Modern Service Networking

Securing services with a Service Mesh.

Modern Service Networking

Erik Veld

Developer Advocate at HashiCorp

HashiCorp

Terraform

Consul

Packer

Vault

Nomad

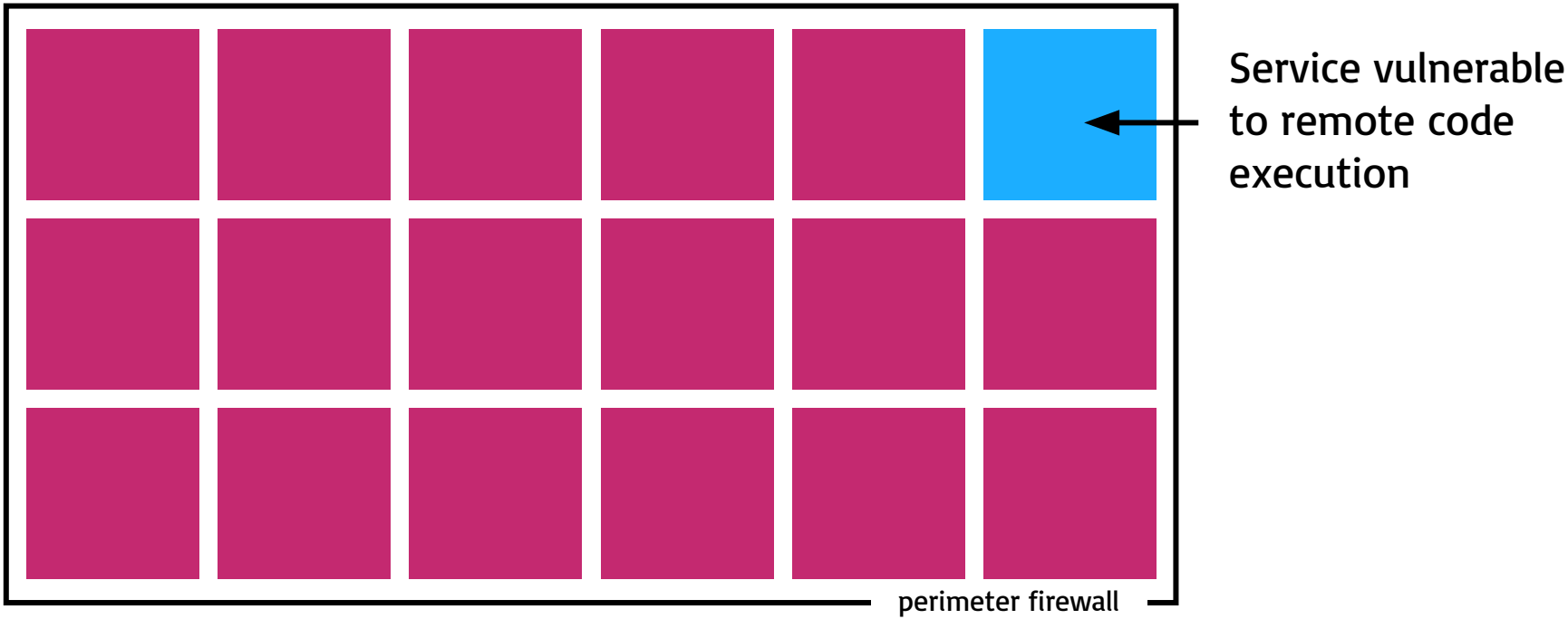
Vagrant

HashiCorp

Why?

Remote Code Execution.

Easy to bypass the perimeter by attacking code.

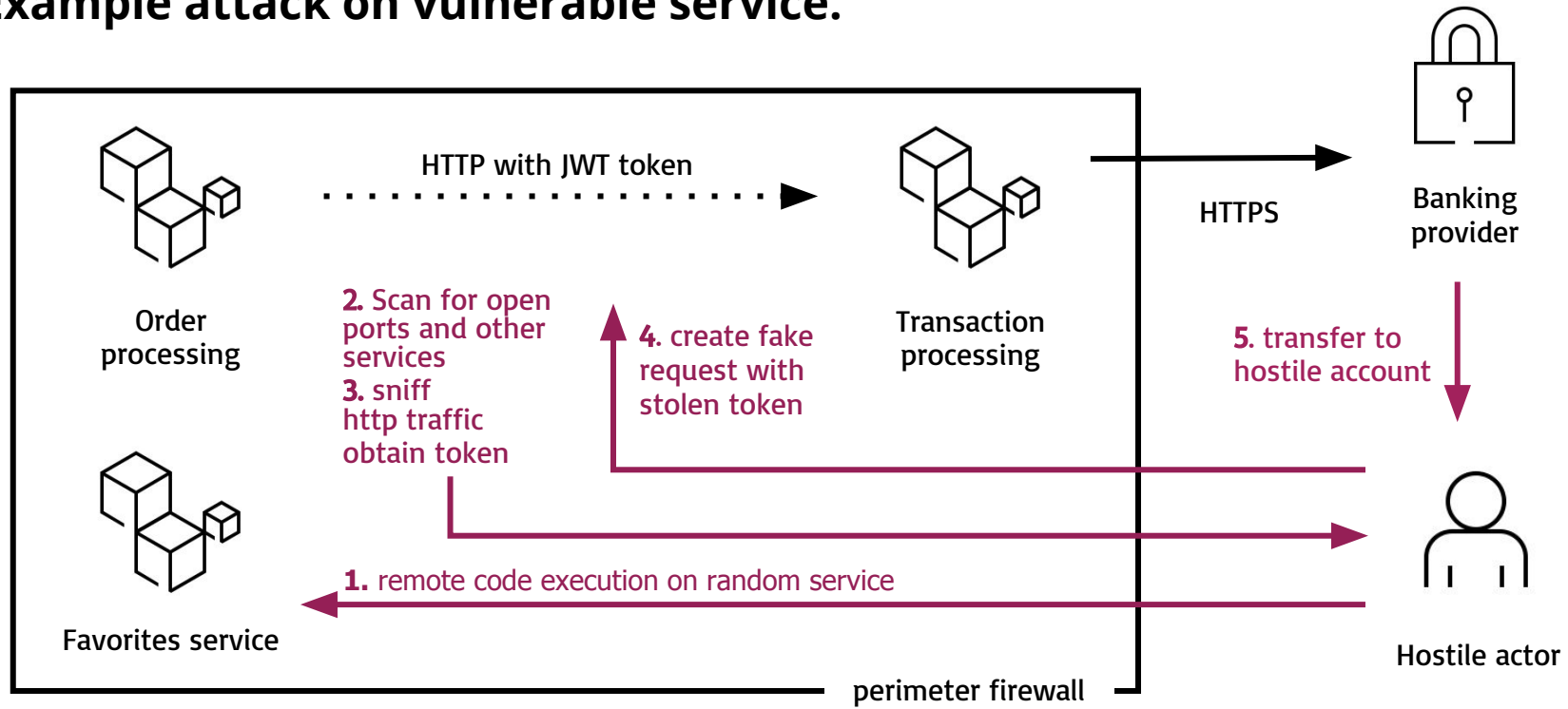


“One of the key phases in most targeted attacks is what’s known as lateral movement. Attackers rarely luck out and manage to immediately compromise the computer”

Symantec Internet Threat Report 2018

Lateral Movement.

Example attack on vulnerable service.



Lateral Movement.

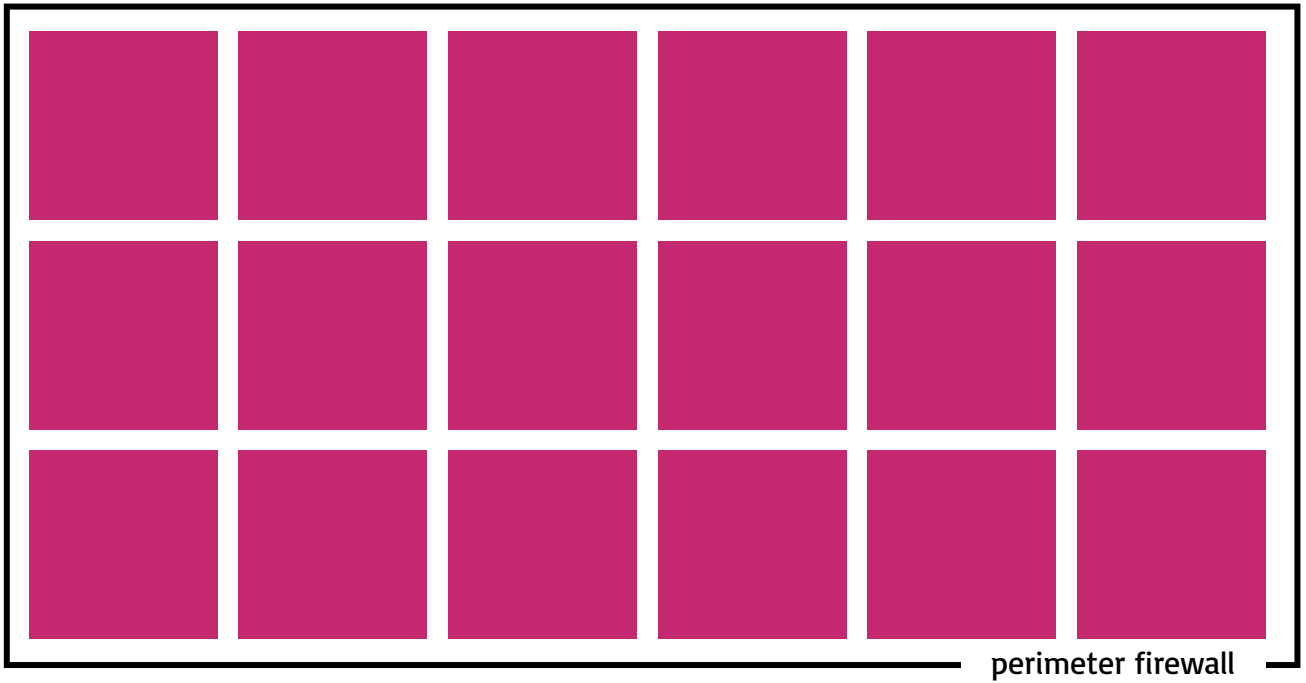
Two factors which allowed lateral movement.

- Open network access
- Traffic between services is not encrypted

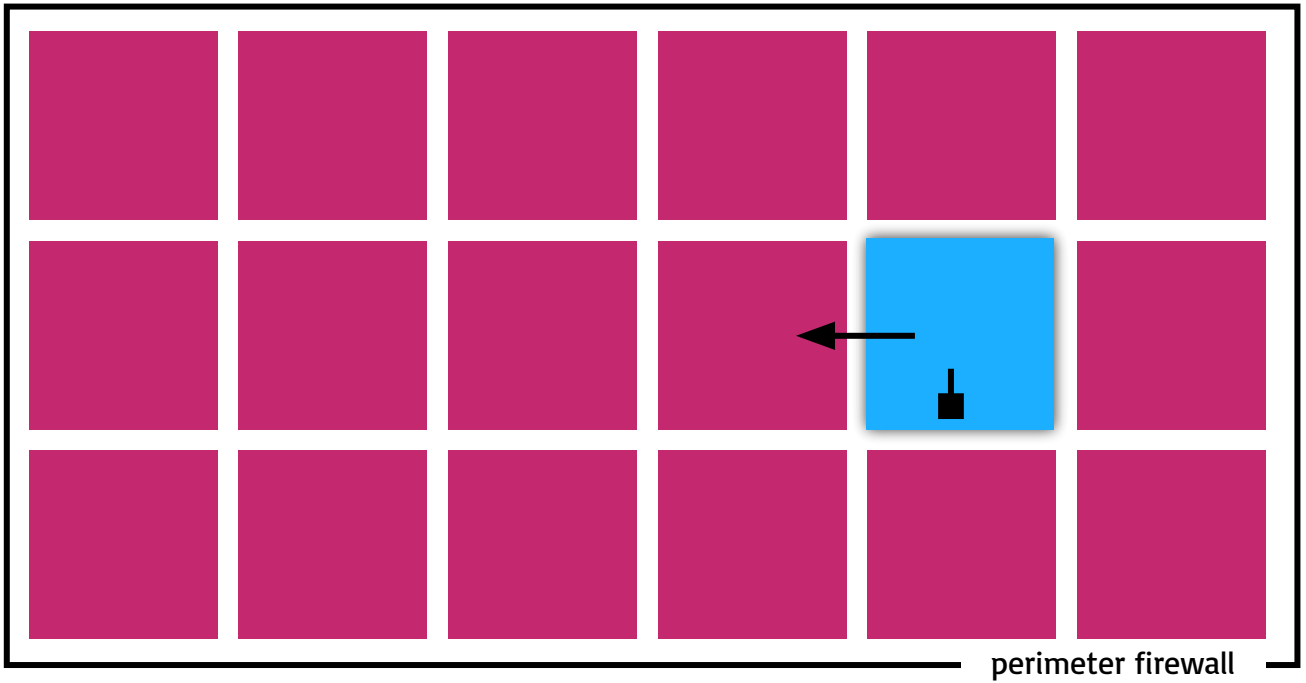
Problem 1:

Open network access.

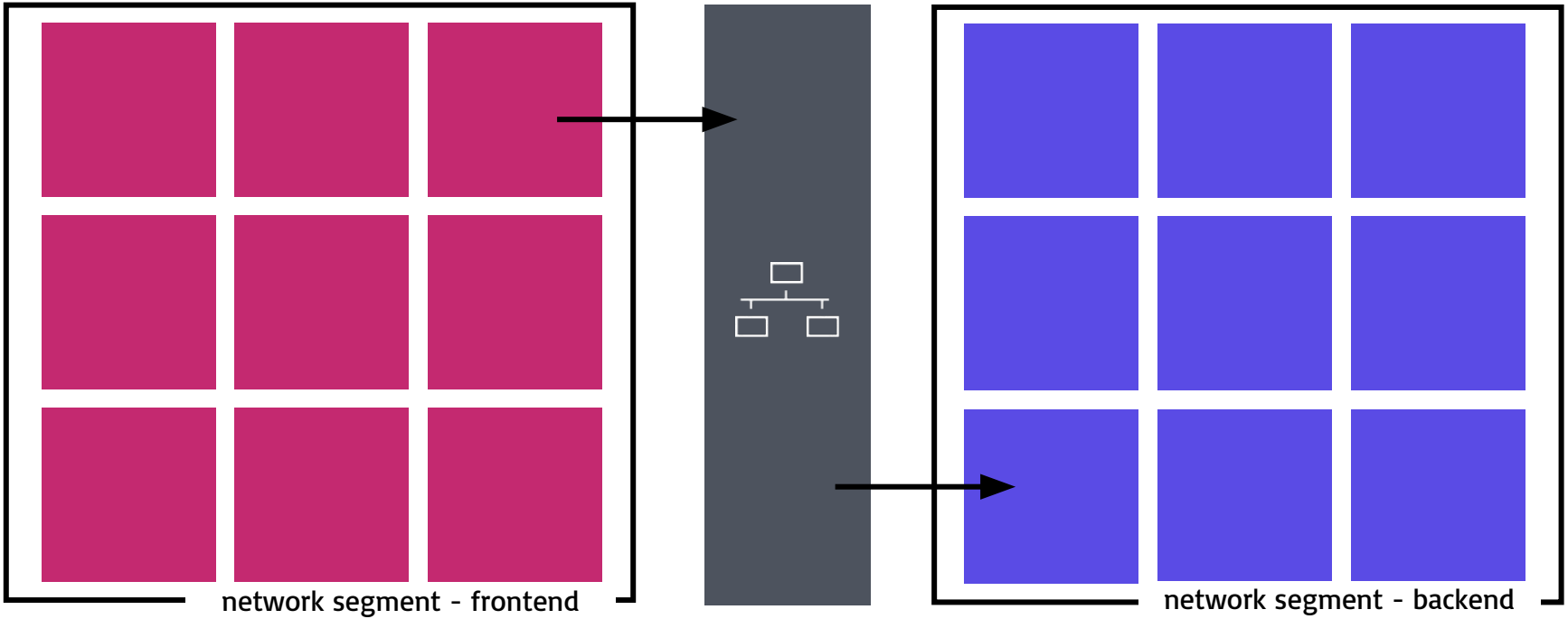
Traditional approach to security was a perimeter firewall.



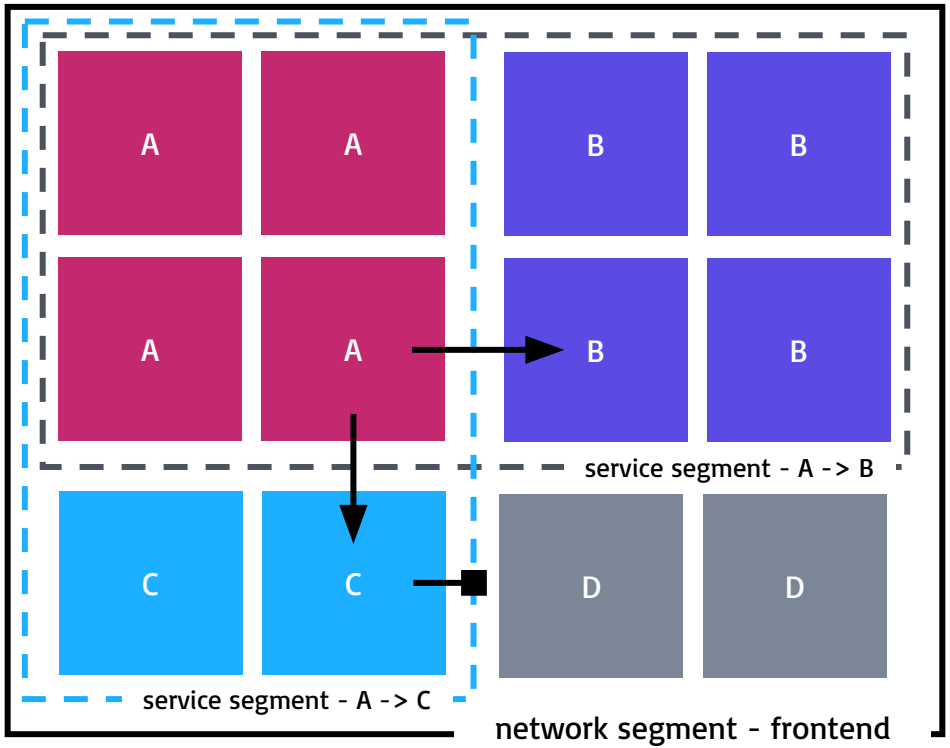
We need internal network isolation.



Network segmentation.

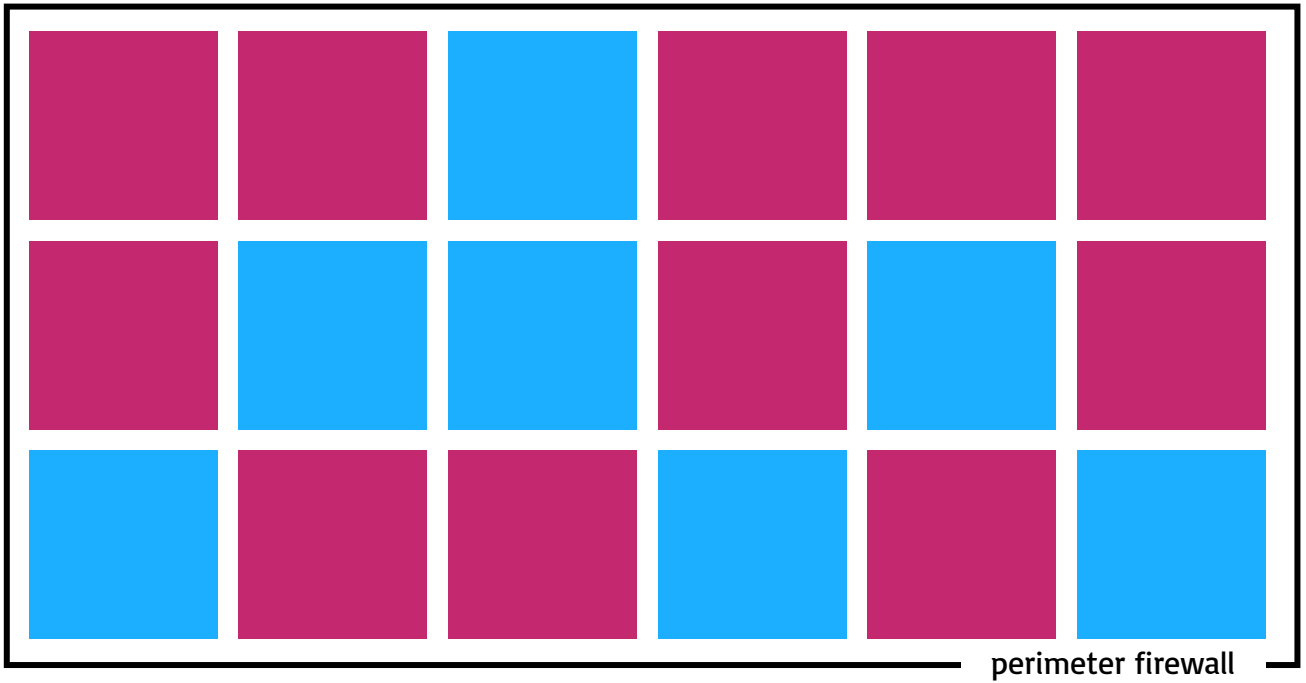


Service segmentation.



Problem.

Dynamic environments result in constantly changing ip addresses and ingress ports.



Definition.

What is a dynamic environment?

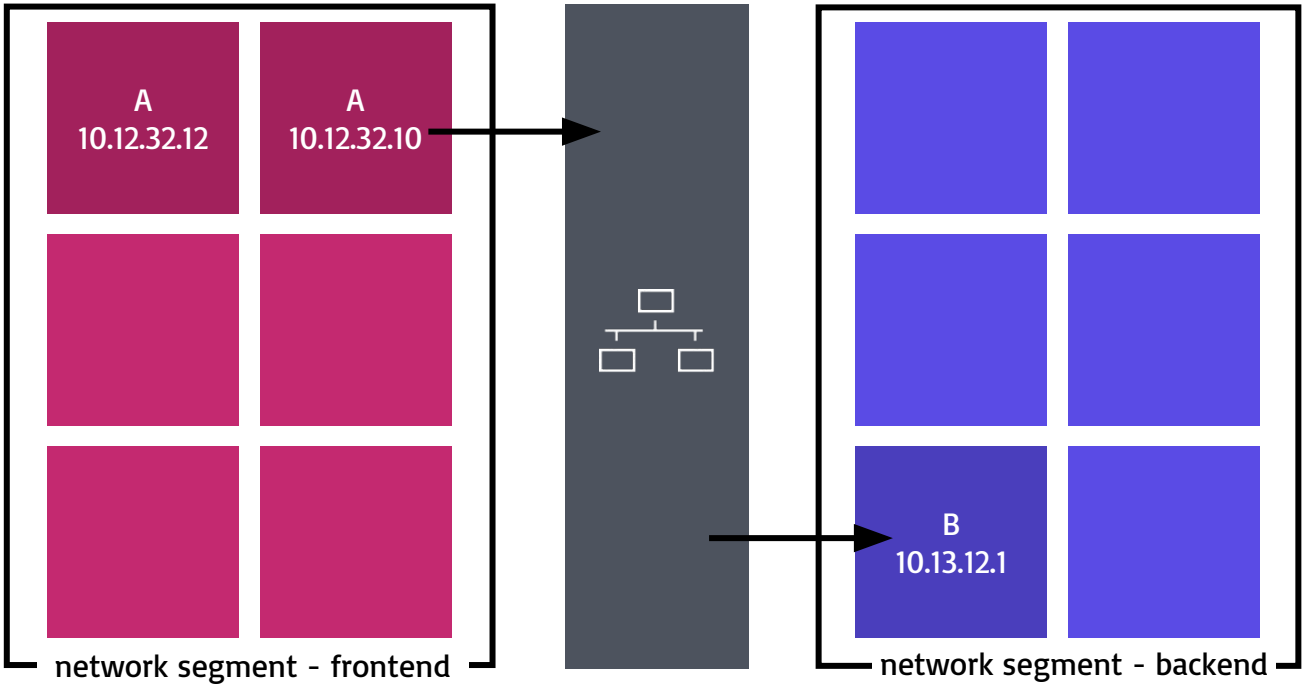
- Applications and infrastructure subject to frequent changes
- Subject to auto scaling or automated instance replacement
- Applications running in a scheduler like Kubernetes

Problem.

Problems with network and service segmentation in dynamic environments.

- Application deployment is disconnected from the network configuration
- Applications are scheduled in a modern scheduler

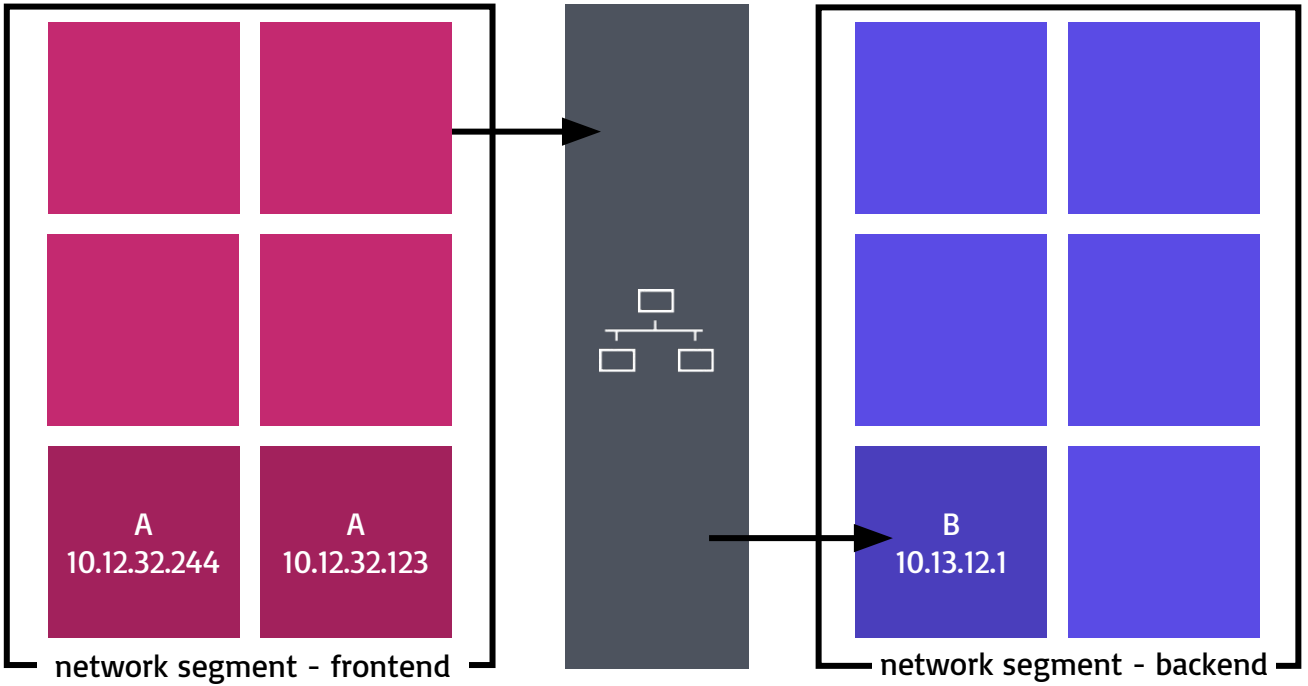
Application deployment is disconnected from the network configuration.



routing rules:
10.12.32.10 -> 10.13.12.1
10.12.32.12 -> 10.13.12.1

firewall:
B allow port 9912

Application deployment is disconnected from the network configuration.

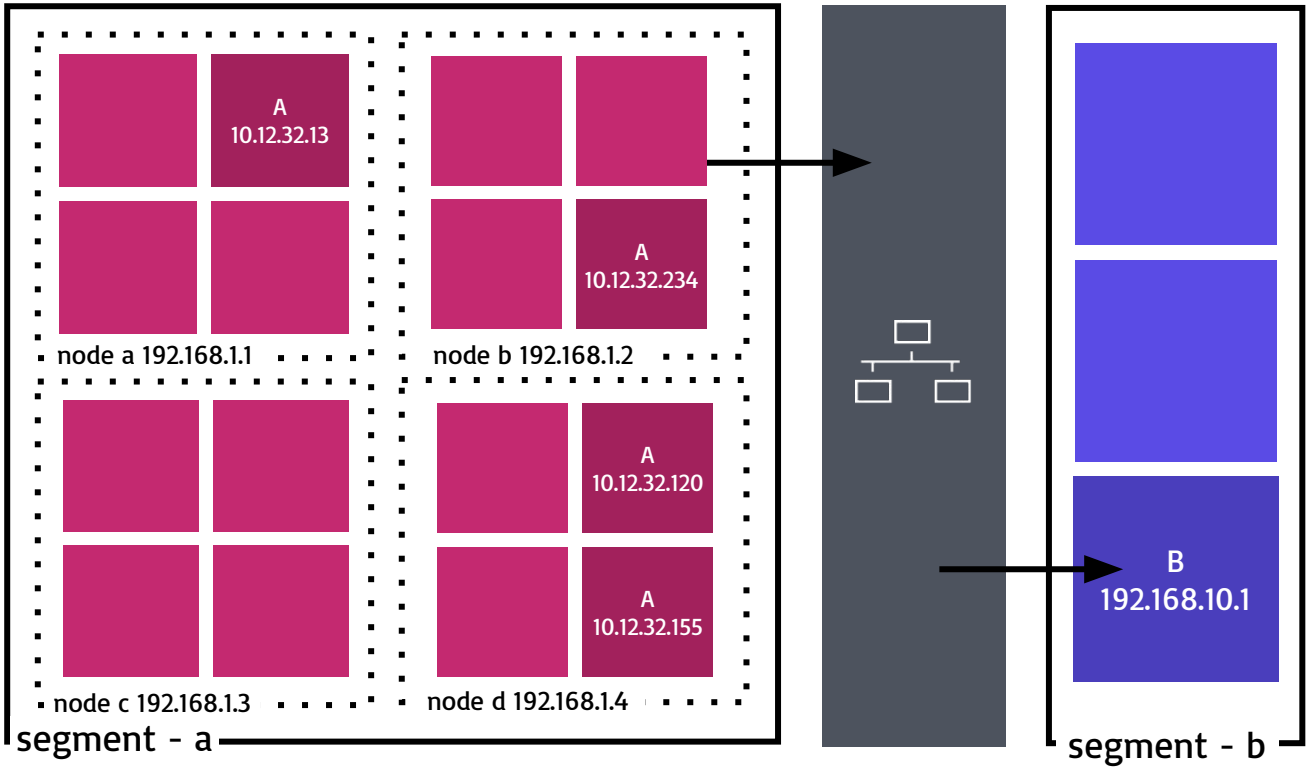


routing rules:
10.12.32.244 -> 10.13.12.1
10.12.32.123 -> 10.13.12.1

firewall:
B allow port 9912

Applications is redeployed routing rules must also change

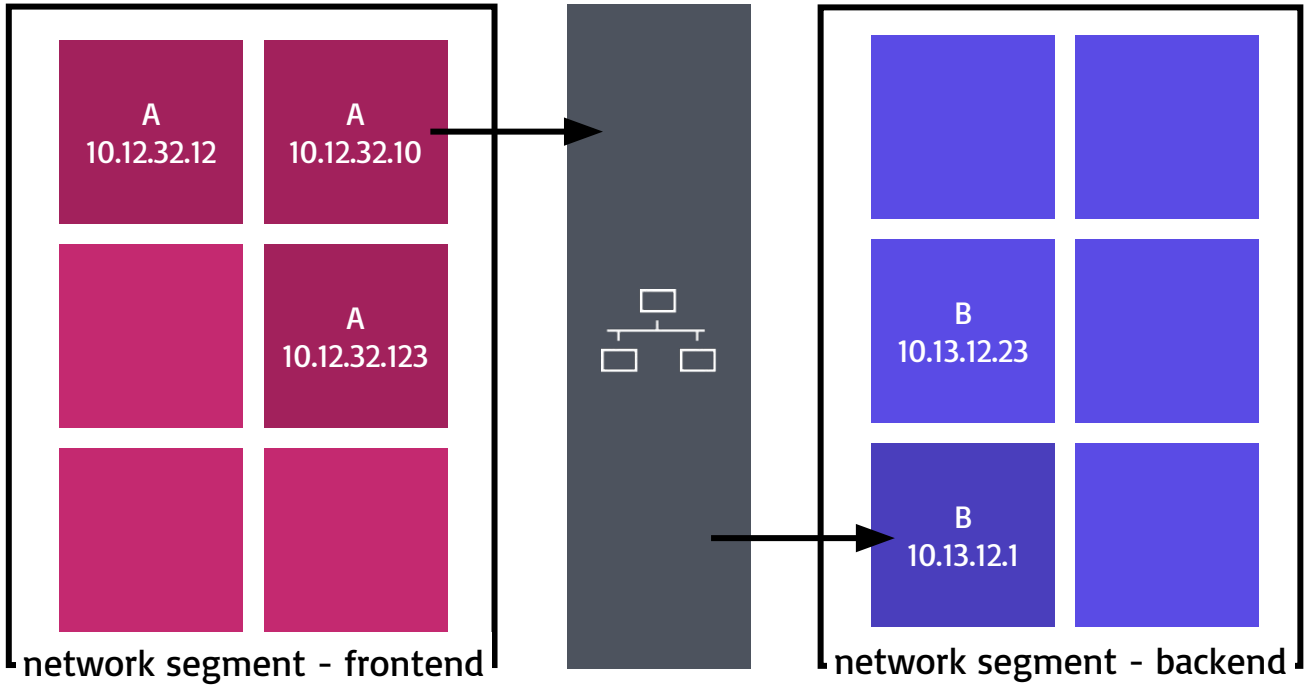
Applications are scheduled in a modern scheduler.



desired rules:
192.168.1.1 -> 192.168.10.1
192.168.1.2 -> 192.168.10.1
192.168.1.4 -> 192.168.10.1

reality:
192.168.10.0/24 ->
192.168.10.1

Network / Service segmentation with intention based security.



routing rules:

~~10.12.32.10 -> 10.13.12.1~~
~~10.12.32.12 -> 10.13.12.1~~
~~10.12.32.12 -> 10.13.12.1~~
~~10.12.32.10 -> 10.13.12.23~~
~~10.12.32.12 -> 10.13.12.23~~
~~10.12.32.12 -> 10.13.12.23~~

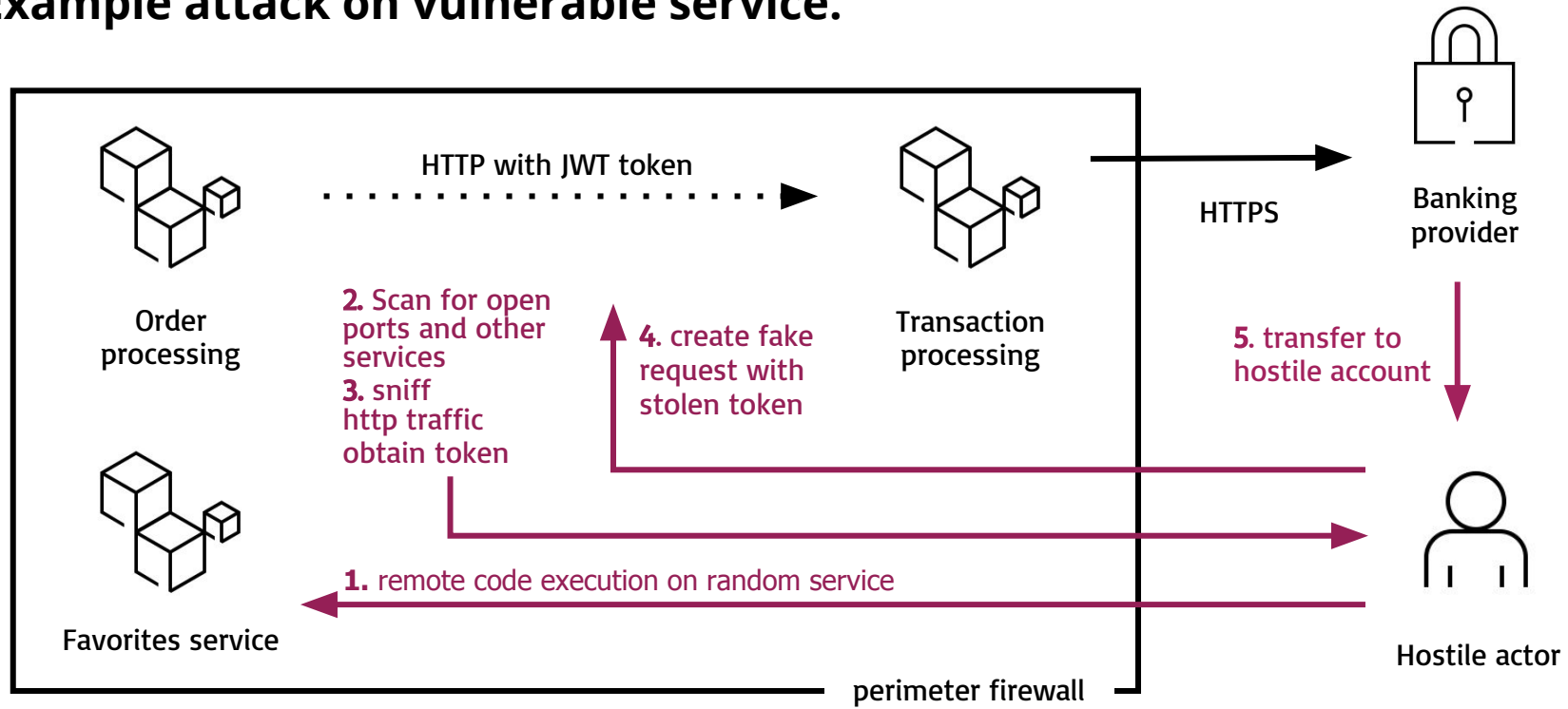
intentions:
service a -> service b

Problem 2:

Traffic between services is not encrypted.

Lateral Movement.

Example attack on vulnerable service.



"it takes on average 180 days, to detect a breach"

OWASP

Implementing and managing certificates is hard.

- You need to manage a Certificate Authority (CA)
- Have to distribute, and rotate certificates and keys
- Application code needs to be changed to handle TLS termination

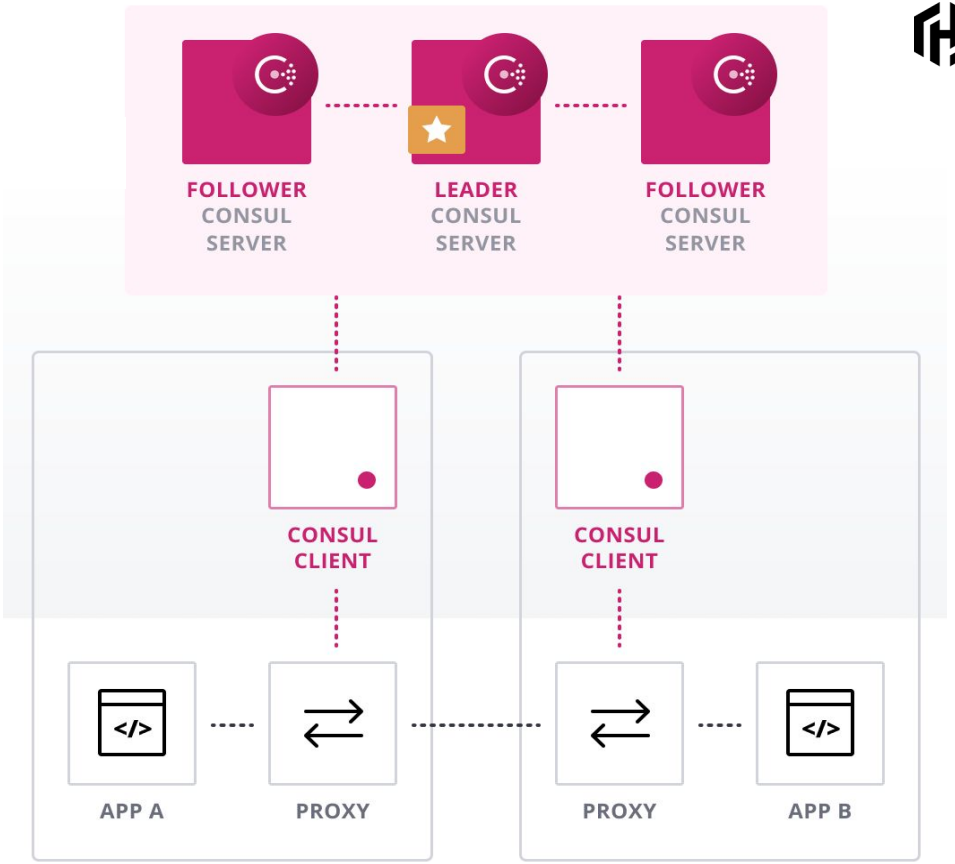
Reality is.. a service mesh can do all this for you.

What is a Service Mesh?

Service Mesh

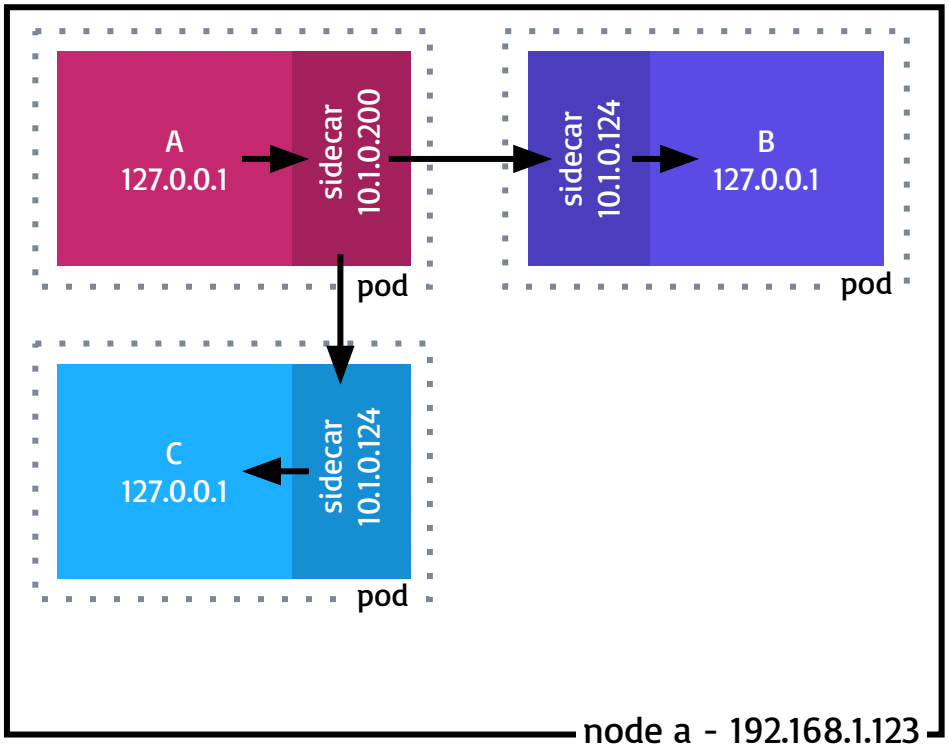
The **Control Plane** is responsible for Service Discovery, Authorization and configuring the proxies of the Data Plane.

The **Data Plane** is responsible for connecting services and routing data between them.



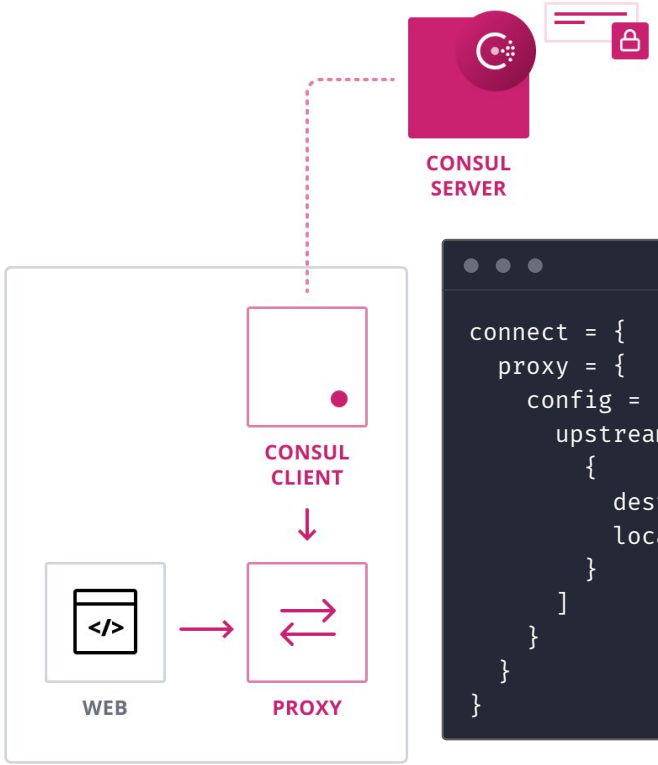
Sidecar pattern

- Service **binds to localhost** (http)
- Sidecar **accepts inbound connections** (https)
- Service and sidecar run in **isolated environment** e.g. k8s pod
- **0** code changes



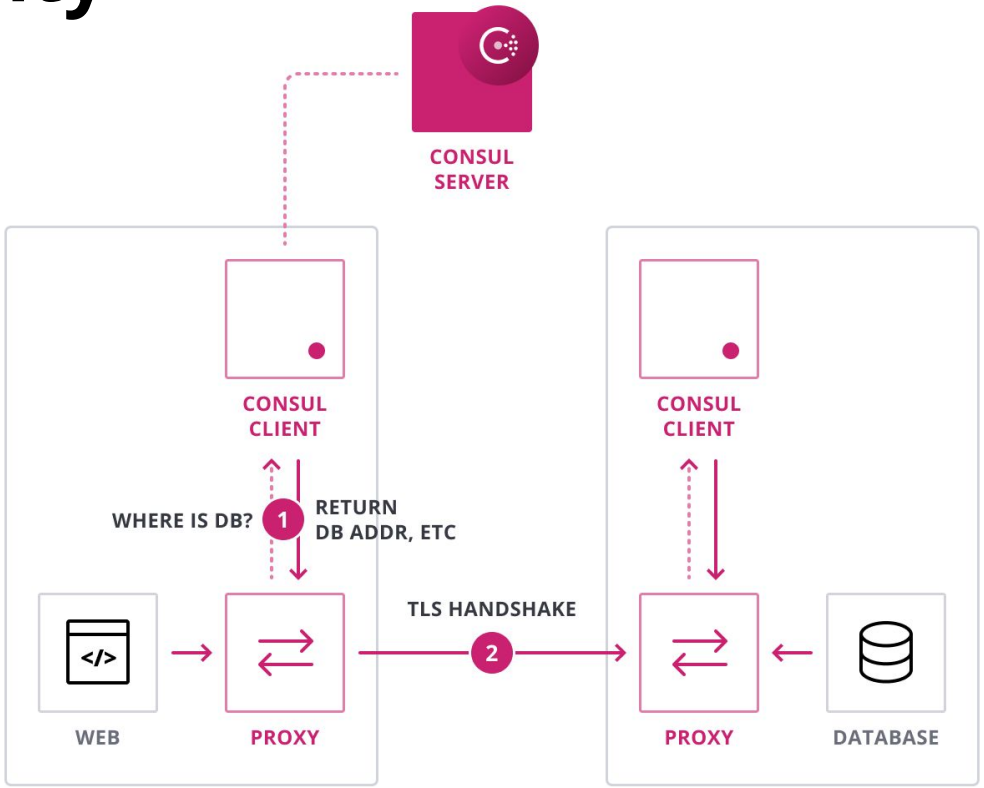
Configuration

- A proxy is **co-located** with the service instance that proxies inbound traffic.
- The Client agent **instantiates** the proxy and **registers** it as a service.
- The proxy is **configured** with a port that is used for the service and ports for any upstream destination that the service wants to connect to.



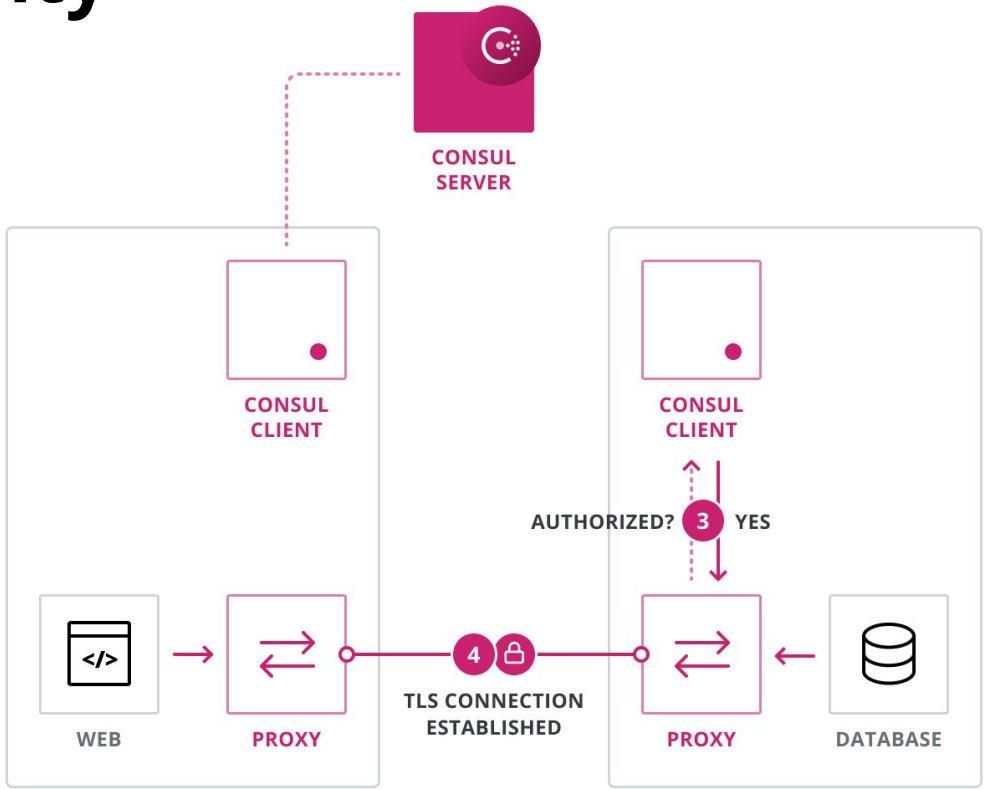
Service based security

- The local agent also returns the URI for the **expected identity** of the service it is connected to.
- Proxies between web and database start TLS handshake to **authenticate the identity**.



Service based security

- The DB proxy sends the **authorization request** to its local agent.
- The local agent **authorizes the connection** based on locally cached intention.
- **Mutual TLS** is established.



Intentions

The service access graph defines which services are **allowed or denied from communicating through intentions**.

Demo.

Summary.

- ~~You need to manage a Certificate Authority (CA)~~
- ~~Have to distribute, and rotate certificates and keys~~
- ~~Application code needs to be change to handle TLS termination~~
- Consul Connect has a **built in CA**, capable of leveraging HashiCorp **Vault**
- Consul Connect **manages key and certificate rotation**
- Sidecar proxy **terminates TLS** and **implements Authz**, no code changes

Thank You

eveld@hashicorp.com

@erikveld