coming from?

- Introduced 2010 by John Kindervag (Forrester Research)
- Never Trust, always verify
  - It’s a data-centric world with shifting threats and perimeters
  - Network zones are not trustworthy simply because they are within an enterprise perimeter
- Several adoptions in last few years
  - Forrester ZTX - Zero Trust eXtended
  - Gartner CARTA - Continuous Adaptive Risk and Trust Assessment
  - Google BeyondCorp
  - Microsoft Zero Trust

ZT – Key Principles

- Never Trust, Always Verify!
- **Never Trust**
  - Assume breach
  - No static defense of traditional network perimeter
  - No trust in networks
  - No trust in interfaces
  - No trust in users
  - No trust …

ZT – Key Principles

- Never Trust, Always Verify!
- **Always verify**
  - Authenticate, authoricate and encrypt EACH individual access request to a workload
  - Least privilege (-> segmentation)
  - Adaptive access control based on available context-information (subject, client, target, …)

ZT – Evolution Example

- Login

1. Username is not enough
   -> Password and nowadays 2nd factor needed
2. Enterprise perimeter location is not enough
   -> context-based access management needed
3. Login once is not enough
   -> Continuous monitoring and (re-) authentication based on the current risk level

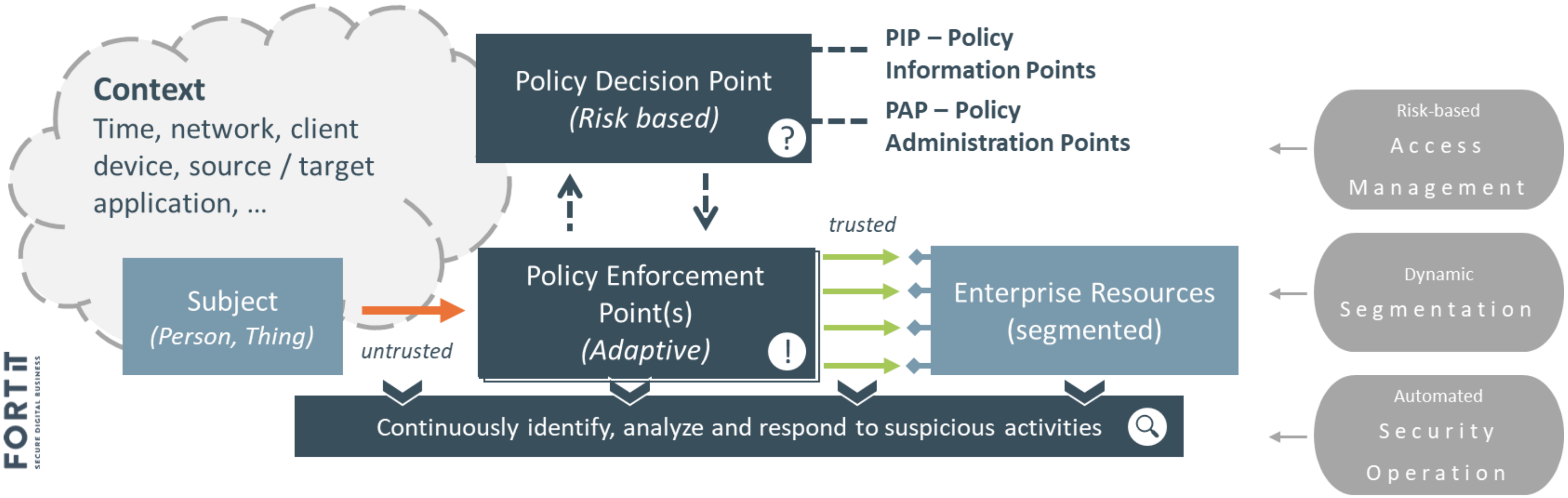
- Information Points for access decision
  - Who, What, When, Where, Why, How, …
  - Layer 7 information needed; Lower layer control useful (software defined X)

ZT – Where to start?

- Access Management
  - Adaptive authentication & authorization based on risk levels determined by context information
- Segmentation
  - Segments getting smaller and closer to the workload
  - Dynamic micro-segmentation based on policies (workload, data)
  - Establish authenticate-before-access principles
- Security Operation
  - Identify, analyze and respond to suspicious activities and incidents
  - Provide context information for access decisions

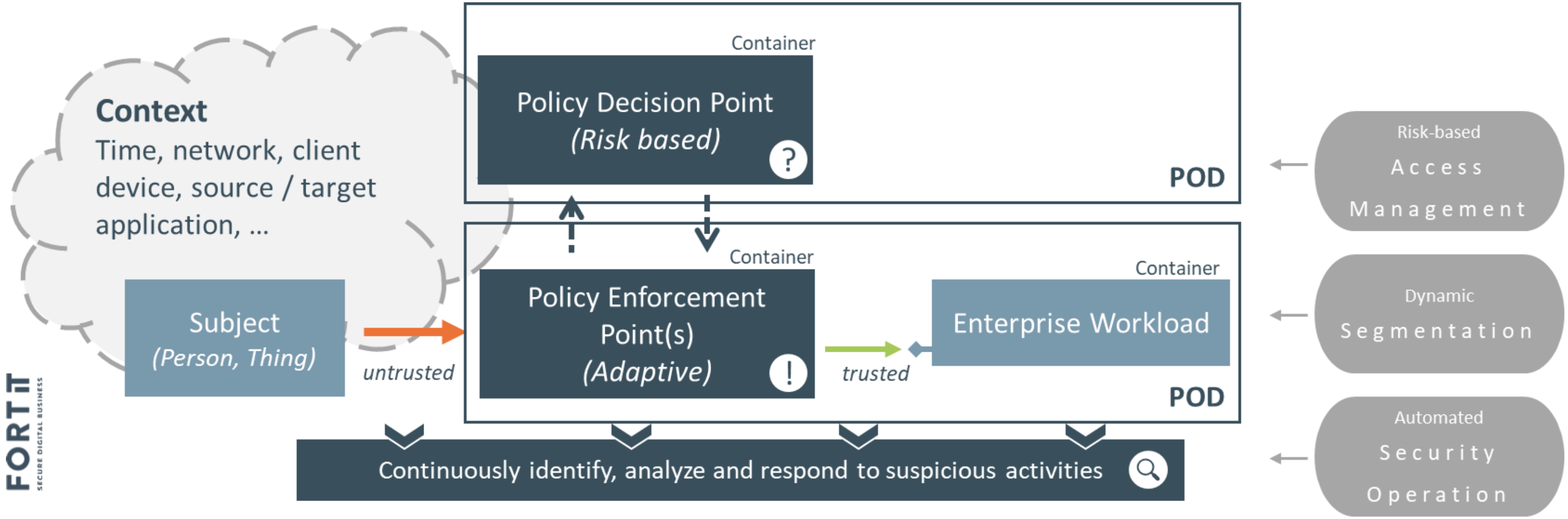
ZT – Architecture

- Based on the (old) XACML architecture
- XACML Wikipedia: “The standard defines a declarative fine-grained, attribute-based access control policy language, an architecture, and a processing model describing how to evaluate access requests according to the rules defined in policies.”



ZT – Architecture

- Zero Trust Access Management with Kubernetes / Containers



Agenda

First lesson

- Zero Trust Security Model

Second lesson

- Secrets Management
- Hyperscaler Cloud Security

What is Secrets Management?

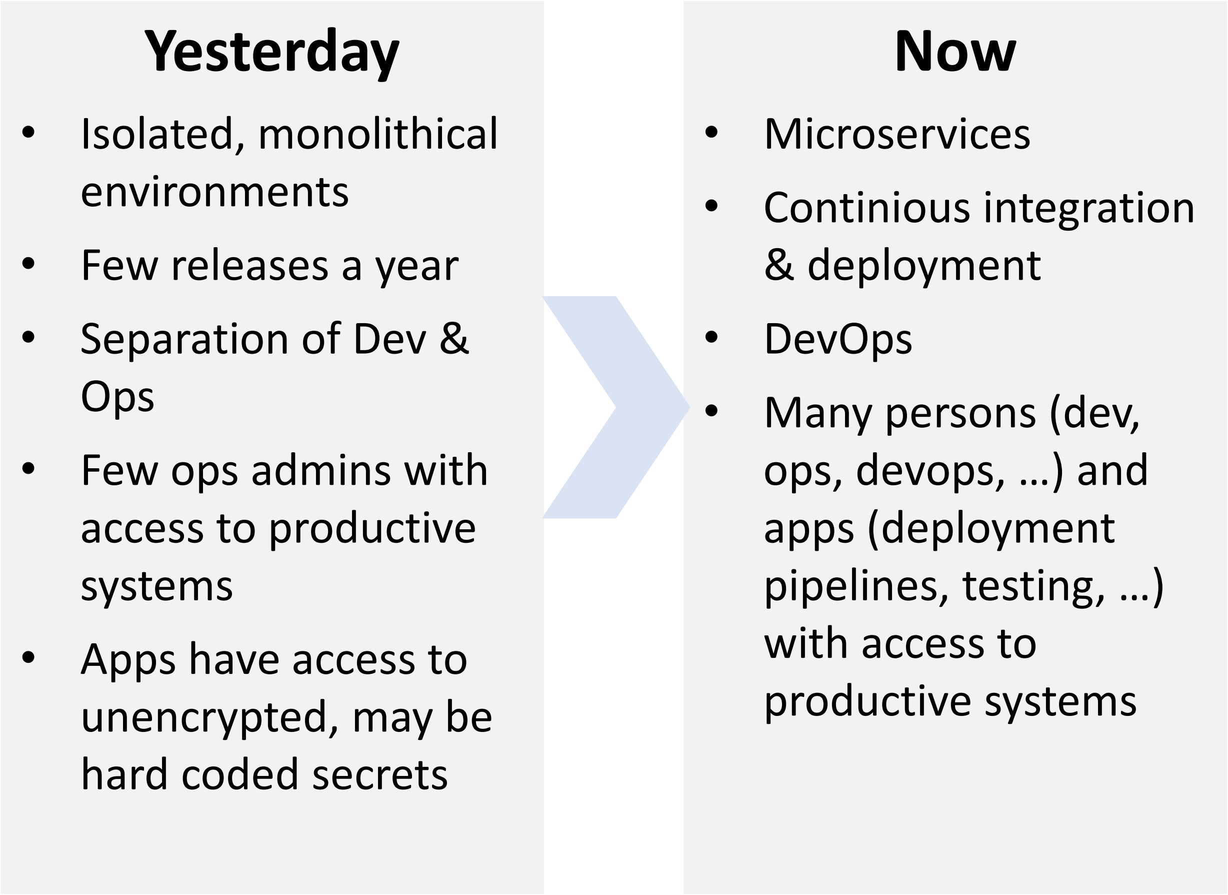
- Where do you need / use secrets?

- Where are they stored?

What is Secrets Management?

- Secrets Management
  - Managing digital secrets, including passwords, APIs keys and tokens
  - Used in services, applications, privileged accounts
- What impact has e.g. DevOps on secrets management?
- What impact has e.g. Zero Trust on secrets management?

The world changed



Yesterday

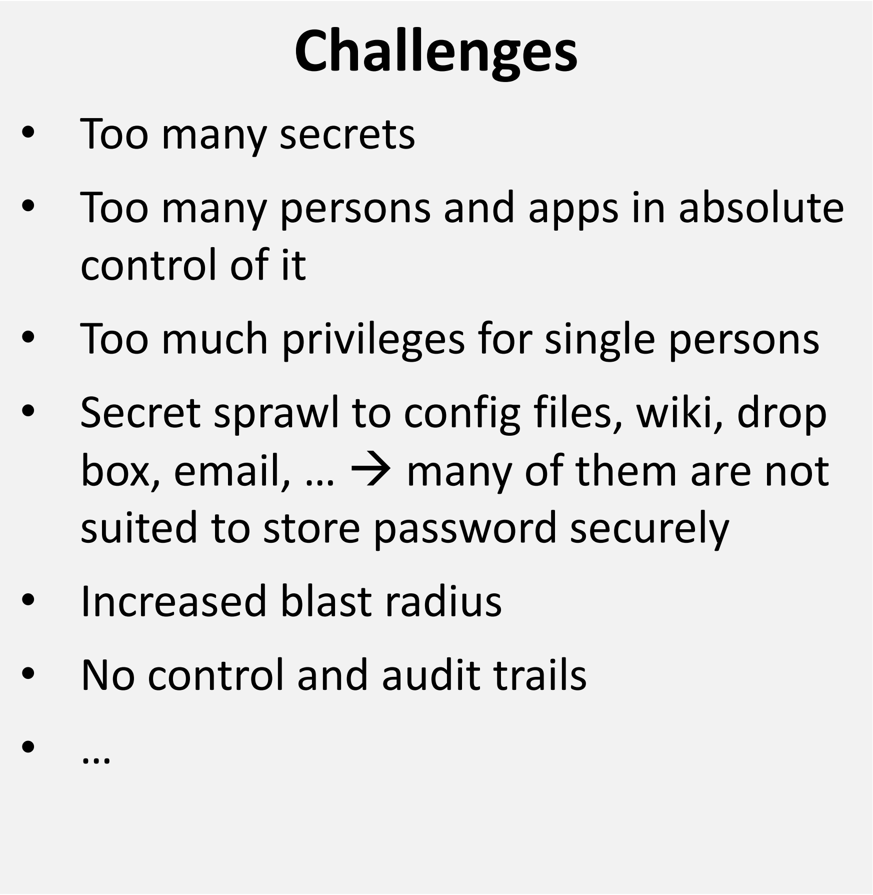
- Isolated, monolithical environments
- Few releases a year
- Separation of Dev & Ops
- Few ops admins with access to productive systems
- Apps have access to unencrypted, may be hard coded secrets

Now

- Microservices
- Continious integration & deployment
- DevOps
- Many persons (dev, ops, devops, …) and apps (deployment pipelines, testing, …) with access to productive systems

Challenges?

Challenges



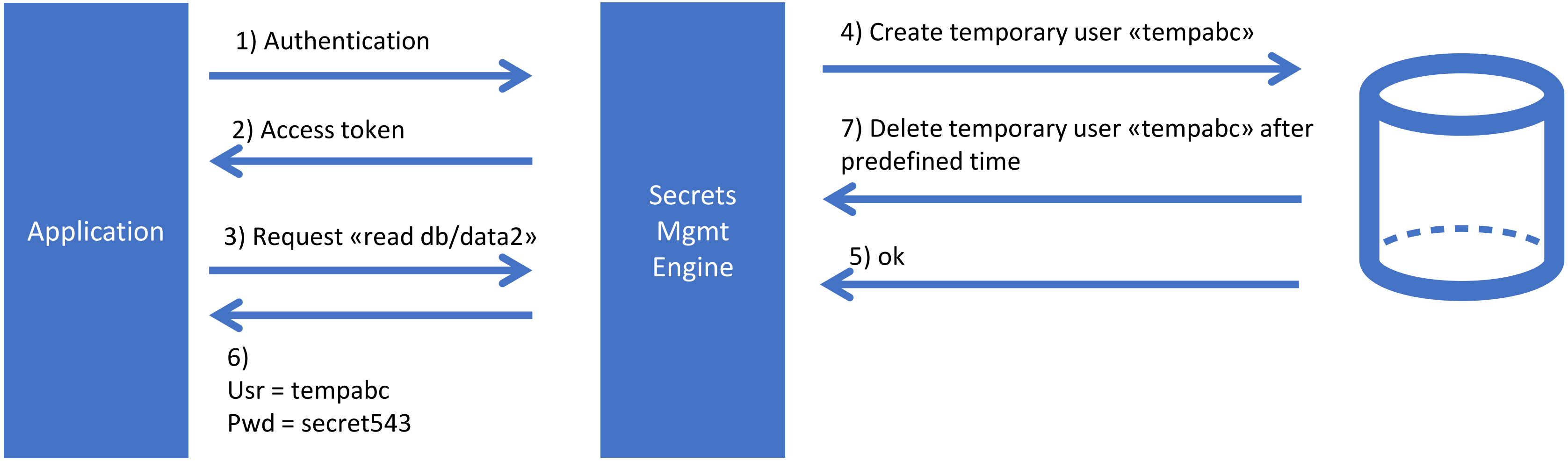
Challenges

- Too many secrets
- Too many persons and apps in absolute control of it
- Too much privileges for single persons
- Secret sprawl to config files, wiki, drop box, email, … → many of them are not suited to store password securely
- Increased blast radius
- No control and audit trails
- …

Action fields & goals

- Management
  - Secrets management is automated in a central service (e.g. interfaces to tool chain)
- Usage
  - Secrets are generated dynamically and have a limited validity
  - All secrets are encrypted (at rest & in transit)
- Monitoring
  - Every person & app only gets access to the secrets they need (least privilege)
  - Usage-monitoring - every usage of secrets is monitored
  - Security breaches (secrets) can be isolated and traced back

Authentication example



Application

1) Authentication

2) Access token

3) Request «read db/data2»

6)
Usr = tempabc
Pwd = secret543

Secrets
Mgmt
Engine

4) Create temporary user «tempabc»

7) Delete temporary user «tempabc» after predefined time

5) ok

Based on Vault by HashiCorp

Global Hyperscaler Cloud Security

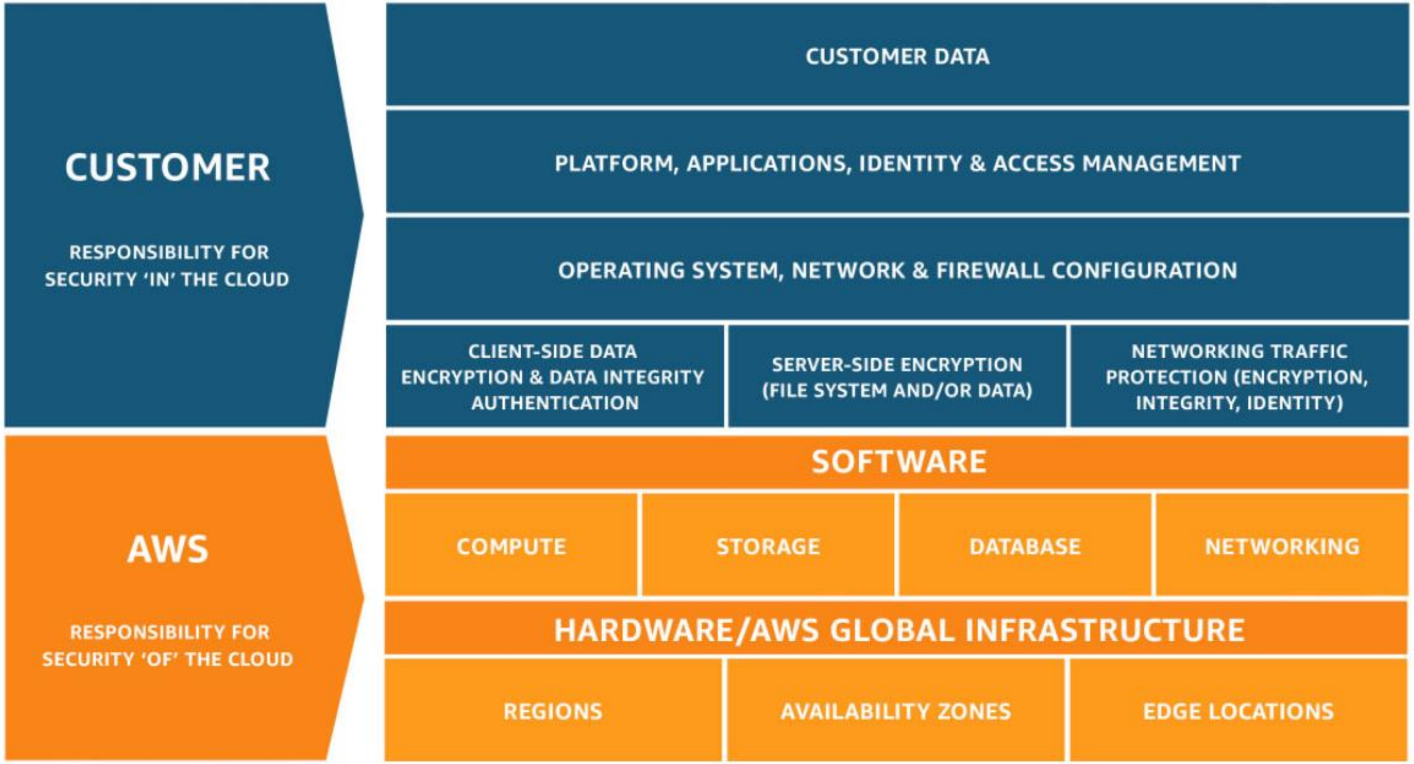
- What is Cloud Security?

- Who is responsible for what? Is it black / white?

- Is it easier? How could the comparison with development / security framework help?

Global Hyperscaler Cloud Security

- **Cloud Provider**: protecting the infrastructure as composed of the hardware, software, networking, and facilities
- **Customer**: The rest - including secure configuration of all used services


CUSTOMER
RESPONSIBILITY FOR SECURITY 'IN' THE CLOUD
AWS
RESPONSIBILITY FOR SECURITY 'OF' THE CLOUD
CUSTOMER DATA
PLATFORM, APPLICATIONS, IDENTITY & ACCESS MANAGEMENT
OPERATING SYSTEM, NETWORK & FIREWALL CONFIGURATION
CLIENT-SIDE DATA ENCRYPTION & DATA INTEGRITY AUTHENTICATION
SERVER-SIDE ENCRYPTION (FILE SYSTEM AND/OR DATA)
NETWORKING TRAFFIC PROTECTION (ENCRYPTION, INTEGRITY, IDENTITY)
SOFTWARE
COMPUTE
STORAGE
DATABASE
NETWORKING
HARDWARE/AWS GLOBAL INFRASTRUCTURE
REGIONS
AVAILABILITY ZONES
EDGE LOCATIONS

Shared Responsibility Model

Source: aws.amazon.com

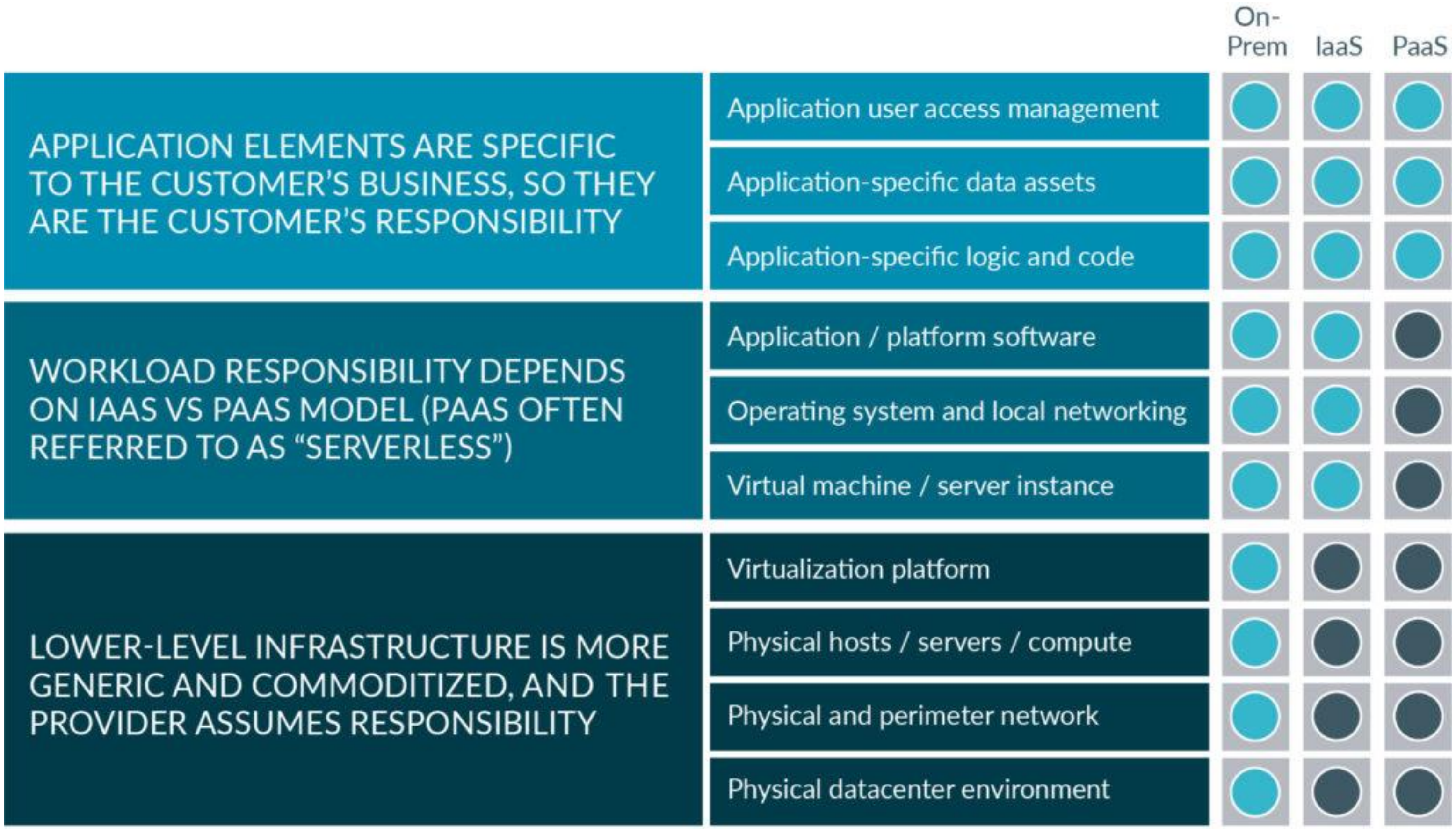
Who is responsible for the customer-configured DMZ network zone for application X (IaaS)?

«Common» Cloud Security Services

- **Physical Security**: Protecting physical assets at a geographic location
- **Infrastructure security**: Segmentation, hardening, security patches etc.
- **Data and Access Security**: Authentication, Authorization, Encryption etc.

- Cloud providers have little control over the third aspect, data and access security
- Application-level security is typically the customers' responsibility
- Most of the breaches occur because this third part is not very well secured

On-Premises, IaaS, PaaS



		On-Prem	IaaS	PaaS
APPLICATION ELEMENTS ARE SPECIFIC TO THE CUSTOMER'S BUSINESS, SO THEY ARE THE CUSTOMER'S RESPONSIBILITY	Application user access management	○	○	○
	Application-specific data assets	○	○	○
	Application-specific logic and code	○	○	○
WORKLOAD RESPONSIBILITY DEPENDS ON IAAS VS PAAS MODEL (PAAS OFTEN REFERRED TO AS "SERVERLESS")	Application / platform software	○	○	●
	Operating system and local networking	○	○	●
	Virtual machine / server instance	○	○	●
LOWER-LEVEL INFRASTRUCTURE IS MORE GENERIC AND COMMODITIZED, AND THE PROVIDER ASSUMES RESPONSIBILITY	Virtualization platform	○	●	●
	Physical hosts / servers / compute	○	●	●
	Physical and perimeter network	○	●	●
	Physical datacenter environment	○	●	●

○ Customer ● Provider

Source: cloudsecurityalliance.org

«Common» Cloud Security Services

- Monitoring & Logging
- Identity & Access Management
- Compliance Detector & Manager
- …

z.B.

AWS Shield	DDoS Protection
Amazon GuardDuty	Threat detection
AWS Security Hub	Security alerts and security situation

- **It’s complex…**
- You must understand and configure it right!
- Otherwise it will cost you much
  - In sense of operation costs
  - In sense of security incidents