
cisco
You make possible

A multi-cloud segmentation journey through big data

With Tetration for Network Engineers

Remi Philippe
Tim Garner

BRKACI-2040

cisco Live!
Barcelona | January 27-31, 2020

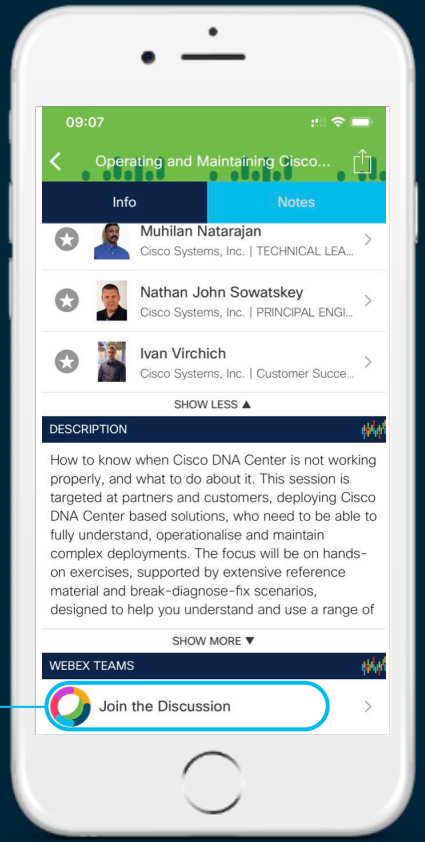
Cisco Webex Teams

Questions?

Use Cisco Webex Teams to chat with the speaker after the session

How

1. Find this session in the Cisco Events Mobile App
2. Click “Join the Discussion”
3. Install Webex Teams or go directly to the team space
4. Enter messages/questions in the team space



Why are you here?

The goal of this session is simple: we begin with an existing environment, full of applications and shared services, all running on top of a mixture of infrastructures, on-premises, and multiple clouds, deployed on bare metals, VMs, and containers - it's messy, just like the world you work on. We then take a freshly deployed Tetration and together walk through the step-by-step process of defining and enforcing segmentation policy through a combination of hierarchical "ground rules" and automated application policy discovery.

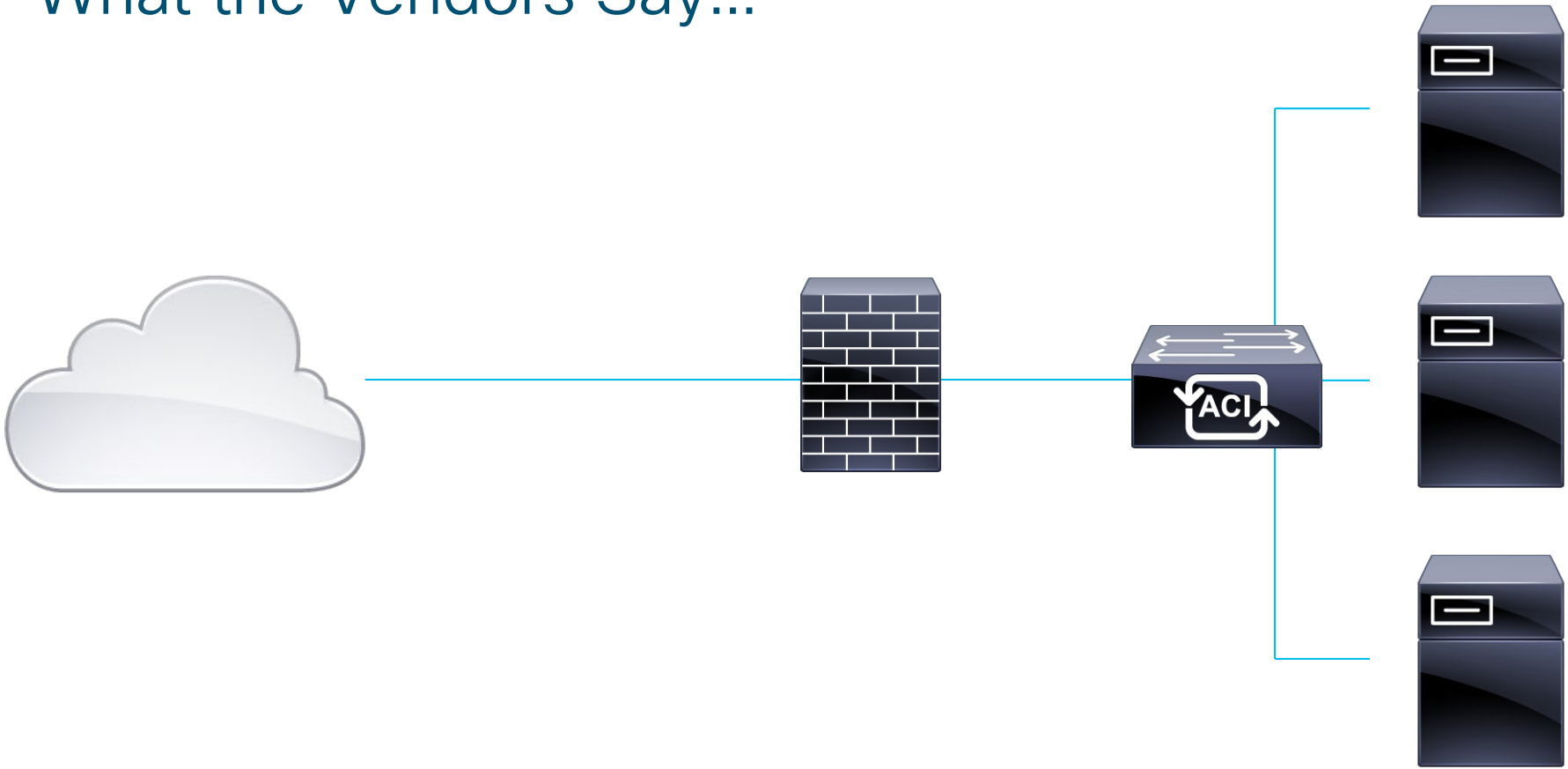
Agenda

- Why do we segment workloads?
  - What makes it hard across multiple networks
  - What makes it hard across multiple workload types (bare metals, VMs, containers)
  - How does networking keep a seat at the table
- Gathering the necessary information for building policy
  - Who is talking to whom: Agents, ERSPAN, NetFlow, NSEL
  - What defines the whom: vCenter, AWS, Kubernetes, Load balancers, Campus
- Segmenting…
  - Workloads in the same VLAN
  - Workloads across VLANs
  - Workloads in the cloud
- Learnings from the real world

Segmentation in the Real World

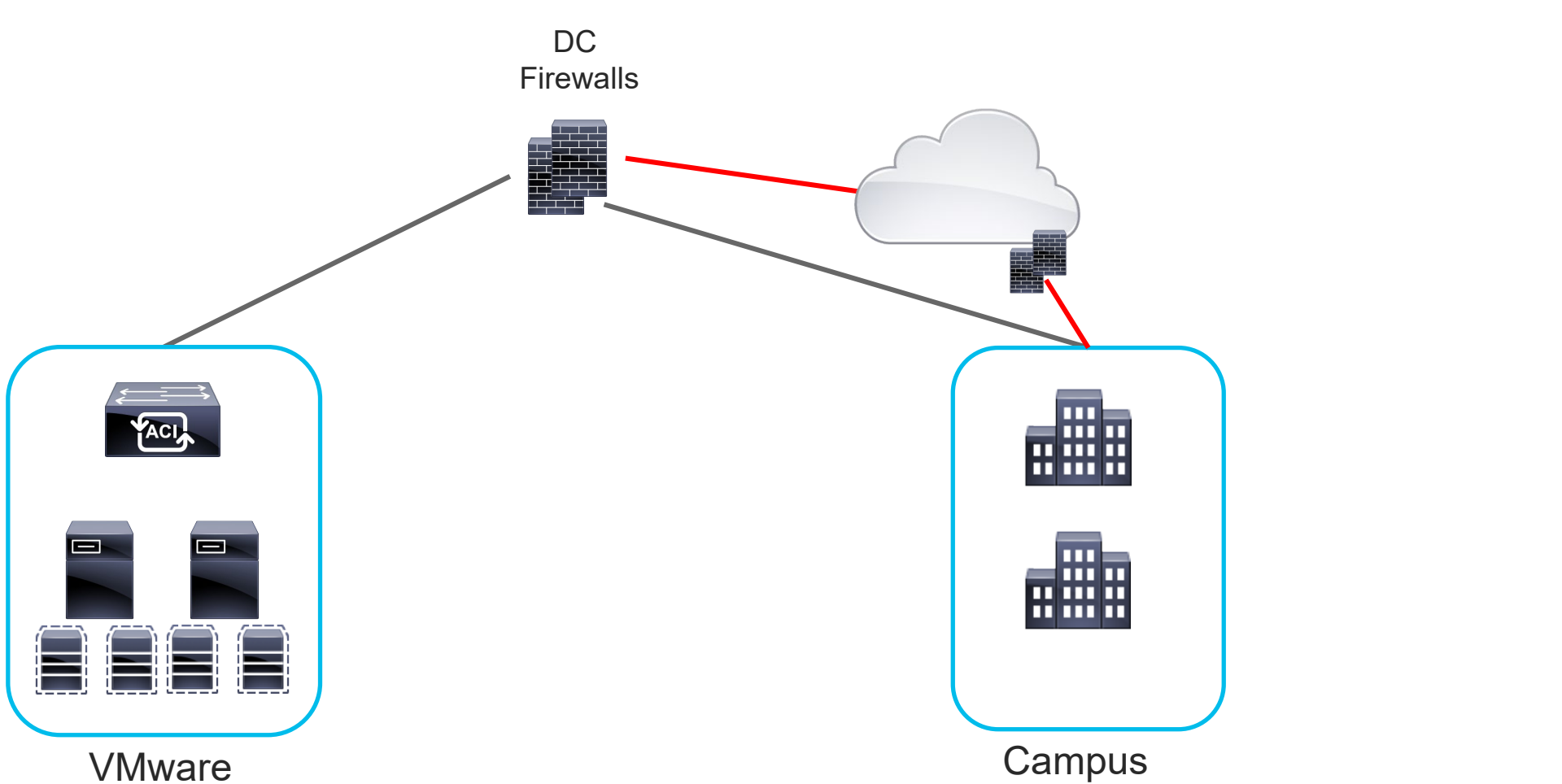
...One ACL at a time?!

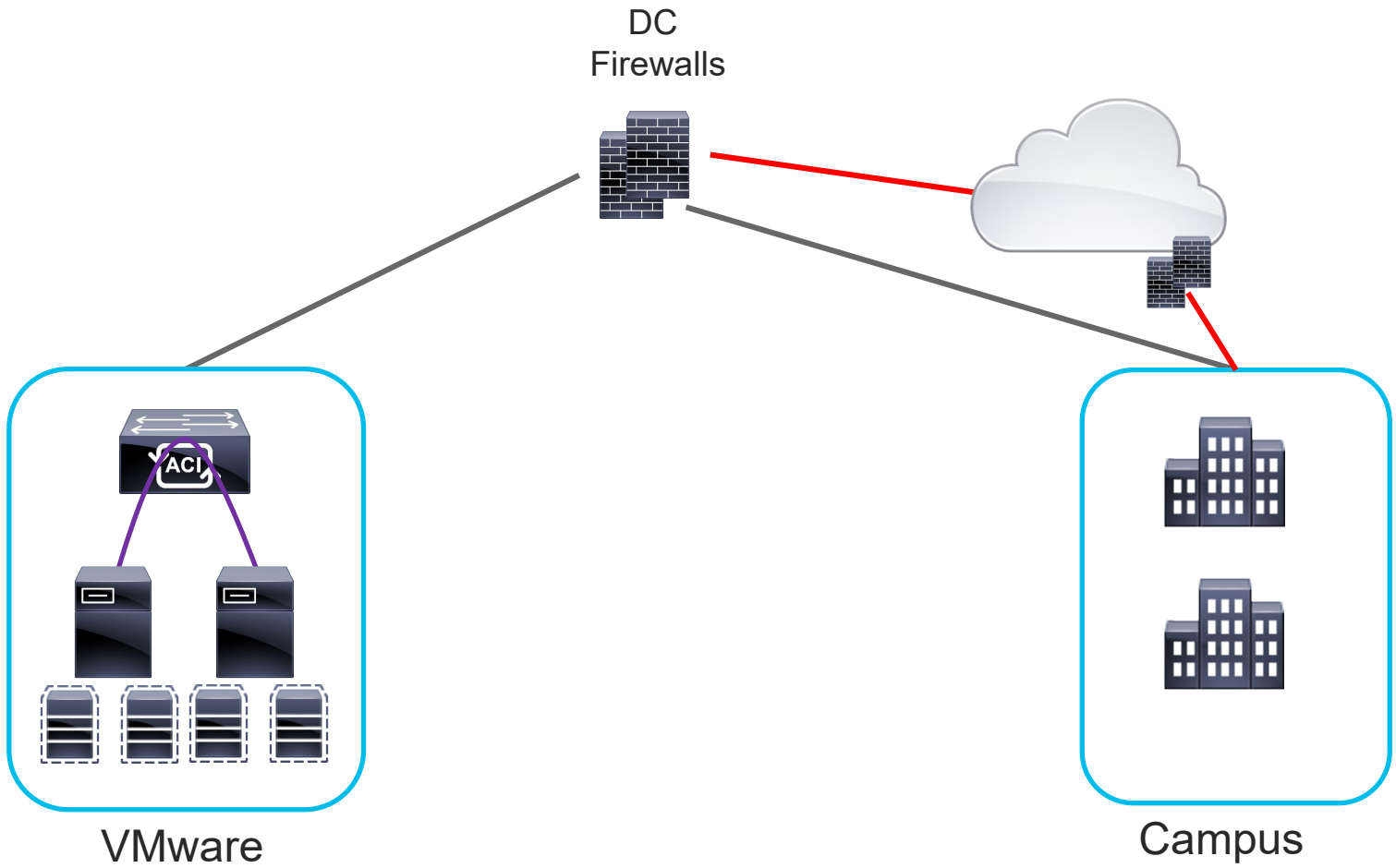
What the Vendors Say...

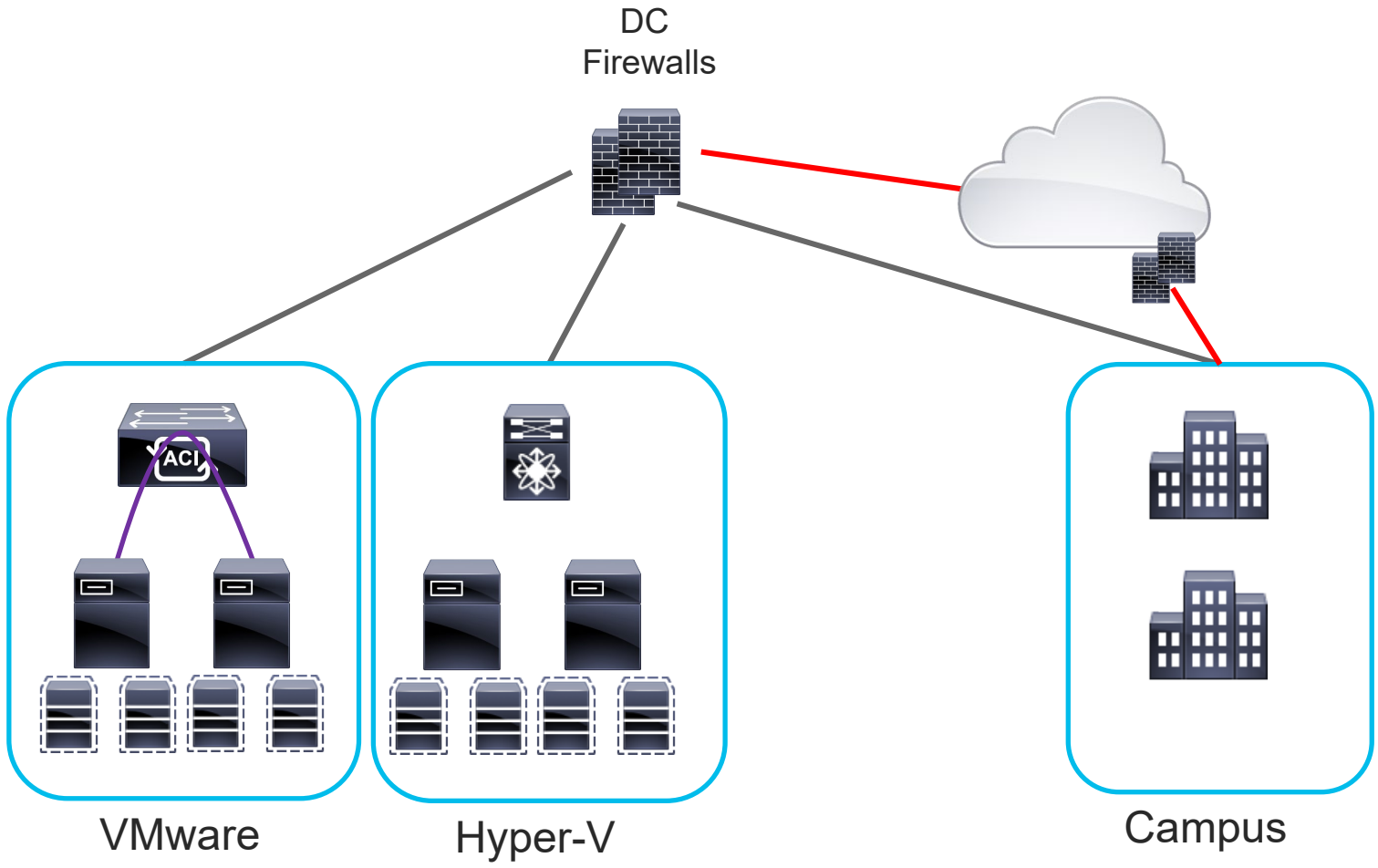


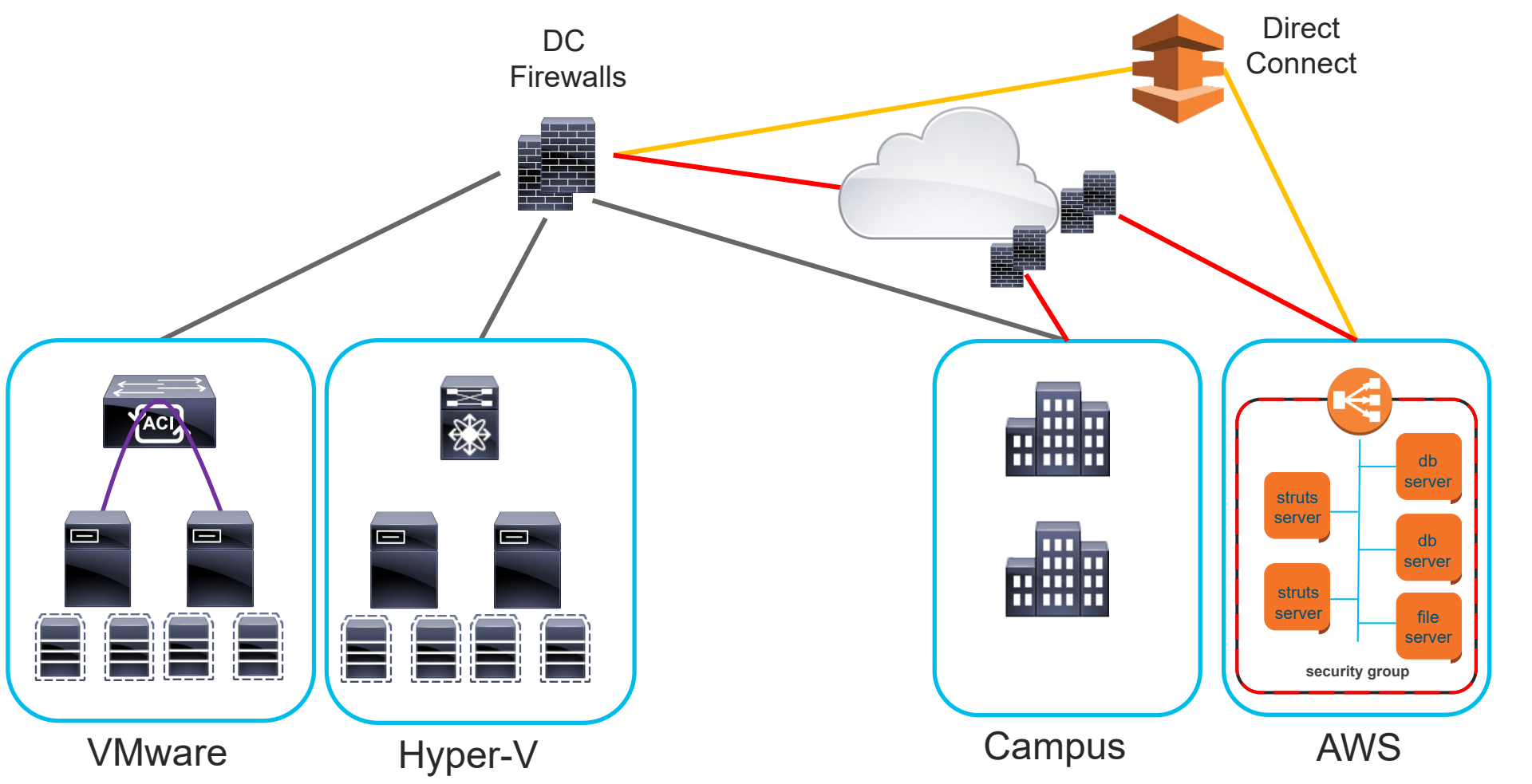
Now the **REAL** World

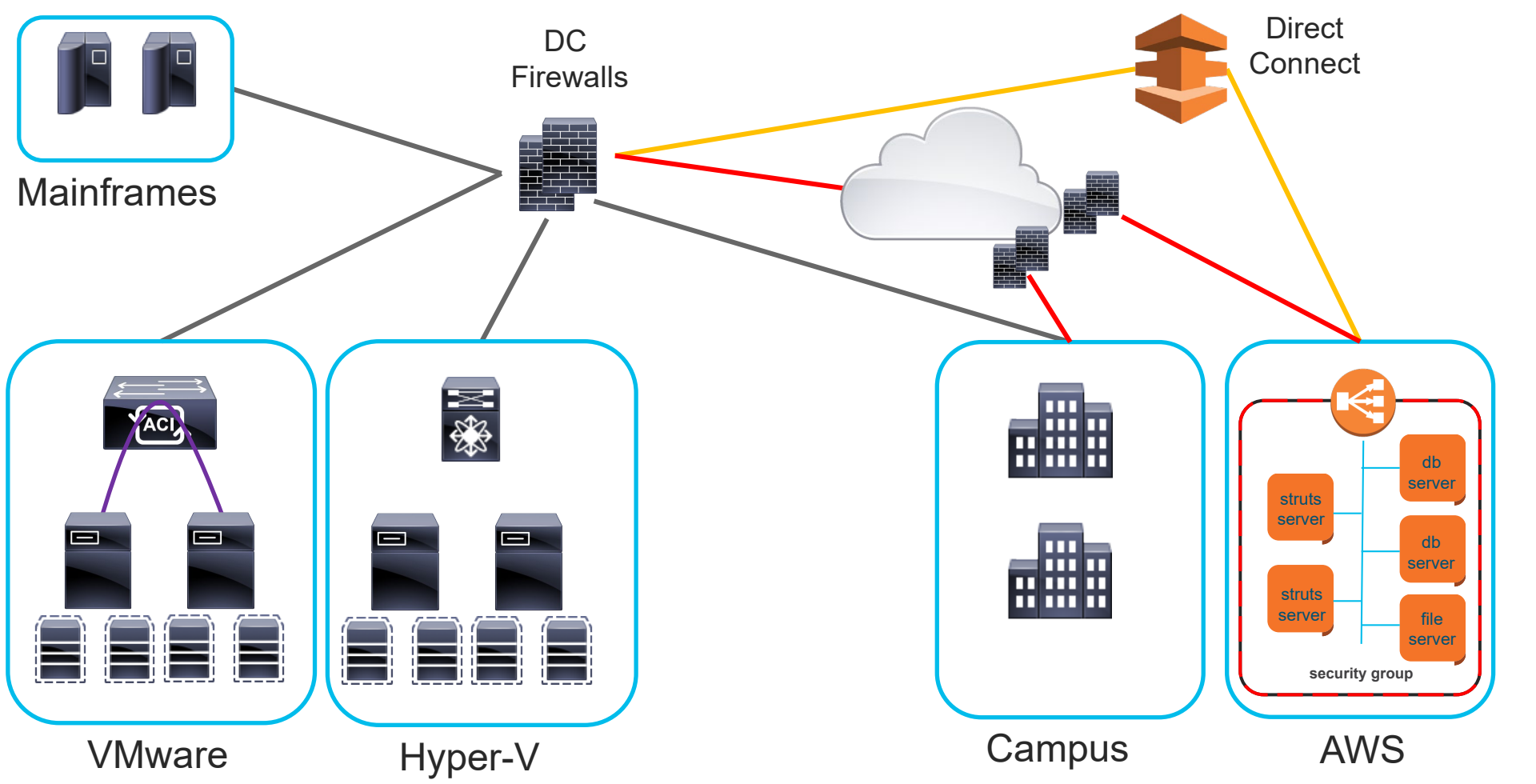
...And Security **Impossible**?!

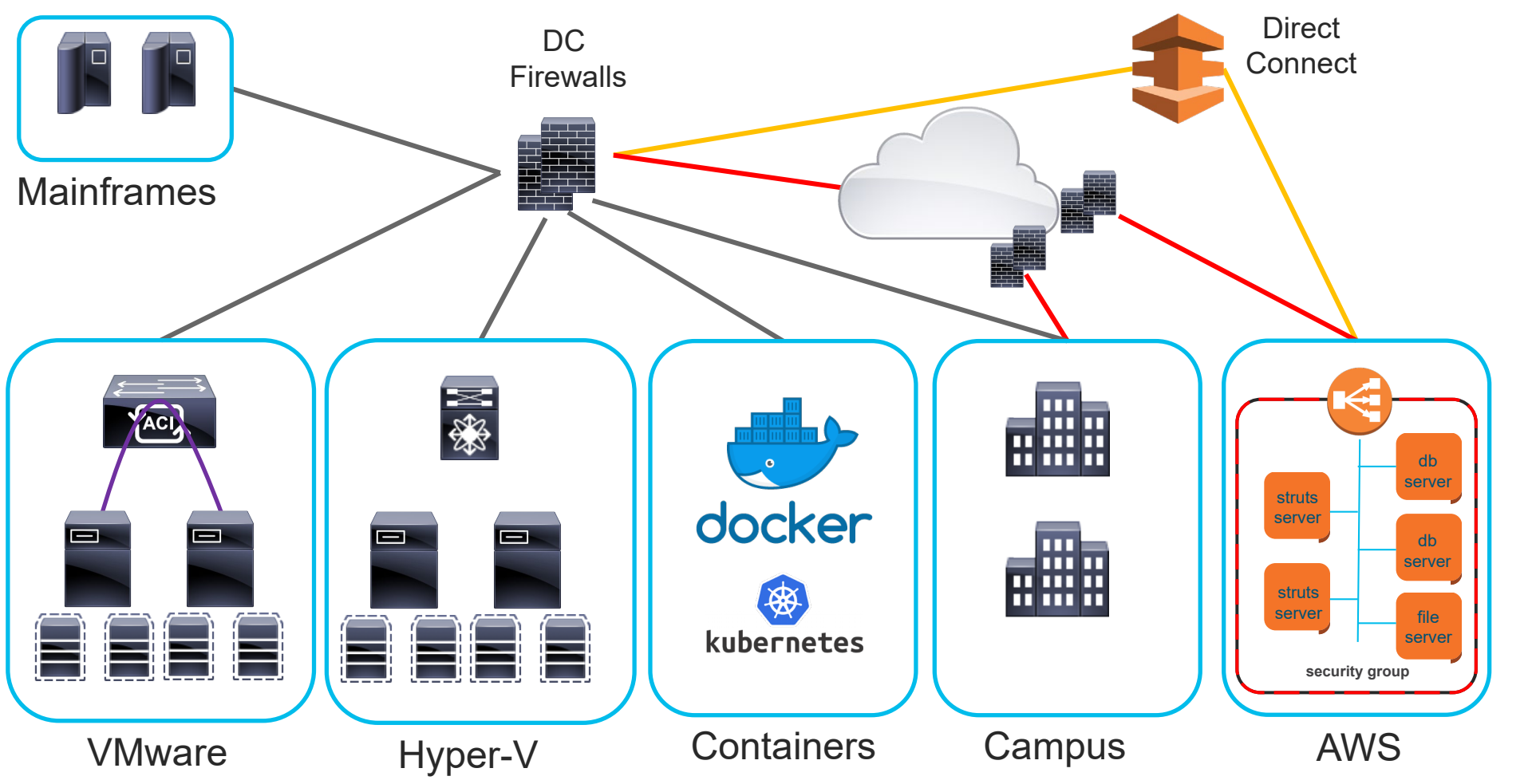












The Network Guy

- I've got a few ACLs for you

```
access-list 101 permit tcp 192.168.212.0 0.0.0.255 10.0.0.0 0.255.255.255 eq telnet
access-list 101 permit tcp 192.168.212.0 0.0.0.255 10.0.0.0 0.255.255.255 eq ftp
access-list 101 permit tcp 192.168.212.0 0.0.0.255 10.0.0.0 0.255.255.255 eq http
access-list 101 deny ip 192.168.212.0 0.0.0.255 10.0.0.0 0.255.255.255
access-list 101 permit icmp any 10.0.0.0 0.255.255.255 administratively-prohibited
access-list 101 permit icmp any 10.0.0.0 0.255.255.255 echo-reply
access-list 101 permit icmp any 10.0.0.0 0.255.255.255 packet-too-big
access-list 101 permit icmp any 10.0.0.0 0.255.255.255 time-exceeded
access-list 101 permit icmp any 10.0.0.0 0.255.255.255 unreachable
access-list 101 permit icmp 172.16.20.0 0.0.255.255
access-list 101 deny icmp any any
access-list 101 permit ip 202.33.42.0 0.0.0.255 any
access-list 101 permit ip 202.33.73.0 0.0.0.255 any
access-list 101 permit ip 202.33.48.0 0.0.0.255 any
access-list 101 permit ip 202.33.75.0 0.0.0.255 any
access-list 101 deny ip 202.33.0.0 0.0.255.255 any
access-list 101 deny tcp 210.120.122.0 0.0.0.255 10.2.2.0 0.255.255.255 eq www
access-list 101 deny tcp 210.120.183.0 0.0.0.255 10.2.2.0 0.255.255.255 eq www
access-list 101 deny tcp 210.120.114.0 0.0.0.255 10.2.2.0 0.255.255.255 eq www
access-list 101 deny tcp 210.120.175.0 0.0.0.255 10.2.2.0 0.255.255.255 eq www
access-list 101 deny tcp 210.120.136.0 0.0.0.255 10.2.2.0 0.255.255.255 eq www
access-list 101 deny tcp 210.120.177.0 0.0.0.255 10.2.2.0 0.255.255.255 eq www
access-list 101 permit tcp any 10.2.2.0 0.255.255.255 eq www
access-list 101 deny tcp any any eq www
access-list 101 permit tcp any any
access-list 101 deny ip 195.10.45.0 0.0.0.255 any
access-list 101 permit ip any any
```

The Network Guy

- I've got a few ACLs for you

```
access-list 101 permit tcp 192.168.212.0 0.0.0.255 10.0.0.0 0.255.255.255 eq telnet
access-list 101 permit tcp 192.168.212.0 0.0.0.255 10.0.0.0 0.255.255.255 eq ftp
access-list 101 permit tcp 192.168.212.0 0.0.0.255 10.0.0.0 0.255.255.255 eq http
access-list 101 deny ip 192.168.212.0 0.0.0.255 10.0.0.0 0.255.255.255
access-list 101 permit icmp any 10.0.0.0 0.255.255.255 administratively-prohibited
access-list 101 permit icmp any 10.0.0.0 0.255.255.255 echo-reply
access-list 101 permit icmp any 10.0.0.0 0.255.255.255 packet-too-big
access-list 101 permit icmp any 10.0.0.0 0.255.255.255 time-exceeded
access-list 101 permit icmp any 10.0.0.0 0.255.255.255 unreachable
access-list 101 permit icmp 172.16.20.0 0.0.255.255
access-list 101 deny icmp any any
access-list 101 permit ip 202.33.42.0 0.0.0.255 any
access-list 101 permit ip 202.33.73.0 0.0.0.255 any
access-list 101 permit ip 202.33.48.0 0.0.0.255 any
access-list 101 permit ip 202.33.75.0 0.0.0.255 any
access-list 101 deny ip 202.33.0.0 0.0.255.255 any
access-list 101 deny tcp 210.120.122.0 0.0.0.255 10.2.2.0 0.255.255.255 eq www
access-list 101 deny tcp 210.120.183.0 0.0.0.255 10.2.2.0 0.255.255.255 eq www
access-list 101 deny tcp 210.120.114.0 0.0.0.255 10.2.2.0 0.255.255.255 eq www
access-list 101 deny tcp 210.120.175.0 0.0.0.255 10.2.2.0 0.255.255.255 eq www
access-list 101 deny tcp 210.120.136.0 0.0.0.255 10.2.2.0 0.255.255.255 eq www
access-list 101 deny tcp 210.120.177.0 0.0.0.255 10.2.2.0 0.255.255.255 eq www
access-list 101 permit tcp any 10.2.2.0 0.255.255.255 eq www
access-list 101 deny tcp any any eq www
access-list 101 permit tcp any any
access-list 101 deny ip 195.10.45.0 0.0.0.255 any
access-list 101 permit ip any any
```

CYA Protection

The Mainframe Guy

- Talk to the network folks. Not touching the mainframe

The Virtualization Guy

- I have an UI! Give me 30 mins to click through the 10 screens in 2 browsers.

The Kubernetes Guy

- It's easy, just match my POD selector and allow traffic in my CNI

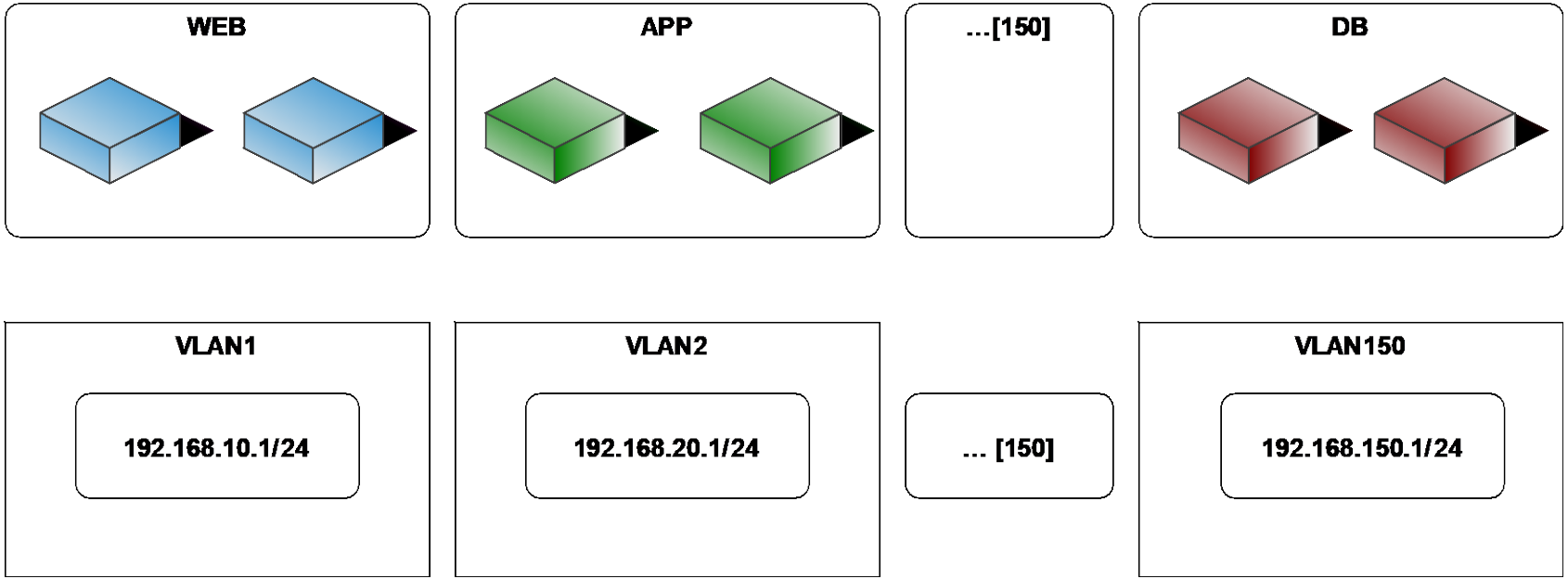
```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - ipBlock:
        cidr: 172.17.0.0/16
        except:
        - 172.17.1.0/24
    - namespaceSelector:
        matchLabels:
          project: myproject
    - podSelector:
        matchLabels:
          role: frontend
    ports:
    - protocol: TCP
      port: 6379
  egress:
  - to:
    - ipBlock:
        cidr: 10.0.0.0/24
    ports:
    - protocol: TCP
      port: 5978
```

The Security Girl

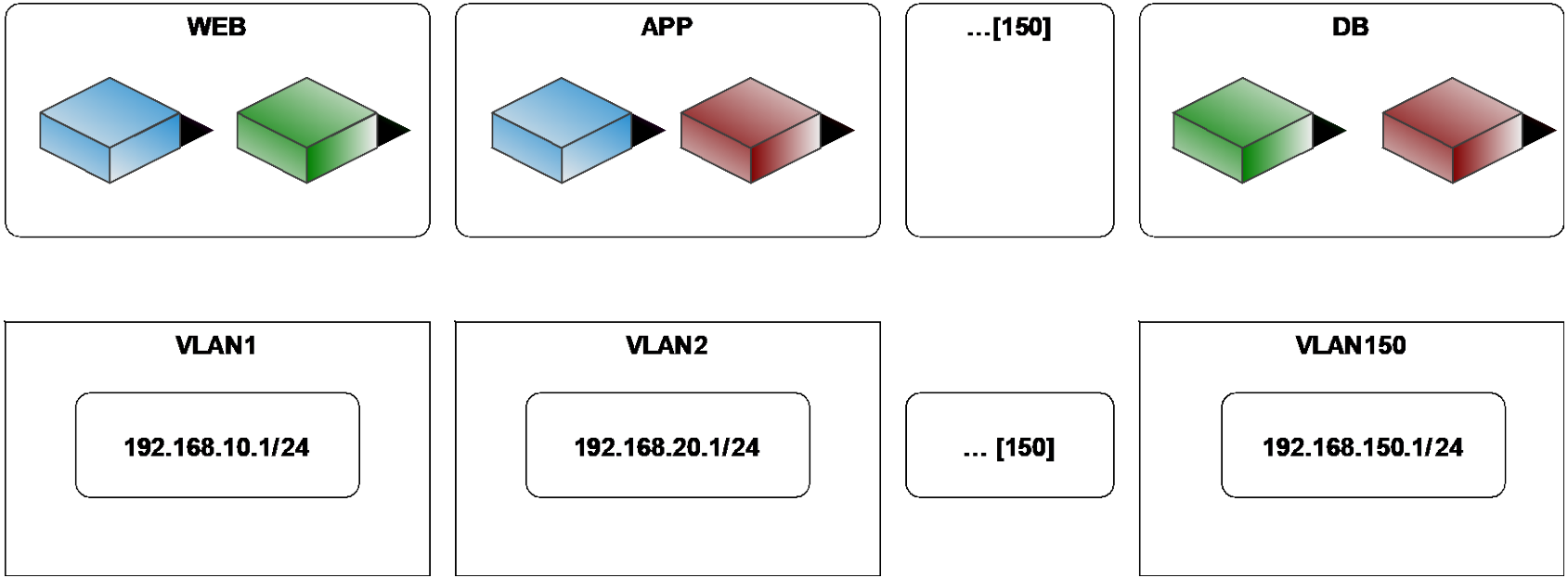
Bad Guys

Segmentation concerns of network engineers

What we hope for

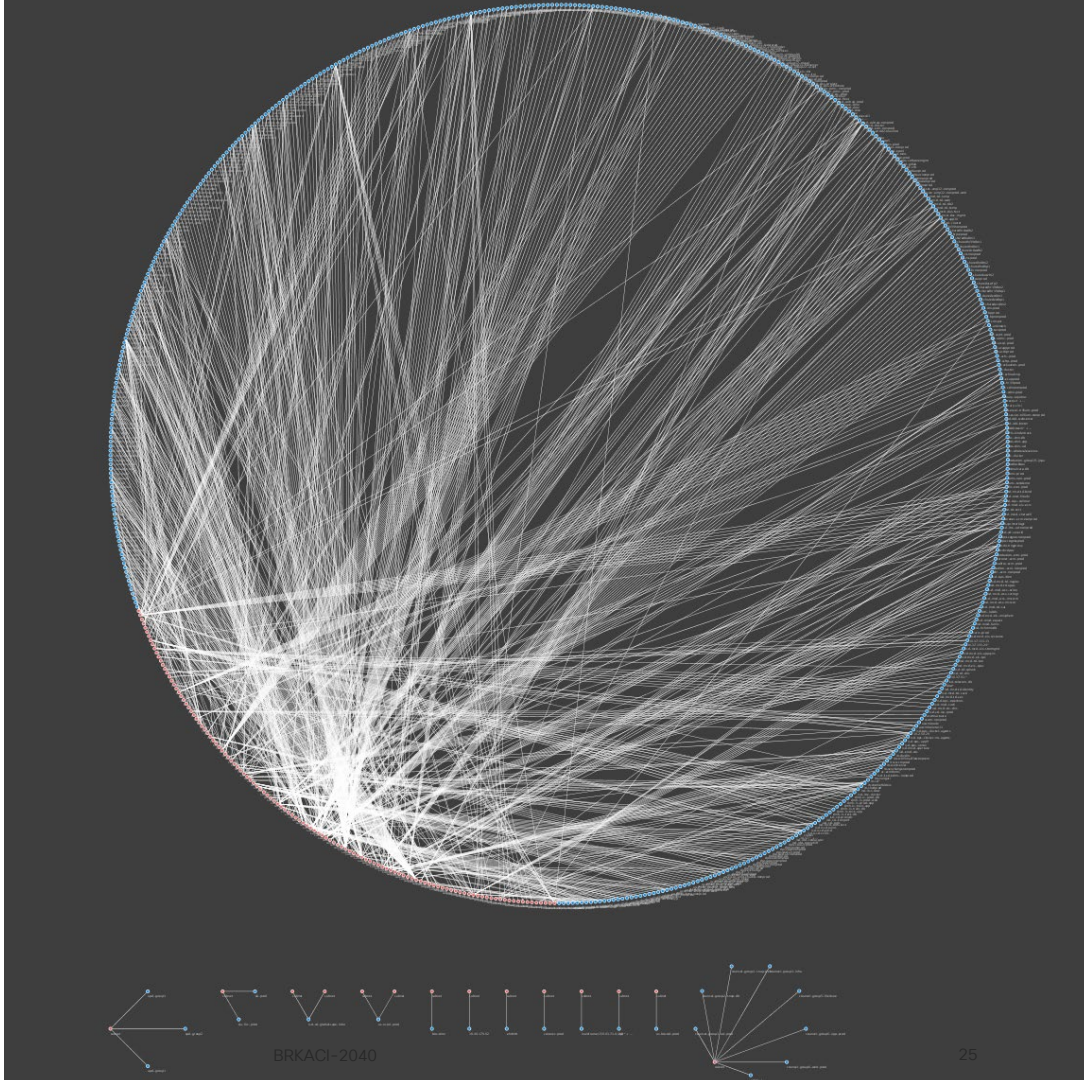


What life gives us



Forwarding and Policy

it gets complicated

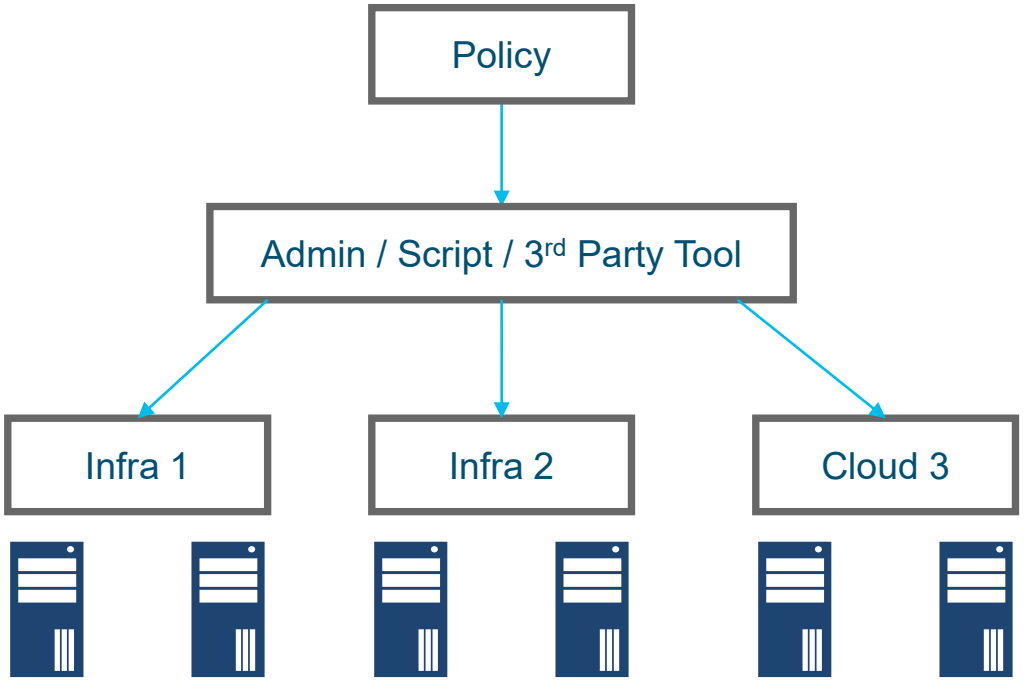


Common Problems

- There is no guarantee of placement control on applications
- There are many existing subnets across different network infrastructures
- Application owners have been assigned addresses at random from one OR more subnets with no logical order
- Applications are not able to re-address their applications
- Traffic engineering and service stitching required to enforce policies
- Inconsistent capabilities of different policy enforcement points
- How to deal with stateful traffic during network convergence or failovers

A new approach

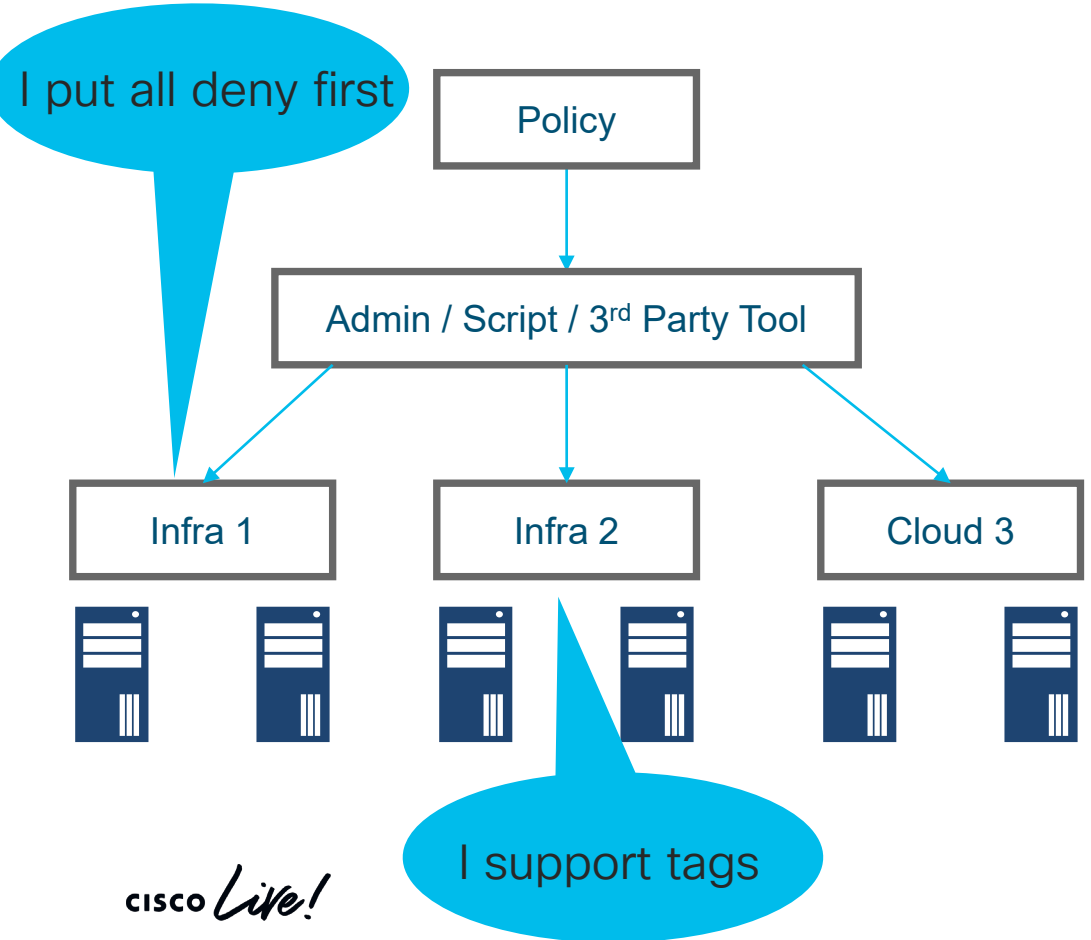
Infrastructure enforced policy



Administrator is responsible for converting logical policy into infra specific configs

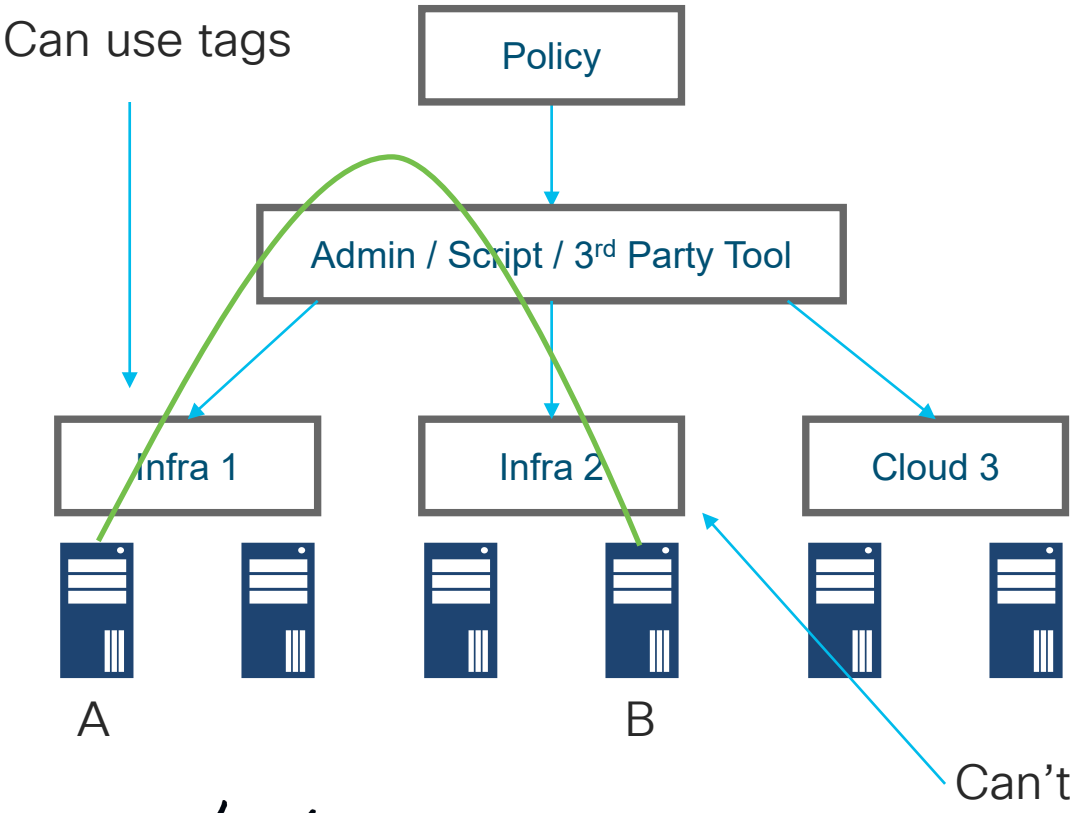
But...

Infrastructure enforced policy



Each infrastructure has unique capabilities and scale limits

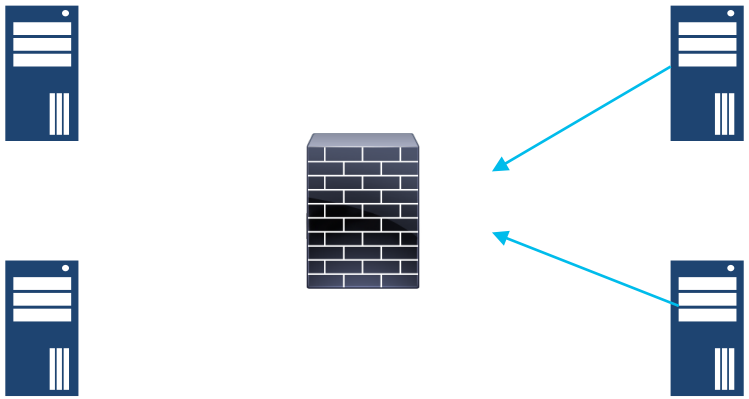
Infrastructure enforced policy



Administrator must guarantee policy is converted correctly and continues to be compliant with logical intent

How do you guarantee membership?
Who is responsible?

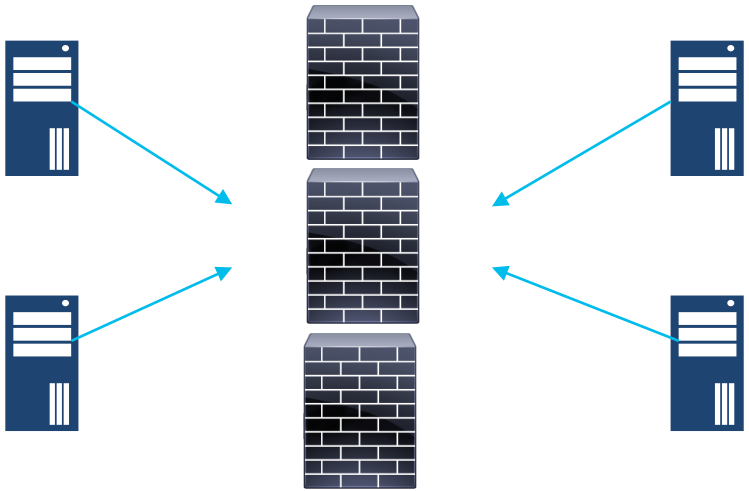
Infrastructure enforced policy



East to West segmentation generates comprehensive dynamic policy sets

Security capabilities become tied to infrastructure capabilities

Infrastructure enforced policy

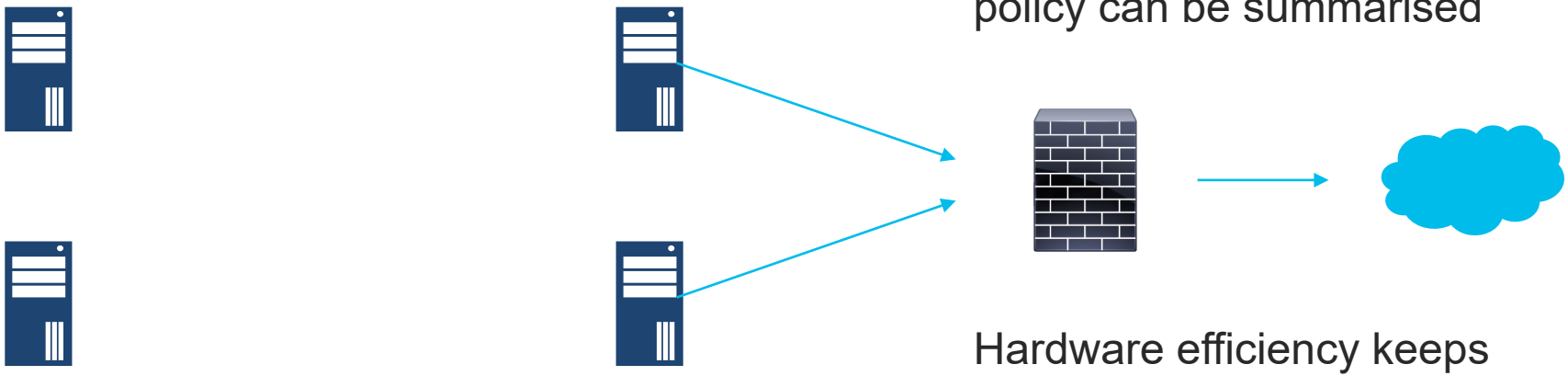


East to West segmentation generates comprehensive dynamic policy sets

Security capabilities become tied to infrastructure capabilities

Larger and larger devices are required in the infra to handle demand

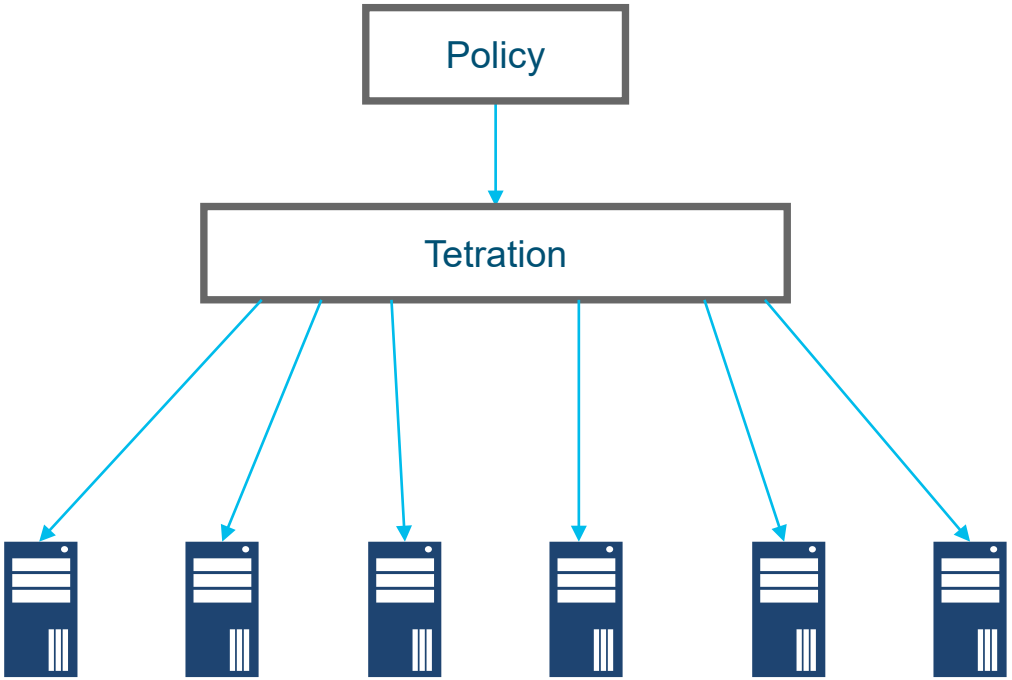
Infrastructure enforced policy



Works extremely well at choke points and where policy can be summarised

Hardware efficiency keeps unwanted traffic out of the datacentre without impact to workloads

Host enforced policy

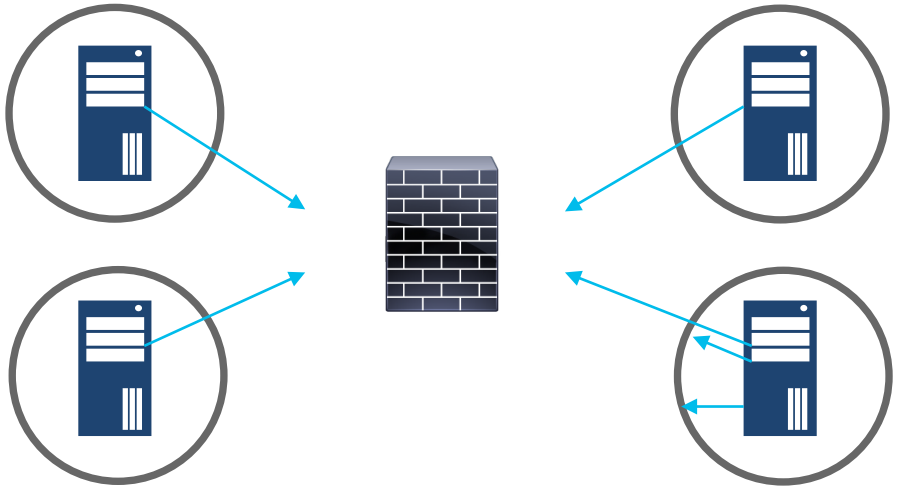


Tetration is responsible for converting logical policy into host specific configs

All hosts have the same capabilities and scale limits

Tetration guarantees policy is converted correctly and continues monitor and main compliance with logical intent

Enables a lightweight firewall on each workload

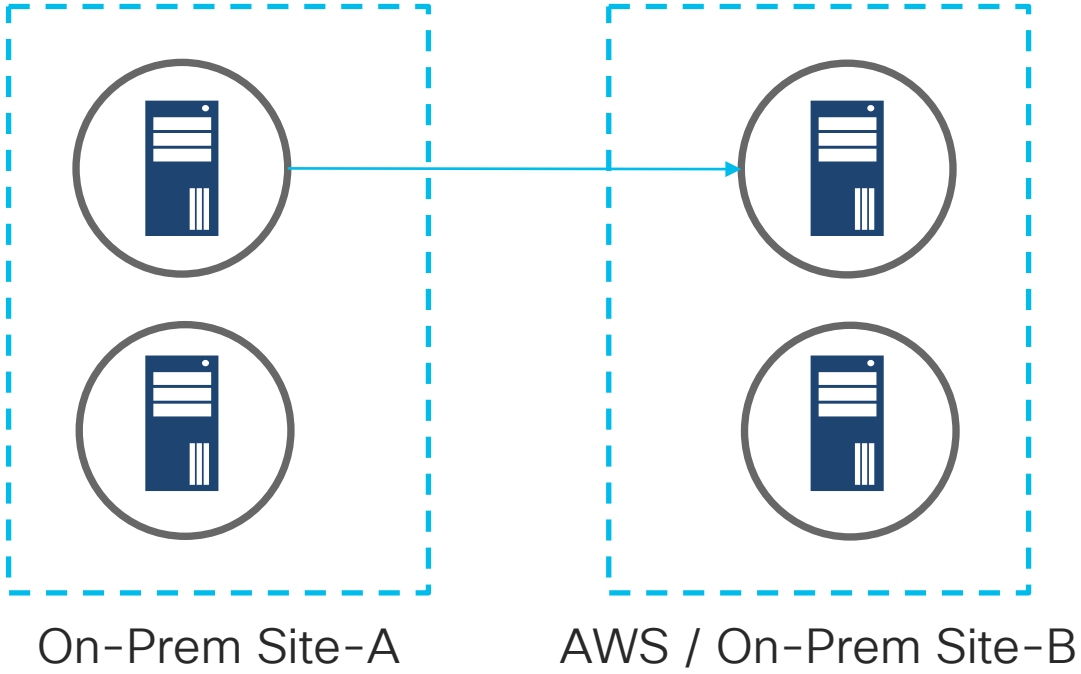


Each workload runs a lightweight firewall capable of advanced enforcement

Tetration can lighten the amount of east to west policies in the infrastructure

OUTPUT policies protects workloads by preventing traffic from getting out…

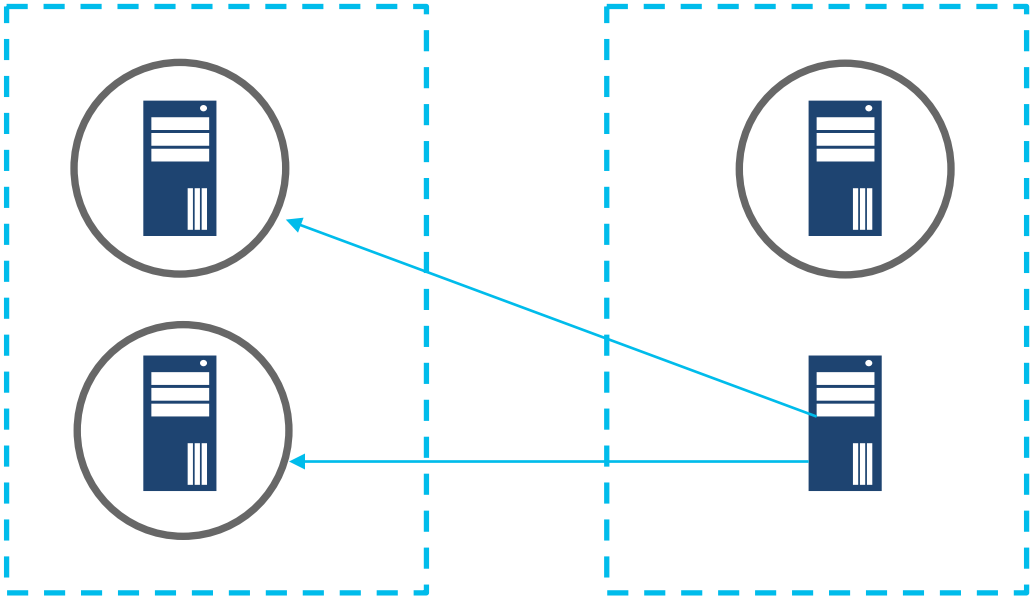
Enables a lightweight firewall on each workload



The workload firewalls are agnostic to the underlying infrastructure –

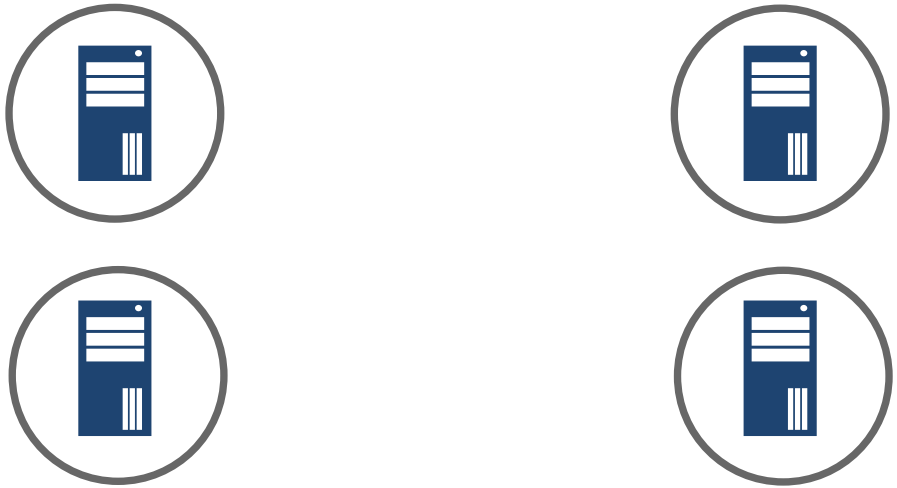
You design your policy in one view and it's enforced the same way globally

Breaches are immediately contained



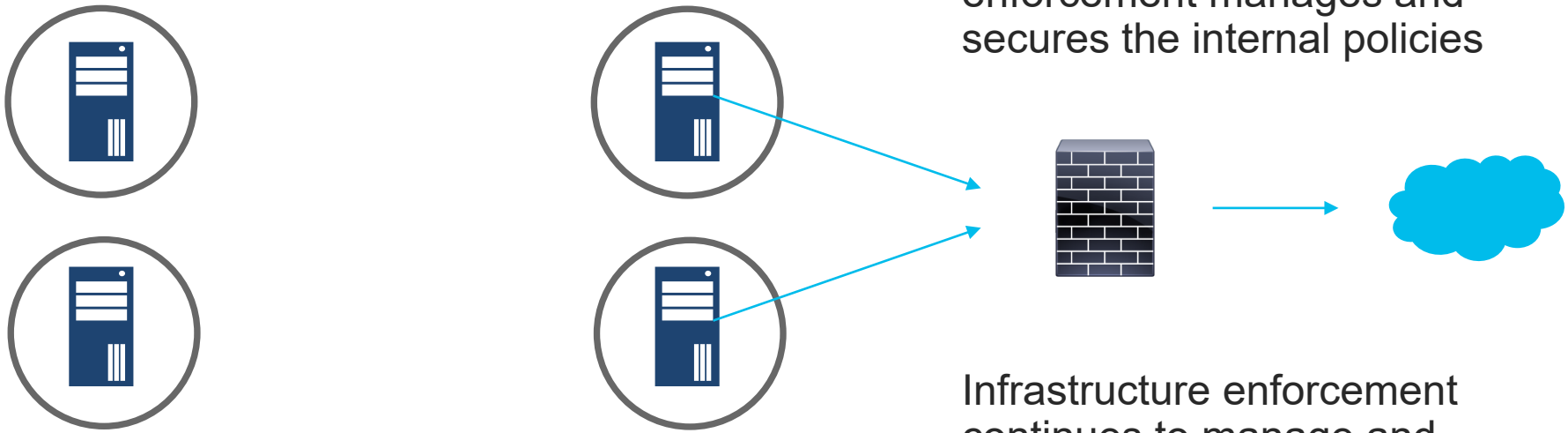
…And INPUT protects other workloads in case of breach (of workload or process)!

Multiple layers of defense



Tetration host based enforcement manages and secures the internal policies

Multiple layers of defense



Tetration host based enforcement manages and secures the internal policies

Infrastructure enforcement continues to manage and secure the edge policies

How does network engineering keep a seat at the table?

Unfortunately, policy is seen as a pain

- That Excel Spreadsheet I have to fill in
- Those approvers I need to chase
- Waiting for the next change window

< your best complaint here>

So what do users do?

How does network engineering keep a seat at the table?

- Developers want policy as code
  - Enforced in the CNI
- DevsecOps may want trust-based computing
  - TLS encrypted
- App Owner set the host firewall
  - Windows GPO anyone?
- And network gets /32 to /24 rules (at best)
  - Ask once so make it count!

How about we build a solution to make everyone happy?
Using Hardware and Software where they make the most sense?

Big Data?
Where is this Big Data?

...If you know where to look?!

Source of Data

▸	18:19:39	2 938996165657 eni-0149f2702f5ce87c0 172.30.0.93 173.255.206.154 38624 123 17 1 76 1559067579 1559067580 ACCEPT OK
▸	18:19:39	2 938996165657 eni-0149f2702f5ce87c0 173.255.206.154 172.30.0.93 123 38624 17 1 76 1559067579 1559067580 ACCEPT OK
▸	18:19:39	2 938996165657 eni-0149f2702f5ce87c0 185.244.25.141 172.30.0.93 45612 8088 6 1 40 1559067579 1559067580 REJECT OK

- In order to design a segmentation policy, we need data
  - In the form of flow data
    - IP X talked to IP Y over port Z

Source of Data

▸	18:19:39	2 938996165657 eni-0149f2702f5ce87c0 172.30.0.93 173.255.206.154 38624 123 17 1 76 1559067579 1559067580 ACCEPT OK
▸	18:19:39	2 938996165657 eni-0149f2702f5ce87c0 173.255.206.154 172.30.0.93 123 38624 17 1 76 1559067579 1559067580 ACCEPT OK
▸	18:19:39	2 938996165657 eni-0149f2702f5ce87c0 185.244.25.141 172.30.0.93 45612 8088 6 1 40 1559067579 1559067580 REJECT OK

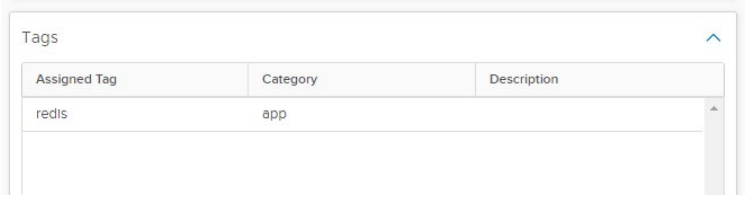
- In order to design a segmentation policy, we need data
  - In the form of flow data
    - IP X talked to IP Y over port Z
  - In the form of context
    - port Z is being served by HTTPD

```
cisco@sjc15-jenkins1:~$ sudo netstat -anlp | grep java
tcp6       0      0 :::8080                 :::*                    LISTEN      1806/java
udp6       0      0 :::5353                 :::*                                1806/java
udp6       0      0 :::33848                :::*                                1806/java
cisco@sjc15-jenkins1:~$ ps -ef | grep 1806
jenkins   1806  1804  0 May24 ?        00:04:38 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/jenkins/jenkins.war --webroot=/var/cache/jenkins/war --
httpPort=8080
cisco@sjc15-jenkins1:~$
```

Source of Data

▸	18:19:39	2 938996165657 eni-0149f2702f5ce87c0 172.30.0.93 173.255.206.154 38624 123 17 1 76 1559067579 1559067580 ACCEPT OK
▸	18:19:39	2 938996165657 eni-0149f2702f5ce87c0 173.255.206.154 172.30.0.93 123 38624 17 1 76 1559067579 1559067580 ACCEPT OK
▸	18:19:39	2 938996165657 eni-0149f2702f5ce87c0 185.244.25.141 172.30.0.93 45612 8088 6 1 40 1559067579 1559067580 REJECT OK

- In order to design a segmentation policy, we need data
  - In the form of flow data
    - IP X talked to IP Y over port Z
  - In the form of context
    - port Z is being served by HTTPD
  - In the form of user data
    - IP X has tag App=SAP…




```
cisco@sjc15-jenkins1:~$ sudo netstat -anlp | grep java
tcp6       0      0 :::8080                 :::*                    LISTEN      1806/java
udp6       0      0 :::5353                 :::*                                1806/java
udp6       0      0 :::33848                :::*                                1806/java
cisco@sjc15-jenkins1:~$ ps -ef | grep 1806
jenkins   1806  1804  0 May24 ?       00:04:38 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/jenkins/jenkins.war --webroot=/var/cache/jenkins/war --
httpPort=8080
cisco@sjc15-jenkins1:~$
```

See a Problem?

- We're looking for 3 different type of data
  - Context
  - Flow
  - Process
- All come in different shape and form
  - AWS is different from Azure and GCP
  - vSphere vs Hyper-V vs KVM
  - Sampled Netflow vs Streaming Telemetry
- First Step is to combine this data in a usable way
  - **Good news, Tetration does that for you!**

What Happens?

Flow Data	IP: 1.2.3.4 Src Port: 45862 Protocol: TCP Flags: SYN, ACK	IP: 4.3.2.1 Dst Port: 80 Protocol: TCP Flags: SYN, ACK

Collected from AnyConnect

Collected by the Software Agent

What Happens?

Flow Data	IP: 1.2.3.4 Src Port: 45862 Protocol: TCP Flags: SYN, ACK	IP: 4.3.2.1 Dst Port: 80 Protocol: TCP Flags: SYN, ACK

Why are flags Important?

What Happens?

Flow Data	IP: 1.2.3.4 Src Port: 45862 Protocol: TCP Flags: SYN, ACK	IP: 4.3.2.1 Dst Port: 80 Protocol: TCP Flags: RST, ACK

What if the flags were like this?
Should I consider that a legit flow?

What Happens?

Containers anyone?
How do you know what process is running behind a port?

Process Data	Socket: TCP/45862 State: ESTABLISHED Process: Chrome Hash: 0xabcdef01	Socket: TCP/80 State: LISTEN Process: HTTPD Hash: 0xabcdef02

What Happens?

Process Data	Socket: TCP/45862 State: ESTABLISHED Process: Chrome Hash: 0xabcdef01	Socket: TCP/80 State: LISTEN Process: HTTPD Hash: 0xabcdef02

How do you know if you're whitelisting a malware?

What Happens?

Do you know who this is?

```
inet6 fe80::e4bf:83ff:fe1e:3276%awdl0
```

Context Data

AnyConnect User: remi
ISE Posture: Trusted
Location: Paris, FR
AD Group: Engineering

App: Intranet
Location: SJC
Data: Internal
ESX: host-1

What Happens?

Do you know where this workload is?

Context Data	AnyConnect User: remi ISE Posture: Trusted Location: Paris, FR AD Group: Engineering	App: Intranet Location: SJC Data: Internal ESX: host-1

What Happens?

What does Service Now say?

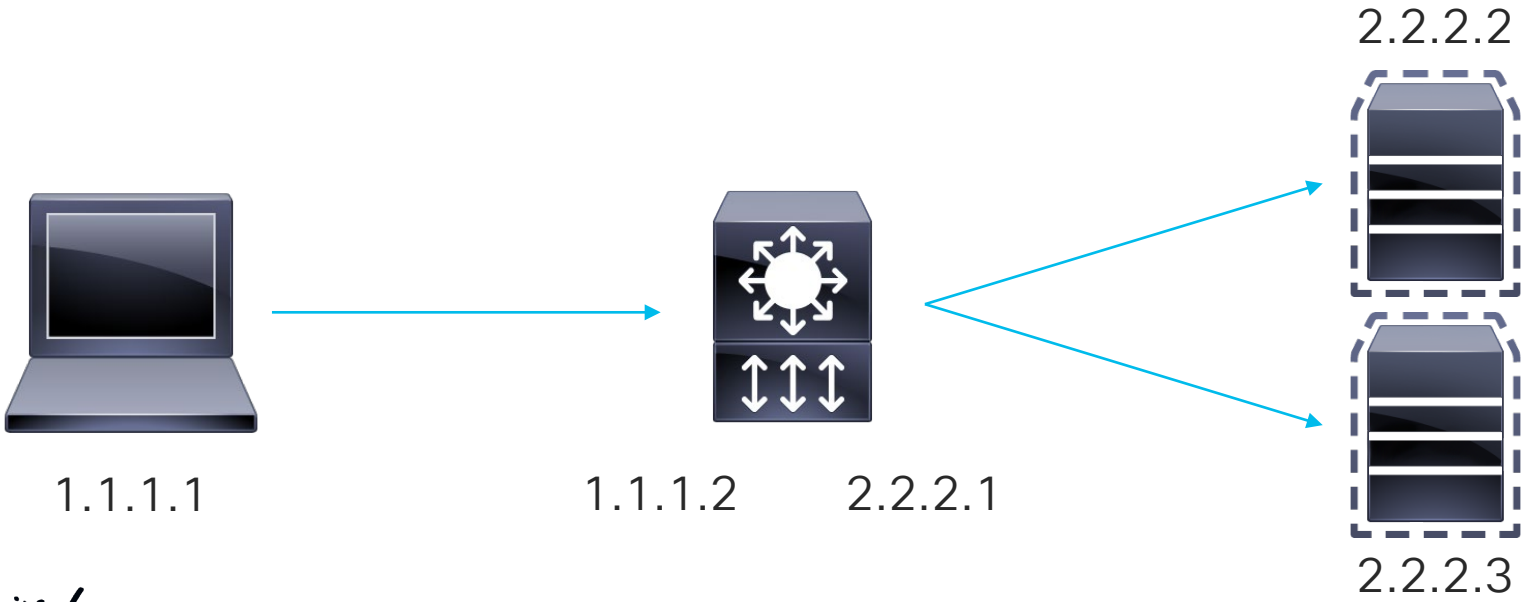
Context Data	AnyConnect User: remi ISE Posture: Trusted Location: Paris, FR AD Group: Engineering	App: Intranet Location: SJC Data: Internal ESX: host-1

Now the Complete **fully correlated** Picture

Flow Data	IP: 1.2.3.4 Src Port: 45862 Protocol: TCP Flags: SYN, ACK	IP: 4.3.2.1 Dst Port: 80 Protocol: TCP Flags: SYN, ACK
Process Data	Socket: TCP/45862 State: ESTABLISHED Process: Chrome Hash: 0xabcdef01	Socket: TCP/80 State: LISTEN Process: HTTPD Hash: 0xabcdef02
Context Data	AnyConnect User: remi ISE Posture: Trusted Location: Paris, FR AD Group: Engineering	App: Intranet Location: SJC Data: Internal ESX: host-1

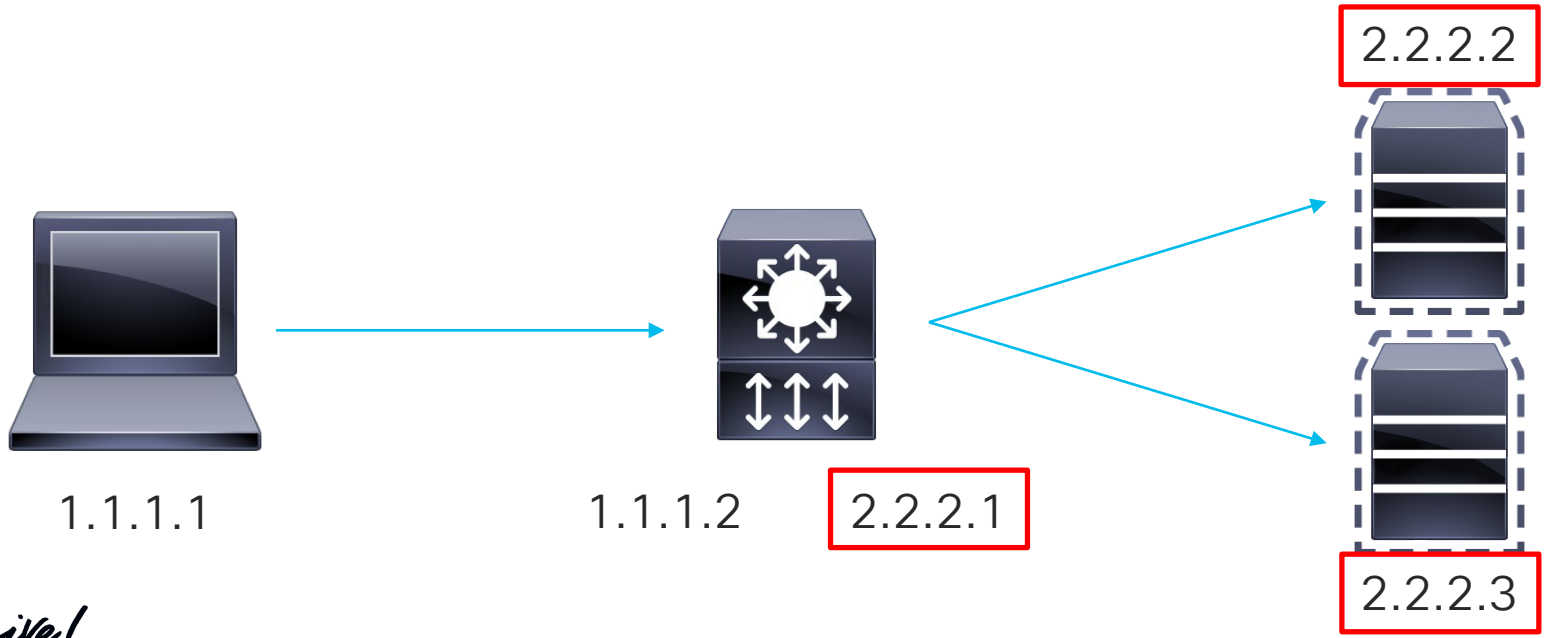
Oops, we forgot the Load Balancers?

- So Far™, we've never seen a network without a load balancer
  - They have the tendency to NAT and "break" flows



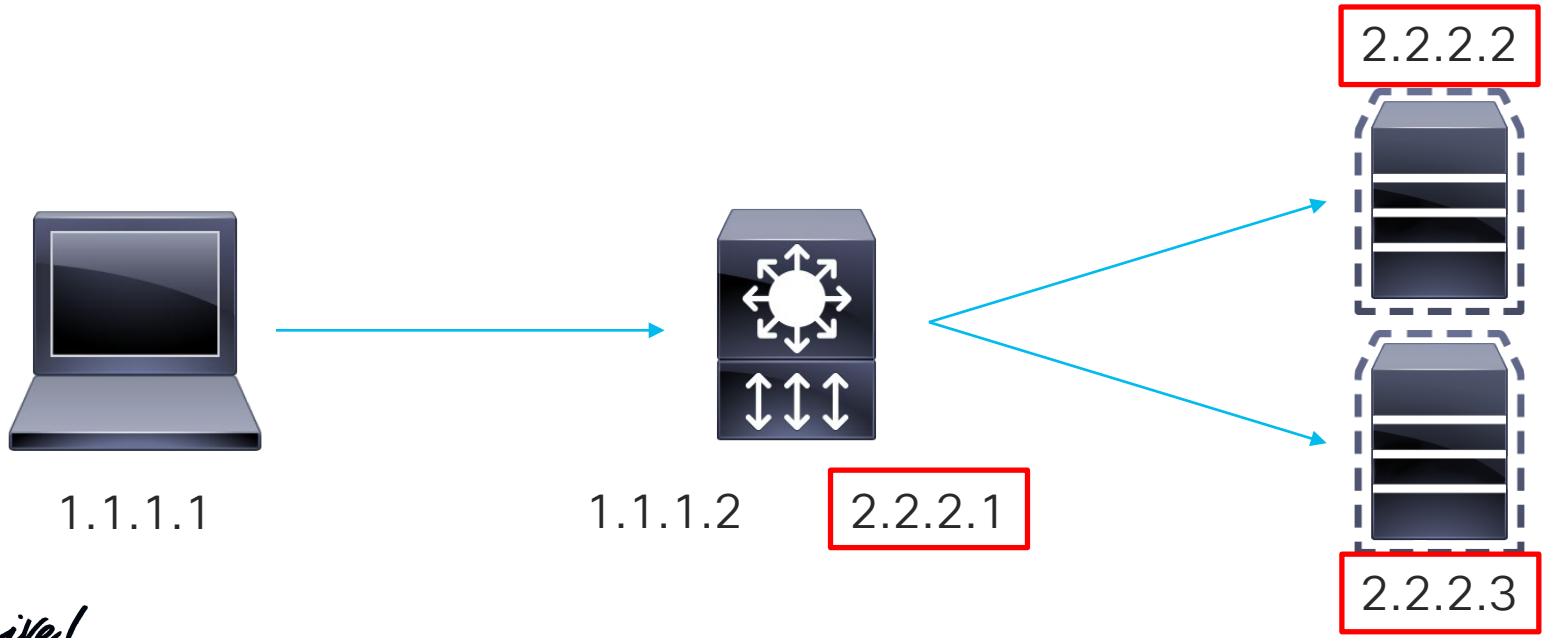
Oops, we forgot the Load Balancers?

- Without visibility at the SLB level, what does the server see?



Oops, we forgot the Load Balancers?

- Is our goal to define 2.2.2.1 -> 2.2.2.2/3 or…
- Is it to define 1.1.1.1 -> 2.2.2.2/3?

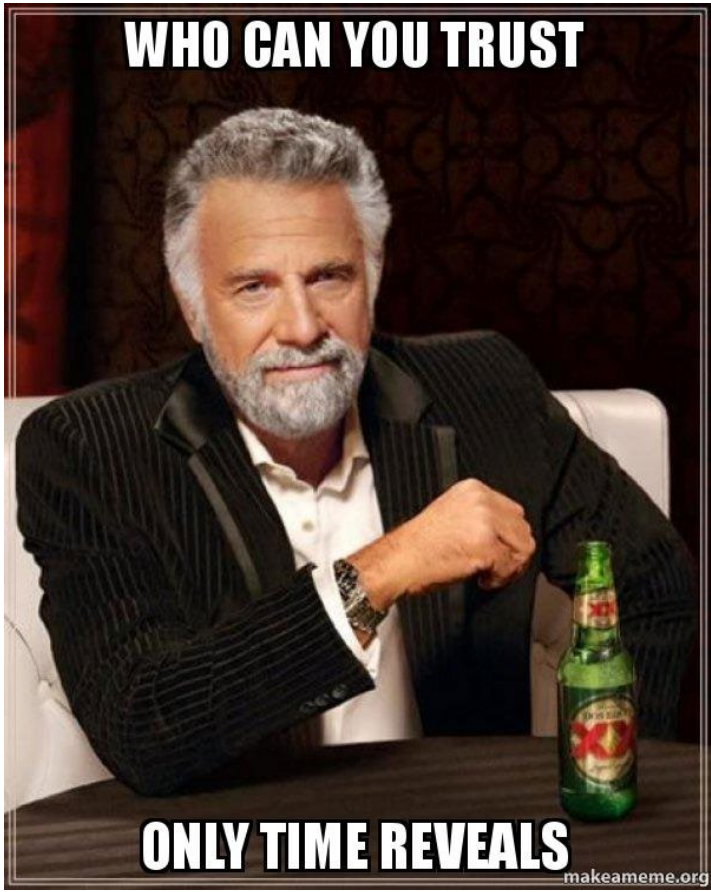


Now the Complete **fully correlated** Picture with LB

Flow Data	IP: 1.2.3.4 Src Port: 45862 Protocol: TCP Flags: SYN, ACK		IP: 4.3.2.1 Dst Port: 80 Protocol: TCP Flags: SYN, ACK
Process Data	Socket: TCP/45862 State: ESTABLISHED Process: Chrome Hash: 0xabcdef01		Socket: TCP/80 State: LISTEN Process: HTTPD Hash: 0xabcdef02
Context Data	AnyConnect User: remi ISE Posture: Trusted Location: Paris, FR AD Group: Engineering	VIP: 1.1.1.2 Pool: Servers	App: Intranet Location: SJC Data: Internal ESX: host-1

Which Context Data to Use?

- Some data is pulled at the source and can be trusted
  - Flow Data
- Data you use depends on who you trust
- At the end of the day, your segmentation will rely on these tags
  - Do you trust the kube guy to set his tags?
  - Trust the load balancer team?
  - Trust the CMDB data (quality)?
- One thing you should NOT trust
  - Excel files… Here's why…



Which Context Data to Use?

- Some data is pulled at the source and can be trusted
  - Flow Data
- D
- A
  - Do you trust the kube guy to set his tags?
  - Trust the load balancer team?
  - Trust the CMDB data (quality)?
- One thing you should NOT trust
  - Excel files… Here's why…

IP	Application	City	Country	Line of Busin	Location	Region	TA_bogon_ip	TA_zeus
0.0.0.0/2			France		MyDC	NA		
0.0.0.0/8							TRUE	

Let’s Tag!
Demo

Ok, where are we at?

- At this point, we have:
  - Communication Data (flows)
  - Annotated with Machine Data (process)
  - Combined with Context (cmdb, orchestrator)
- We now need to build up a policy

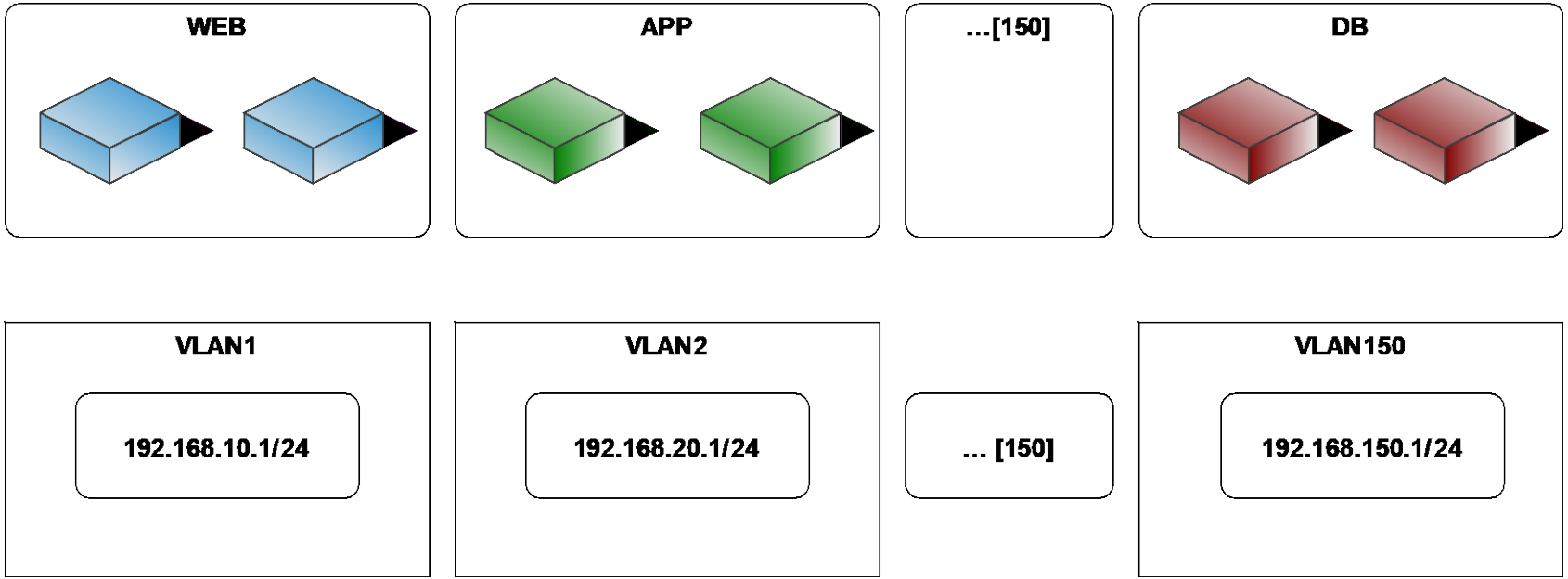
Policy Use Cases

Four common segmentation scenarios

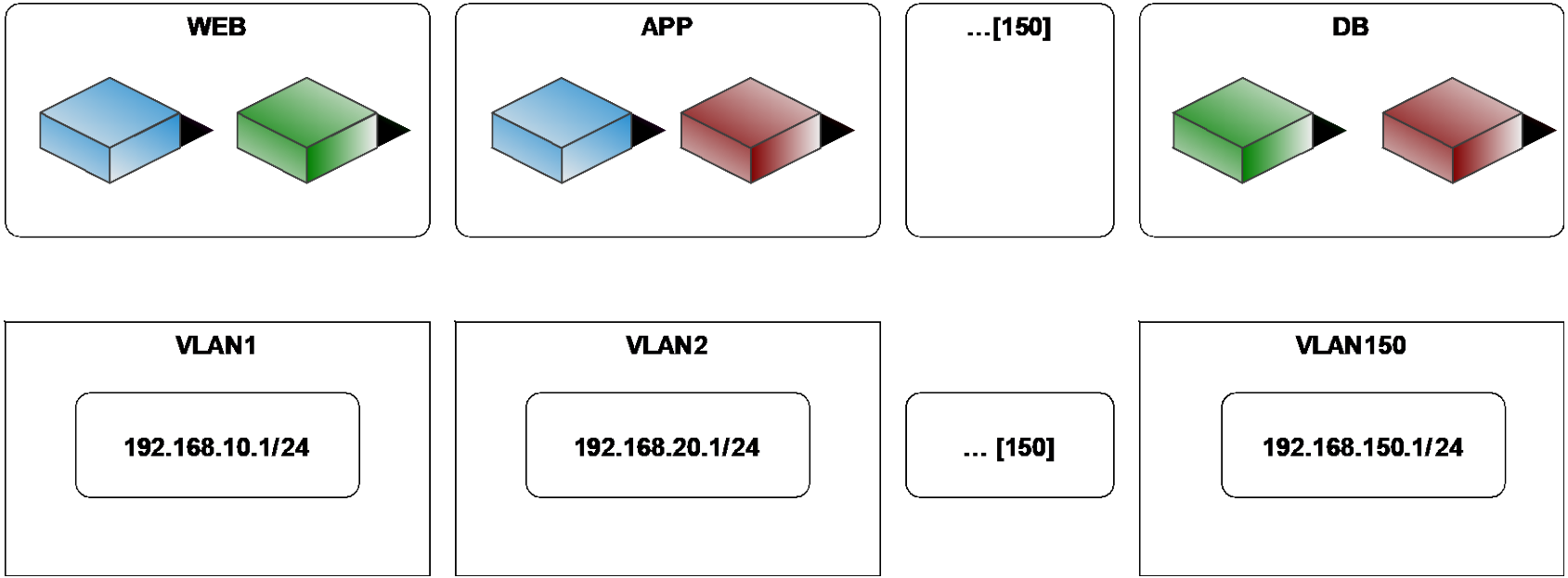
… a demonstration of

- Workloads in the same VLAN
- Workloads across VLANs
- Workloads in the cloud
- Workloads across an on-prem network and a cloud network

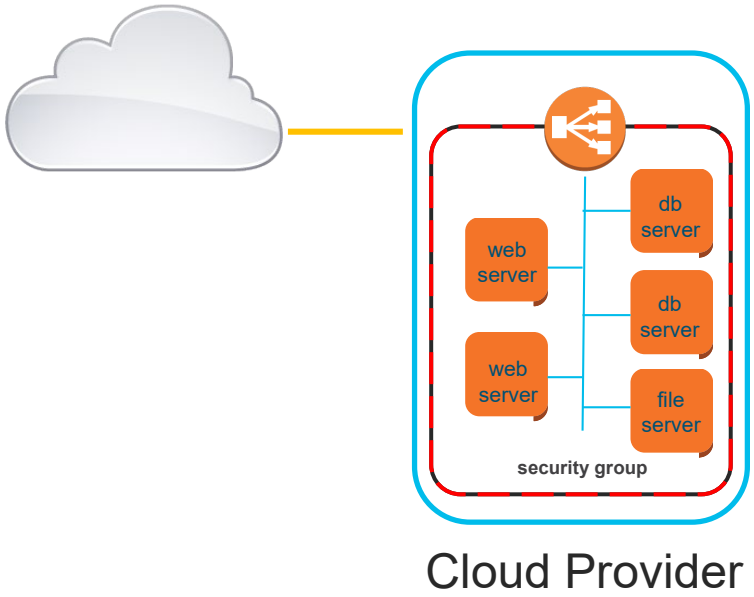
Demo: Workloads within the same VLAN



Demo: Workloads across VLANs



Demo: Workloads in the cloud



In Summary

...And Secure!

What did we achieve?

- We started with an environment running existing “applications” in production
- The environment spanned on-premises and cloud networks
- We discussed the complexity of attempting to enforce policy traditionally
- Host based policy enforcement was proposed as the solution to our problem
- We looked at the way Tetration can help us gather the data to build policy
- Using Tetration, we built our segmentation policies for all applications, no matter where they were running or how they connected to the network
- We tested the segmentation policy to ensure we would would not damage production traffic
- With one click we enforced our segmentations rules globally
- Great success!

Complete your online session survey

- Please complete your session survey after each session. Your feedback is very important.
- Complete a minimum of 4 session surveys and the Overall Conference survey (starting on Thursday) to receive your Cisco Live t-shirt.
- All surveys can be taken in the Cisco Events Mobile App or by logging in to the Content Catalog on ciscolive.com/emea.

Cisco Live sessions will be available for viewing on demand after the event at ciscolive.com.

cisco Live!

Continue your education

Demos in the Cisco Showcase

Walk-In Labs

Meet the Engineer 1:1 meetings

Related sessions

Thank you

You make **possible**