Zero Trust Architecture (Networking)

The search for truth and observability

Greg Maples
Principal Security Architect
greg.maples@gigamon.com

Gigamon®

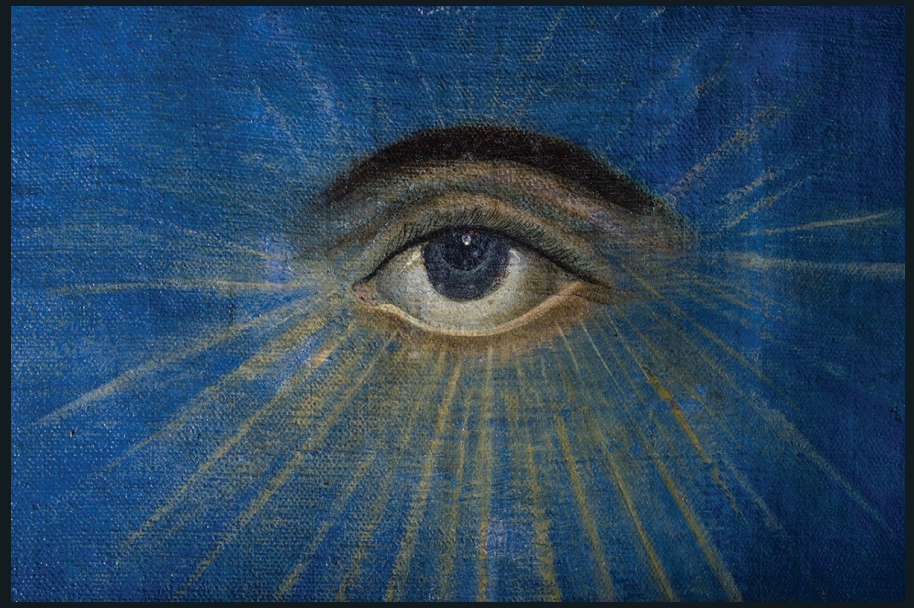
Agenda

+ Introduction
+ Objective Truth
+ Sources of Data – TAP vs SPAN
+ OBS - Comparison and Differentiation
+ Wrap Up
+ Q&A

What is Zero Trust Architecture?

A very brief history

- John Kindervag, an analyst at Forrester, coined the term "zero trust“ in 2010. His assertion was that an organization should not inherently trust anything outside *or inside* its perimeter footprint. He suggested that you must always verify everything that tries to connect to your network.[1] A zero trust mindset posits that there is no ‘trusted network’, all networks and traffic are untrusted no matter the origin.
- NIST adopted a ZTA standard in 2020
  https://www.nist.gov/publications/zero-trust-architecture
- DOD formally adopted a ZTA standard in 2021
  https://dodcio.defense.gov/Portals/0/Documents/Library/(U)ZT_RA_v1.1(U)_Mar21.pdf
- The impact of ZTA on industry has been interesting. Virtually all organizations embrace the *idea* of ZTA, but implementation lags because of costs, legacy infrastructure, and traditional infrastructure conservatism.



1. https://en.wikipedia.org/wiki/Zero_trust_security_model

What Do We Mean by

Truth?

What is Objective Reality?

The search for ultimate truth – Unanswered in human history

- It is impossible to establish an external reference framework for the measurement of 'reality'. All measurement exists within the framework of the system of measurement.
- Philosophy has examined this since the beginning. https://www.oxford-royale.com/articles/4-debates-in-philosophy-everyone-should-know-about/
- It is important to understand that the data being measured or tested is only as reliable as the method of measurement is reliable.
- Why this PHIL-101 discussion matters is that as a practitioner, you must understand that you will eventually have to place trust – somewhere – the choice is being naïve about it or not

How deep does the rabbit hole go?

Where can you find truth?

- Software – Obvious Problems
- BIOS Bootloader – Compromises are common
  https://www.schneier.com/blog/archives/2015/03/bios_hacking.html
- Firmware – Compromised by state actors
  https://www.wired.com/2015/02/kapersky-discovers-equation-group/
- CPU Instruction Set – “god” instructions
  https://www.eejournal.com/article/clever-hack-finds-mystery-cpu-instructions/
  (Did NSA hack chip designs? Hint: Yes)

What's at the bottom of the rabbit hole?

Gödel, Escher, Bach: An Eternal Golden Braid

- Provability:
  - In order to have software 'proven', there must be a provability algorithm.
    This algorithm is either automated or manual.
    If it is automated, see initial point above.
    If it is manual, then it is subject to human mistakes.
- Incompleteness[1]: (Per Stanford Reference)
  - Gödel's two incompleteness theorems are among the most important results in modern logic, and have deep implications for various issues. They concern the limits of provability in formal axiomatic theories.
    The first incompleteness theorem states that in any consistent formal system FF within which a certain amount of arithmetic can be carried out, there are statements of the language of FF which can neither be proved nor disproved in FF.
    According to the second incompleteness theorem, such a formal system cannot prove that the system itself is consistent

https://plato.stanford.edu/entries/goedel-incompleteness/
https://www.schneier.com/blog/archives/2015/03/bios_hacking.html

Nothing Can Be Trusted!

In Software, there is *no* objective reality to compare against

- Any facility can be corrupted
- Hashing and checksums can be altered/replaced.
  Names, Sizes, Signatures can be compromised.
  Validations can be short-circuited.
- The tools used to check on things can be compromised
  - ‘top’ and ‘ps’ hacked to not show processes
  - Memory and network tools replaced
  - How good are you at reading ‘od’?

Example: Syslog can't necessarily be trusted

Remember, owned is owned

Before:

```
*.info;mail.none;authpriv.none;cron.none        /var/log/messages
authpriv.*                                      /var/log/secure
mail.*                                          -/var/log/maillog
cron.*                                          /var/log/cron
*.emerg                                         *
uucp,news.crit                                  /var/log/spooler
local7.*                                        /var/log/boot.log
```

After:

```
auth.*                                          |/var/adm/syslog.pipe
authpriv.*                                      |/var/adm/authlog.pipe
*.info;mail.none;cron.none                      /var/log/messages
mail.*                                          -/var/log/maillog
cron.*                                          /var/log/cron
*.emerg                                         *
uucp,news.crit                                  /var/log/spooler
local7.*                                        /var/log/boot.log
```

Even backups can't necessarily be trusted

Remember, owned is owned

- How to corrupt backups
  - Force a pipe into the device node path for reading or writing
  - Replace the named binary of the backup
  - Alter the backup config file
  - Suppress syslog and warning messages
- *Then* corrupt the testing/reporting process
  - *Yes, everything is fine and working*
  - *Yes, all the backups are good*
- *Then* encrypt the valid backups and rename them

Places that practitioners place implicit trust

You are placing trust, even if you don't know it

- Firewall / NGFW / Policy software
  Is this software externally validated by an independent international commission?
  Of course not.
- Routing/Switching fabric software & "hardware" (actually firmware)
  When was the last time that your switch OS was audited and validated by hand?
  Never.
  - What if the OC192 module you installed was compromised?
  - How would you know?
- Patches
  Who wrote the patch? Who validated it? Where was it regressed?

The cloud is even worse

If AWS was copying your data…

- Once you're on a cloud deployment, you have zero access to the underlaying environment.
  - You have to trust the vendor
  - You assume that data theft is not occurring
  - You assume that no government agency has gotten a warrant for your data
- Even just basic virtualization eliminates layers of visibility

So, what now?

Where do you choose to trust?

- If you can't trust software, firmware, syslog, backups, UEFI, patches, or anything else… then where do you begin to establish a baseline of trust in your network?
- Begin at the bottom – Physics
  Although you can argue that even physics and test measurement can be hacked, you have to start somewhere. The best place is at the bottom.
  - Electricity
  - Signals
  - Photons
- If your strategy begins anywhere else, you're making assumptions that may bite back

What Do We Mean by

Visibility?

“Truth” in deployments

Data will transit

- Transit is your key:
  - The days of ‘sneaker net’ are long gone, all data will be in transit at some point
  - Data in transit can be observed
  - Remote access can be observed
  - Remote control can be observed
  - Exfiltration can be observed
- But you must understand and prepare for intercept
  - Architecture and TAP/SPAN

Once you have access to the network…

You can begin to establish *visibility* into your data

Visibility

/vɪzɪˈbɪlɪti/
noun

The ability to collect packets from any location in the network (physical, private or public cloud), aggregate them, transform them so that they improve the effectiveness of the attached packet-consuming tool(s), and feed them to those tool(s) to maximize monitoring efficiency, cost-effectiveness and reliability.

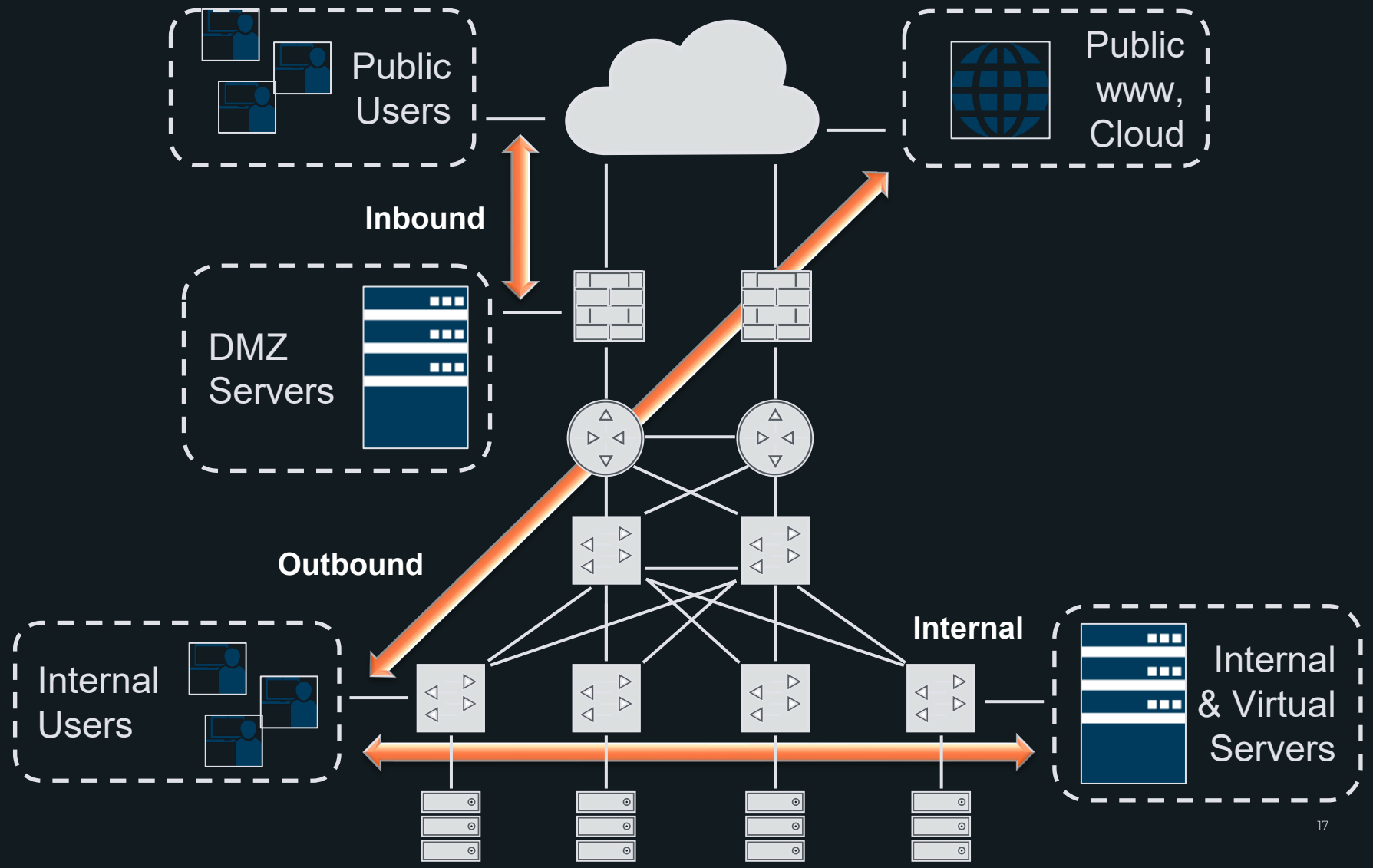
Traffic Visibility Architecture

Where will you place inspection points?

- In front of FW?
- In front of core switching?

What will you pay for access?

- Can you afford to inspect E/W traffic?
- Can you inspect DC to DC or cloud to DC?



Here's what 800-207 has to say

Exactly what you'd think… You have to see the traffic

3.4 Network/Environment Components

In a ZT environment, there should be a separation (logical or possibly physical) of the communication flows used to control and configure the network and application/service communication flows used to perform the actual work of the organization. This is often broken down to a control plane for network control communication and a data plane for application/service communication flows [Gilman]. The control plane is used by various infrastructure components (both enterprise-owned and from service providers) to maintain and configure assets; judge, grant, or deny access to resources; and perform any necessary operations to set up communication paths between resources. The data plane is used for actual communication between software components. This communication channel may not be possible before the paths have been established via the control plane. For example, the control plane could be used by the PA and PEP to set up the communication path between the subject and the enterprise resource. The application/service workload would then use the data plane path that was established.

3.4.1 Network Requirements to Support ZTA

1. Enterprise assets have basic network connectivity. The local area network (LAN) enterprise controlled or not, provides basic routing and infrastructure (e.g., DNS). The remote enterprise asset may not necessarily use all infrastructure services.
2. The enterprise must be able to distinguish between what assets are owned or managed by the enterprise and the devices' current security posture. This is determined by enterprise-issued credentials and not using information that cannot be authenticated information (e.g., network MAC addresses that can be spoofed).
3. The enterprise can observe all network traffic. The enterprise records packets seen on the data plane, even if it is not able to perform application layer inspection (i.e., OSI layer 7) on all packets. The enterprise filters out metadata about the connection (e.g., destination, time, device identity) to dynamically update policies and inform the PE as it evaluates access requests.
4. Enterprise resources should not be reachable without accessing a PEP. Enterprise resources do not accept arbitrary incoming connections from the Internet. Resources accept custom-configured connections only after a client has been authenticated and authorized. These communication paths are set up by the PEP. Resources may not even be discoverable without accessing a PEP. This prevents attackers from identifying targets via scanning and/or launching DoS attacks against resources located behind PEPs. Note that not all resources should be hidden this way; some network infrastructure components (e.g., DNS servers) must be accessible.
5. The data plane and control plane are logically separate. The policy engine, policy administrator, and PEPs communicate on a network that is logically separate and not directly accessible by enterprise assets and requests. The data plane is used for application/service data traffic. The policy engine, policy administrator, and PEPs use the control plane to communicate and manage communication paths between assets. The PEPs must be able to send and receive messages from both the data and control planes.
6. Enterprise assets can reach the PEP component. Enterprise subjects must be able to access the PEP component to gain access to resources. This could take the form of a web portal, network device, or software agent on the enterprise asset that enables the connection.
7. The PEP is the only component that accesses the policy administrator as part of a business flow. Each PEP operating on the enterprise network has a connection to the policy administrator to establish communication paths from clients to resources. All enterprise business process traffic passes through one or more PEPs.
8. Remote enterprise assets should be able to access enterprise resources without needing to traverse enterprise network infrastructure first. For example, a remote subject should not be required to use a VPN back to the enterprise network (i.e., virtual private network [VPN]) to access services utilized by the enterprise and hosted by a public cloud provider (e.g., email).
9. The infrastructure used to support the ZTA access decision process should be made scalable to account for changes in process load. The PE(s), PA(s), and PEPs used in a ZTA become the key components in any business process. Delay or inability to reach a PEP (or inability of the PEPs to reach the PA/PE) negatively impacts the ability to perform the workflow. An enterprise implementing a ZTA needs to provision the components for the expected workload or be able to rapidly scale the infrastructure to handle increased usage when needed.
10. Enterprise assets may not be able to reach certain PEPs due to policy or observable factors. For example, there may be a policy stating that mobile assets may not be able to reach certain resources if the requesting asset is located outside of the enterprise's home country. These factors could be based on location (geolocation or network location), device type, or other criteria.

Key Points:

- Separate Management traffic from Production
- Use management plane to define production architecture
- You must be able to identify the devices on your network and who controls them, and base the trust granted on that information
- The enterprise can observe (see into) all network traffic.
- You record packets seen on the data plane
- even if you can't perform application layer inspection on all packets.
- You gather metadata about the connection to dynamically update traffic policies and log access tests.
- There are no ways into the network that bypass policy and access enforcement
- Your assets should be able to access things like email without having to needlessly transit your network. For example, forcing remote email hosted in the cloud to transit your VPN first.

Digging a little deeper on 3.4.1

Some Implications

Thinking Points:

- Separate management plane – *normal practice*
- Software defined network architecture – *expensive and difficult to implement*
- Device identification and access control – *many possible vendors*

- *You can see into all network traffic.*
  You record packets seen on the data plane,
  even if you can't perform application layer inspection on all packets.

- You gather metadata about the connection to dynamically update traffic policies and log access requests. – *This is normal best practice (Splunk, for example)*
- There are no ways into the network that bypass policy and access enforcement - *Obviously*
- Your assets should be able to access things like email without having to needlessly transit your network. For example, forcing remote email hosted in the cloud to transit your VPN first.

Inspection points

Inbound Services

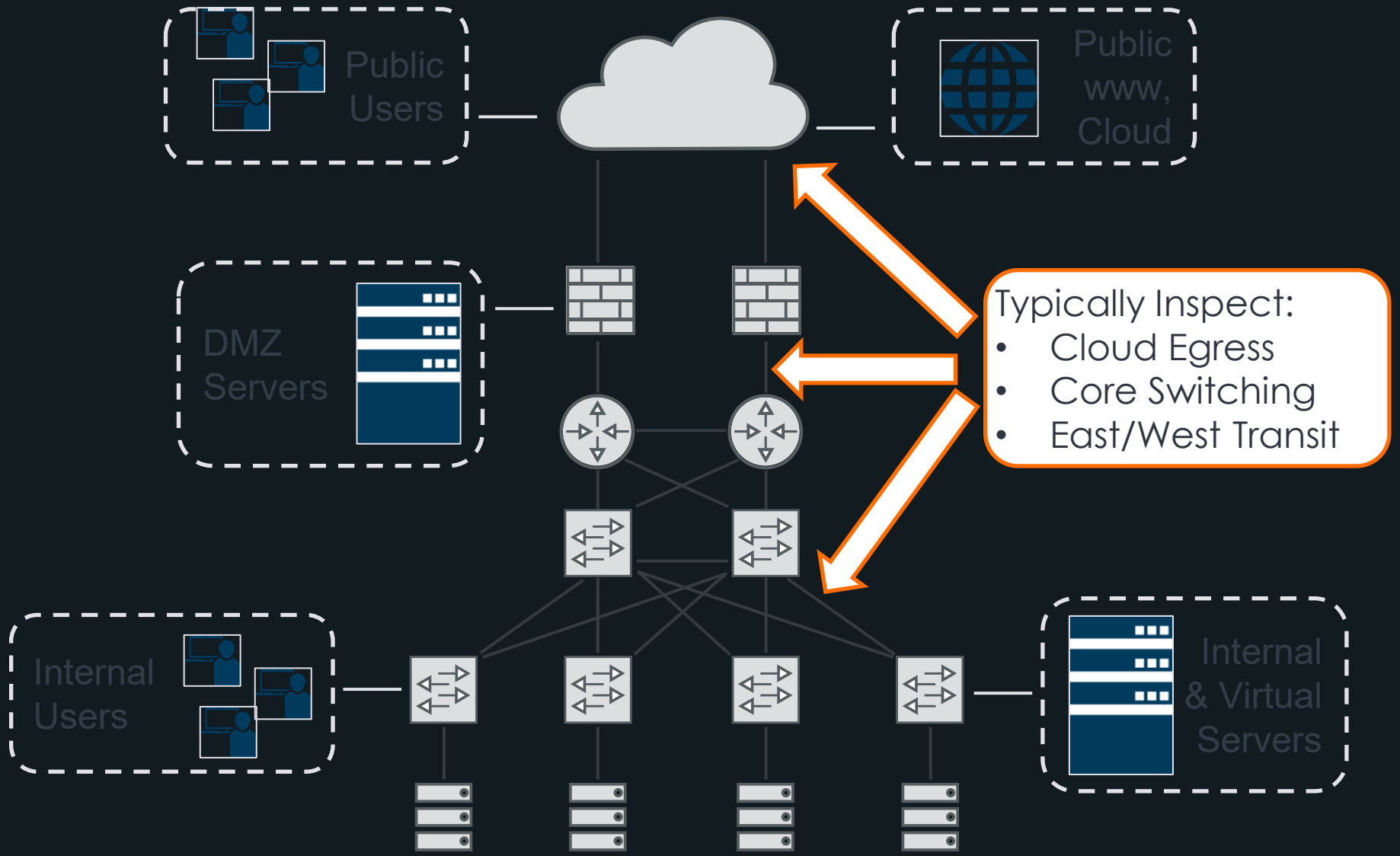
- Public facing web servers
- Decryption Zone

Outbound Services

- Enterprise user traffic to Internet
- Content Inspection

Internal Service

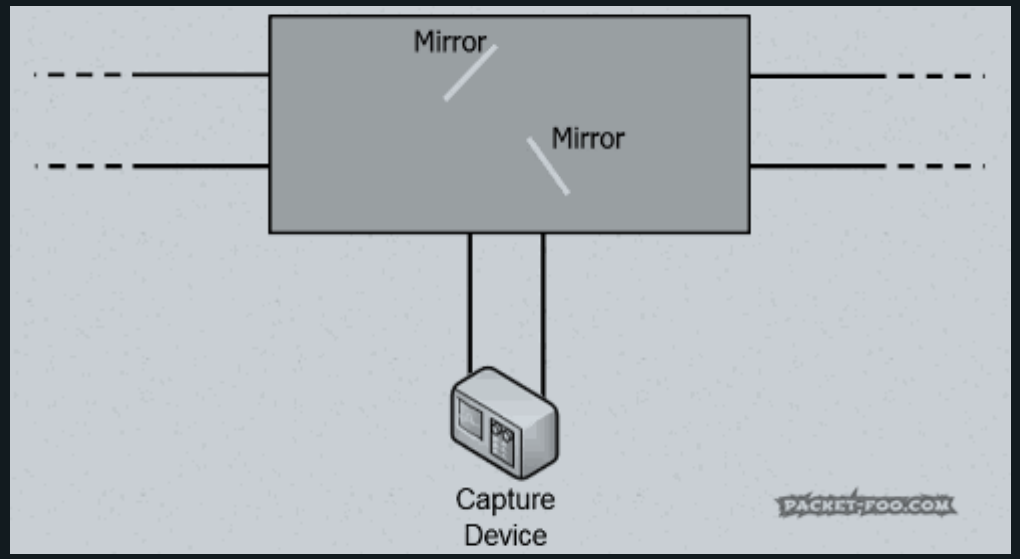
- User traffic to internal servers
- East/West Inspection



How to inspect?

Well, we have a recommendation…

+ TAP vs SPAN
  - ‘SPAN port’ – out of scope for this discussion, essentially a software created copy of traffic
  - If you use a SPAN, it is often less expensive, except for the toll on your switch CPU
  - SPANs are often acceptable in lesser risk environment
+ A TAP is different
  - Optics don’t lie
  - Optical TAPs are passive – can’t be discovered unless someone is measuring the light
    (And if they can do that, you have big problems)



SPAN -Switch Port Analyzer

ZTA Visibility Requirements

All Traffic, even if you can't decrypt

5.4 Visibility on the Network

As mentioned in Section 3.4.1, all traffic is inspected and logged on the network and analyzed to identify and react to potential attacks against the enterprise. However, as also mentioned, some (possibly the majority) of the traffic on the enterprise network may be opaque to layer 3 network analysis tools. This traffic may originate from non enterprise-owned assets (e.g., contracted services that use the enterprise infrastructure to access the internet) or applications/services that are resistant to passive monitoring. The enterprise that cannot perform deep packet inspection or examine the encrypted traffic and must use other methods to assess a possible attacker on the network. That does not mean that the enterprise is unable to analyze encrypted traffic that it sees on the network. The enterprise can collect metadata (e.g., source and destination addresses, etc.) about the encrypted traffic and use that to detect an active attacker or possible malware communicating on the network. Machine learning techniques [Anderson] can be used to analyze traffic that cannot be decrypted and examined. Employing this type of machine learning would allow the enterprise to categorize traffic as valid or possibly malicious and subject to remediation.

You may not be able to see into every packet, but you must at least SEE each one!

- Decrypt if you can
- Record metadata at the very least
- Pay close attention to traffic you don't understand

Notes from NIST CSWP 20

"Planning for a Zero Trust Architecture: A Planning Guide for Federal Administrators" – May 6, 2022

1.1.3 Tenets that Apply to Data Flows

I. **All communication is secured regardless of network location.**
In zero trust, the network is always considered contested. A ZTA should be designed with the assumption that an attacker is present on the network and could observe/modify communications. […]

II. **Access to individual enterprise resources is granted on a per-session basis.**
[...]

III. **Access to resources is determined by dynamic policy—including the observable state of client identity, application/service, and the requesting asset—and may include other behavioral and environmental attributes.** […]

IV. **The enterprise collects as much information as possible about the current state of assets, network infrastructure and communications and uses it to improve its security posture.** […] This requires the enterprise to monitor all traffic to the extent feasible and restricted (or required) by policy, regulation or legal requirement. […]

So then, what is

Observability?

Observability

Comes from **Control Theory** (19th century, deriving from the work of Maxwell)

- Understanding dynamic systems (originally steam-powered machines) to enter a desired state, based on input, while minimizing undesired outcomes like overshoot, delay or instability.
- Feedback loops are fundamental to control theory.
- Observability is the ability to figure out a system's internal state from its external inputs and outputs.
- Monitoring is possible if you have observability.
- Visibility, as we define it, is observability using the collection of network packets which are used to derive knowledge of the internal state of a monitored system.
- Example: determining that a server is running malware from visibility (and thus observability) of the Command and Control (C2) channel to the malware's threat actor
- In cloud environments, organizations are looking to other techniques to give observability

What Could Observability Look At?

+ There are many types of "outputs" one can observe
+ Below is a partial list of some of the types of data which can be used for security analytics

+ Network traffic delivered as raw packets
+ Summarized network traffic (e.g., NetFlow)
+ Session or content-derived metadata
+ Events and logging information transmitted via network protocols
+ Log files stored on systems or in databases
+ Polled statistics (e.g., SNMP)
+ Environmental data and telemetry
+ Asset, threats and vulnerability data
+ Human-sourced data (e.g. suspicious activity reports, non-compliance reports)
+ Probably many more….

The "Three Pillars of Observability"(1) in Distributed Systems

Event Logs

- Generated by a system in response to an event
  - Plaintext (free form)
  - Structured (e.g. JSON)
  - Binary (e.g. binlogs)
- **Pros**: simple, well supported, good ecosystem
- **Cons**: system has to log the event, can cause performance issues, too granular, poor at dealing with cascade failures
- **Ecosystem**: Kafka, SIEM

Metrics

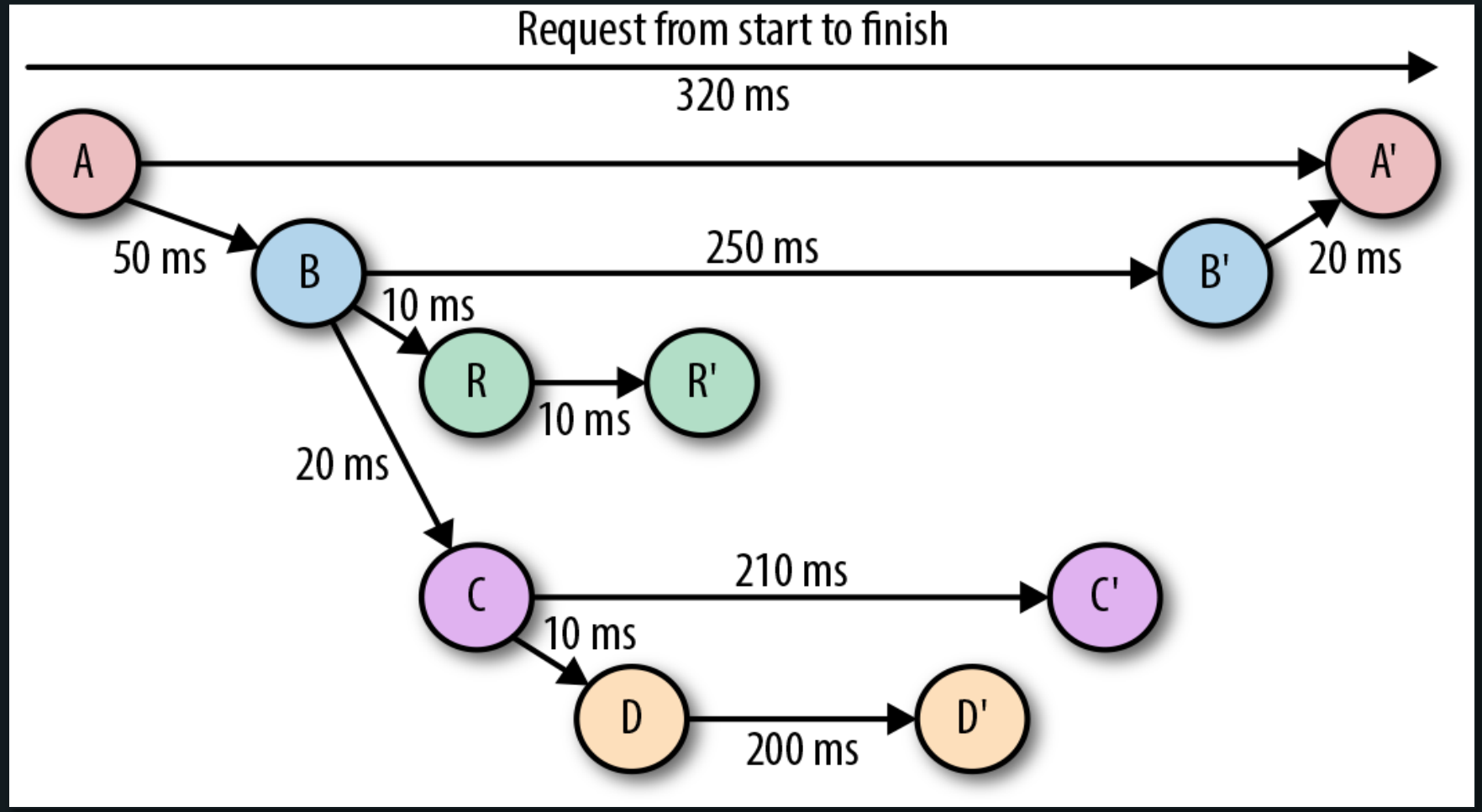
- A numeric representation of data over time
- Usually a name/value pair
- **Pros**: extremely powerful for mathematical, statistical and problabilistic modelling; generally, storage efficient; great for alerting
- **Cons**: harder to search on, requires the system to provide the metric (system scoped)
- **Ecosystems**: Graphite, Prometheus, ElasticSearch, SNMP

Distributed Tracing

- As a request flows through a distributed system, boundaries are identified (e.g. RPC calls, applications, proxies, frameworks) and a timestamp is generated as these flow through the system.
- **Pros**: very powerful for identifying problems in a distributed system, interservice dependency analysis, capacity planning
- **Cons**: challenging to do, instrumenting your code isn't enough (libraries and proprietary code)
- **Ecosystem**: Zipkin, Virsec and Jaegar

(1) Distributed Systems Observability by Cindy Sridharan, O'Reilly Press

Deep Diving Into Distributed Tracing



(1) Distributed Systems Observability by Cindy Sridharan, O'Reilly Press

Observability vs. Visibility

Event/Metric/Tracing vs. Visibility

- Where does our definition of visibility fit?
- Tools attached to a visibility node treat network nodes (servers, clients, containers, VMs etc.) as "opaque" observability systems
- Tools observe events (network transactions) and gather metrics, derived from network traffic
- No visibility of tracing inside those network nodes
  - Arguably, some can consume logging and SNMP to give limited internal visibility
  - Others may deploy agents to supplement network visibility
- Distributed tracing is the opposite (complement) of network visibility/observability

Challenges of the Cloud Environment

Traditional Workloads

- Typically centralized, monolithic
- Static, tightly change controlled
- "Pets"
- Capacity sized to expected maximum loads
- Well established architectures and processes
- Security is core
- Waterfall development
- Hardware

Cloud Workloads

- Typically distributed
- Dynamic and ephemeral, FaaS
- "Livestock"
- Elastic so that services come and go on demand
- Developing architectures and processes
- Security is best effort mixed with some denial/hubris
- Agile moving to CI/CD
- Hypervisor/Container/Microservices

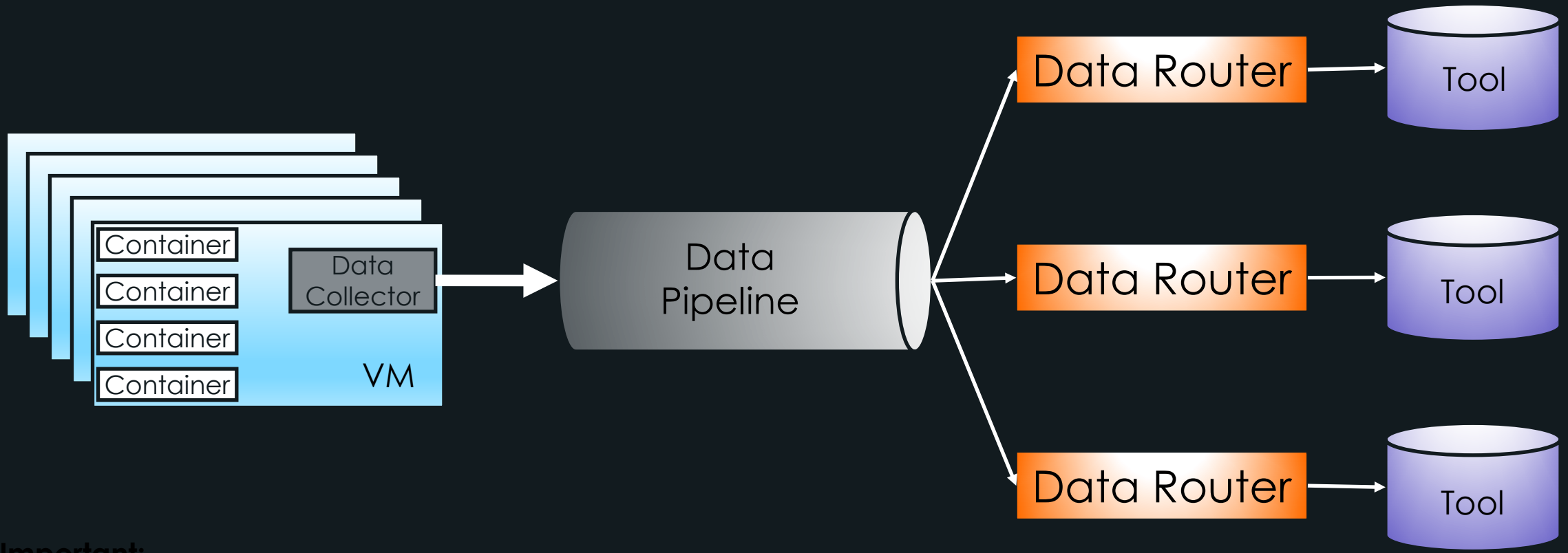
A Small Digression: Challenges of Cloud Security

- Lack of defined cloud security architectures (esp. following “lift and shift”).
- Poor identity, credential and cryptographic key management.
- Hybrid deployments can sometimes end up “worst of all worlds” (almost all deployments are hybrid).
- Misconfiguration and malicious insider threat.
- Elastic and ephemeral workloads, sometimes deployed at large scale.
- Vulnerabilities in infrastructure and services.
- Legal and regulatory compliance (including data sovereignty and residency).
- Different controls across different Cloud Service Providers (or accept the risk of a single cloud strategy).
- Infosec skills are already in short supply – Infosec+Cloud is even harder to hire.

So, what is an

OBSERVABILITY PIPELINE?

Model: Data Collection – Pathing – Tool Distribution



Important:
This diagram is descriptive, actual architectural implementation varies by vendor

In Summary

Summary

"Give me a lever long enough and a fulcrum on which to place it, and I shall move the world. "
— **Archimedes**

Truth

The closest you can come to objective measurement – Physics

Access

Measure/Copy data in transit, not at rest

Visibility

Be certain to look at all the important transit points

Observability

Build a system that has designed-in feedback loops

Zero Trust Architecture (Networking)

Thank You!

Greg Maples
Principal Security Architect
greg.maples@gigamon.com

Gigamon®