Scaling Trust with Millions of Containers: Microsegmentation Strategies for Authorization

About Me

- Drupal
  - Security Team
  - Database Maintainer
- Service Mgmt for RHEL/Ubuntu
  - Committer
  - Scalable CGroups Management
- Pantheon
  - CTO and Co-founder

Pros and Cons of Microservice Architecture

PANTHEON®

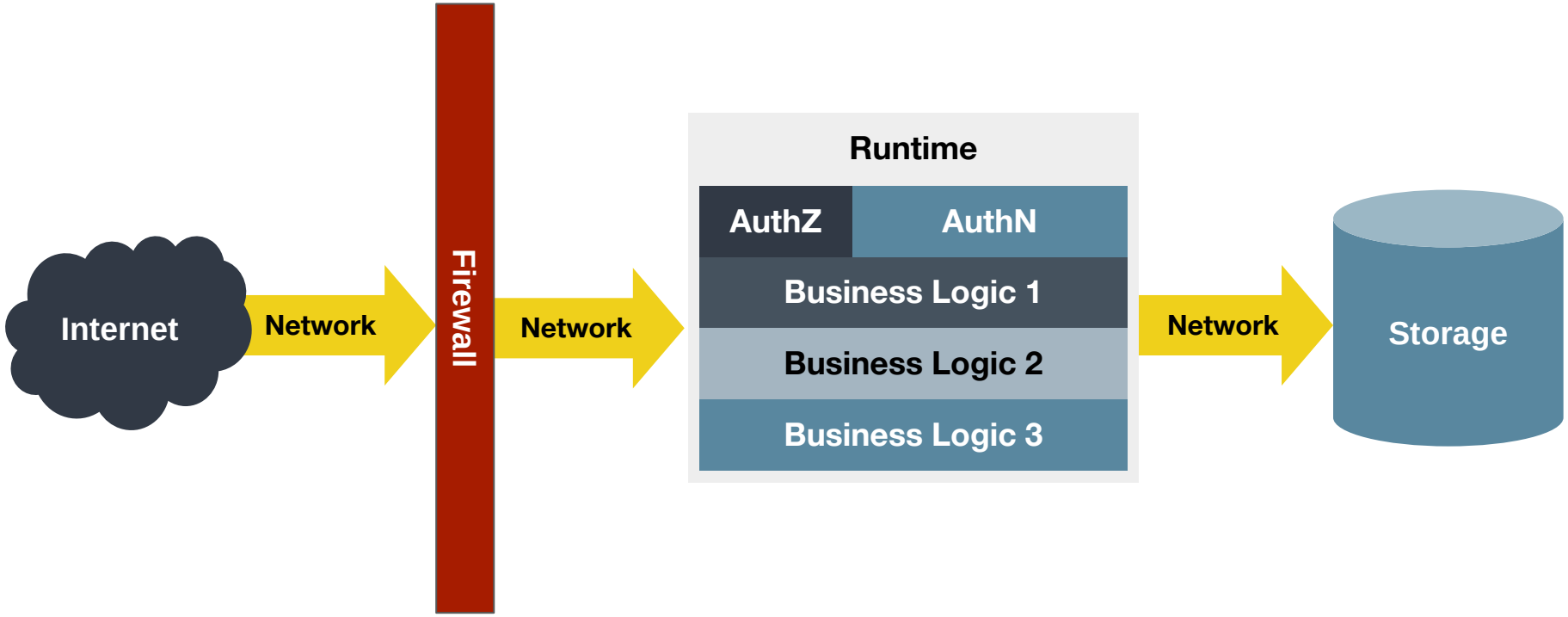
Pros

- Language/framework diversity
- Lower-risk experimentation
- Compartmentalized testability

Cons

- Language/framework diversity
- Change management
- **New attack surfaces**
- **Single point of failure (SPoF) risks for authorization**

Traditional, Monolithic Web Architecture



Monoliths Have Important Security Upsides

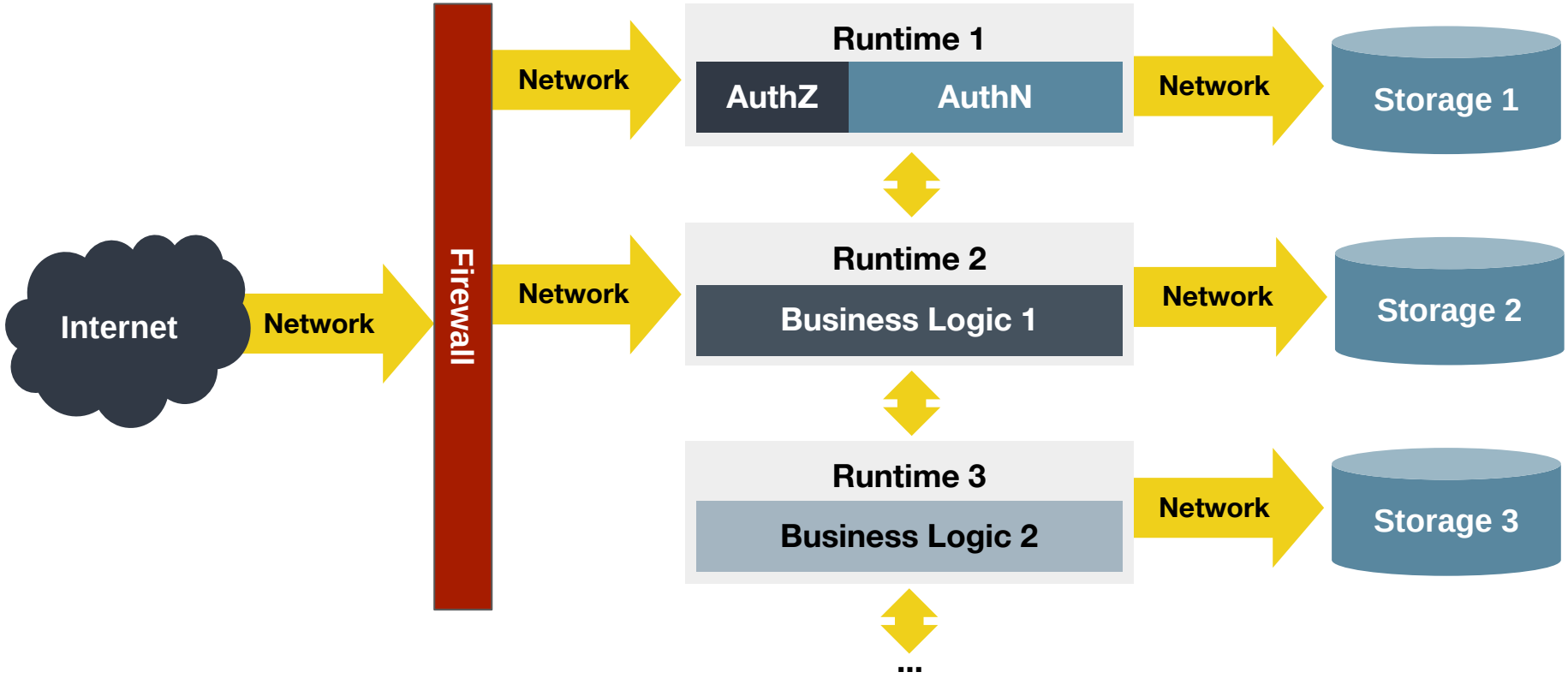
PANTHEON®

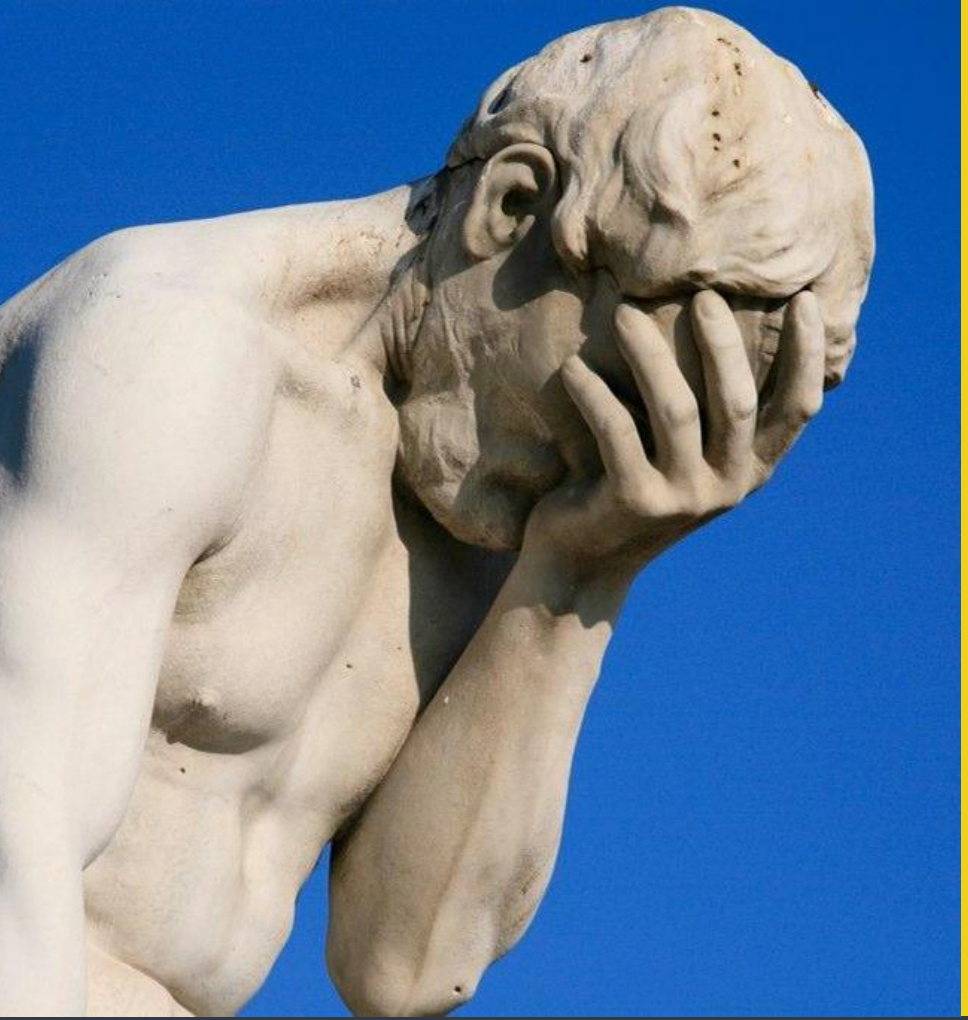
- Runtimes can enforce caller/callee relationships
- Limited serialization and parsing
- Easy, frequent, fine-grained authorization checks
- A singular “edge” to clients
- Centralized logging

We lose these with microservices...

...unless we’re careful.

A Naive Microservice Deployment





This is worse than before.

Micro*services* do not guarantee micro*segmentation*.

The Attack Surface Has Expanded

PANTHEON®

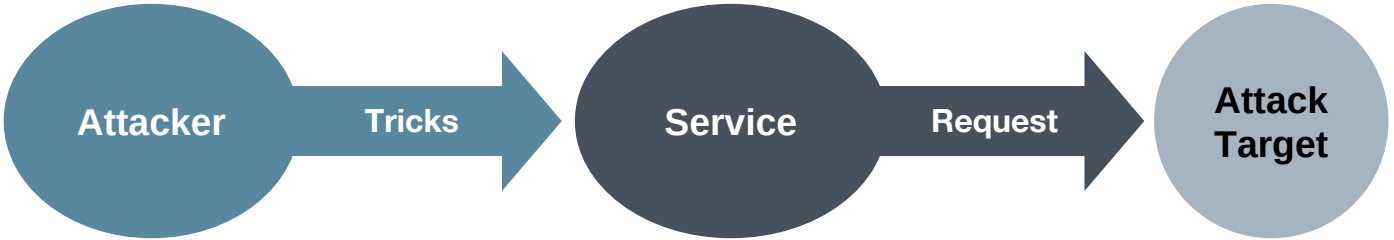
Inherent Challenges

- Lots of serialization and parsing
- Every stack has unique vulnerabilities
- Anything can (try to) send a packet to anything

Common Anti-Patterns = Risk

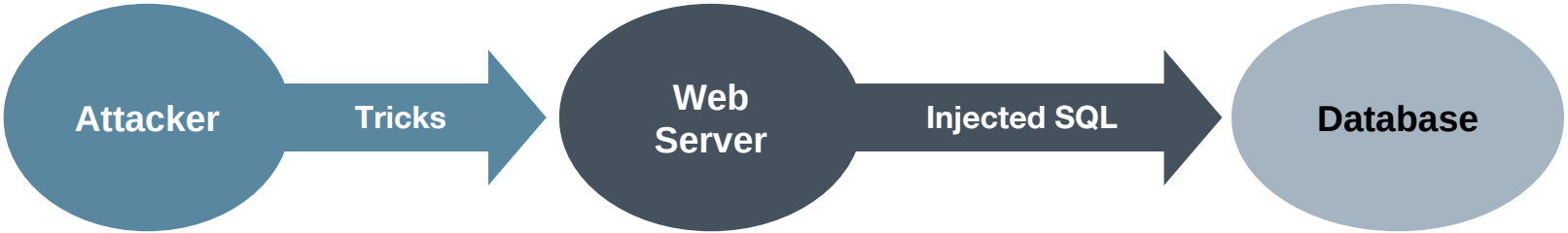
- **All in the Family:** Trusting everything behind the firewall
- **God Proxies:** Trusting frontend systems to behave

The Confused Deputy Problem

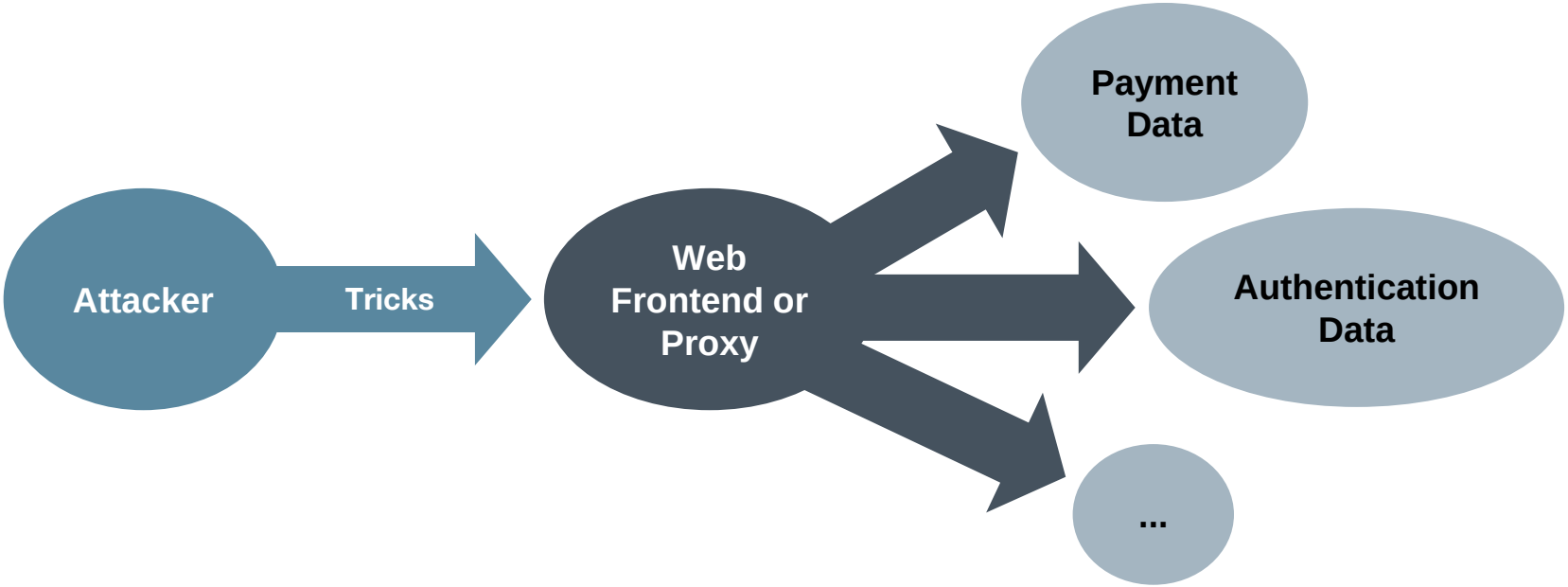


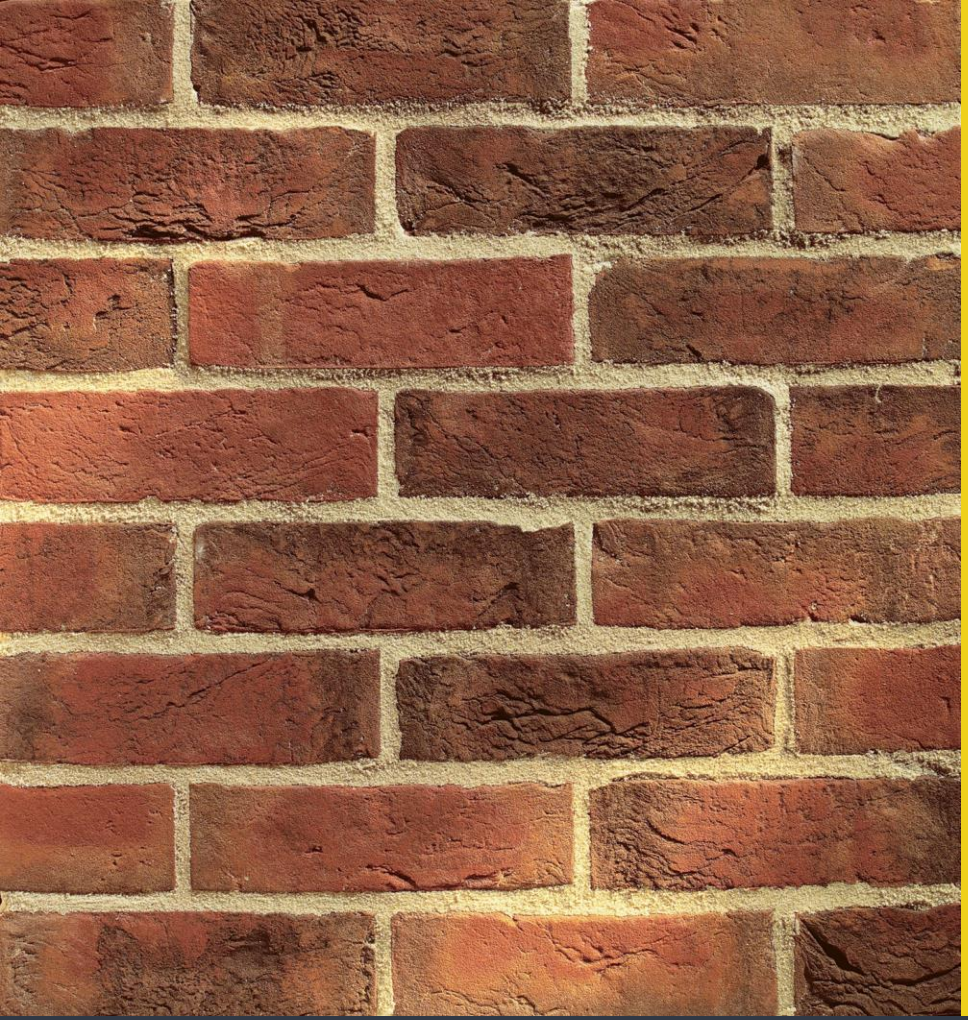
An Old Problem

A Current Problem



Back with a Vengeance in Microservices





The Big Question

“A Foot in the Door”

If an attacker gains the ability

to make network requests

behind the firewall

what happens?

Getting Behind the Firewall

PANTHEON®

Many Methods

- Tricking a Proxy
- VPN Access
- Arbitrary Exec on Any Service
- Employee Device Compromise

Many Successful Hacks

- Sony
- Panama Papers
- Stratfor
- DNC Emails
- Equifax

...all relied on using a vulnerability in **one system** to compromise **another**

Locking Down Access

Phase One: mTLS and Firewalls

Goal: Only allow each service to interact with the **services** they have a need to access.

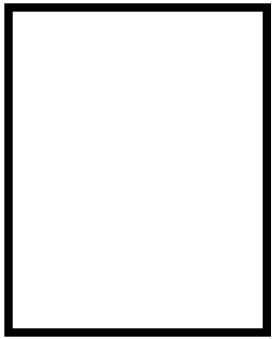
Allowed
☐ Disallowed

		To			
		AuthN/IdP Service	**LDAP**	**Billing Service**	**Cardholder Data**
From	**Web Frontend**		☐		☐
	AuthN/IdP Service	☐		☐	☐
	LDAP	☐	☐	☐	☐
	Billing Service	☐	☐	☐	
	Cardholder DB	☐	☐	☐	☐

Complements: Firewalls and mTLS

PANTHEON®

	Firewalls and SDNs	Mutual TLS (mTLS)
Enforced	Beneath the application	By the application or proxy
Complexity	Lower	High
Scaling	Easy (if Static) Tough (if Dynamic)	Easy
Ideal Use	Coarse Segmentation	Fine or Micro Segmentation



We’re back to the **same surface area** (or less) than a monolithic design.

But, what about attacks that traverse **established, allowed** paths?

With Just Firewalls and mTLS

Problem: The web frontend can access anyone's data if tricked into doing so.

Allowed
☐ Disallowed

		To		
		Alice's Billing Records	**Bob's Billing Records**	**Eve's Billing Records**
From	**Alice → Web Frontend**		☐	☐
	Bob → Web Frontend	☐		☐
	Eve *Attacks* → Web Frontend	!	!	

Phase Two: Capabilities

Goal: Only allow each service to interact with the **data** they have an **immediate** need to access.

Allowed
☐ Disallowed

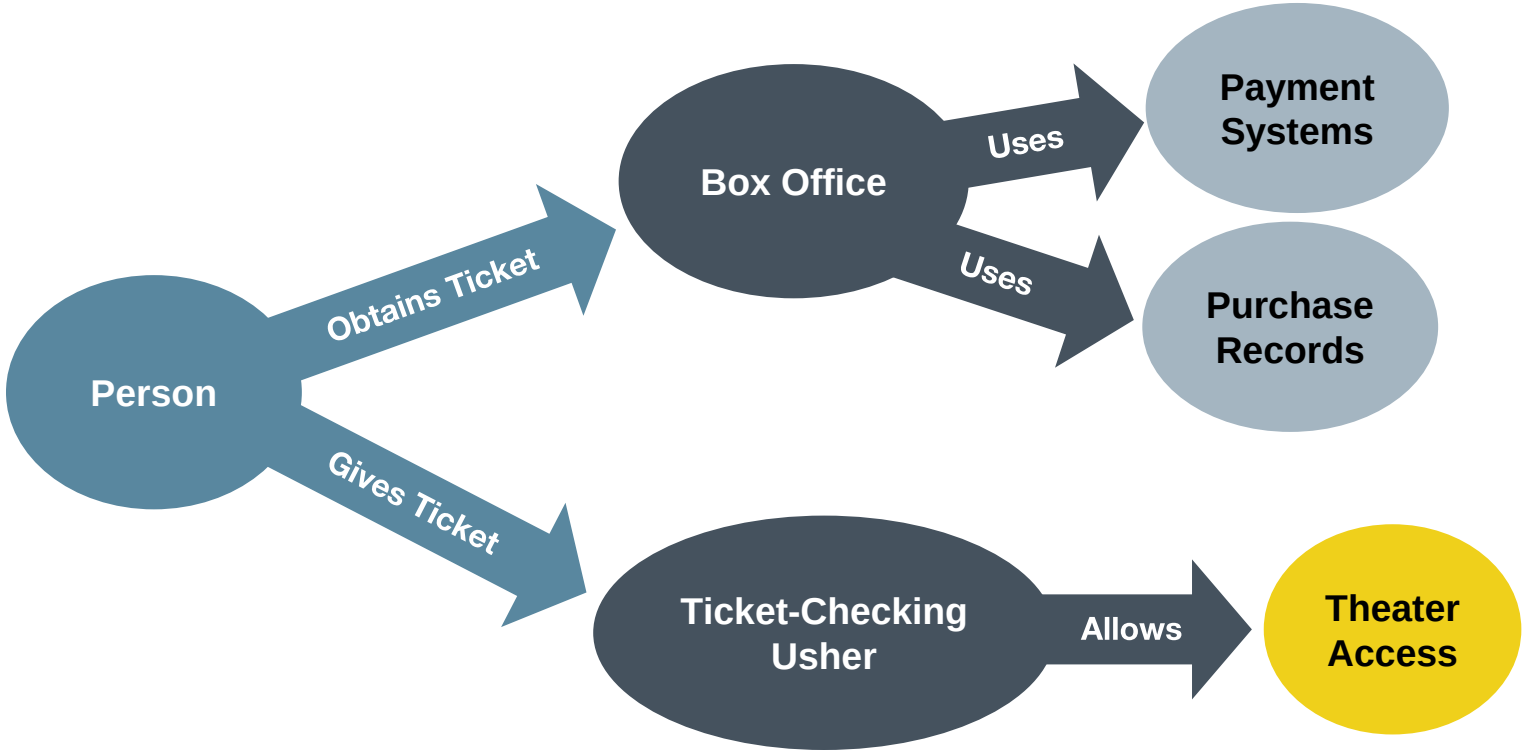
		To		
		Alice's Billing Records	**Bob's Billing Records**	**Eve's Billing Records**
From	**Alice → Web Frontend**		☐	☐
	Bob → Web Frontend	☐		☐
	Eve *Attacks* → Web Frontend	☐	☐	

Ambient Authority vs. Capabilities

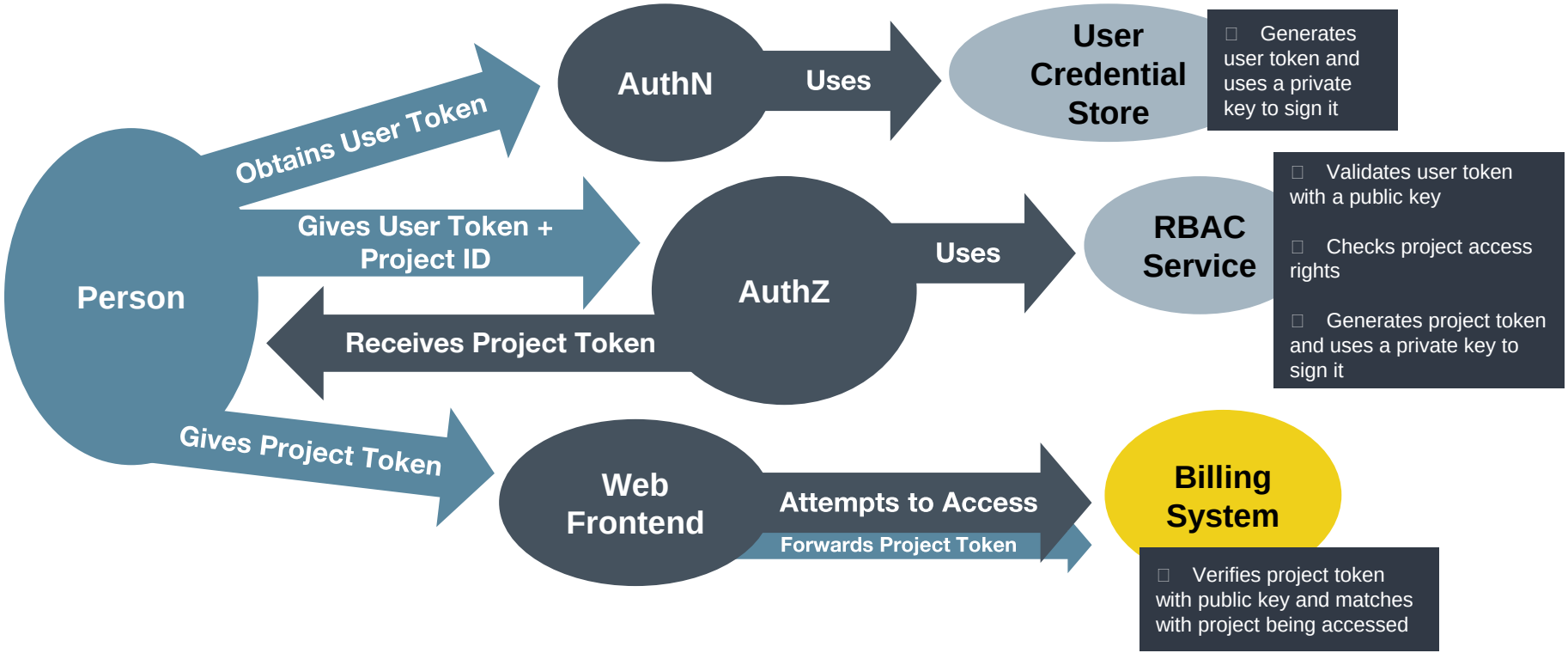
PANTHEON®

	Ambient Authority	Capabilities
Locus	Actors	Objects
Authorization	Lookups for Actor+Verb+Object	Implied by possession of token for Verb+Object
Scaling Impact	Lookups on every request	Renewals at time intervals
Each Microservice Needs...	Client for session and authorization lookups (or even business logic)	Public key and parser
Biggest Risk	Actors can be tricked	Revocation

Capabilities in Action: A Familiar Scenario



Applied to Web Systems



Capabilities in the Wild: S3 Signed URLs

https://s3.amazonaws.com
/#{bucket}
/#{path}
?AWSAccessKeyId=#{key-id}
&Expires=#{expiration}
&Signature=#{signature}

Properties:

- **Key ID** allows audit trail of use
- Possibly to **forward** without the recipient having authority
- Authorizations map to specific **objects** and **actions**

HTTP + JWT in Action

PANTHEON®

```
Authorization: Bearer
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkw
IiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9.EkN-
DOsnsuRjRO6BxXemmJDm3HbxrbRzXglbN2S4sOkopdU4IsDxTI8jO19W_A4K8
ZPJijNLis4EZsHeY559a4DFOd50_OqgHGuERTqYZyuhtF39yxJPAjUESwxk2J
5k_4zM3O-vtd1Ghyo4IbqKKSy6J9mTniYJPenn5-HIirE
```

JWT Decoded

Header:

```
{
  "alg": "RS256",
  "typ": "JWT"
}
```

Payload:

```
{
  "sub": "project-39839",
  "access": "ro",
  "name": "Jane Doe",
  "email": "jane@doe.com"
}
```

Signature: (not human readable)

Properties:

- All the benefits of signed URLs
- **Fully validatable** en route
- Strong hashing (HMAC SHA-2)
- Widespread **library support**

Revisiting the Equifax Breach

PANTHEON®

How the Attack Worked

1. **Exploit** of Apache Struts
2. Arbitrary **code execution** from the web frontend
3. Used the **ambient authority** of the frontend to obtain bulk records from deeper systems

If They Had Capability Security

- **Web frontend vulnerability:** Would only affect current visitors
- **Capability token system vulnerability:** Possible to throttle in the web frontend

Scaling Security to Millions of Containers

PANTHEON®

Firewalls and mTLS

- Firewalls for coarse segments
- Service-to-service mTLS
- Short-lived certs (no revocation)
- Internal, automated certificate authority

Capability Tokens

- Self-evident authorization (no lookups)
- Forwardable with queues and sub-requests
- Contain actor data only for logging and audit purposes

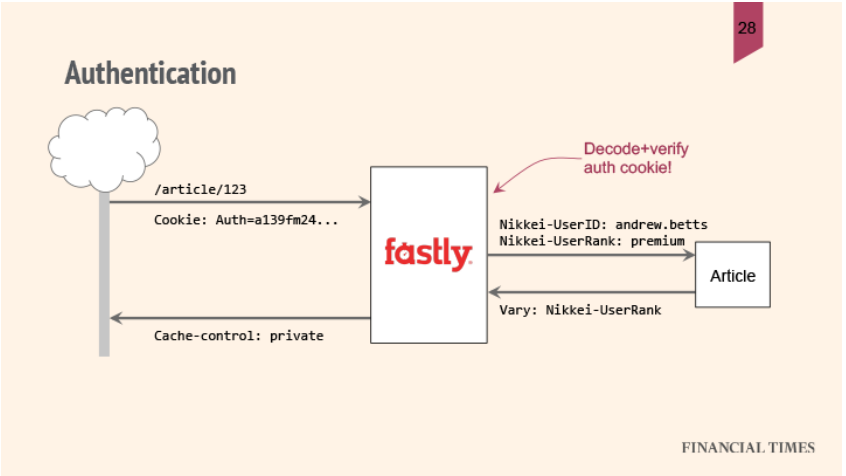
Questions?

@DavidStrauss

Bonus Capabilities in the Wild: FT Cookies

Financial Times + Paywall AuthN in CDN

1. **User** authenticates.
2. Authorization system maps **user** to **subscription level**.
3. User receives a **capability token** with the signed authorization information in a **JWT-format cookie**.
4. The **CDN can validate** the user's subscription as they load paywalled articles and still **hit the cache**.



CDNs create familiar challenges: Capability tokens allow proxies to *validate* without being *fully trusted*.

Complements: Firewalls and mTLS

PANTHEON®

Firewalls and SDNs

- Imposed beneath the application layer
- Sometimes imposed beneath the virtual machine or even physical machine
- **Simpler** initial setup
- Scaling up requires dynamic SDN lookups and is especially hard in cloud environments
- More useful for **coarse** segmentations

Mutual TLS (mTLS)

- Each side (client and server) has certificates signed by a mutually trusted authority (usually internal)
- **Complex** initial setup
- Application connections use a certificate on each side
- **Scales** effortlessly
- More useful for **fine** segmentations

Ambient Authority versus Capabilities

PANTHEON®

Ambient Authority

- Shallow services have full reign of data in the deeper systems they use
- Shallow services apply access rules to requests made to deeper ones
- Access decisions require constantly looking up and applying RBAC rules, which **scales poorly**
- Shallow services are vulnerable to becoming a confused deputy

vs.

Capabilities

- Clients accessing the the shallower services must supply access tokens
- Shallow services forward the tokens as necessary when invoking deeper services
- Access is self-evident from the tokens, which **scales well**
- Even when tricked, shallow services have limited ability to access or manipulate deeper services