Enabling Zero Trust

The Art of Healthy Skepticism

Ryan Etten
Consulting Architect
retten@redhat.com

Red Hat

What We'll Discuss Today

- What is Zero Trust
- A Bit of History
- Impact on Enterprise IT
- Demystifying Zero Trust Architecture (ZTA)
- Zero Trust Policies
- Implementing ZTA (w/Considerations)
- Wrap-Up and Q&A

Zero Trust Architecture

Zero Trust is NOT “Security”

… it is an architecture that helps you achieve important security goals.

It’s not about building highway walls; it’s about building better gates.

What is Zero Trust?

It’s assuming everything is independently and always exposed to all potential threats

More walls, more gateways: A common paradigm is the idea of moving from a single network perimeter to a layered defense with single, carefully controlled gateways

- We make decisions only when we need to
- We make decisions based on threats and risks
- We make decisions based on “need to access”
- We make decisions based on the current, best information available
- We don’t leave any access that is not needed
- It’s the paradigm we need in our new cloud connected, BYOD, WFH, IoT world

Zero Trust: A New Architecture?

A timeline of the paradigm

2004

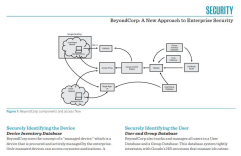
Jericho Forum

The seeds of zero trust with de-perimeterization as a goal. Lists some of the first guiding principles.

2009

"Zero Trust"

Forrester popularized the term to encapsulate early ideas around access controls and perimeter-less security



2014

BeyondCorp by Google

Lays out the first major strategy for now common ZTA approaches and remove the intranet/internet distinction

2015

A nascent ZT market

Spurred by similar goals, more enterprises seek to follow google, and vendors crop up to service these neds

2017

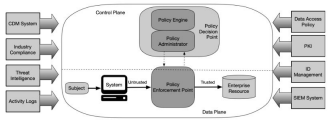
Analysts take notice

The maturing market captures attention and deeper study, and Forrester updates its ZTX ref. Arch and Gartner launches CARTA

2019

NIST SP 800-207

Seeking to provide more guidance to the USG and integrate RMF, NIST releases draft special publication



2021

Executive Order 14028

Identifies ZT as a preferred solution to today's IT security challenges and mandates its use in the USG

NIST Special Publication 800-207: Zero Trust Architecture, nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf

Why is Zero Trust Important?

- Now a mandatory requirement under Presidential EO
- Effective way to reduce data loss and prevent data breaches

Problem with Perimeter Based Security

Demystifying ZTA

- In the past, the solution to this was to implement perimeter security
- This approach assumes every user inside a network is trustworthy and should have access.
- This assumption presents at least two problems:
  - If a bad actor has network access, they can laterally move within the network
  - If an employee is not physically at work, they cannot access the network

Core Principles of ZTA

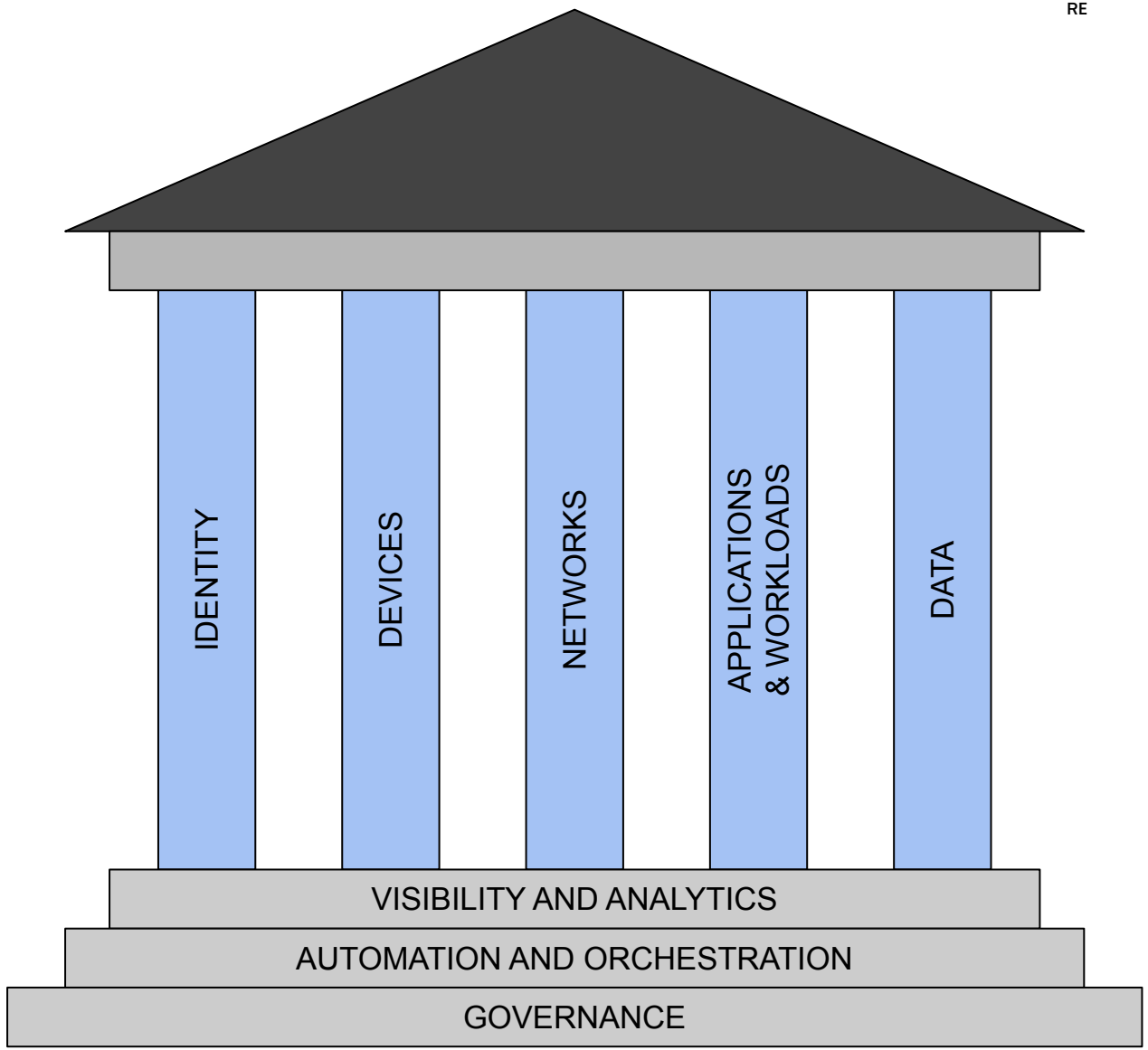
Three Fundamental Tenets to Solve Perimeter Based Problem

- **Explicit verification**
- **Least-privilege access**
- **Assume breach**

- *With no internal network, there is no longer the concept of an outside intruder or remote worker*
- *Individual-based authentication works across devices and on the application side across on-premises resources, SaaS applications, and public cloud (particularly when using an identity management solution Red Hat IdM/SSO)*

CISA Pillars

Zero Trust Maturity Model



Identities are Everywhere

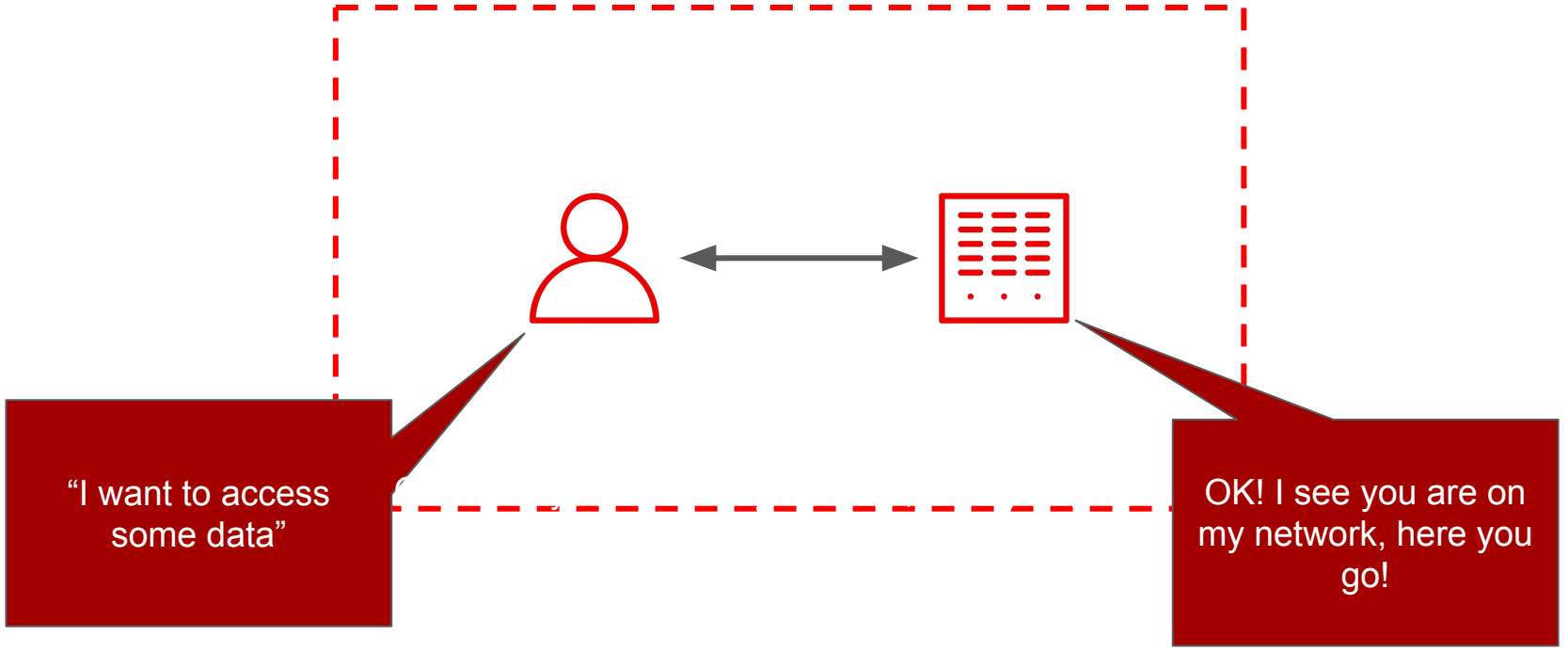
In this typical zero trust architecture, we see the layered security boundaries

Why? Zero trust is about moving access decisions closer to the data and processing

- We see every gateway is now dependent who is accessing it, and why
- Each is dependent on the identity of the actors involved
- Identity becomes one of the most crucial inputs to our decision making processes
    - We'll find we need a suite of Zero-trust control plane elements, each using identity & its attributes to weigh access decisions
- Following Zero Trust principles guides us to how to use Identities the right way

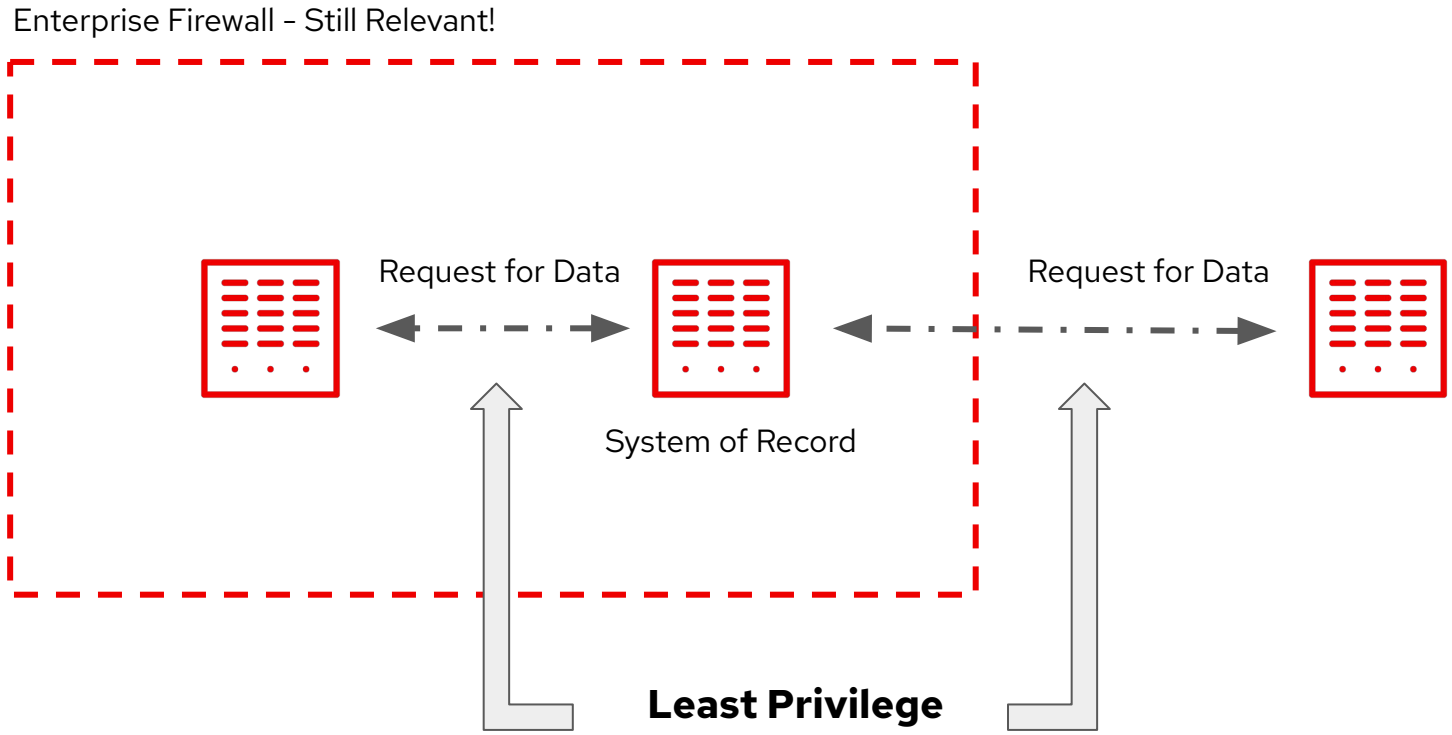
Implicit Trust Today

Users and applications are trusted because they are all inside the same boundary



Explicit Trust is the Goal (Distrust Tomorrow)

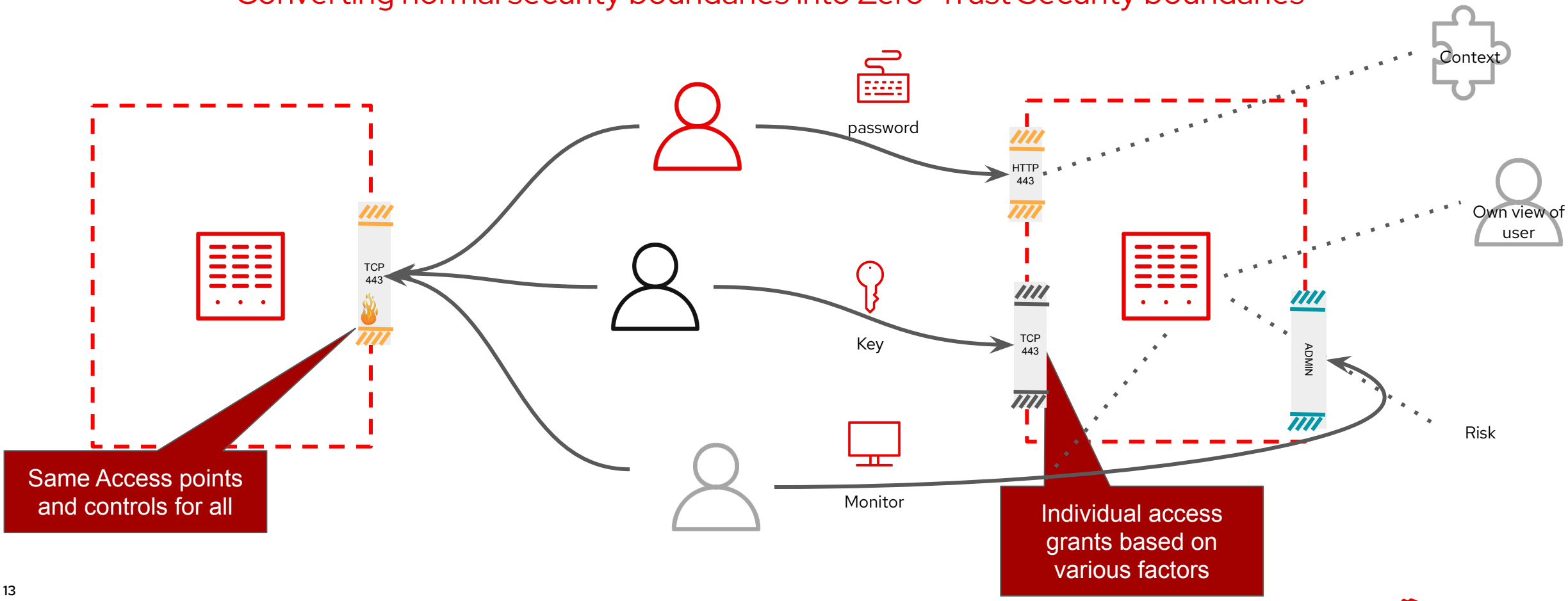
Trust is no longer implicit or by boundary -- but derived from the specifics of each transaction



Least privilege refers to the practice of restricting access to only those resources absolutely necessary—i.e. the "least" privileges necessary for an activity. In ZTAs – Each request for access to a resource needs to be **dynamically** validated, with the fullest **context** possible.

Zero Trust Boundaries

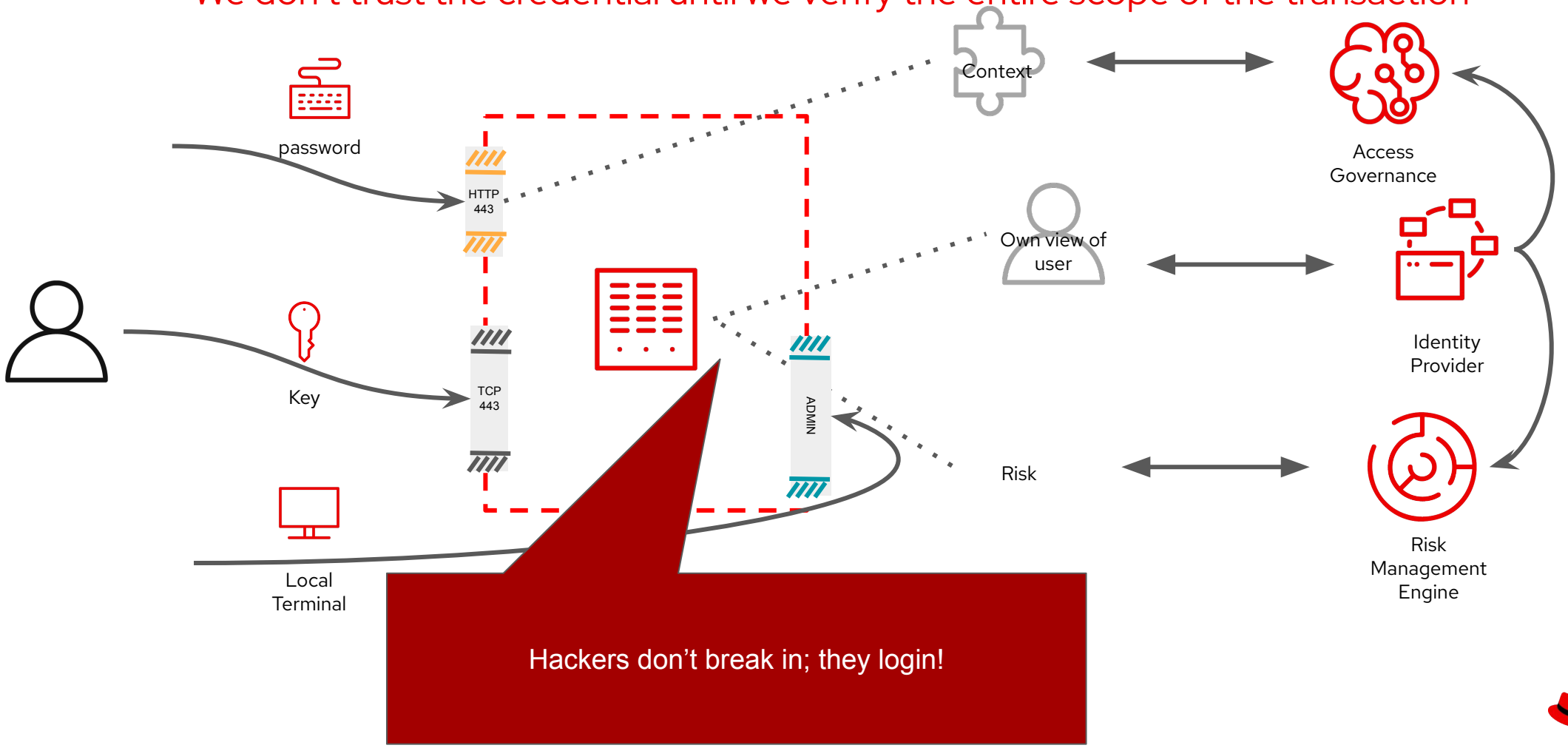
Converting normal security boundaries into Zero-Trust Security boundaries



Red Hat

Identity is key to establishing trust

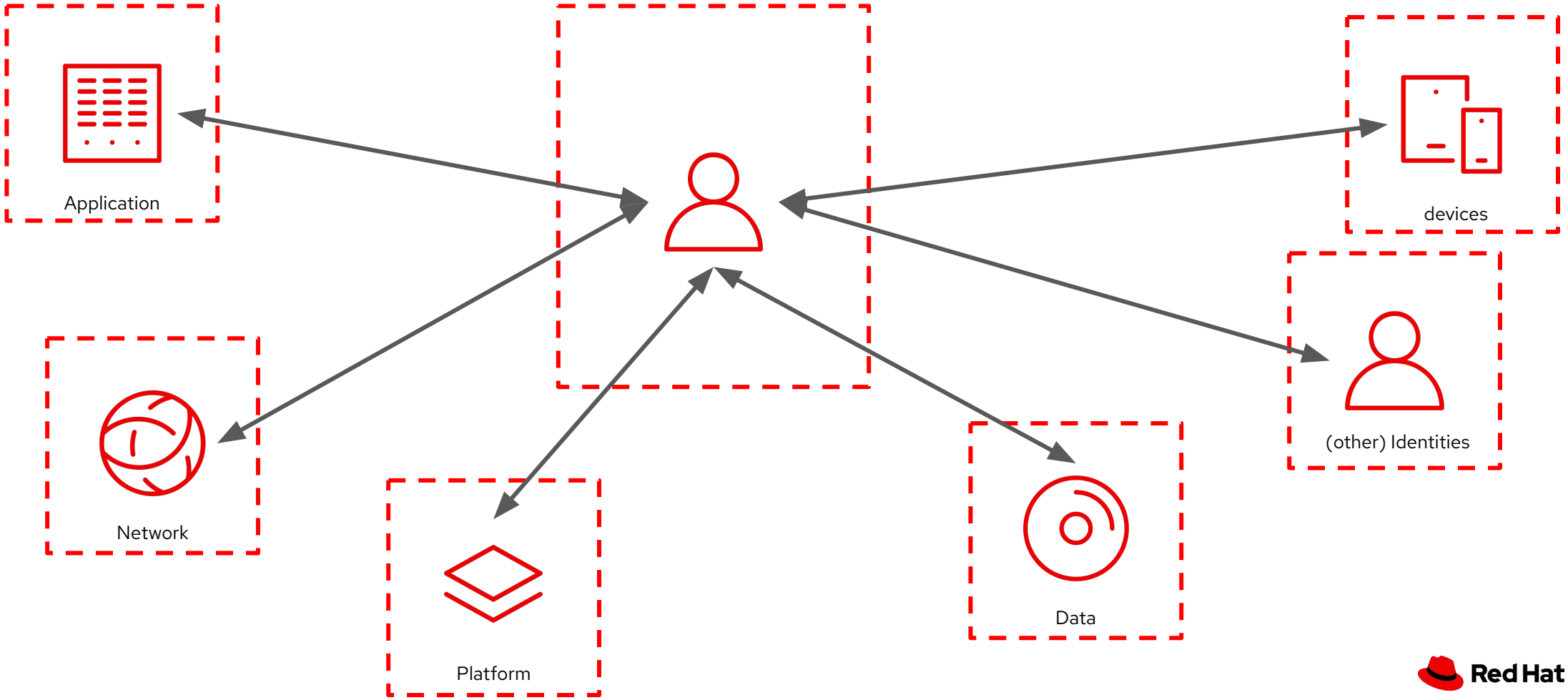
We don't trust the credential until we verify the entire scope of the transaction



Red Hat

Every Interaction, Every Resource

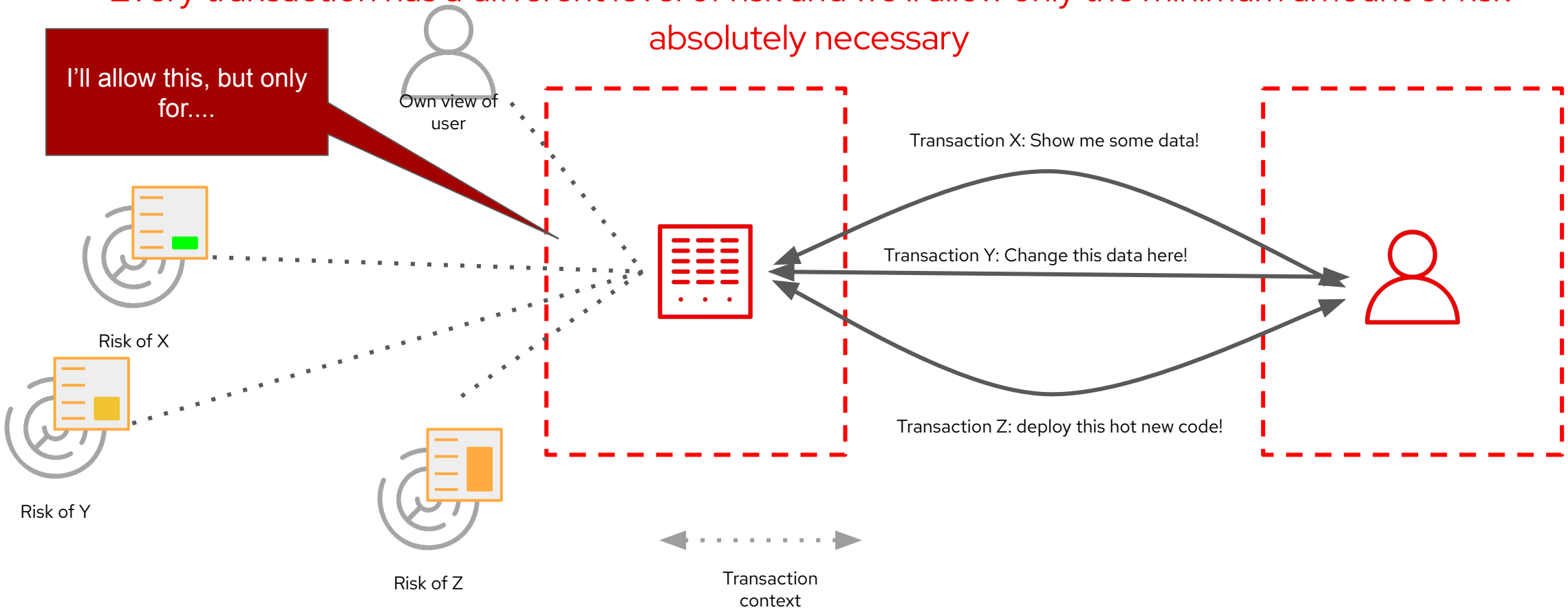
And it's being explicit about every potential trust relationship in your enterprise



Red Hat

And only to the extent necessary

Every transaction has a different level of risk and we'll allow only the minimum amount of risk absolutely necessary



Red Hat

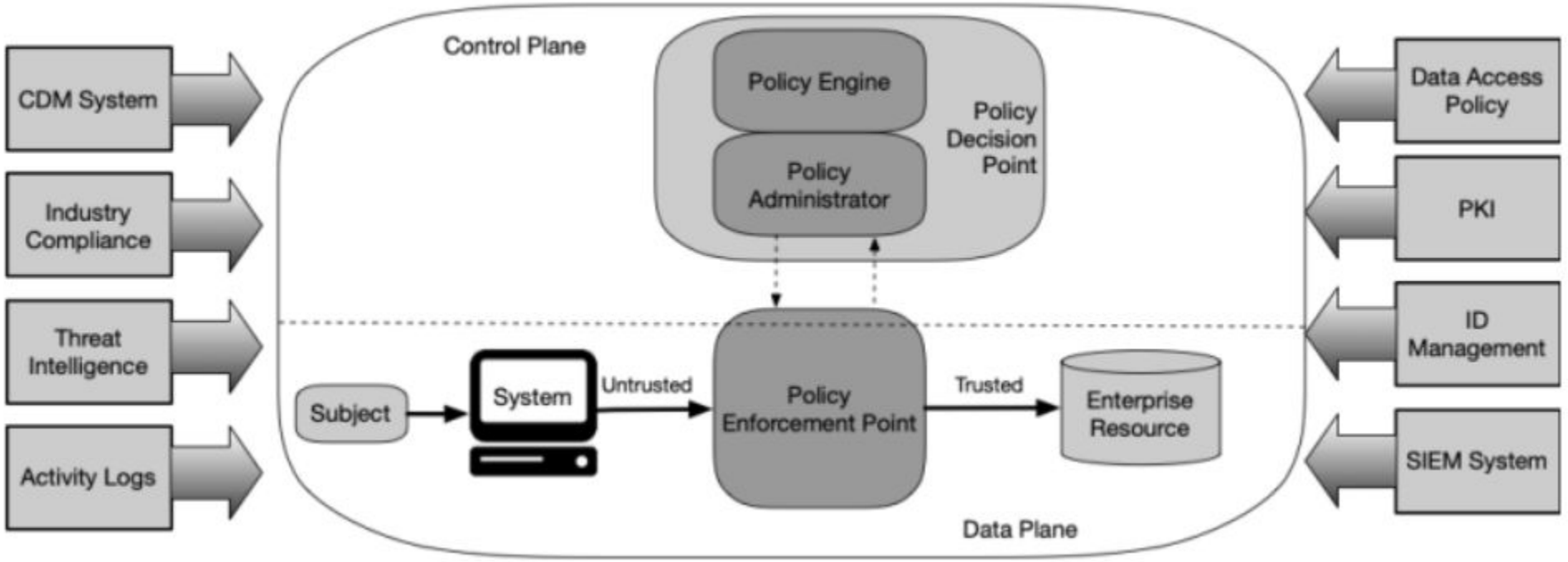
Building Blocks of ZTA

Demystifying ZTA

- (**What**) Is there a business need or context for this connection to exist?
- (**Who**) What is the identity of the user requesting access?
- (**Where**) Can we establish the integrity of the host initiating the connection?
- (**Why**) Is the content of the connection sane?
- (**How**) Do the applications on both sides of the connection make sense?

Zero Trust Architecture

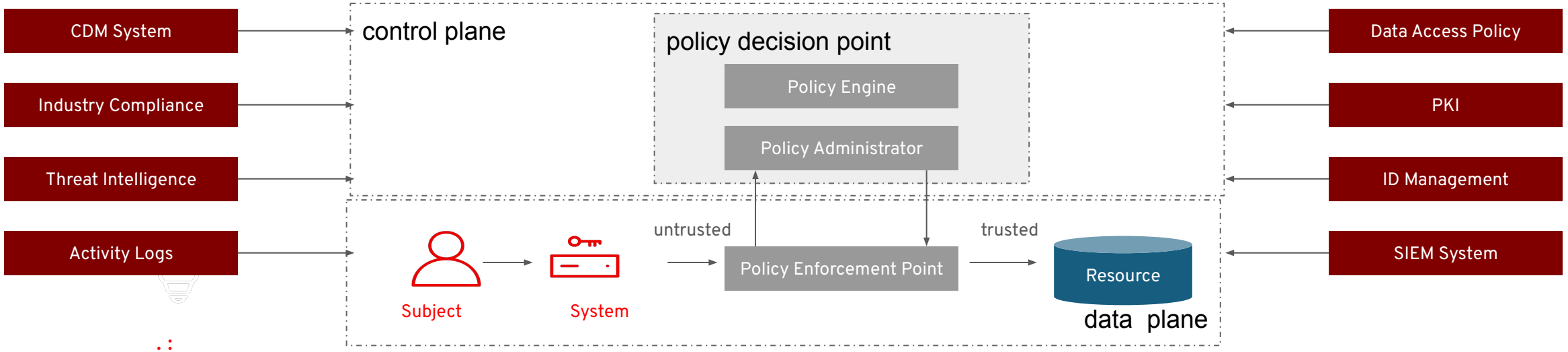
Logical Components



NIST Special Publication 800-207: Zero Trust Architecture, nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf

How do we implement Zero Trust?

The DoD, GSA, and NIST 800-207 reference architectures provides a lot guidance



Expanded ZT Principles:

- **All** data sources and computing services are considered resources
- **All** communication is secured regardless of network location
- Access to individual enterprise resources is granted on a **per-session basis**
- Access to resources is determined by **dynamic policy**
- The enterprise **monitors** and measures the integrity and security posture of all owned and associated assets
- All resource authentication and authorization are dynamic and **strictly enforced** before access is allowed
- 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.

Red Hat

Building a ZTA

Zero Trust is NOT “Security”

Let see how we can realize a system driven from the Zero Trust principles and security model we want to enforce – **deny by default and allow when we can explicitly show trust**

Let's Start building

If Zero Trust is about moving the access decisions closer to the app or data, lets start there

Let's draw a perimeter directly around our app: this is the thing we want to protect

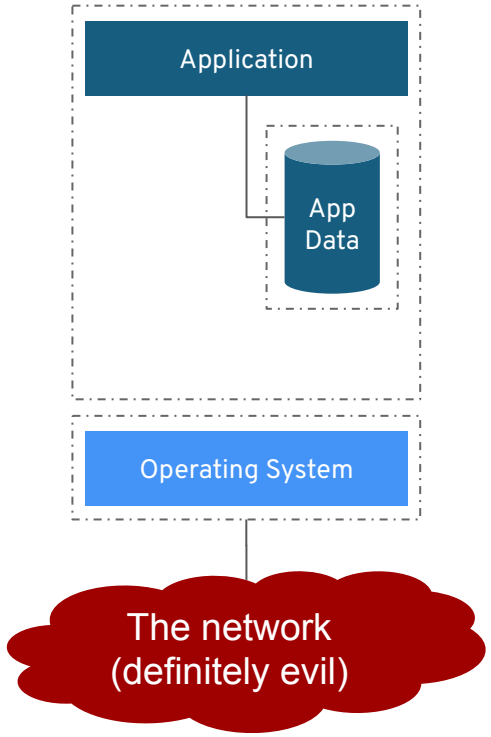
Protecting our app *also* protects the rest of the enterprise by denying attackers its use for further exploitation, lateral movement, implantation, or other vectors

Boundaries

If we aren't going to trust anyone, we need to some boundaries to keep them out

Zero trust principles say don't trust and separate out:

- Apps need infrastructure to operate (an OS, VM or H/W)
- Apps usually need some sort of data -- user data, configuration data, etc
  - This is an example of a common Zero-trust implementation technique -- microsegmentation
- The strength of each boundary is going to depend on traditional security controls applied to each
- Apps that are useful need a way to access these, usually over a network
  - we still protect the network as its critical connectivity, we just don't trust anything on it
- We need ways to get through the boundaries we just built -- gateways

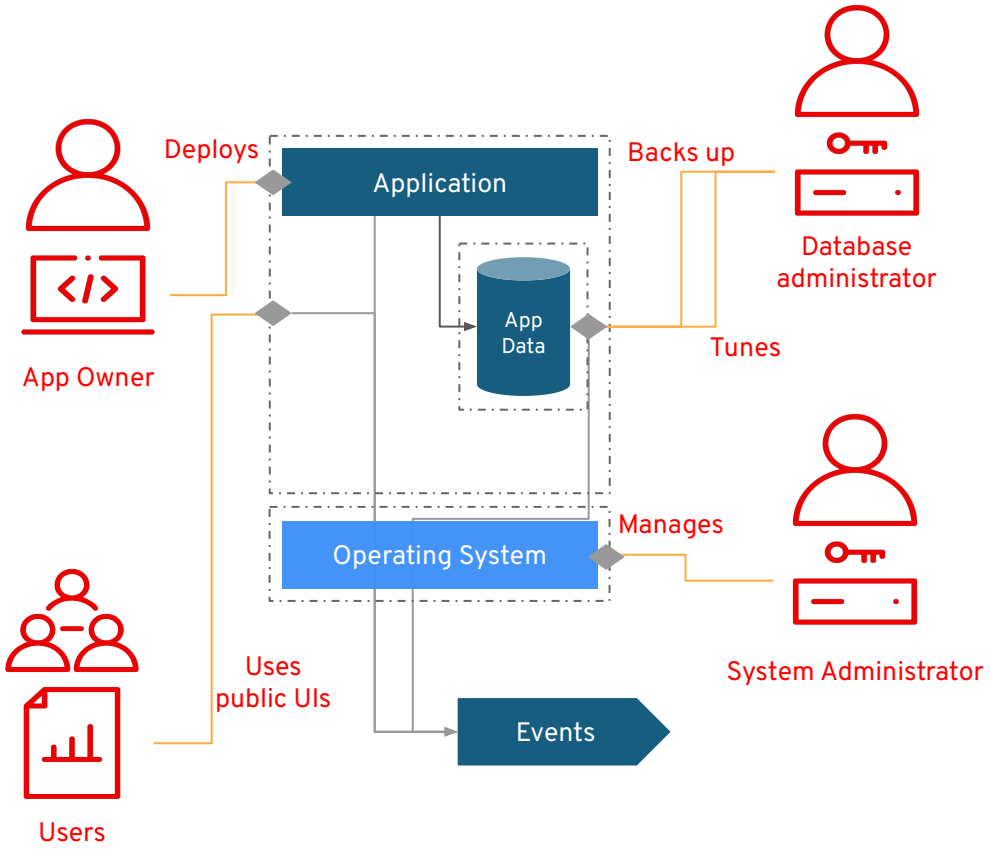


The Actors

But we still need to open up *some sort* of access

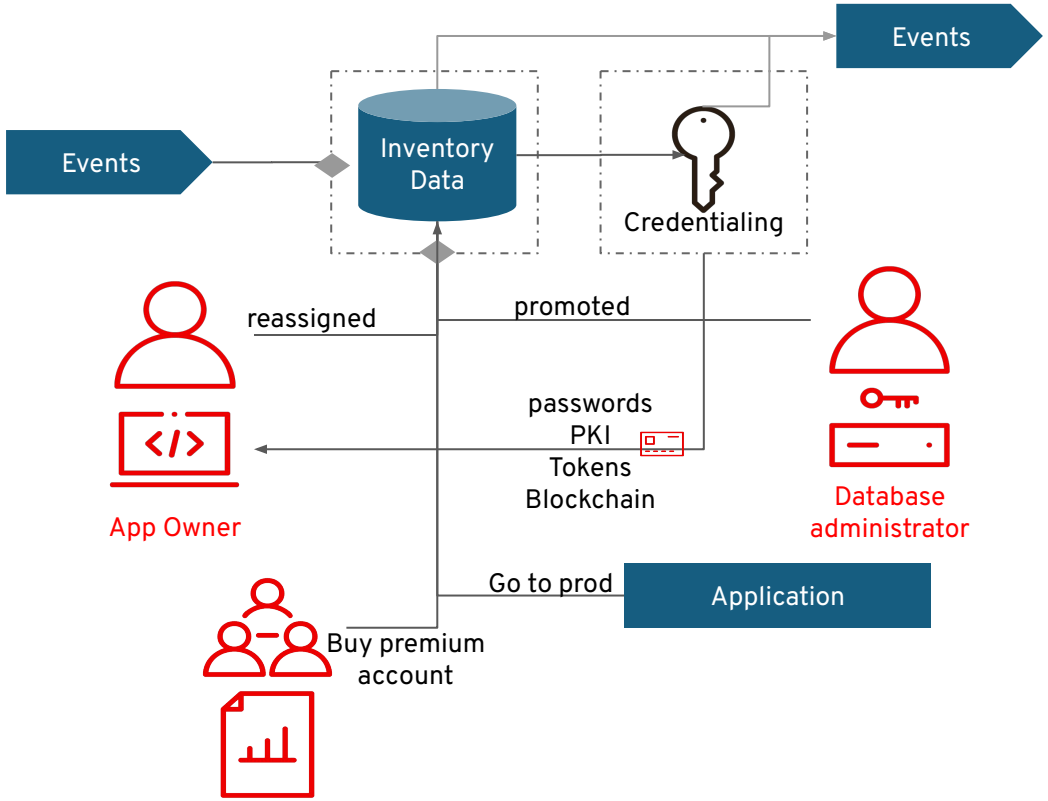
Zero trust principles say we allow access only for what is absolutely necessary for a business function:

- All the participants need to be known if we are going to assess anything
  - They need a means to prove (proofing) their identity
  - and some credentials to assert these
- Each actor is given access based on what they need from the application
  - We will want role separation to handle actions differently
  - The App, Data, and Infra all need their own access policies
- We need a enforcement point at each gateway to assess who the actor is and what they are doing
- Punching all these holes is a risk due to error, drift, etc
  - We want to consolidate gateways and enforcement points



Inventories & Identity Stores

Proving identity and attributing people and systems often requires out-of-band physical interactions



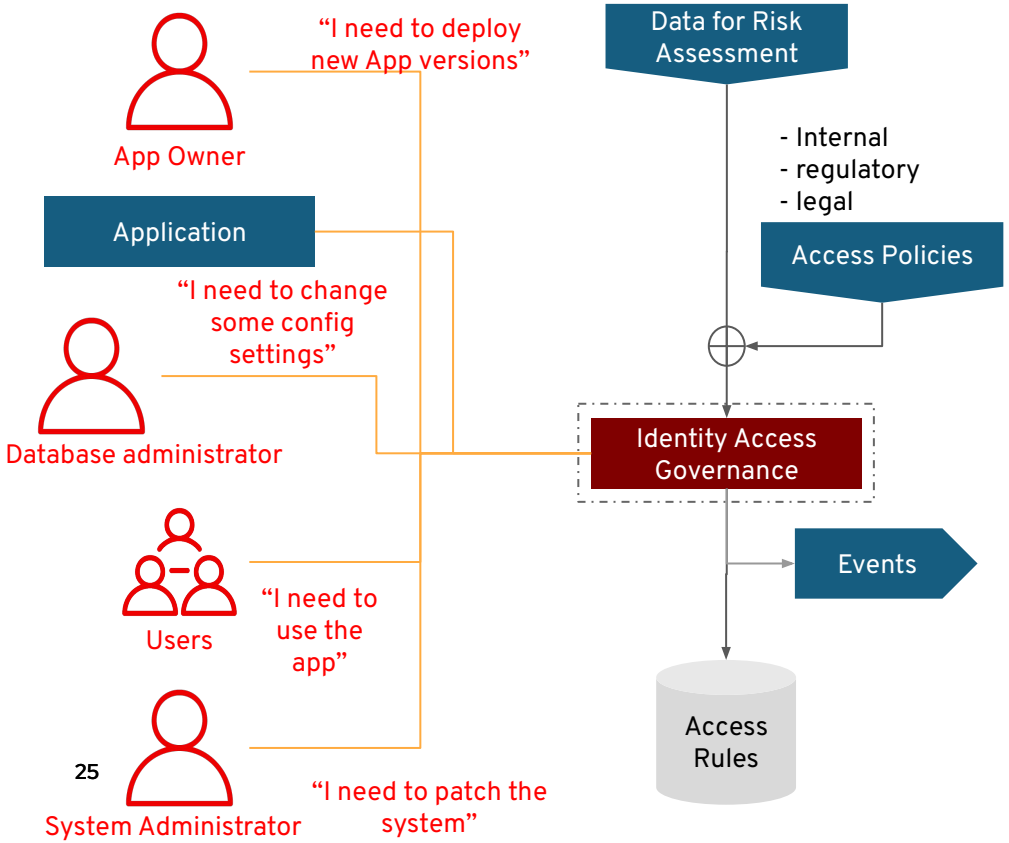
Zero trust principles say we need to know who is interacting with the system to balance need and risk:

- Information often due to some lifecycle event, such as hiring/firing, (de)commissioning, registration
  - We need to watch/listen to these lifecycle events from relevant systems like HR, Xacta, or the App itself
- Multiple credentialing options should be used
  - Ideally together (aka Multi-Factor Authentication or MFA)
  - Make it easy to keep credentials safe (eg. password managers, smart cards, wallets)
- Need to supply, verify, and curate IDs & attribute data
  - HR data is often wrong, people, roles, systems change
  - Strong governance to manage this process

Red Hat

Access Requests

Actors will need to express their needs ; we need to convert these to rules we can act on



Zero trust principles say every action is independently authorized (aka – by default nothing is authorized)

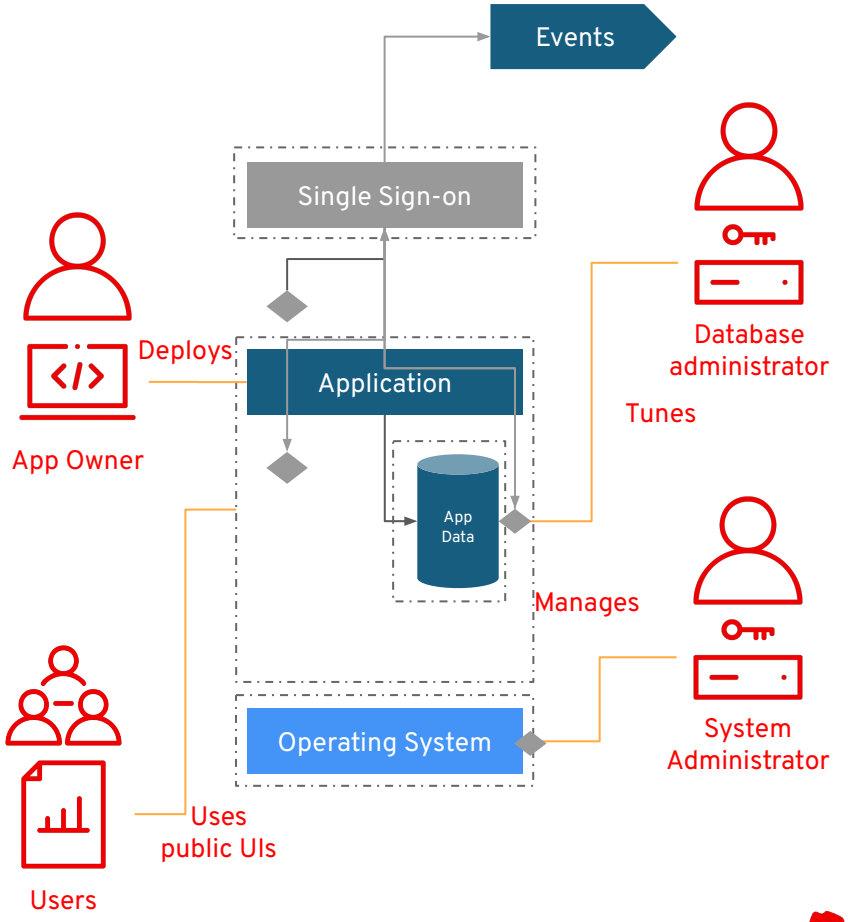
- If nothing is authorized by default, no need for default accounts either: there is no root
  - The rule is No standing privileges
  - Accounts and privileges entailed granted only when needed
- How much privilege is determined by muxing
  - Who you are, what you need to do – the request
  - policies expressing constraints and preferences
  - And a risk assessment of your action
  - We need a tool (say) "identity governance" – to collect & refresh this information on events
- People expect quick/friendly GUIs, or implicit requests; Systems expect continuous access
  - continuously monitor and diagnose that need and revoke if necessary

Control your gateways via SSO

You're now the single enforcement point for access

Zero trust principles say we need consistent enforcement when authorizing access:

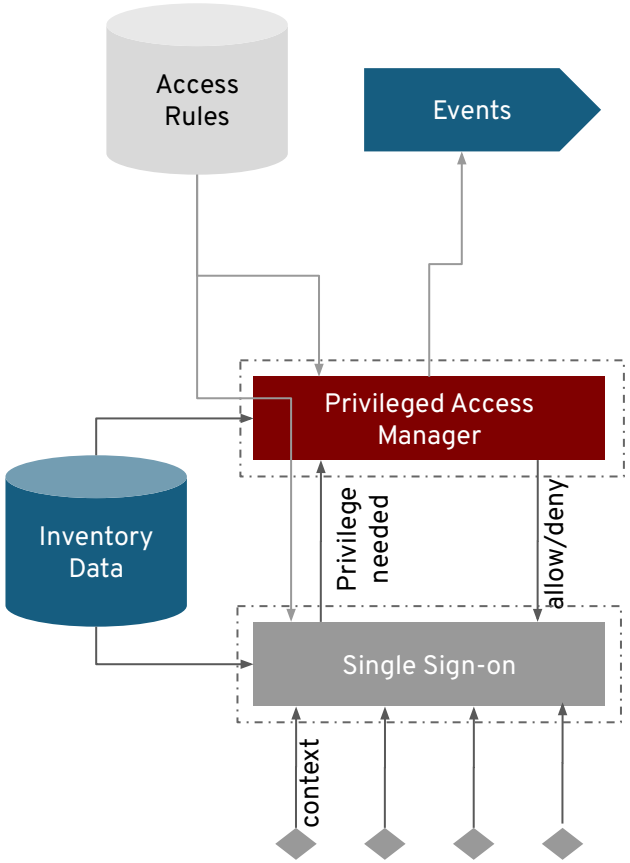
- Single Sign On technologies allow these gateways, wherever located, to enforce a single logic
  - Often embed rules engines for making enforcement decisions at zero privileges and/or known roles (implicit)
  - Often federates access to identity stores for authn
- Can provide the access request GUIs we need
  - Consistent presentation, look and feel
  - Can provide housekeeping GUIs too: registration, account certification, password/credential reset, and others
- Can collect information: context, identity attributes, and activity to feed event systems
  - Funnels all access decisions -- makes compliance/auditing simpler



Red Hat

Managing Privileged Access

Eventually we need to let someone do something interesting (but shut the door when they leave)

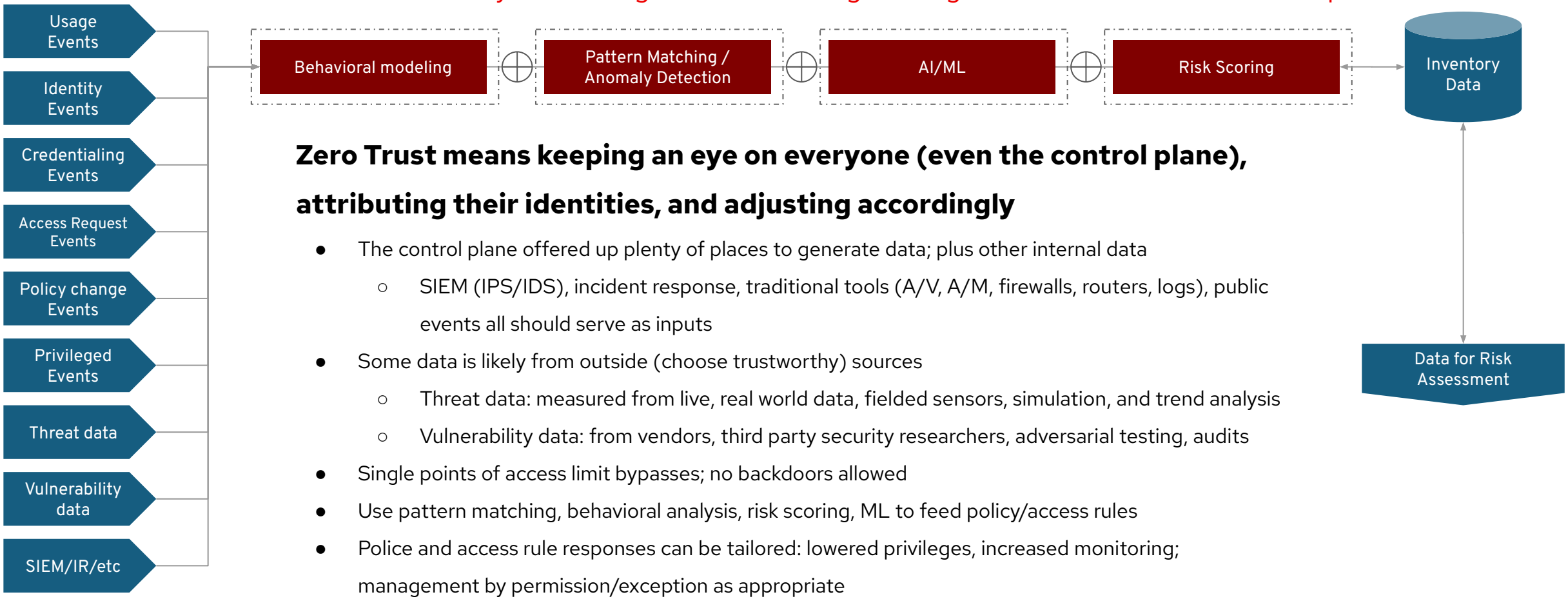


Zero trust requires us to create only -needed access-:

- Different access determinations for different realms: application, data, infra, control
  - The same request may need far fewer data privileges (eg. None or R/O) for some component than others
  - The same request may open associated/close existing accesses on other systems
  - Based on other context: time, location, device types, etc
- Project the needed access resources in the systems
  - New or altered user accounts, roles, group membership
  - May broaden or narrow existing accesses
  - By rule automatically scoped, revoked, time/quant bounded
- Implementing may require proxies, agents, plugins, KMS, or other tools to act on /on behalf of some components
- Allow real-time access to pass via the original gateway

Data Driving Decisions

Continuous Behavior analysis and diagnostic monitoring feeding back to the zero-trust control plane



Zero Trust means keeping an eye on everyone (even the control plane), attributing their identities, and adjusting accordingly

- The control plane offered up plenty of places to generate data; plus other internal data
  - SIEM (IPS/IDS), incident response, traditional tools (A/V, A/M, firewalls, routers, logs), public events all should serve as inputs
- Some data is likely from outside (choose trustworthy) sources
  - Threat data: measured from live, real world data, fielded sensors, simulation, and trend analysis
  - Vulnerability data: from vendors, third party security researchers, adversarial testing, audits
- Single points of access limit bypasses; no backdoors allowed
- Use pattern matching, behavioral analysis, risk scoring, ML to feed policy/access rules
- Police and access rule responses can be tailored: lowered privileges, increased monitoring; management by permission/exception as appropriate

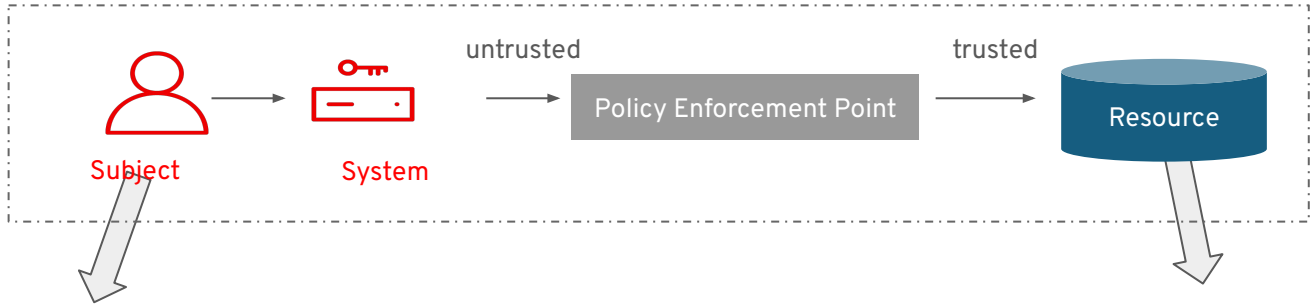
Are we there yet?

Considerations for implementation

How do we know we are done *building a Zero Trust Architecture?*

Enterprises have a lot of resources

ZTA requires that we implement dynamic access controls for every resource

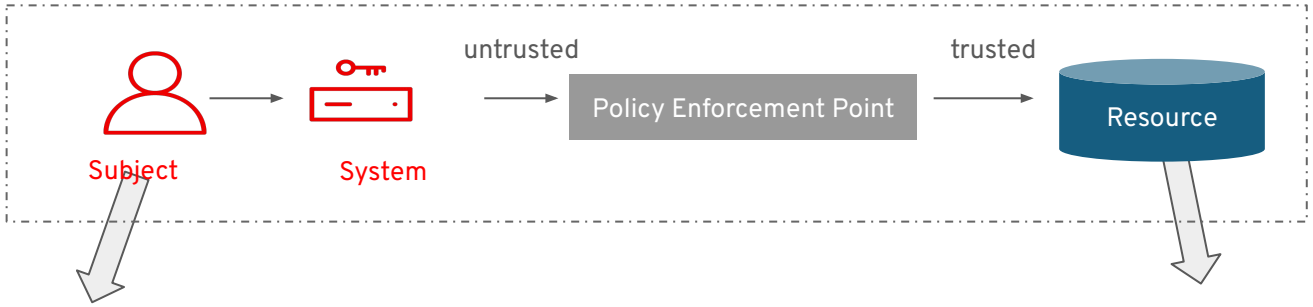


User	Has Dynamic Access Controlled to...	Application
Application	Has Dynamic Access Controlled to...	Database
Application	Has Dynamic Access Controlled to...	Corp Network
Application	⋮	Log System
Anything	Has Dynamic Access Controlled to...	**Everything**

Red Hat

At all levels of abstraction

ZTA requires that we implement dynamic access controls for every resource



Application	Has Dynamic Access Controlled to...	Database
Database	Has Dynamic Access Controlled to...	Host OS
Host OS	Has Dynamic Access Controlled to...	Corp Network
Corp Network	⋮	Edge Routers
Anything	Has Dynamic Access Controlled to...	**Everything**

Red Hat

Building a holistic Enterprise choice for ZTAs requires a look at your components, and above and below

Adoption of ZTA will depend on the ability to **select** technology that supports your overall zero trust goals and strategy, as well as **adopt** or **adapt** legacy technology

Components "above the stack"

- Can request access & services from below the stack
- Tolerant of denials or deferrals
- Presents an enterprise identity and attributes

Your component must itself be "Zero Trust"

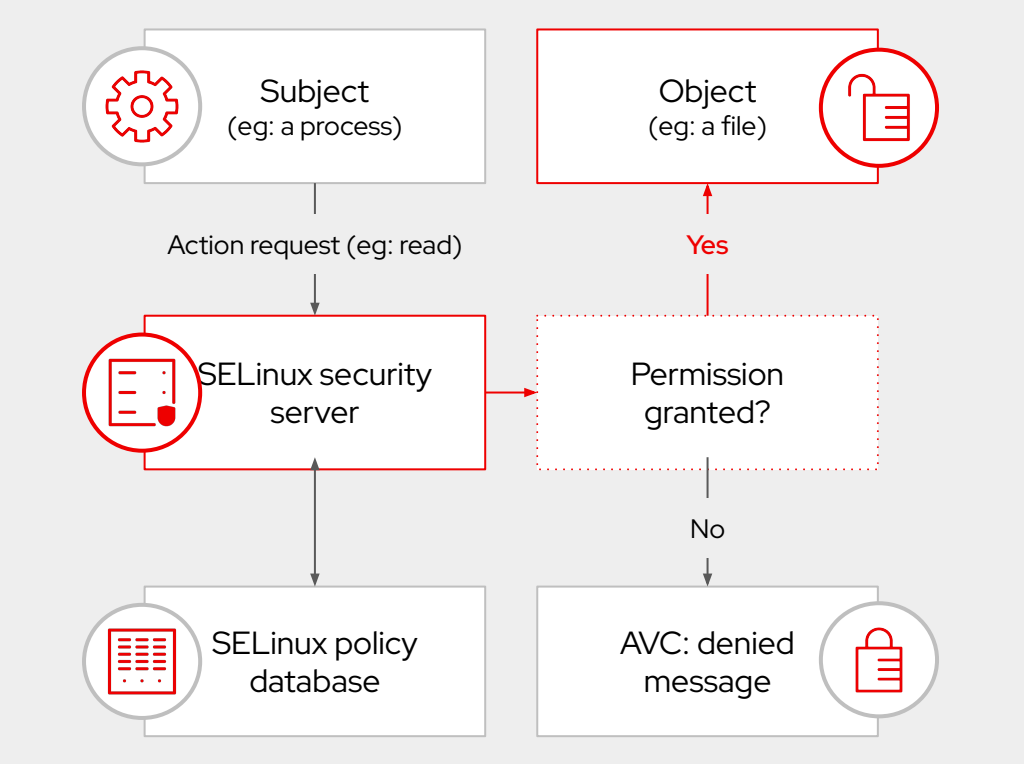
- Can allow or deny access on a need-to-know basis
- Can provide context data to the ZT control plane
- Implements least privilege access model

Components "Below the stack"

- Must capture attempt to access services
- Hardened against improper access / Default deny

RHEL: -the-foundation for a Zero Trust Architecture

RHEL is the best choice for implementing zero trust for your infrastructure



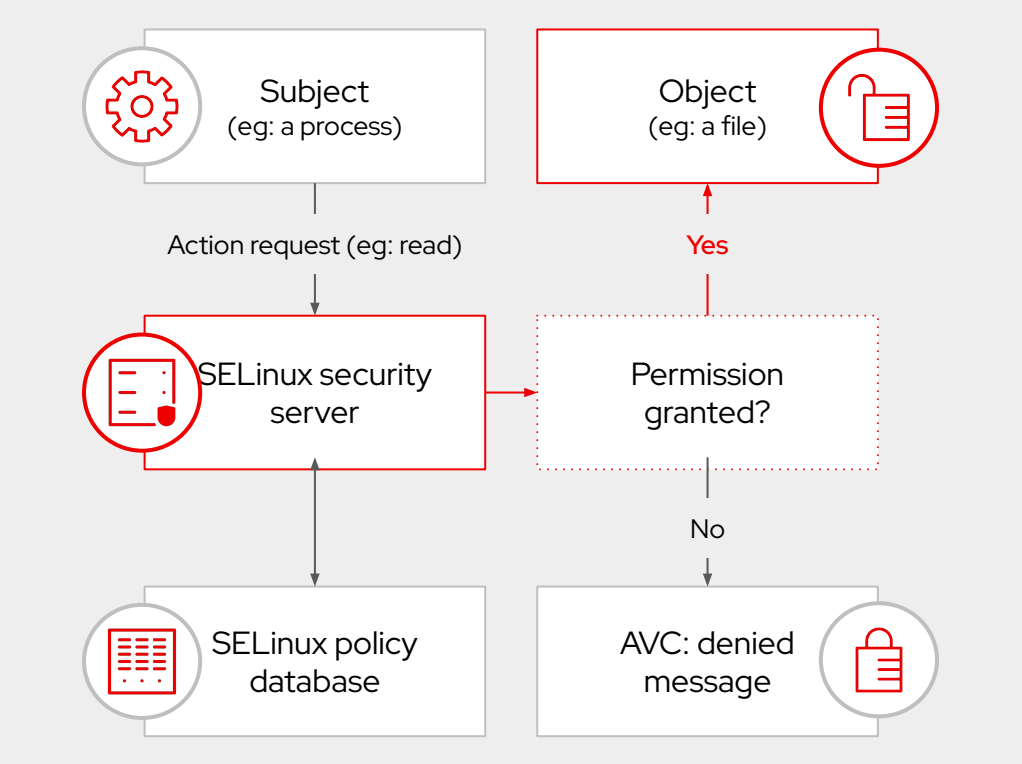
The most trusted operating system for the enterprise, Red Hat Enterprise Linux (RHEL) is a foundation of ZT:

- Advanced resource access (SELinux, ACLs, FAPolicyd)
- Advanced process management, including Linux Containers (aka, Docker)
- Pluggable & Stackable Access Modules & IDM
- H/W and S/W identity and cryptographic attestations (via SecureBoot, KeyLime & TPMs, IMA, RH signin)
- Derivative operating systems via UBI, image builder, RHCOS to further minimize attack surface

Red Hat

Platforms for a Zero Trust Architecture

Considerations for platform implementation



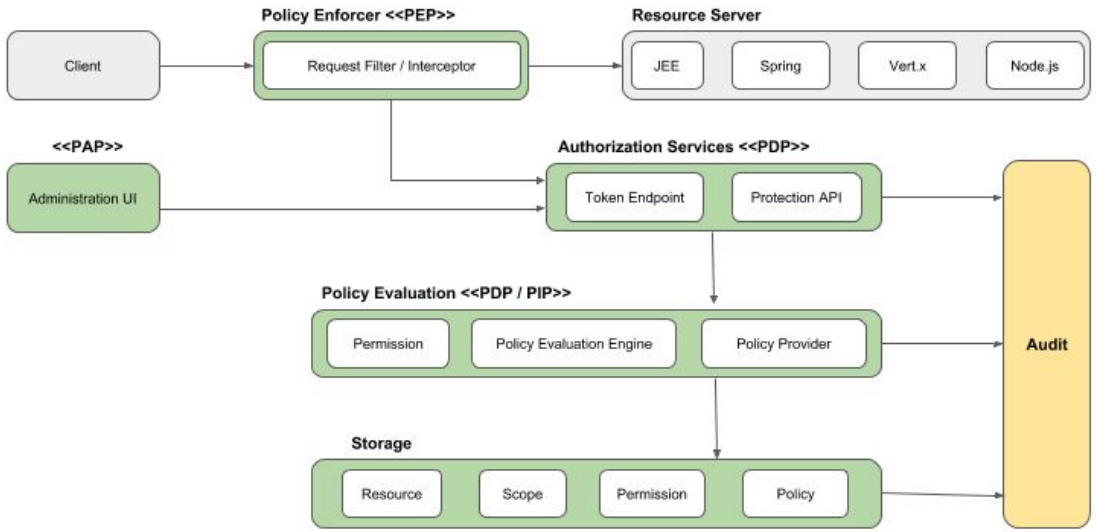
Determine the Operating Systems (OS) that supports the enterprise

- Advanced resource access
- Advanced process management, including Linux or Windows containers
- Pluggable Access Modules, IDM
- H/W and S/W identity and cryptographic attestations (via SecureBoot, KeyLime & TPMs, IMA, RH signing)
- Derivative operating systems via image builder
- Immutable OS to further minimize attack surface

Critical Software - Definition & Explanatory Material, nist.gov/itl/executive-order-improving-nations-cybersecurity/critical-software-definition-explanatory

Platforms for a Zero Trust Architecture

Considerations for access protection



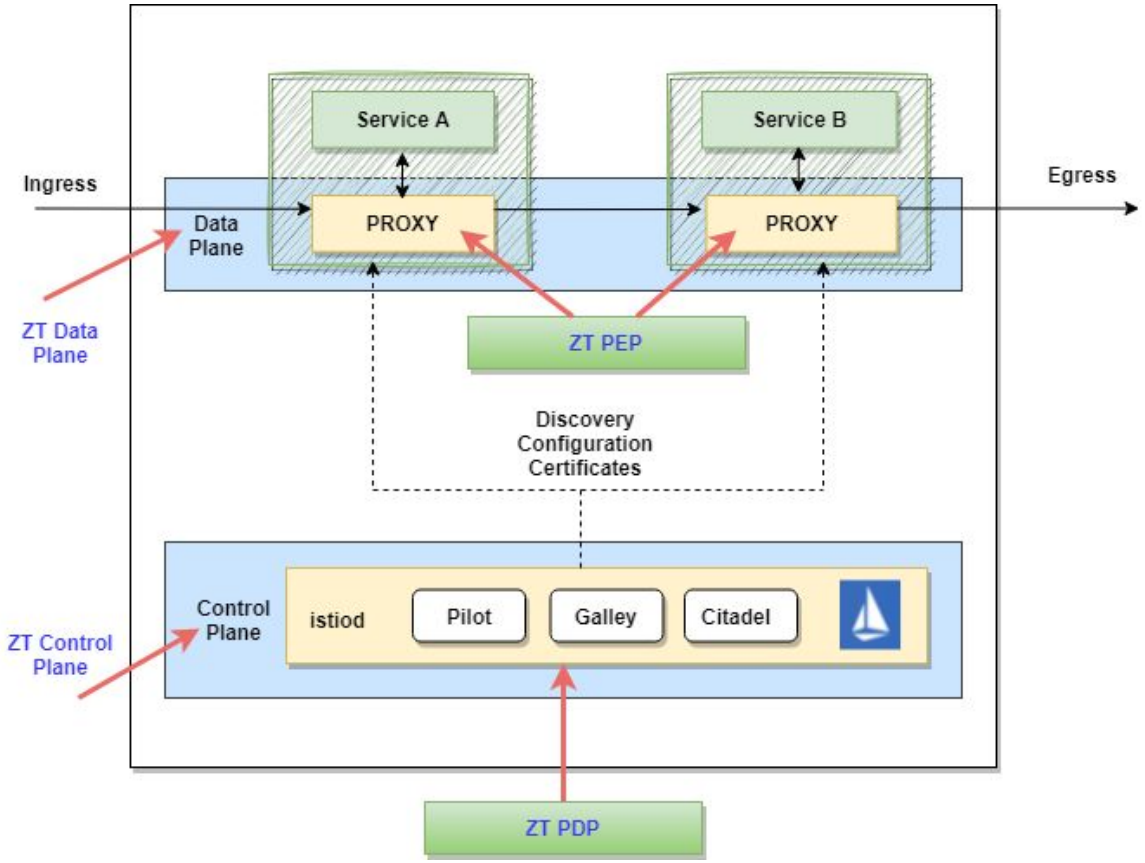
Determine the enforcement of access

- Integrate and manage user identities, roles, and attributes, and build custom user models
- Federate multiple identity providers
- Credential management & MFA to authentication flows
- Define ABAC, RBAC, CBAC, UBAC via full, embedded rules engine
- Collect access events & telemetry to inform future access decisions

Critical Software - Definition & Explanatory Material, nist.gov/itl/executive-order-improving-nations-cybersecurity/critical-software-definition-explanatory

Protecting the container hosts

Considerations for cloud technologies



Application-oriented Zero Trust capabilities *in a cloud native form*:

- Macrosegmentation
  - Built in Software Defined Network (SDN)
  - with least privilege policy managed network segmentation
- Microsegmentation : Service Mesh automates application level cryptographic mTLS segmentation
- Minimal standing privileges & controlled immutability
- Automatic Policy Enforcement Points (PEP): mutating, validating, and admission controllers
- Rich RBAC controls and pluggable external providers
- Collection of auditing, monitoring, eventing, for rich access context decisions

Kubernetes Hardening Guide, media.defense.gov/2021/Aug/03/2002820425/-1/-1/0/CTR_Kubernetes_Hardening_Guidance_1.1_20220315.PDF

Takeaways

Zero Trust Architecture

Visibility informs policy

Trust is neither binary nor permanent

Ownership is not a control

The perimeter is any place where you make an access control decision

Access decisions are based on re-establishing trust every time

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.


linkedin.com/company/red-hat


youtube.com/user/RedHatVideos


facebook.com/redhatinc


twitter.com/RedHat

Red Hat