OWASP
Open Web Application
Security Project

Portland OWASP Training Day 2019

What Is OWASP

OWASP

Projects

Conferences

Chapters

The Portland Chapter

Twitter: @PortlandOWASP

Website: pdxowasp.org

Meetup: OWASP-Portland-Chapter

Slack: owasp.slack.com #chapter-pdx

Next Chapter Meeting

Threat Modeling in 2019 with Adam Shostack

New Relic

Wed, October 9th 6-8pm

https://www.meetup.com/OWASP-Portland-Chapter/

AWS Threat modeling and Automation

Kendra Ash @securelykash - Ryan Nixon @taiidani

Who are you?
QA, DEV,
Security?...

Agenda

Kendra and Ryan

Introduction and Background

Threat Modeling

Support / Adoption

Card Game

Audit Tool

Feedback

Future State

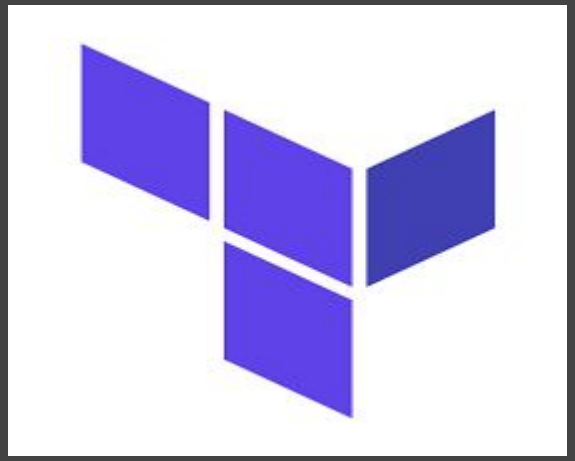
@securelykash @taiidani

Introduction
Who are we?

A Little about us

@securelykash @taiidani

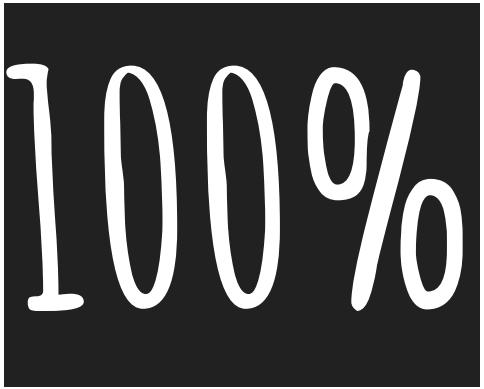

vacasa
Unbeatable revenue
Advanced rate optimization
Local caretaking teams
Dedicated property managers
Vacation rentals you can
vacasa
Love your job.
Better management means better vacations.
vacasa
vacasa
24/7 guest support
Professional care + housekeeping
3D virtual tours
15,000+ vacation rentals around the world
@securelykash @taiidani



TypeScript

Environment at Vacasa

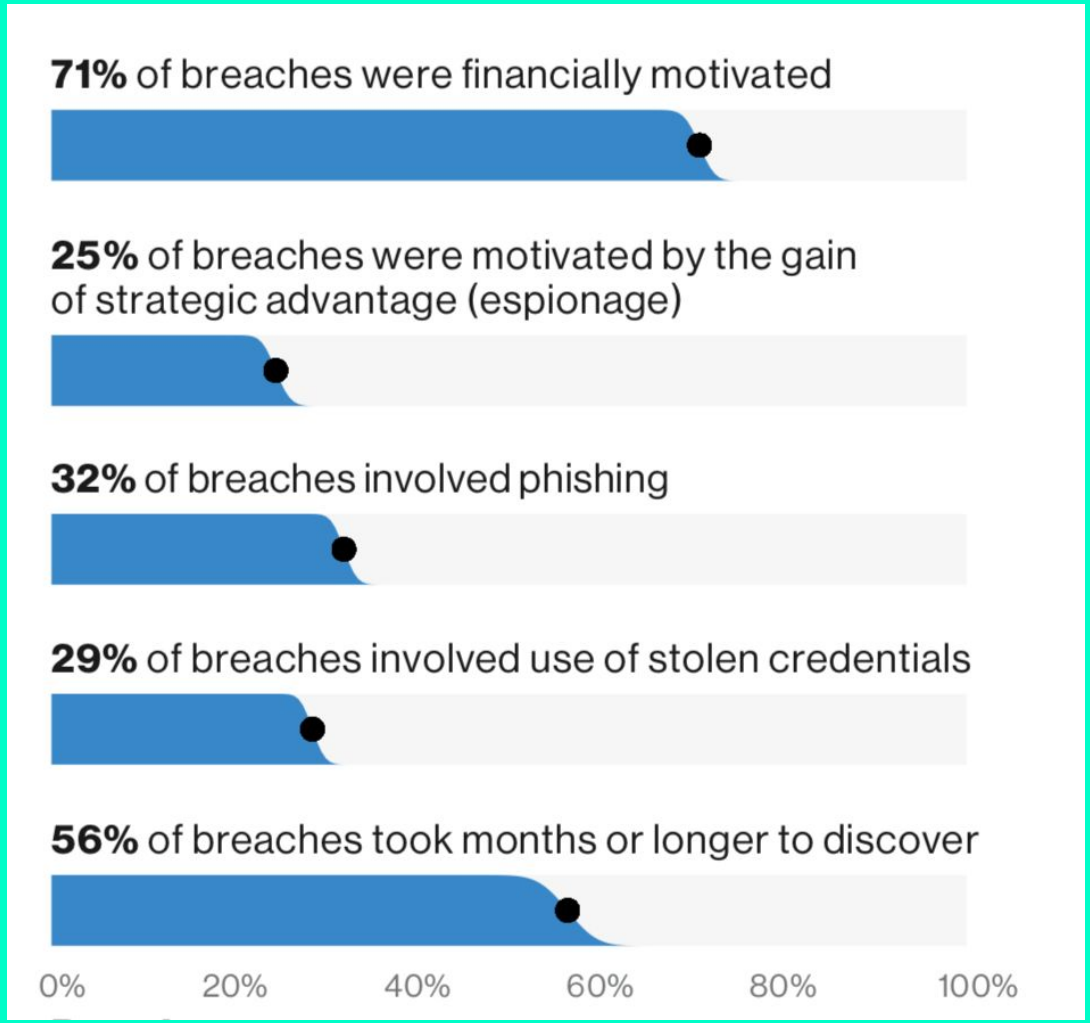
@securelykash



Attempting to make Vacasa a more secure place!

Verizon data breach investigation report for 2019.

2,013 of the 41,686 incidents were confirmed data breaches.



Threat

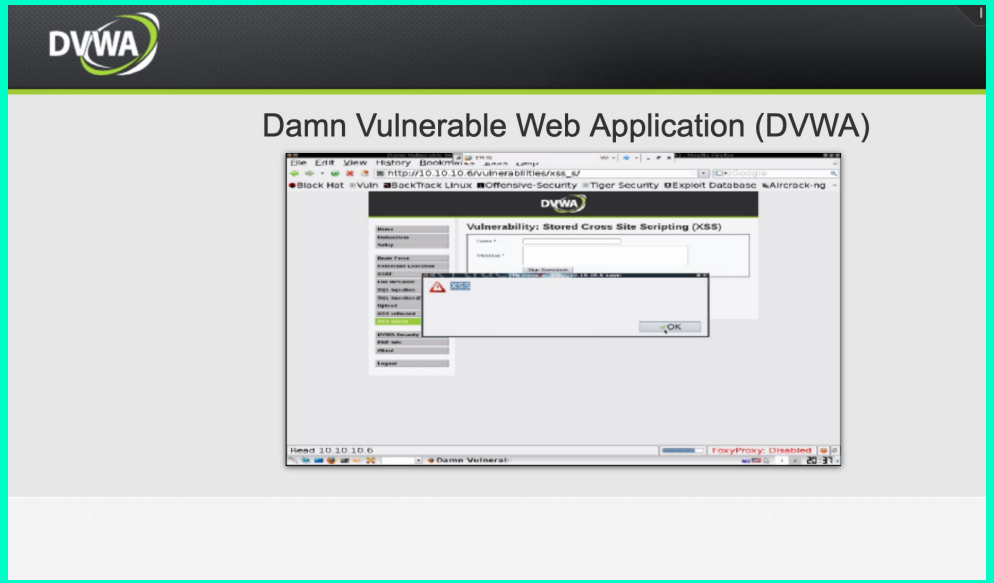
Vary depending on technologies

People

Robots

Mother Nature

THE POSSIBILITY OF A MALICIOUS ATTEMPT TO DAMAGE OR DISRUPT A COMPUTER NETWORK OR SYSTEM



http://www.dvwa.co.uk/

https://bkimminich.gitbooks.io/pwning-owasp-juice-shop/content/

@securelykash

OWASP ServerlessGoat

OWASP ServerlessGoat is a deliberately insecure realistic AWS Lambda serverless maintained by OWASP.

Enter a URL of a Word Doc (.doc) file to convert:

https://www.puresec.io/hubfs/document.doc

PURESEC AND OWASP FOUNDATION.


STRIDE

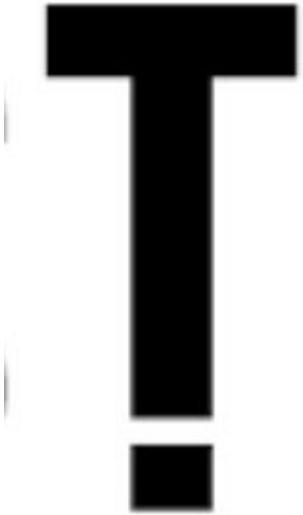
History of STRIDE

Created at Microsoft by: Praerit Garg and Loren Kohnfelder a

When: 1990's

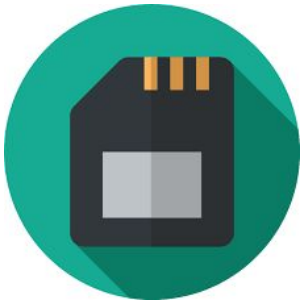

Email spoofing
Jean Quinn
From: Kelly Adams
Sent: 4/11/19 9:13 AM
To: Keith Jones
Subject: Data

SPOOFING



Modifying something: code, configuration, or data.

Tampering with networks, memory, or files.



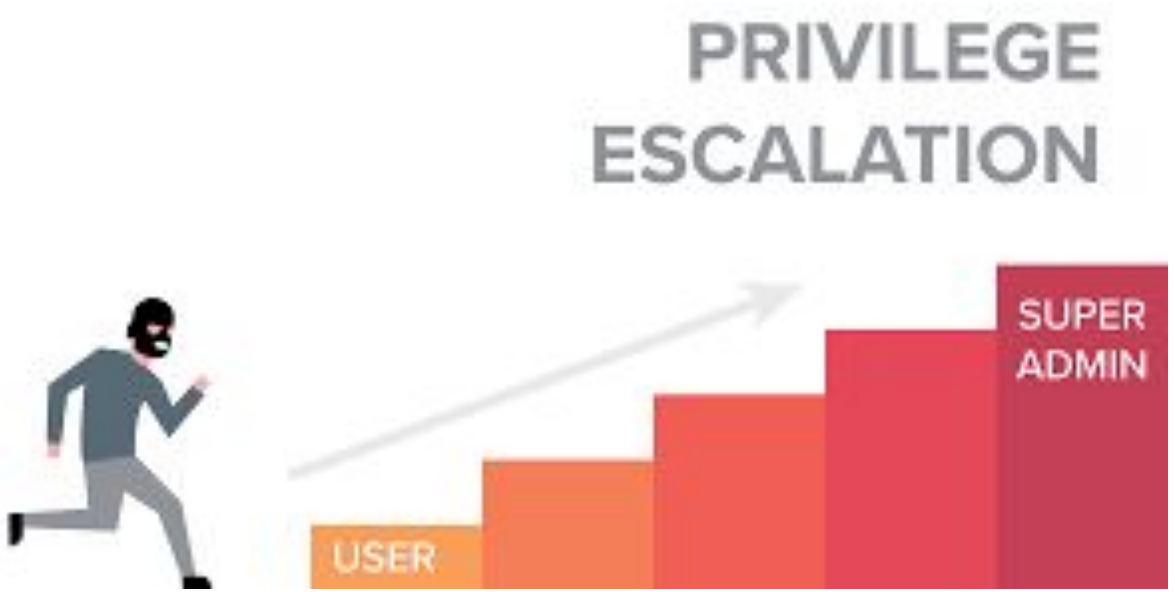
Tampering

R

Repudiation

Information disclosure

Denial of service



Escalation of privileges

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

OWASP TOP 10 – 2013
A1 – Injection
A2 – Broken Authentication and Session Management
A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References
A5 – Security Misconfiguration
A6 – Sensitive Data Exposure
A7 – Missing Function Level Access Control
A8 – Cross-Site Request Forgery (CSRF)
A9 – Using Known Vulnerable Components
A10 – Unvalidated Redirects and Forwards

History

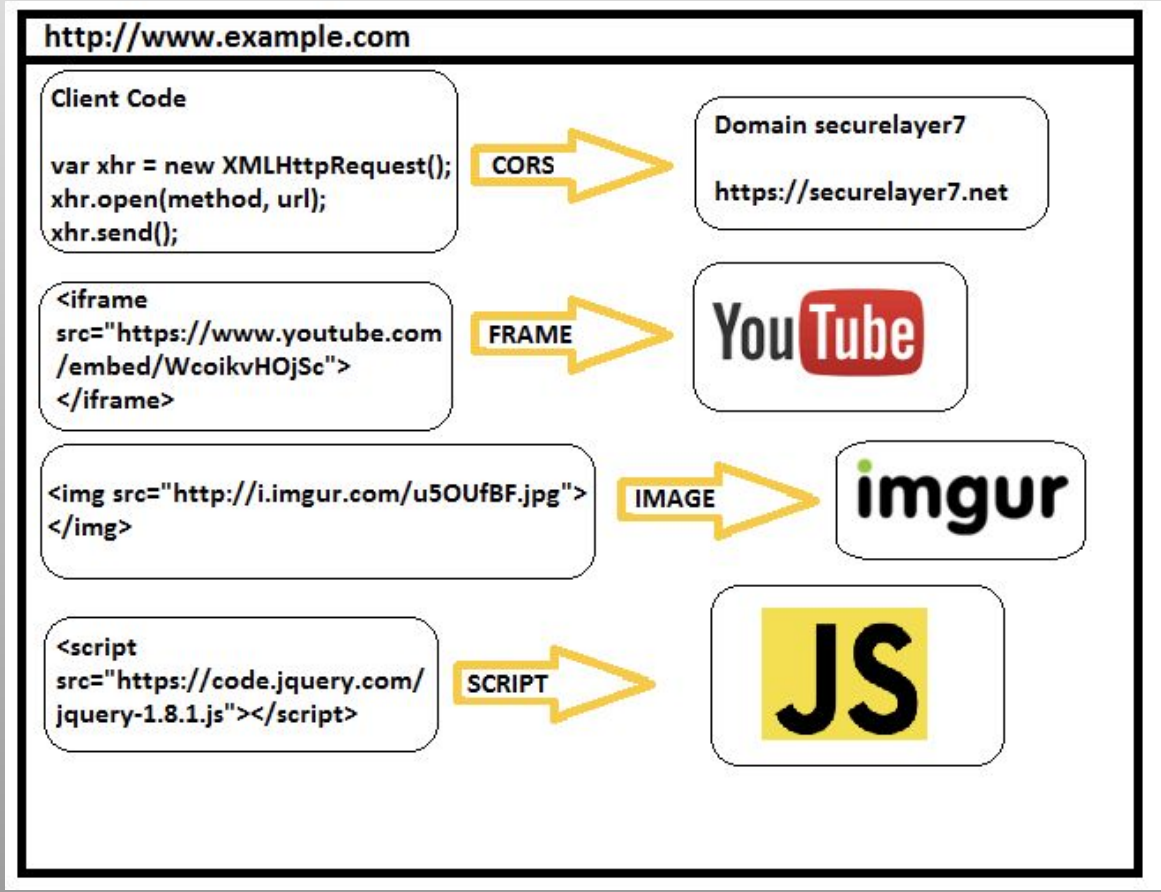
Of OWASP top 10

When: Fall of 2001

Who: Mark Curphey

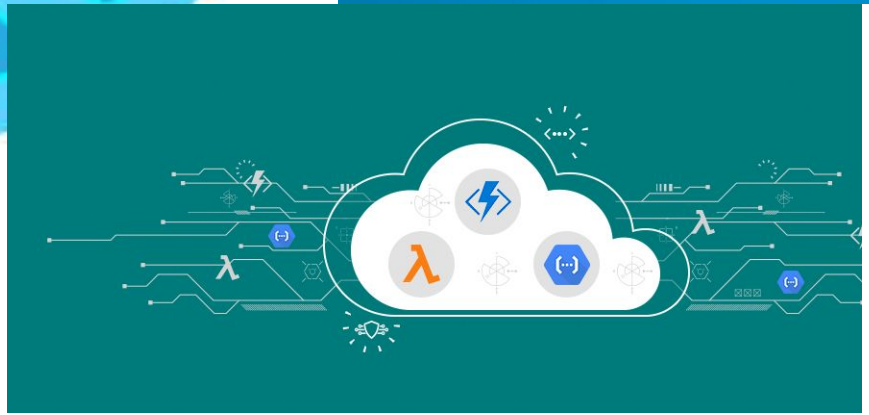
Non profit was started in 2004.

Cross site scripting (xss)



Logging and Monitoring

What do we think this is?



Serverless

Event Injection
Broken Authentication
Sensitive Data Exposure
Insecure Cloud Configurations
Broken Access Control
Denial of Service (DoS)
Overprivileged Functions
Logic Vulnerabilities
Vulnerable Dependencies
Unhandled Exceptions

Serverless top 10

Injection

Code Injection via API Gateway

Ensure deserialization for the data in the request is secure.

https://{string}.execute-api.{region}.amazonaws.com/{st

age}/order

Command Injection via S3 Bucket

Filename

Subject

Messaging Protocol (MQTT)

Injection in Queue System

Who are the messages coming from?

What are in the messages?

Idempotent?

BROKEN AUTHENTICATION

Who has seen examples of broken auth?

Incidents found by Verizon data breach report 2019 are Web Application Attacks.

Web application attack defined as:

Any incident in which a web application was the vector of attack.

This includes exploits of code-level vulnerabilities in the application as well as thwarting **authentication** mechanisms.

EXAMPLES

Endpoints without authentication.

Identity provider tokens not being verified for expiration.

Missing verification of claims / permissions.

10 minute break
If needed?

Who deletes data?

Get all endpoints?

Get all the orders in the database?

Get all the vacation rentals for a homeowner?

Get all reservations?

Get all accounts for the entire company?

Sensitive Data Exposure

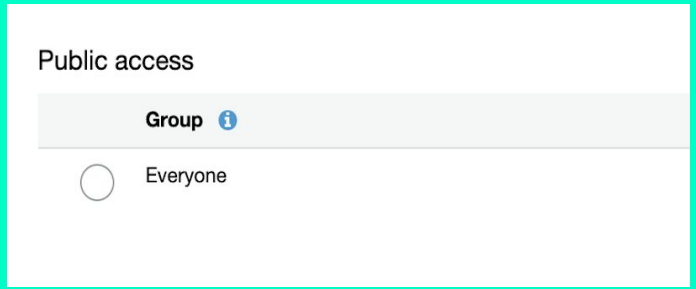
Exposing more data than the user needs to perform a task.

Giving all users admin level access to information.


Insecure cloud configuration

S3 BUCKETS

Accidently given public access?



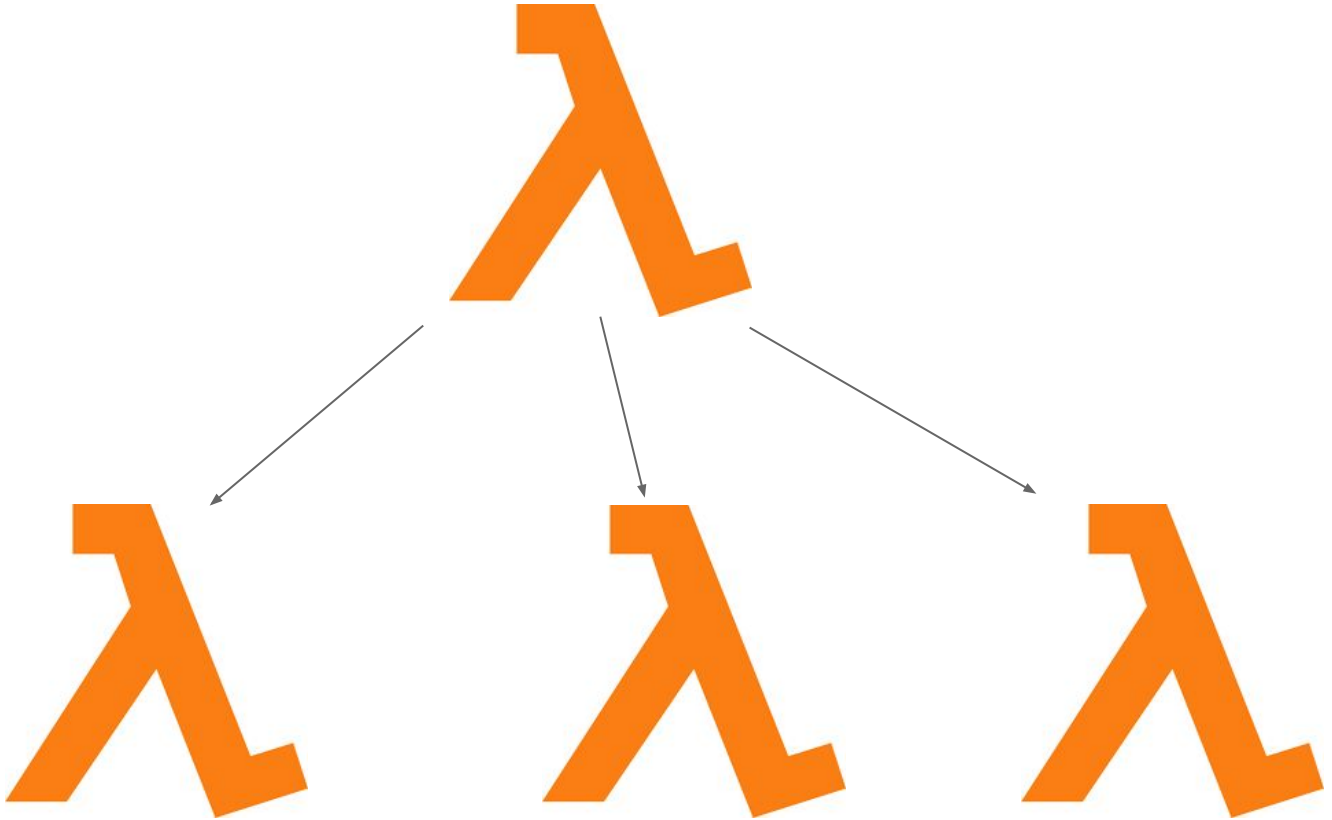
Capital one incident

Rumor has it, the S3 bucket was not the issue.
By: J Cole Morrison

Misconfigured firewall, waf, security group etc.

IAM Role allowed for S3 access to those 700+ Buckets

Lambda functions



How many lambdas does a lambda have permission to invoke?

@securelykash

Broken Access Control

Limit access for all functions.

Follow least privilege.

Single IAM role for each lambda function.

Overprivileged functions

Protego Labs “has found that almost all functions are configured with more permissions than what they actually need.”

Logic Vulnerabilities

When a system allows an attacker to make requests out of order for an established workflow.

Time of check to time of use (TOCTOU), also known as a race condition.

⚠ **We found potential security vulnerabilities in your dependencies.** Dismiss

Some of the dependencies defined in your `package-lock.json` have known security vulnerabilities and should be updated.

Review vulnerable dependencies

Only the owner of this repository can see this message.
Learn more about vulnerability alerts

Vulnerable dependencies

Leaking stack traces, sensitive info, primary keys, etc.

Unhandled exceptions

(error messages)

Not exposing queries, code, or information.

Test and review all error messages.

Unsanitized Logging

Not exposing queries, code, or information.

Review all log messages.

Not exposing personal information.

Subdomain takeover

"A" records that point at Elastic IPs you no longer own.

{attacker choice}.domain.com

5-10 minute
Break if needed

Take 10 minutes to look at the serverless Goat app

Some ideas

Individually look at the serverless goat https://www.serverless-hack.me/

XSS

Open web inspect in the browser

Upload a file

Upload a fake file

Break up into groups of 5

Security UNO

Demonstrate

Use previous discovered threats.

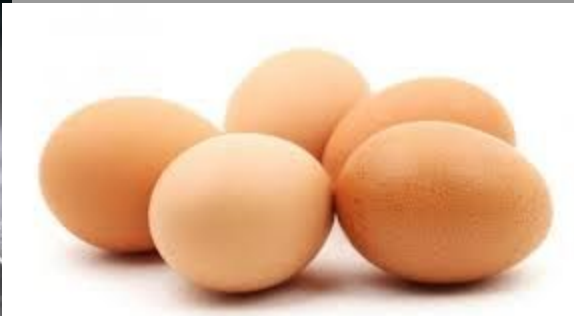
OWASP top 10

Serverless top 10

Play the same number or color.

What did you find?


Who likes chocolate chip cookies?
@securelykash



Ingredients

Ingredients

Dry ingredients

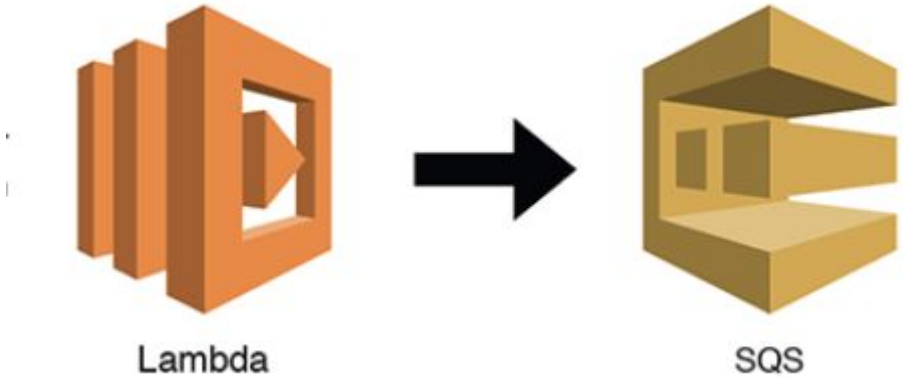
Flour

Baking Powder

AWS model

Single lambda

SQS Queue



Primary ingredients

Butter

Eggs

Vanilla

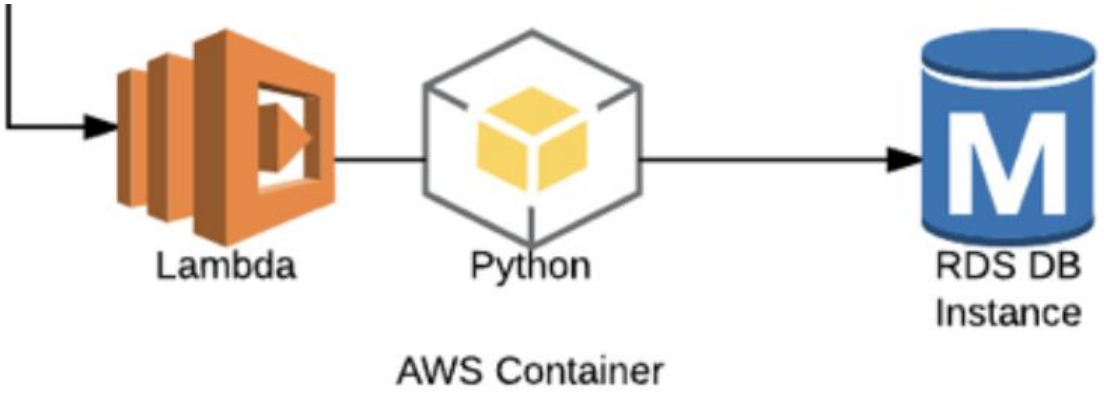
Brown Sugar

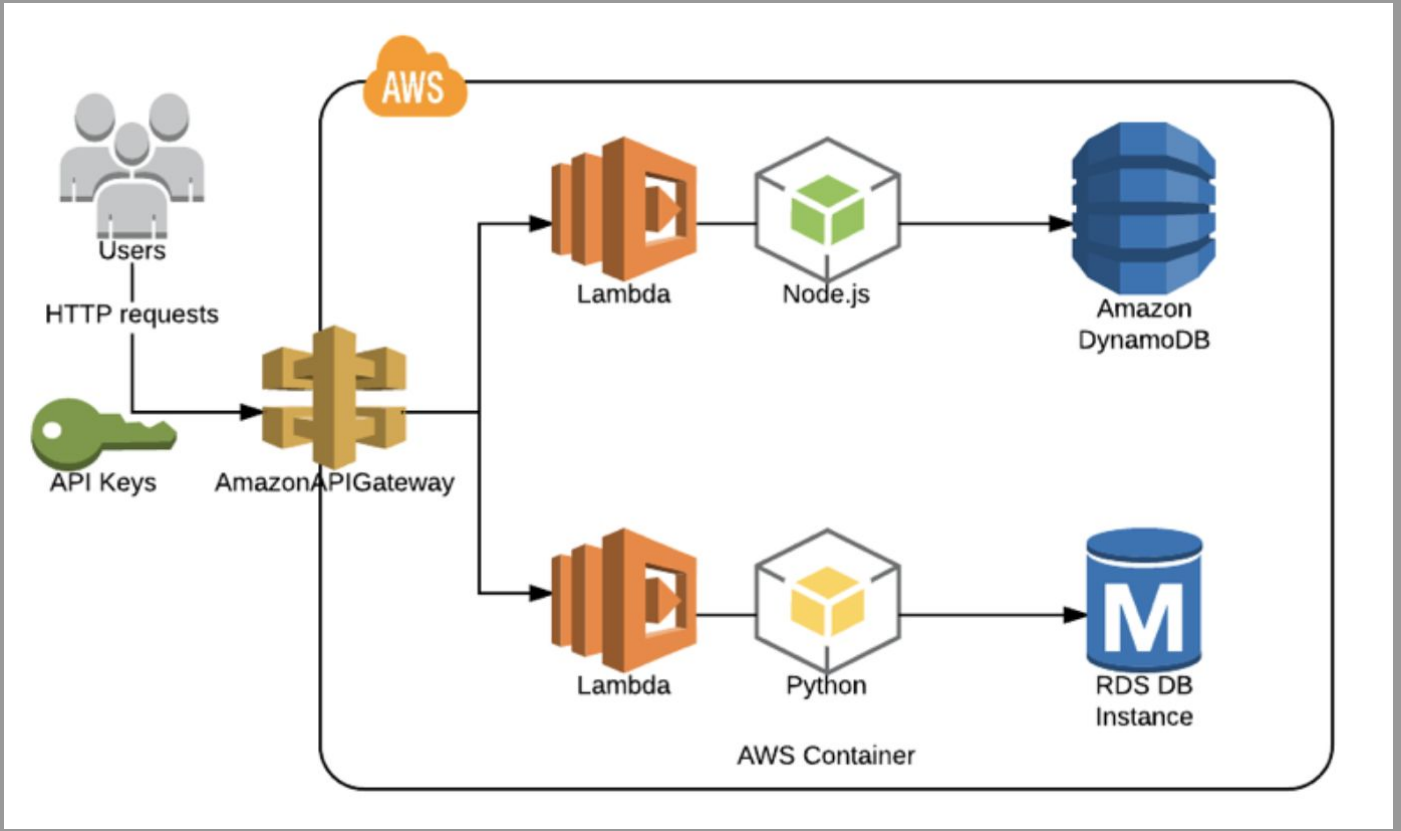
AWS model

API Gateway

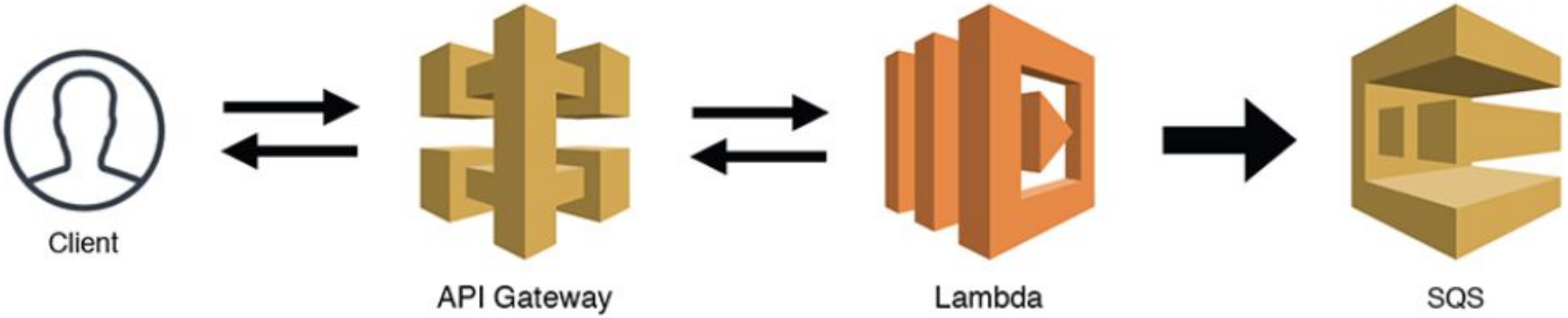
Single lambda

SQS Queue



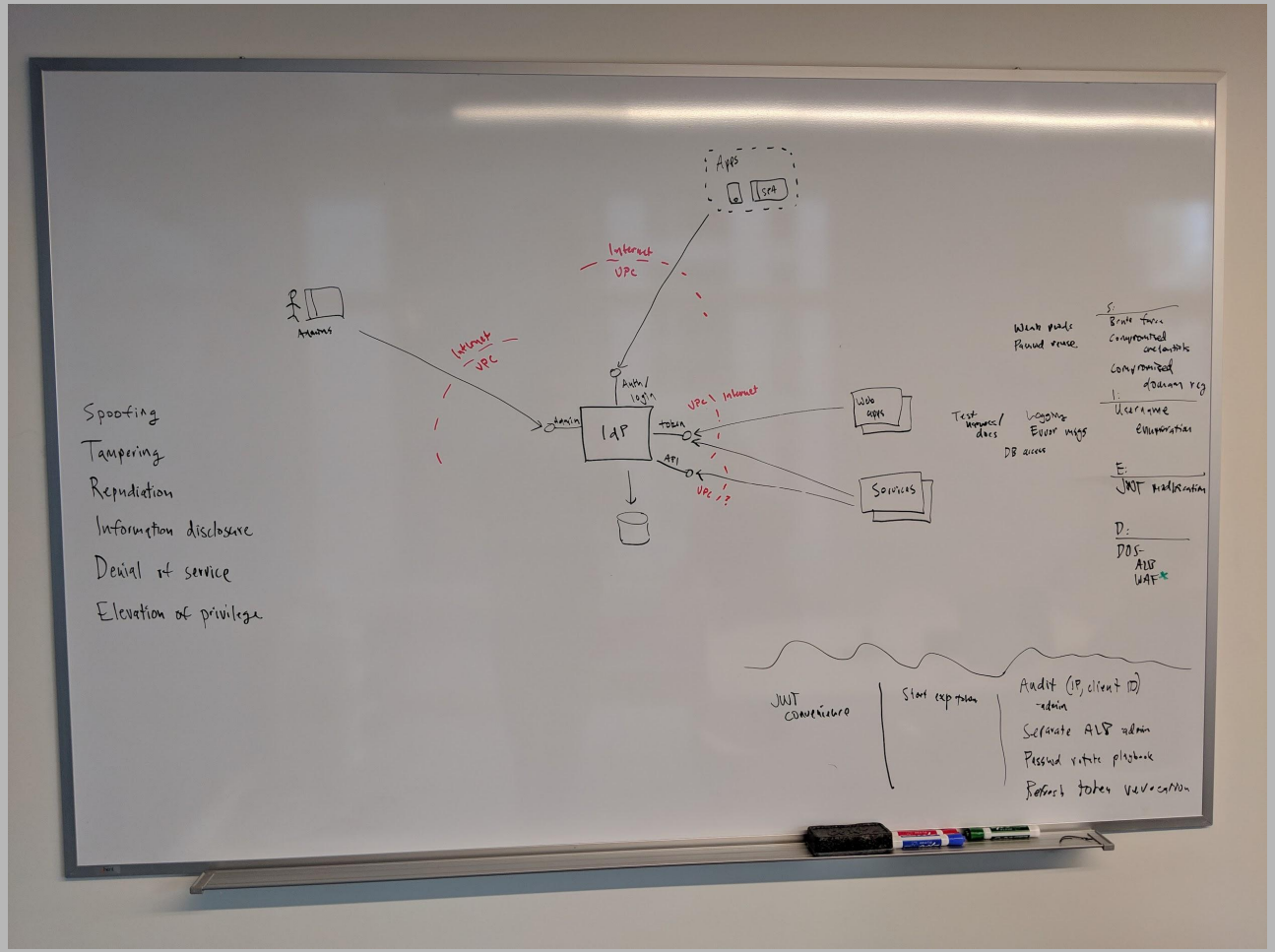


System Level - all the ingredients



System Level - all the ingredients

Example Diagram



@securelykash

Models

System Level

Feature

Epic

Domain

"Representation of a system"

Threat + Model

"Threat Modeling
Is about using
models to find
security problems"

A way to optimize security by identifying threats, vulnerabilities and risks, working towards mitigating or preventing them.

WHEN

Anytime

Design Phase

Every Major Architecture Change

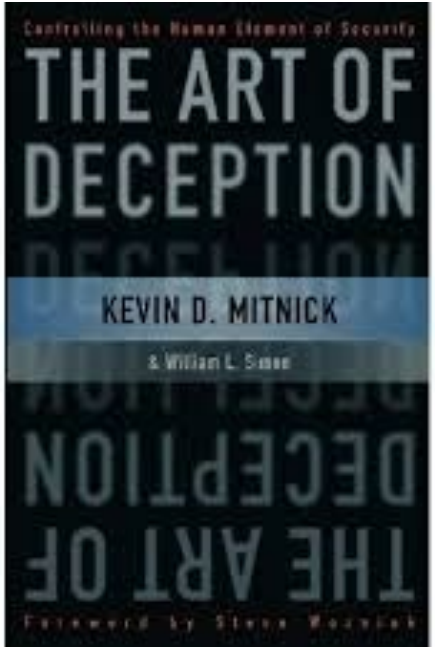
Go Live

Epics

What are you building?

What can go wrong?

@securelykash



Phishing is more than email...Phones and forgotten passwords


Model
Found threats
Discovered Vulnerabilities
@securelykash

MITIGATION

Refers to policies and processes put in place by companies to help prevent security incidents and data breaches as well as limit the extent of damage when security attacks do happen.

MITIGATION

Reducing impact and likelihood to be discovered by an outsider.

Never trust input or make any assumptions about its validity.

Always validate or sanitize input.

Never pass user input directly to any interpreter.

Nutrition Facts Serv. Size: 1 oz (28 g/1 oz), Servings: 8, Amount Per Serving: **Calories** 100, Fat Cal. 35, **Total Fat** 4g (6%DV), Sat. Fat 0g (0%DV), *Trans* Fat 0g, **Cholest.** 20mg (7%DV), **Sodium** 35mg (1%DV), **Total carb.** 15g (5%DV), Fiber 1g (4%DV), Sugars 8g, **Protein** 3g, Vitamin A (0%DV), Vitamin C (0%DV), Calcium (4%DV), Iron (2%DV). Percent Daily Values (DV) are based on a 2,000 calorie diet.

INGREDIENTS: Gluten Free Flour:(Brown rice flour, sweet rice flour, tapioca starch, cornstarch, potato starch), Pure Cane Sugar, Fresh Eggs, Almond Flour, Almonds, (Dried Unblanched), Cranberries (Dried Sweetened), Fresh Orange Juice, Baking Powder; Aluminum Free(Sodium Acid Phosphate), Pure Vanilla Extract (water, alcohol, vanilla extractives), Fresh Orange Peel, Almond Extract

Contains: Eggs, Almonds

What if I was allergic to peanuts(or event injection)?

Mitigation by Thorough logging

Logging of API access keys related to successful/failed logins (**authentication**)

Attempts to invoke serverless functions with inadequate permissions (**authorizations**)

Successful/failed deployment of new serverless functions or configurations (**change**)

Changes to function permissions or execution roles (**change**)

Examples:

Validate all input.

Review all error messages.

Requirements...and product

Making threat modeling fun!

Card games with the whole team!

Escalation of privileges

By: Adam Showstack

Start with a model of the system.

Facilitator, notes keeper and team.

Follow instructions.

Escalation of Privilege

Example cards.

Available on Github

Escalation of privileges

Simplified version

Play like it was UNO.

Match the color.

Discuss each threat (card)

Time

How long does it take to play?

Recommend blocking out 2 hours for a new application.

Questions?

References and Callouts

https://vacasait.atlassian.net/wiki/spaces/ENG/pages/827719808/Secure+Software+Development+Life+Cycle

https://cheatsheetseries.owasp.org/cheatsheets/Access_Control_Cheat_Sheet.html

https://github.com/OWASP/DVSA/blob/master/AWS/LESSONS/README.md

https://www.protego.io/protego-labs-finds-nearly-all-serverless-application-functions-at-risk/
https://start.jcolemorrison.com/the-technical-side-of-the-capital-one-aws-security-breach/

Example Implementation of Continuous Security

Ryan Nixon @taiidani
Kendra Ash @securelykash

Feel free to interrupt me

I tend to rant…

...I’m also an unapologetic cynic ¯_(ツ)_/¯

...and rather holistic in my recommendations

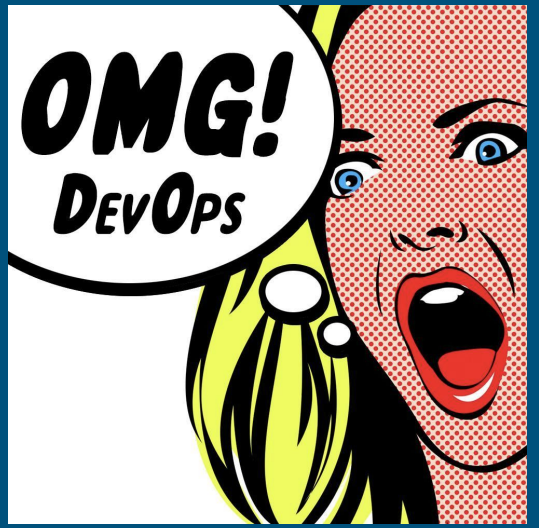
Happy to discuss gray areas and implementation details!

But First, A Story

- What once was…
- A single, flexible deployment pipeline going into a single segmented container orchestration hosting layer. Highly observable. Highly standardized. Highly bottlenecked.
- A centralized DevOps team managing the CI/CD pipeline and all Production ops
- Developers had no direct access to Production, no visibility into the architecture of the app (pre-Infrastructure as Code)
- “Staff up or break up”

But First, A Story

- “The Pivot” - A DevOps Transformation
- AWS Accounts for Everyone!
- You, you, and you. You’re DevOps now. Here’s Root access and tight Product deadlines.
- Who sees where this is going?



Initiate Mutual Grieving Session

The Story, Continued

Engineering

- Cut corners as deadline approached
- Insufficient stability/scalability testing
- No threat modeling
- Self-reviewed Pull Requests
- “We deploy too often for Semantic Versioning.”
- I read about this *amazing* tool on Reddit yesterday. Let’s get it into Production this sprint!

Operations

- Direct deployment to Production, around the pipeline. It’s faster to deploy!
- HTTP is faster than HTTPS. Let’s just use that
- What’s a Postmortem?
- My production application isn’t logging enough, and I don’t use monitoring...how bad could DEBUG=true be?
- If we’re compromised, just “fix it”. We don’t need audit logs.

Kendra's Ingredients!

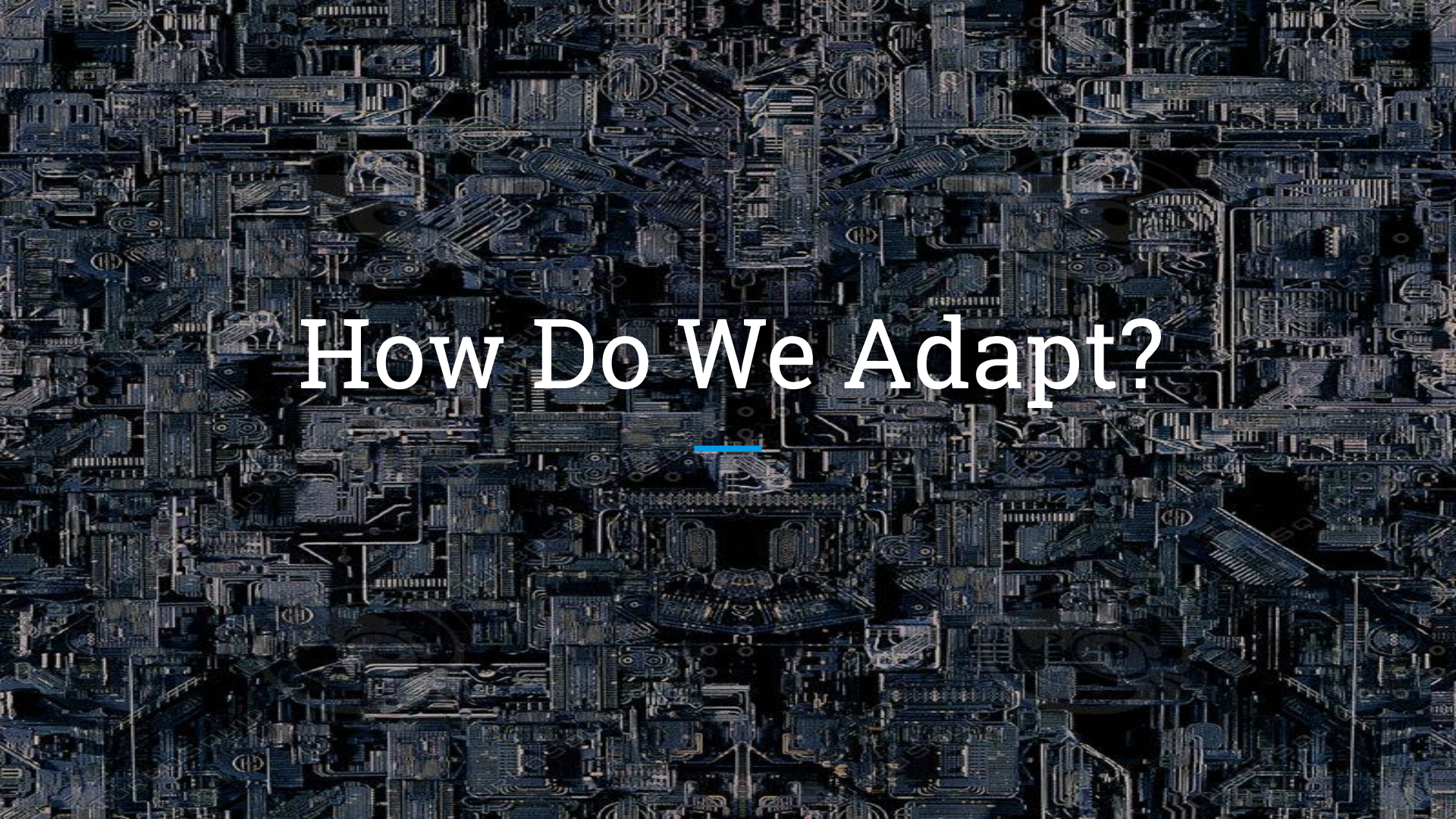
This Could Happen To You!

Engineering

- Faster feature rollouts
- Greenfield keeps developers happy
- Devs understand Production configuration better.
- Incentivizes parity between Local/Dev/Stage/Prod
- Autonomous Engineering teams means less red tape.

Operations

- Saves money on “Slow Ops”
- Isolated teams reduces blast radius of errors
- Devs are on-call for all error types, application and architecture
- Devs are accountable, push back against Product directly rather than Ops trying to


How Do We Adapt?

We Applied Automation

- “Shift left”, a DevOps buzz word.
- Stay out of the bottleneck at all costs.
- Tooling becomes crucial...
  - Scan every Pull Request automatically.
  - Build a “health of the org” report that goes to leadership.
  - Build a self-service “report card” tool that Devs can run.
  - Apply policies *around* the developer infrastructure.
  - Monitor everything.

A Word About Audits

- Audits catch a lot of things, help improve architecture/operations, reduce technical debt, and are extremely helpful to the health of your company!

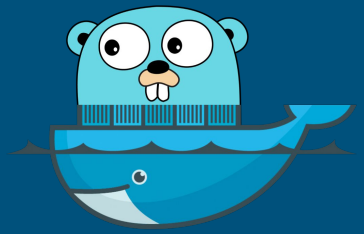
But...

- Having an audit “block” production will slow Engineering down.
- Product can’t put an audit on a gantt chart -- unpredictable delivery times are a problem.
- Engineering has access to Production. What you can’t catch, they can deliver.

You need top-down buy in.

YOU NEED TOP-DOWN BUY IN.

Scan Every Pull Request Automatically



- Introducing Vacabot, a simple Golang-based GitHub App.
- Listens to org-wide webhooks, and reacts.
- Analyzes every Pull Request going out.
- Allows objective, empirical, immediate feedback. Rule sets are visible to whole org and allow contributions.
- Users tend to “just make it pass” which achieves the goal, but may not emphasize the reasons behind it.

✓ **vacabot/audit** — Adding tests is awesome!

✓ **vacabot/audit** — No problems detected. PR passed analysis.

✗ **vacabot/audit** — Test steps are empty.

✗ **vacabot/audit** — Description is empty.

✗ **vacabot/audit** — Please complete all checklists in the PR ✅

CI Siblings

- When every commit is validated alongside the build pipeline, it becomes a natural part of the process.
- Developers don’t have to contact anyone to know how to fix it.
- It encourages standardization! You can get cool auto-labeling of your PR if you follow a standard folder structure.
- With all good CI systems come a few warts…
  - Overrides can be helpful in emergencies.
  - Have some way to re-trigger the build without forcing the user to push again.

Demo

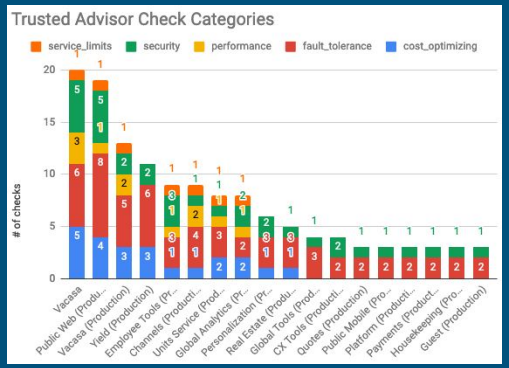
Build a “Health of the Org” Report

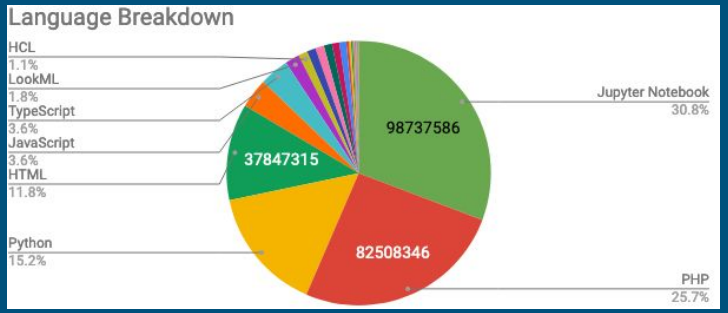
- Central DevOps Monthly Report that goes out to the primary Slack announcements channel every month.
- Visible to all developers and leadership.
- Includes checks such as…
  - IAM accounts needing rotation
  - GitHub reported security issues
  - GitHub abandoned repositories (no pushes in > 6 months)
  - Highly alerting PagerDuty services
  - AWS misconfigurations:
    - Public S3 buckets
    - RDS encryption-at-rest

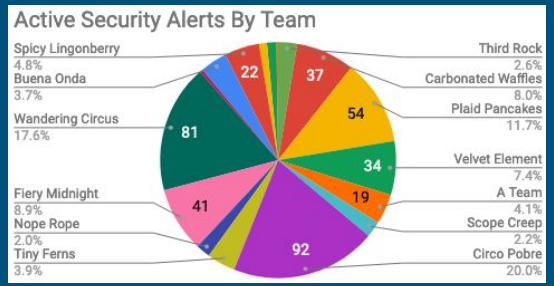
Build a “Health of the Org” Report

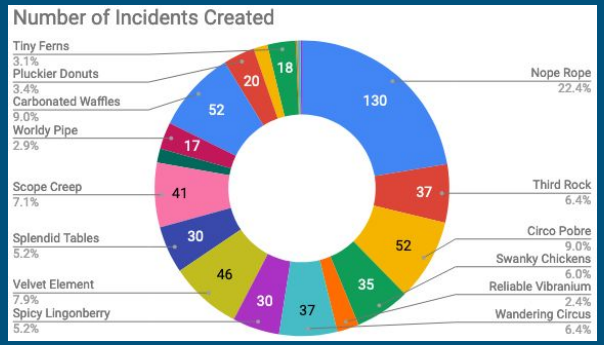
- Get this as *low* in the org as possible. Developers care about their applications, and their on-call schedules -- let them know something’s up.
- Get this as *high* in the org as possible. Without top-down buy in you will only create friction between Engineering and Product.
- Gameify it. Pit teams against each other:
  - “Who can get the highest average code coverage among their repositories?”
  - “Why is my team burning out from PagerDuty when <other team> never gets paged?”
  - “I don’t want to be #1 on a list showing abandoned Pull Requests. Let’s either merge or close ours this sprint.”
  - “Why do we have the fattest repository in the company? Can we reduce our code bloat?”

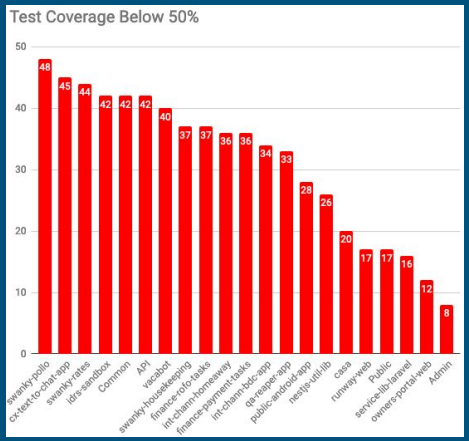
Build a “Health of the Org” Report

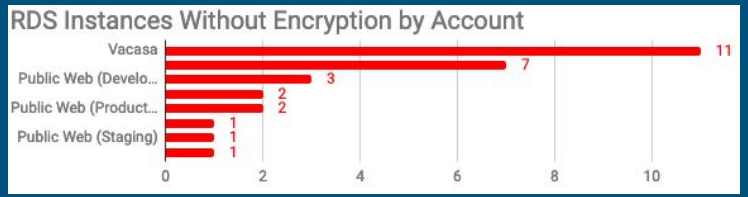












Demo

Build a self-service “report card” tool

- Introducing policy-audit-app, a simple Golang-based CLI tool.
- Scans your GitHub repository, Sentry project, CodeClimate project, and AWS account programmatically
- Produces:
  - A pass/fail result for each check.
  - A “requirement level” that corresponds to the company policies.
  - A “priority level” indicating the threat it could cause to the company if exploited.
  - A Markdown-formatted output that can be placed into version control for tracking the final report.
- Perfect for giving to team leads and leadership!

Policy Audit App

- AWS
  - IAM policies with wide-open permissions
  - IAM users needing rotation
  - S3 bucket misconfigurations
  - Databases without backups enabled
  - Resources without encryption at rest enabled
  - EC2 instances in public subnets
  - LBs and API Gateways without access logs
  - Secrets that haven't been rotated recently
  - Etc. etc.
- GitHub
  - Active security vulnerabilities
  - Branch protections
  - User access / ownership
  - Required files such as README
- CodeClimate
  - Code Coverage levels
- Sentry
  - Appropriate integrations such as PagerDuty

Demo

Shameless Plugs

- Custom Scripted
  - DangerCI
  - Peril
- Static Code Analysis
  - Sonarqube
  - Code Climate
  - GitHub Dependency Scanning
- Infrastructure Validation
  - Atlantis
  - Terraform Enterprise + Sentinel
  - AWS Config
- Or if you have developer resources on hand, build your own!

The Goal

- Monitor everything that you can.
- Automate all the things. Add yourself to every level of the SDLC.
- Produce data that is actionable by leadership and/or Engineering teams.
- Avoid blocking -- focus on identifying risk that Engineering will have to accept or fix in order to deliver.
- Standardization makes a happy company.

Questions?

Thank you!

Survey

http://bit.ly/op2019td-m1