MAXIMILIAN BURKHARDT / APRIL 22, 2019

Information Security IRL

IMPLEMENTING SECURITY AS AN ENGINEER IN 2019

A Very Brief Intro

UC Berkeley ➡️ iSEC Partners (pentesting) ➡️ Airbnb (defense)

Agenda

1. Why we need people like you!
2. What is security?
3. Making security happen
4. Places to go with your infosec career

Some XSS History

19 YEARS AGO

2 CA-2000-02: Malicious HTML Tags Embedded in Client Web Requests

This advisory is being published jointly by the CERT Coordination Center, DoD-CERT, the DoD Joint Task Force for Computer Network Defense (JTF-CND), the Federal Computer Incident Response Capability (FedCIRC), and the National Infrastructure Protection Center (NIPC).

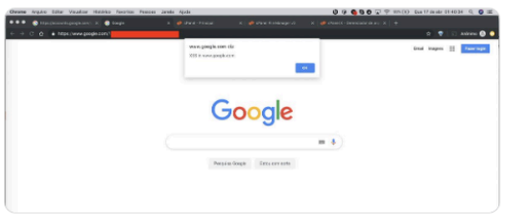
Original release date: February 2, 2000
Last revised: February 3, 2000

A complete revision history is at the end of this file.

Source: https://resources.sei.cmu.edu/asset_files/WhitePaper/2000_019_001_496188.pdf

Some XSS History

LAST WEEK

> My first tweet.
>
> A sweet XSS in Google main page and almost every subdomain.
>
> =)
>
> 
>
>
> 10:59 PM - 16 Apr 2019

Source: https://twitter.com/WHHackersBR/status/1118393568656334850 (not necessarily reputable)

What's the Deal?

- Google has one of the best security teams out there
- From 2015-2016 they paid out $1.2 million for XSS bugs via bug bounties

Source: https://security.googleblog.com/2016/09/reshaping-web-defenses-with-strict.html

Security isn’t scaling.
But everything else is.

The old guidance used to be: pentest everything, make sure skilled humans look at it
There aren’t enough skilled humans
How many of you have interviewed / interned at Equifax, Experian, or Transunion? And yet we trust these companies with all of our data

It's Not Just Old Problems

WE KEEP THINGS INTERESTING

- Containerization / Kubernetes is bringing new problems (and opportunities)
- Blockchain?
- Some crazy dark magic too: Spectre, Meltdown, Rowhammer

So Why Get Involved?

IT'S NOT ALL FIRES

- There's huge opportunities for changing how the industry does security
- We get to work at the bleeding edge

We just need innovators and fresh thinking
ML and security is one of the hotter topics right now
Anything is on the table if it can change the paradigm

Security is Creative

- Different tech stacks
- Different threat models
- Different budgets
- Different company cultures

One-size-fits-all doesn’t work
You can take security principles you learn here and end up implementing them in vastly different ways, depending on what your environment needs
Creativity is necessary both on the attacking side and the defending one. Usually, the most creative wins.

Today I’m going to talk about a creative approach to network security that I think has major promise for changing the game. But first, let’s try to lay the groundwork of what we mean by “security.”

WHAT IS SECURITY?

Describes a really wide range of work and problems

“A system is secure if it behaves precisely in the manner intended — and does nothing more”

— Ivan Arce

... which is not very helpful.

What is Security?

AN ATTEMPT AT MORE USABLE DEFINITIONS

- It's a strategy to address risks to your system
- All about defining what the threats are and responding appropriately

How to Mitigate Risk

IN THE BROADEST POSSIBLE TERMS

- Be threat-agnostic
  - Build protection close to the assets
  - Assume some defenses will fail
- Put in specific defenses against single points of failure
- Self-assess constantly
  - Human review is still really useful!
  - Bug bounties are great at this too

You can go get a Ph.D in this, and we're not going to spend a ton of time on risk models

threat-agnostic: you don't know what's coming at you

So How Is This So Difficult?

HONESTLY

- What's the hangup?
- "In theory, there's no difference between theory and practice. In practice, there is."
- Modern information systems are built on growth, and if security opposes growth, it won't happen.

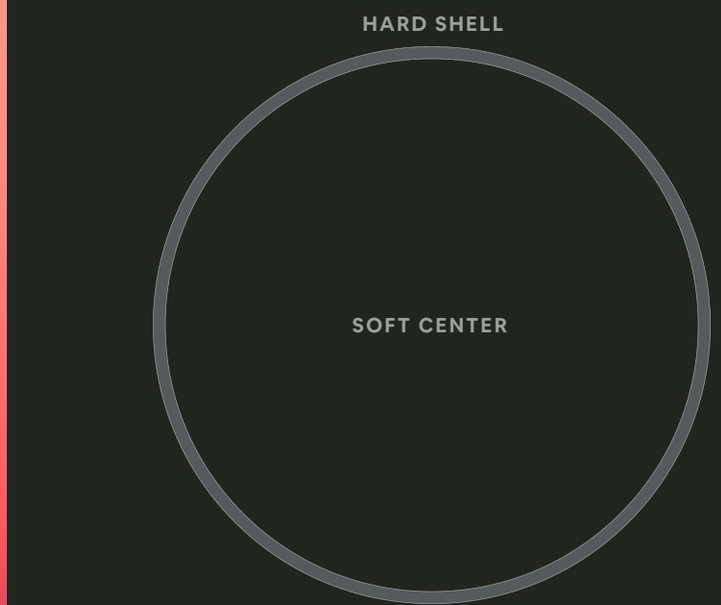
This risk analysis methodology has been around for a long time. So why hasn't it led to more effective security programs?

MAKING SECURITY HAPPEN

Natural Evolution of a Network

As complexity and size grows, security tends to focus on the most immediate threats: external attackers.

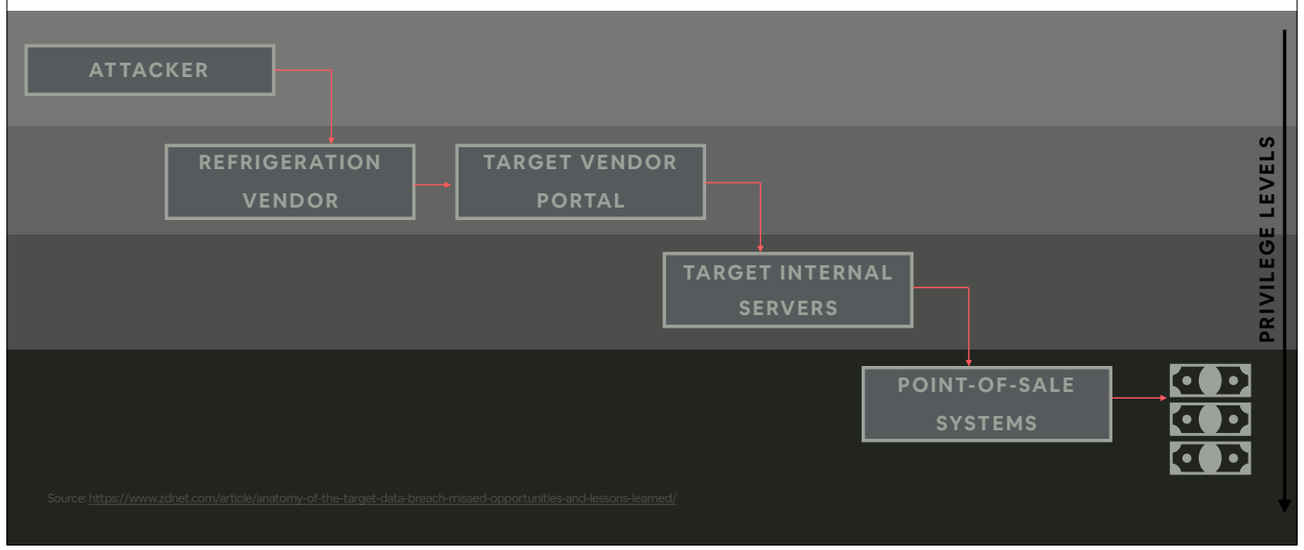
Focus on the perimeter leads to strong outward-facing defenses, but defense-in-depth suffers.



Segmentation remains a good idea
Lots of internal services demand complex connectivity

Case in Point: Target

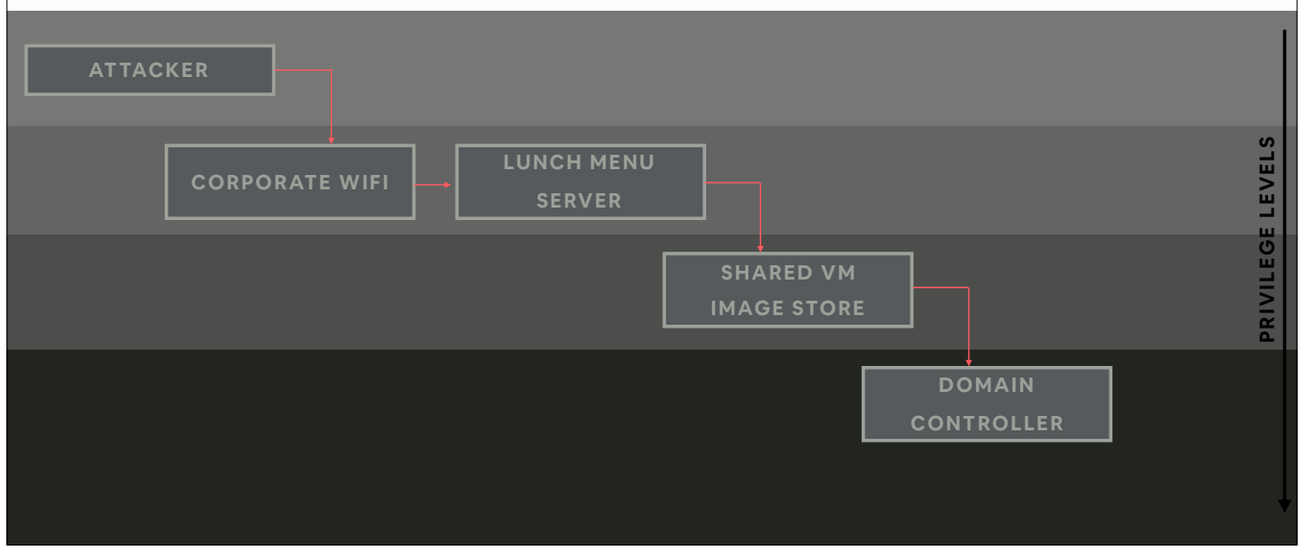
OUR FAVORITE RETAIL BREACH



Source: https://www.zdnet.com/article/anatomy-of-the-target-data-breach-missed-opportunities-and-lessons-learned/

My Personal Favorite

AN UNNAMED SILICON VALLEY COMPANY



Turns out it’s hard to add segmentation.

Easy to preach, hard to do.

Software engineers don’t like configuring firewalls
Actually, most people don’t like configuring firewalls

Change is hard

- Earlier this year, Airbnb had:
  - ~2500 services
  - ~20000 network nodes
  - ~1100 engineers
  - Hundreds of deploys per day
- Developer productivity is a big concern

It’s difficult to change defensive architecture when you already have a network at scale!

How do you switch to a secure internal network?

And *not* halt all development or start over?

People talk about being a security ninja, but the real trick is to apply that to defense and offense.

Defensive Theory

- Don’t build security around the development process; unify security development and software development processes
- “Agile Security”, “DevSecOps”, “SecDevOps”
- Let’s integrate network security with software engineering!

This isn’t a new idea, but it’s most commonly applied to application-layer security projects
There’s a saying that goes around saying that it’s better for developers to be lazy. This is even more true for security engineers — you won’t out-work the attackers, so you need something that scales.

Requirements for Sneaky Network Security

- Solution needs to stay out of the way of engineers
- Security should be there by default, and it should be hard to have an insecure configuration
- Should be as agnostic as possible to how a network service is hosted or what protocols it uses

Use mutual TLS in service discovery for authentication & confidentiality

Discover access lists automatically for zero-config security

Pillars of the Approach

TLS in Service Discovery Proxies

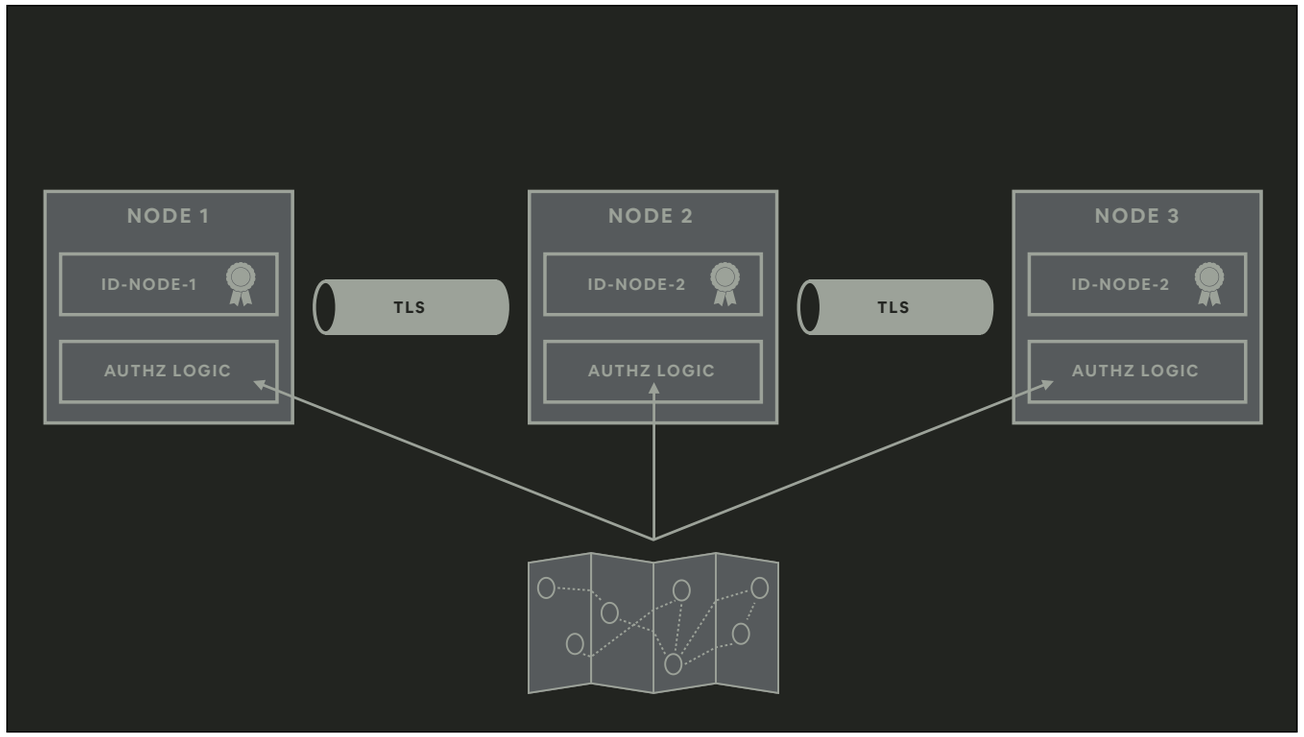
Implement TLS invisibly with proxies deployed as part of your service mesh

Identity Bound to Nodes

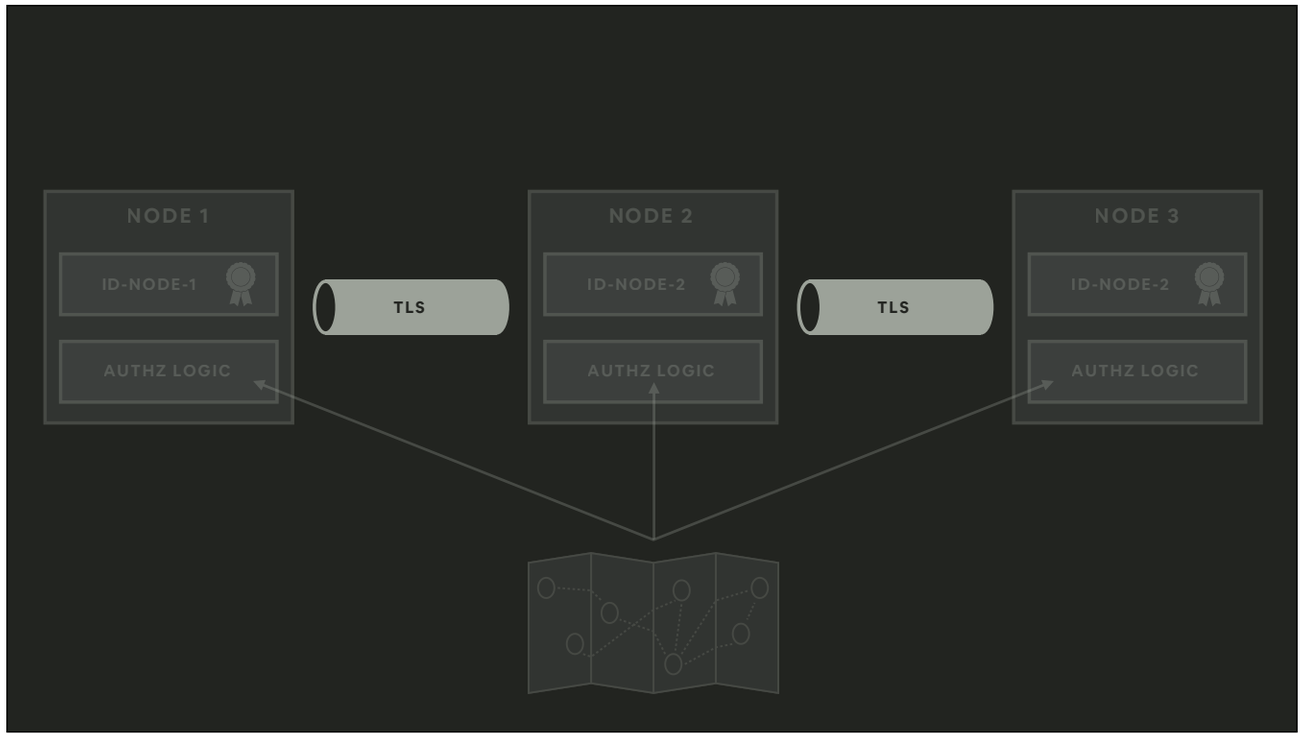
Create certs for nodes in the network based on a strong concept of identity

Generated Authorization Map

Automatically generate authorization rules by analyzing service dependencies


NODE 1
ID-NODE-1
AUTHZ LOGIC
TLS
NODE 2
ID-NODE-2
AUTHZ LOGIC
TLS
NODE 3
ID-NODE-2
AUTHZ LOGIC

PILLAR 1: TLS


NODE 1
ID-NODE-1
AUTHZ LOGIC
TLS
NODE 2
ID-NODE-2
AUTHZ LOGIC
TLS
NODE 3
ID-NODE-2
AUTHZ LOGIC

Mutual TLS

- “Traditional” TLS has the client verifying the identity of the server
- The protocol is flexible enough to support two-way verification
- Allows for strong two-way authentication based on signed key material

Service Discovery

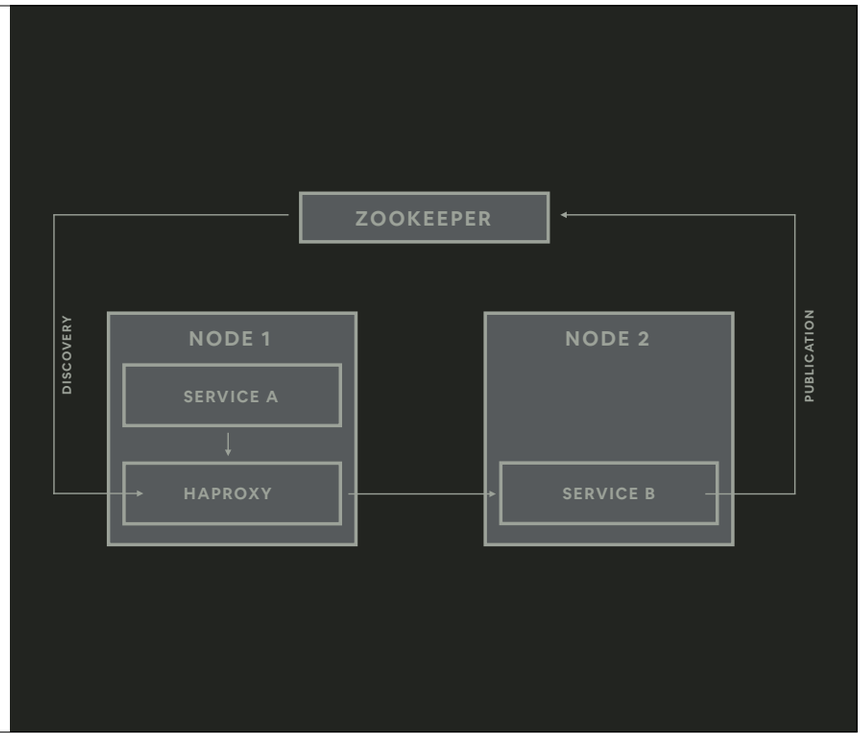
- System for one node in a network to discover other nodes, based on identity or function
- Can be problematic for security if done wrong: it's a map of the network
- Airbnb uses the SmartStack framework, so we used that

SmartStack

An open source service discovery system that uses Zookeeper to share service data and HAProxy to route traffic.

SmartStack is decentralized, and nodes can publish or consume as they please. Security was not originally a design factor!

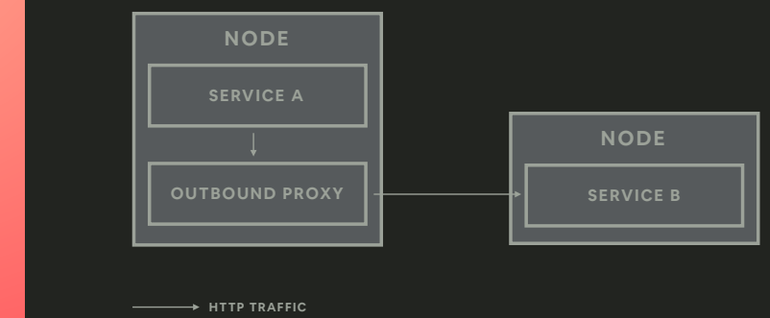
https://bit.ly/smart-stack



Connecting to a Service

THE OLD WAY

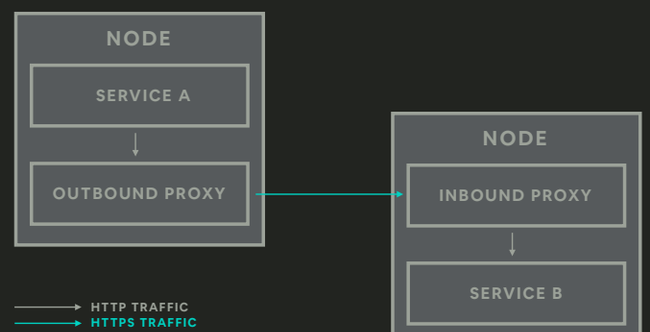
The client sends requests to its local reverse proxy, which sends them to an appropriate backend.



The Secure Shim

"MAGICALLY" INSERTING TLS

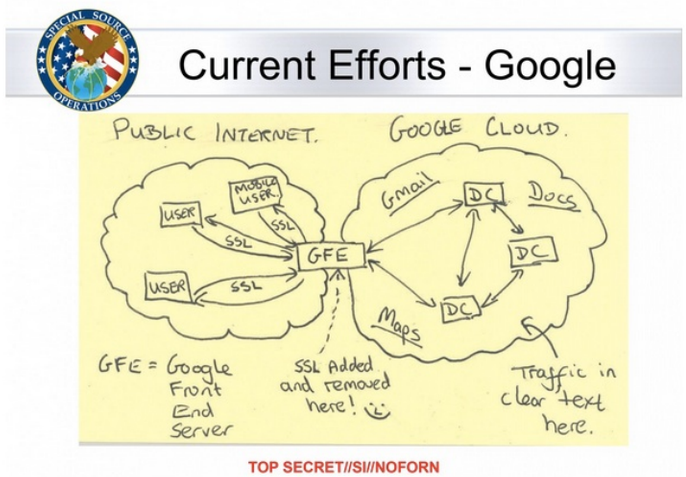
We insert a new reverse proxy in front of the *receiving* service that can catch mutual TLS connections and forward to the underlying service.



Key Benefits to this Approach

- The sending and receiving services do not change — traffic looks about the same to them
- The two service discovery proxies can handle authorization, so security only has to build these controls once
- Having proxies surround your service communications is generally useful (universal metrics, tracing, etc.)

Do the Opposite of What the NSA Wants

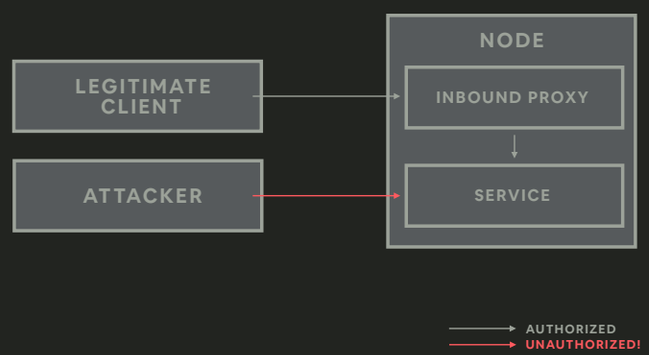


TOP SECRET//SI//NOFORN

Proxy Exclusivity

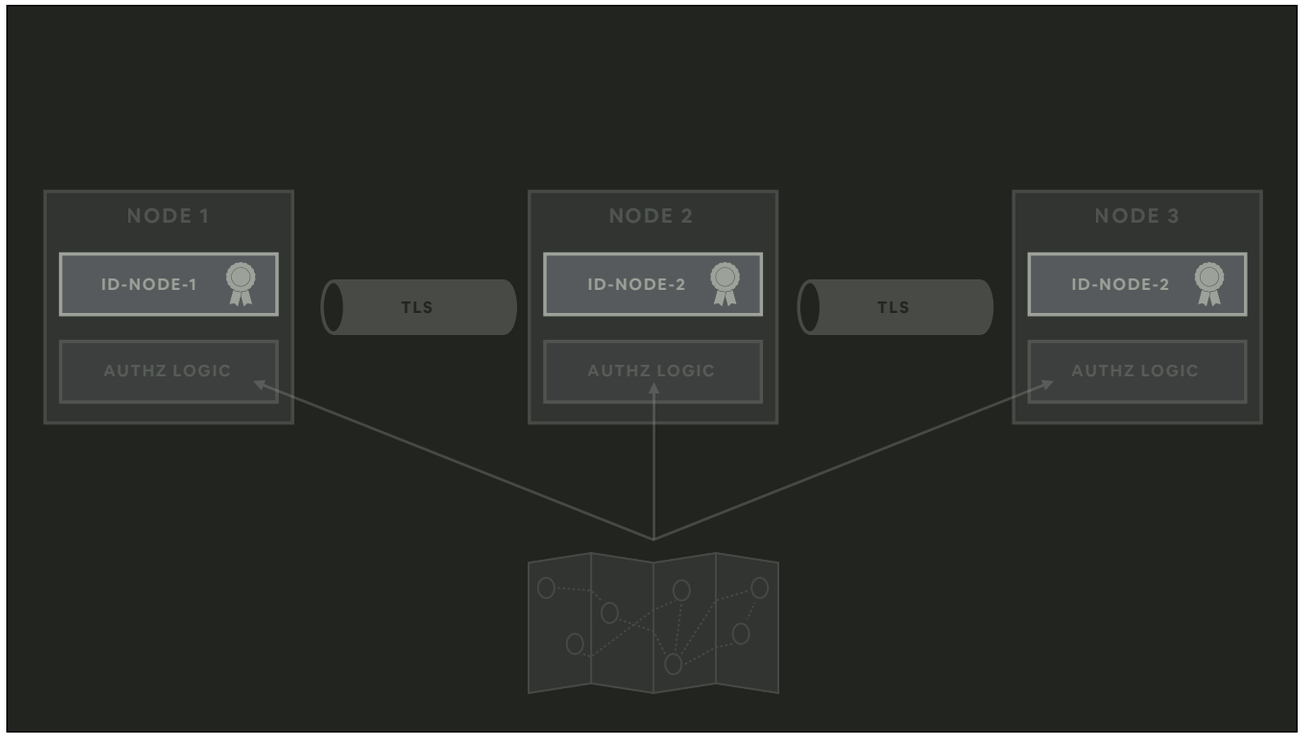
Since the inbound proxy does the authentication, all clients must be forced to use it.

In our network, we dealt with this by binding underlying services to localhost.



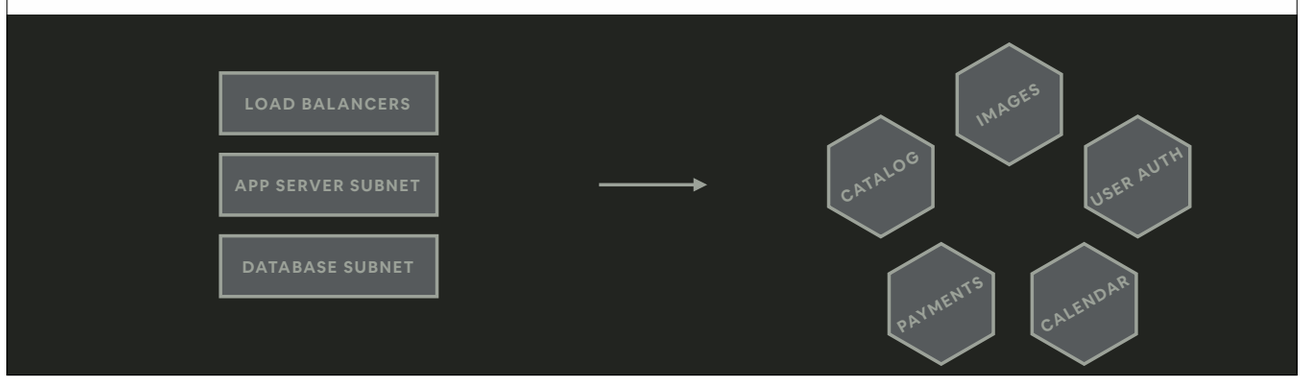
Binding to localhost may not work for your network, especially if you have multiple privilege levels on a single machine. In that case, firewalls or use of local domain sockets may be necessary.

PILLAR 2: IDENTITY BINDING


NODE 1
ID-NODE-1
AUTHZ LOGIC
TLS
NODE 2
ID-NODE-2
AUTHZ LOGIC
TLS
NODE 3
ID-NODE-2
AUTHZ LOGIC

Segmentation

- What if we think about segmentation on the *service* level, not the subnet level?
- Allow the payment config page to call the payment backend service — but don't allow the Slack bot to!



A node in the network should only be able to talk to what it needs to

PUTTING IT TOGETHER

We've got proxies that understand TLS on both sides of our service communication, and TLS is great at verifying identities.

We just need to strongly identify each node in terms of a TLS certificate.

Identifying Nodes

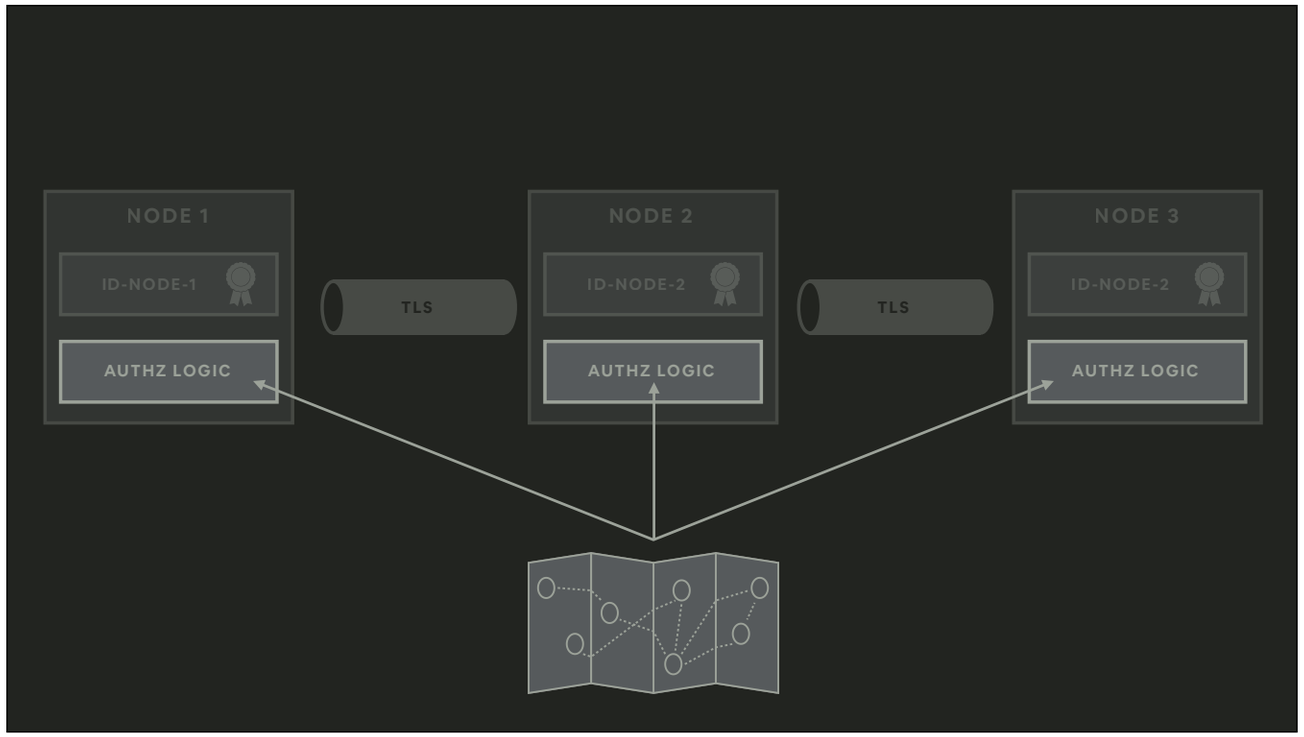
- Need to find an identity that:
  - Is sufficiently varied (more zones are better)
  - Can’t be changed by a node (otherwise nodes can move between zones)
  - Can be detected automatically
  - Can be represented in an X.509 SubjectAlternativeName

Most modern networks have some “role” concept that works well for this

Building Authorization into our Service Discovery

1. Give everything an identity, and distribute certificates that allow nodes to prove it
2. Build a map of what identities should be able to access what services
3. Distribute that map to relevant service discovery proxies, and tell them to enforce it

PILLAR 3: AUTHORIZATION MAP


NODE 1
ID-NODE-1
AUTHZ LOGIC
TLS
NODE 2
ID-NODE-2
AUTHZ LOGIC
TLS
NODE 3
ID-NODE-2
AUTHZ LOGIC

Building a Trustworthy Map

- How do you find out what needs to talk to what?
  - Make a configurable list
  - **Infer it from existing code**
- We assume that if you can merge peer-reviewed code to our config management, you're authorized to make changes.

Inferring the Map

Our Chef repo already had dependency information to make service discovery work.

We can parse this continually and update the map we use for authentication.

CHEF/ROLES/SERVICE1.RB:

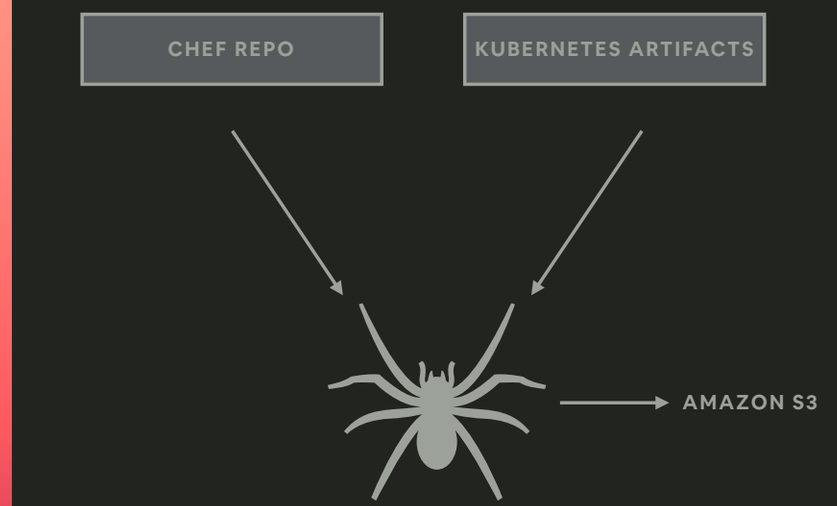
```
name        'service1'

default_attributes({
  'haproxy' => {
    'enabled_services' => [
      'production-db',
      'production-cache',
      'monitoring-service',
    ],
  },
})
```

Arachne

COMPUTING THE WEB

We built a service called Arachne, which processes Chef code and Kubernetes definitions to determine what service dependencies are allowed. It distributes this to nodes via AWS S3.



IT'S ALL ABOUT THE THREAT MODEL

The barriers you put in place to changing the map depend on how you think about the risk of insider threats.

We trust our engineers a lot and let their changes to service config affect access control
Depending on your structure, you may want more controls
The only requirement is that you can reasonably efficiently generate allow-lists for your services

Fine-grained Authorization

Our receiving proxy can inject headers into HTTP streams, allowing us to signal authorization information to application code.

Services can now implement highly detailed access control lists based on caller identity, without implementing any authentication logic themselves.

```
X-Forwarded-Client-Cert:
 Hash=aaf555637e540420d816ef68d048444e9dea
  9a8dfaca1de2a9ac57557a2a4db4;
 Subject="CN=ClientService,OU=Security,
  O=Airbnb,L=San Francisco,ST=California,C=US"
```

Downsides

THERE'S NO FREE LUNCH

- You constantly need to synchronize the an allow-list to your nodes. Caching allows you to use less bandwidth at the expense of greater update latency.
- If TLS has a problem, you have even more problems than you used to
- Adding additional reverse proxies can introduce complexity in traffic flow & signaling
- You need to be able to run software on the nodes receiving traffic, which may not be possible for some vendor software or hosted services
- You'll need to implement certificate revocation, which is usually tricky

ROLLING IT OUT

The Technical Details

WHAT WE MADE THESE COMPONENTS OUT OF

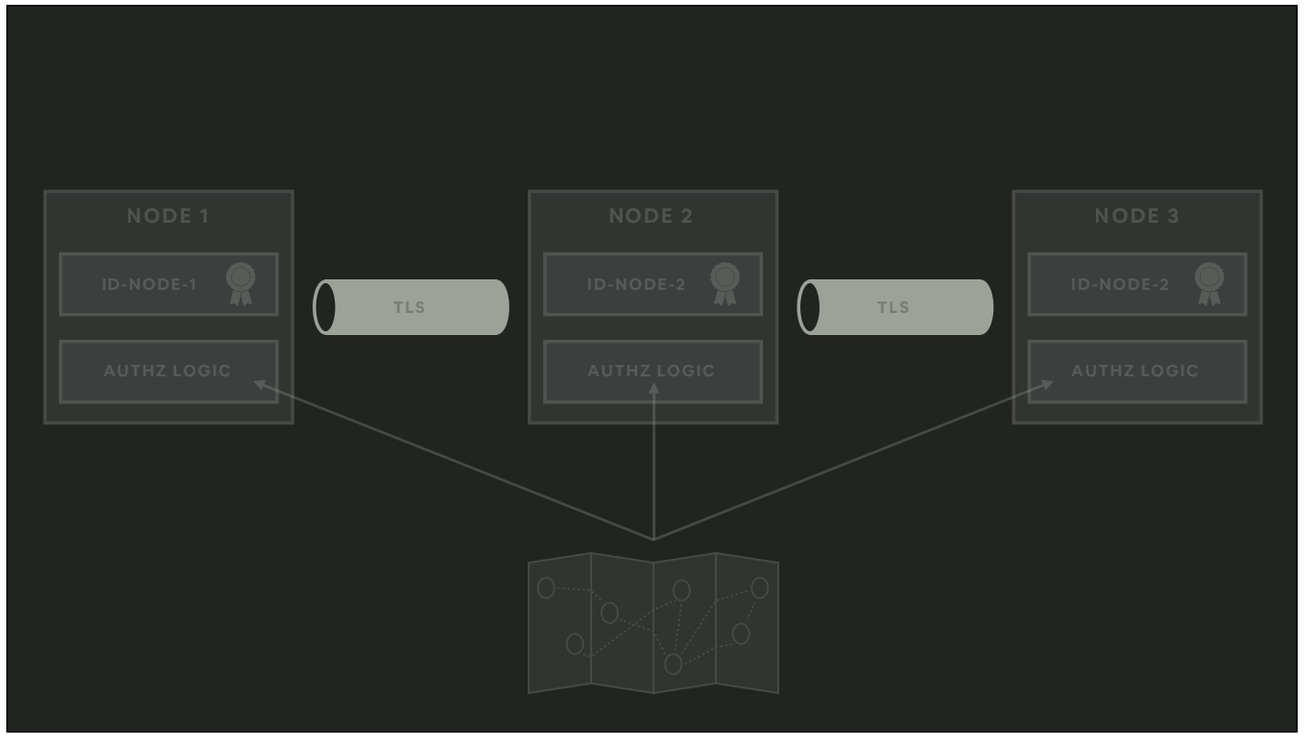
- We used the Envoy proxy to handle incoming TLS connections on the server side
- We gave each node an identifying certificate with a SAN based on its AWS IAM role
- Arachne is a continually running Ruby script inspecting our Chef repo and Kubernetes artifacts
- “Webfiles” (the authorization map) are uploaded to and downloaded from S3
- Average time between topology change and Webfile update: about 4 minutes (usually long before changes are actually deployed)

Availability Considerations

- We relied heavily on caching the output of our map generator
- Incidents in our map generator don't affect production traffic, unless there's a topology change

The Plan

1. Compute authorization map and verify correctness
2. Give everything an identifying certificate
3. Install the receiving proxy everywhere and start listening
4. Configure some services to start using the system
5. Cut everything over
6. Bind services to localhost so the secure proxy *must* be used


NODE 1
ID-NODE-1
AUTHZ LOGIC
TLS
NODE 2
ID-NODE-2
AUTHZ LOGIC
TLS
NODE 3
ID-NODE-2
AUTHZ LOGIC

Things That Went Well

- We went from 14.8% TLS internally to 70.1% in one night
- We ensured there were non-security benefits, getting wider organizational support
- We could disable the system selectively when services had problems with it

And internal use of TLS continued to grow quickly after Day 0.
Non-Security benefits:
- Easier configuration
- Performance
- More metrics available

Things That Didn't Go So Well

- Routing traffic through an inbound proxy can lead to unexpected application behavior
  - All traffic is suddenly from 127.0.0.1
  - Can interfere with stateful things like websockets
- The testing process focused more on inbound effects of the switch, rather than outbound
  - Well-tested: what if all my clients start using TLS?
  - Not well-tested: what if all the services I rely on start demanding TLS?
- Binding services to localhost took longer than expected
  - Inconsistencies in our service configuration made this require many changes

There can also be issues with signaling when services go down. Since there's now a proxy on the receiving end, clients trying to talk to a node that has services which are down (but not yet deregistered from service discovery) will receive layer 7 error responses, not TCP errors. Clients need to realize what's going on, terminate the connection, and try another available node.

Performance

- In our environment, services often got *faster*
- 95th percentile latency went down by as much as 80% in some cases
- Service discovery processes restart infrequently, so they get more benefit from TLS session caching

TLS session resumption rate usually close to 100%, meaning minimal overhead even for services that previously communicated in plaintext

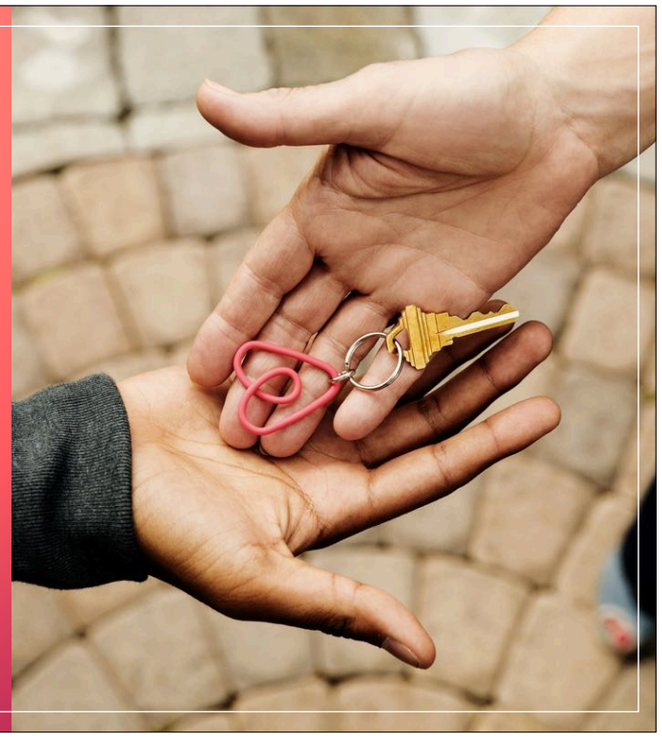
IN SUMMARY

Switching to deeply authenticated networks is possible, because you can make them invisible and fast.

Because of authorization maps, engineers might not even notice that they changed access control rules to talk to a new service. But attackers will find themselves unable to talk to most of the network if they land on a host with an identity that doesn't have access to much.

Istio and Consul also implement this.

DEFENSIVE TECH I LOVE



I believe that project was a good example of using modern tech and a little ingenuity to push security forward.
There are people throughout the industry doing cool stuff like this. Here are three projects I think really exemplify this spirit.

Sites Without Passwords

- Use OAuth2 for social login plus email “magic links”
- Users clearly can’t manage passwords — why let them?
- A user losing control of their email means game over even if they have a password

U2F Security Tokens

- A serious proposal to dealing with phishing
  - Uses web origin in challenge-response protocol
  - Phishing sites can't "proxy" a challenge for a real site
- Easy-to-use crypto

Learn more: https://www.yubico.com/solutions/fido-u2f/

U2F tokens are a good example of why you shouldn't be afraid to consider introducing a totally new type of device to defeat an attack
These problems are serious enough that corporations, governments, and individuals are willing to seriously pay for effective countermeasures

Sandboxes by Default

- They’re everywhere
  - Mobile Apps
  - Containerized software downloads
- Give control of devices back to the users who own them
- The face of malware on iOS and Android is fundamentally different than the old days

Installing an app shouldn’t mean you give full access to your microphone to some developer whose identity you have no concept of

YOUR INFOSEC CAREER

There are a lot of ways to do this

LOTS OF PEOPLE HUNT SECURITY FLAWS

- ~~Criminals~~
- ~~Hacktivists~~

Can't recommend these

- Security researchers
- Pentesters
- Academics
- Defenders
- Governments

Playing Offense

PENTESTING, SECURITY RESEARCH, RED TEAM

- Builds a diverse skillset
- Really fun when your exploits land

This is where I started out. It was a really great place to kick off my post-college journey into security.

Playing Defense

BLUE TEAM, SECURITY PRODUCT ENGINEERING

- You're up against hard problems
- You get to eliminate threats and bug classes, one by one

I'm going to spend a bit more time on discussing your options on the defense side, because it's what I've seen the most diversity in.

At Airbnb we strive to think of new ways to deal with the problems we face. We want to make an environment where engineers at every level have the space to experiment with solutions.

Roles in Defense

SOFTWARE ENGINEER

- Write the tools that implement what you've learned here
  - Crypto toolkits
  - Frameworks to eliminate common bugs
  - Systems to analyze user activity for malicious indicators

Source: https://medium.com/airbnb-engineering/one-step-forward-in-data-protection-8071e2258d16

Roles in Defense

SECURITY ENGINEER

- Know the game — both sides of it
- Guide the development of software to mitigate security risk from the beginning

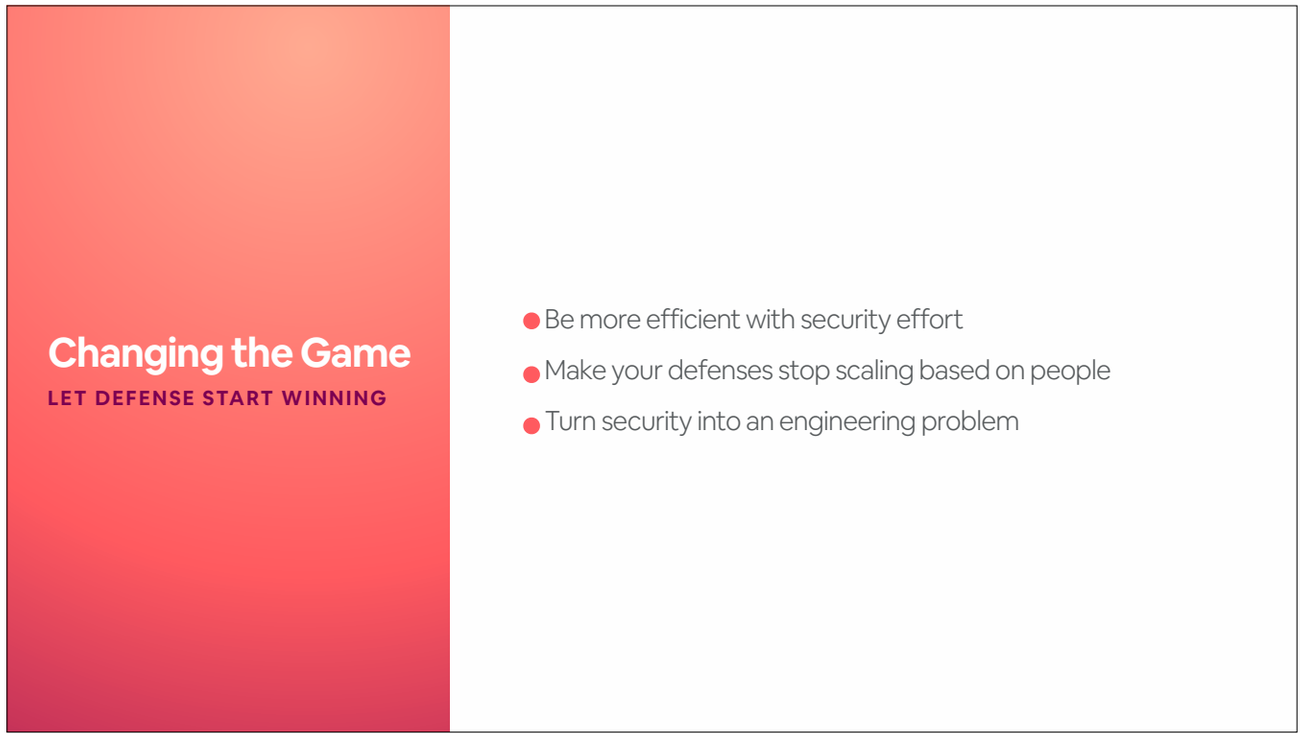
Apply your bug-hunting skills to drive defense forward

Roles in Defense

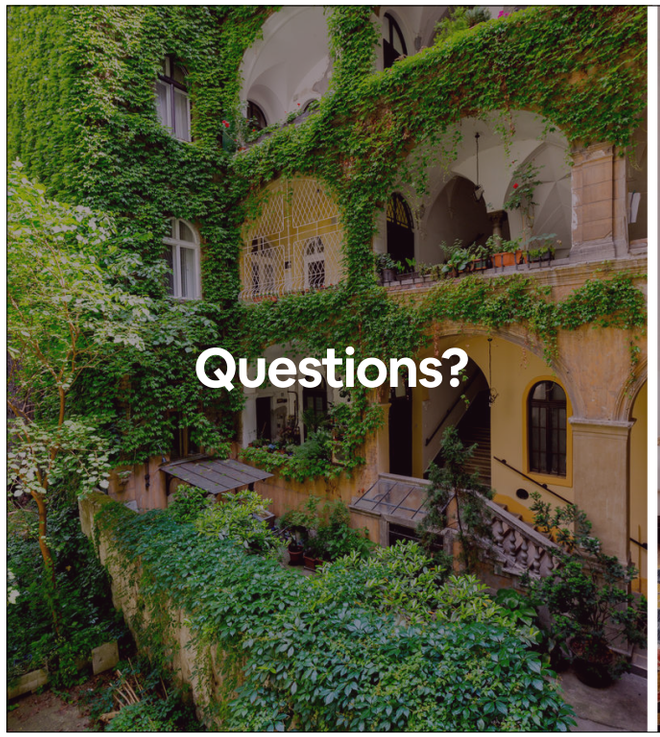
INTRUSION DETECTION ENGINEER

- Don't give the adversary a moment's rest during their attack
- Extract valuable signals, identify events, and respond with speed

DFIR — Data Forensics and Incident Response



Regardless of whether you work on the offensive or defensive side of this, I think we can all agree that we want things to get compromised less. Regular people shouldn’t have their credit cards stolen every month, elections should stay uninfluenced, and nobody should need to fear that the photos on their phone will end up on the internet.


Questions?


Stay Connected
@maxb (Twitter)
max.burkhardt@airbnb.com


airbnb