

DevSecOps cloud-Native[®]





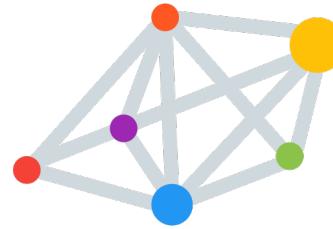
Security in the Cloud

01 RECENT BREACHES

02 WHO IS RESPONSIBLE?

03 WHAT SHOULD WE DO?

04 WHAT CAN WE DO?



200+ Cloud-Resources

"To understand security, we need to consider the big picture, i.e., the full application stack."



BROUGHT TO YOU BY
CODESHIELD.IO



About Me



Johannes Späth, PhD

Co-Founder of CodeShield



Fraunhofer

5+ years experience in applied research
(IT-Security & program analysis / SAST)



Research Intern 2016
Brisbane, Australia

HEINZ NIXDORF INSTITUT
UNIVERSITÄT PADERBORN

PhD in Computer Science



AWS Community Builder 2021



BROUGHT TO YOU BY
CODESHIELD.IO

01

SECURITY IN THE CLOUD RECENT BREACHES



BROUGHT TO YOU BY
CODESHIELD.IO

01

Recent Cloud-Security Breaches

BHIM 04/2020

personal and payment information of 7 Mio. were exposed, due to a misconfigured S3 bucket.



AutoClerk 10/2019

personal data of thousands of hotel guests and members of the US government, military exposed, due to an open elasticsearch database.

CAPITAL ONE 07/2019

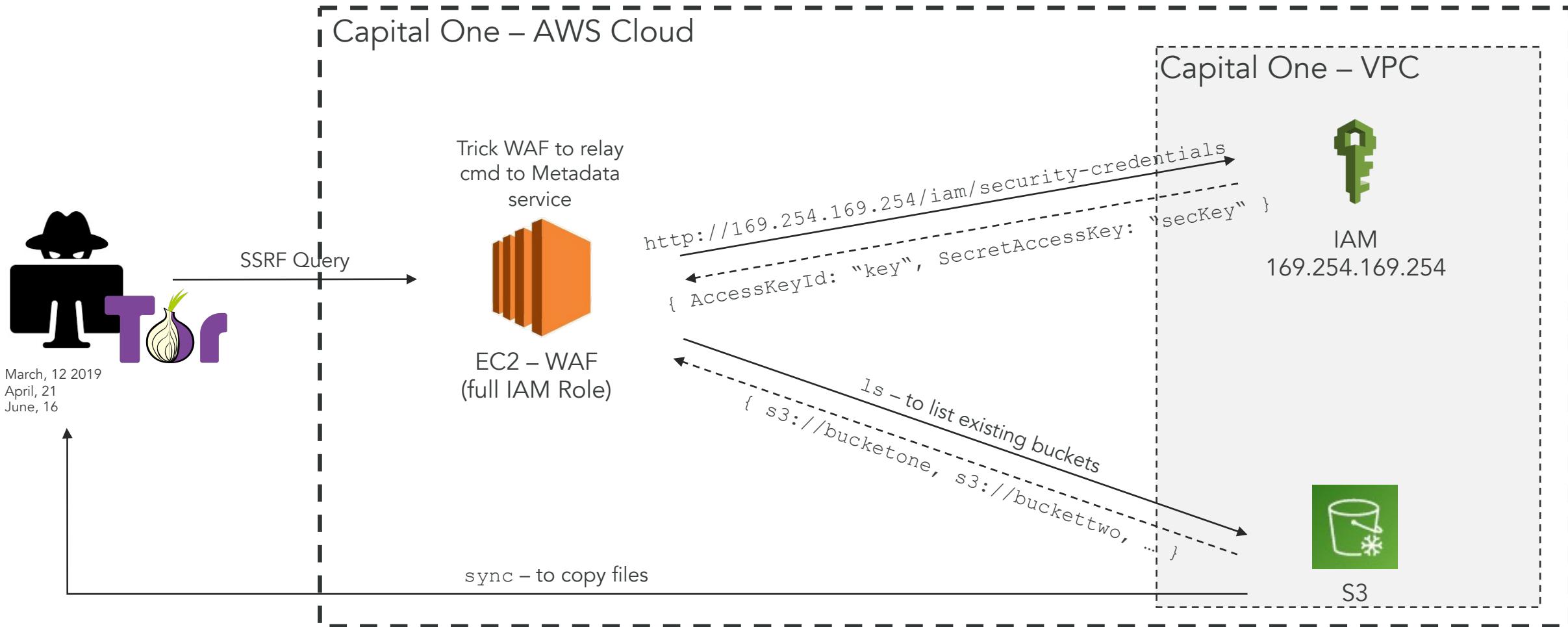
80,000 account numbers, 140,000 Social Security numbers, 1 million Canadian Social Insurance Numbers exposed, due to a Server-Side-Request Forgery attack.



BROUGHT TO YOU BY
CODESHIELD.IO

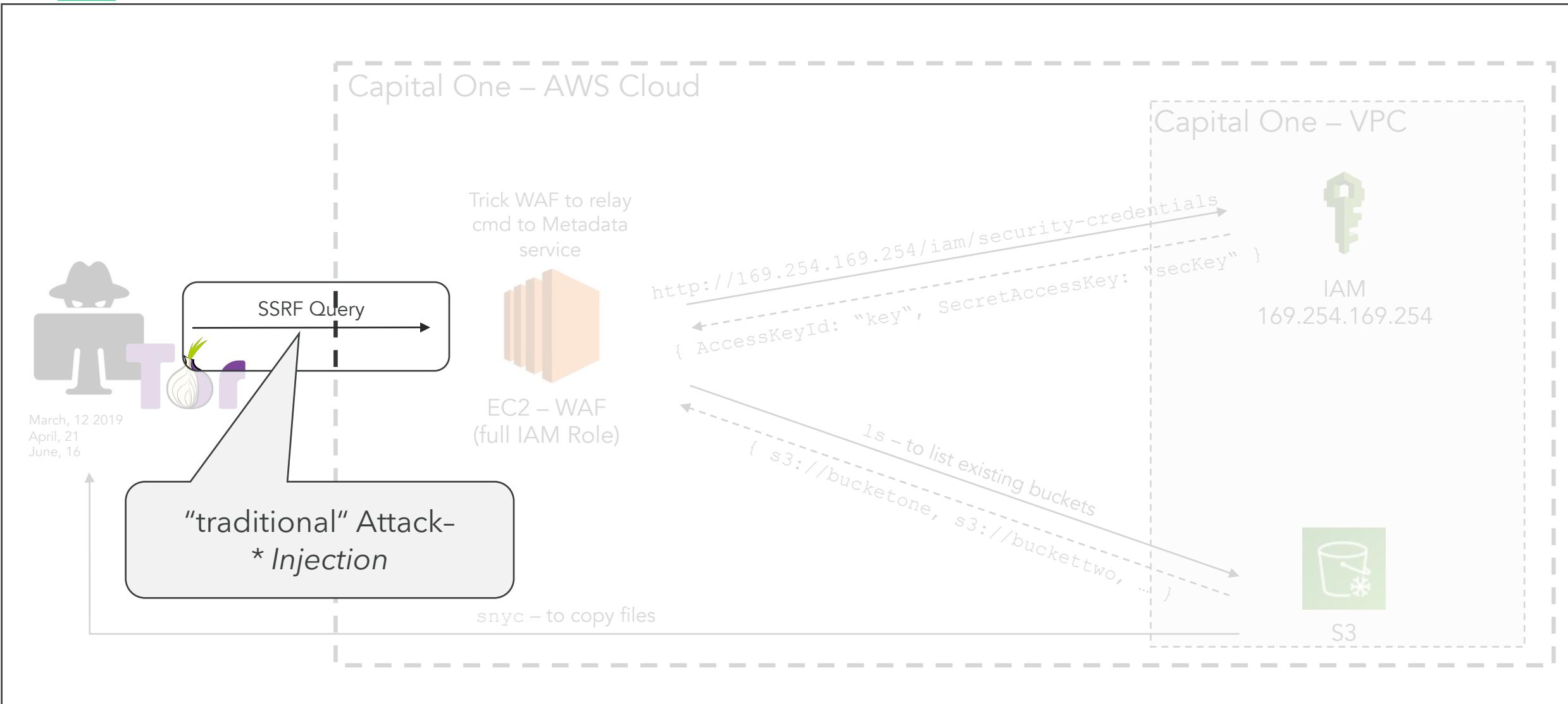
01

Capital One Hack - What happened?



01

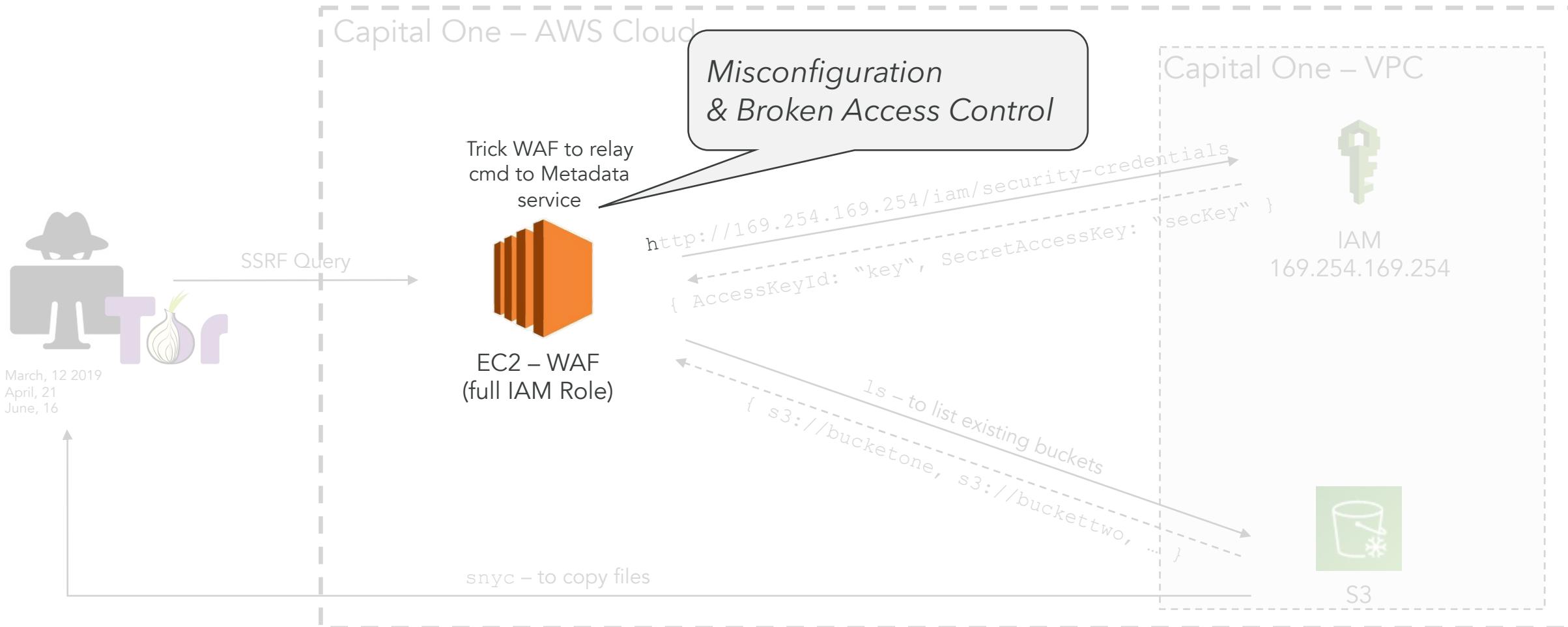
Capital One Hack - What happened?



BROUGHT TO YOU BY
CODESHIELD.IO

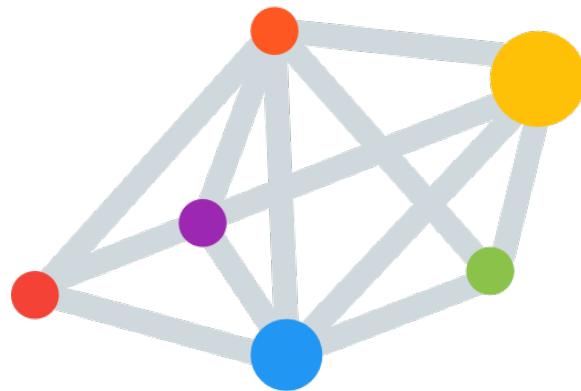
01

Capital One Hack - What happened?



01

Capital One Hack - Lessons Learned



Security must be considered **across** all components



BROUGHT TO YOU BY
CODESHIELD.IO

02

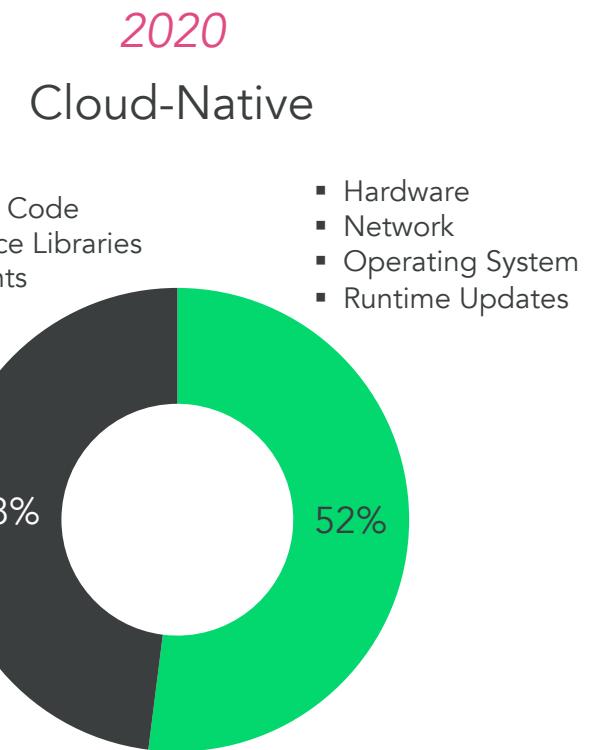
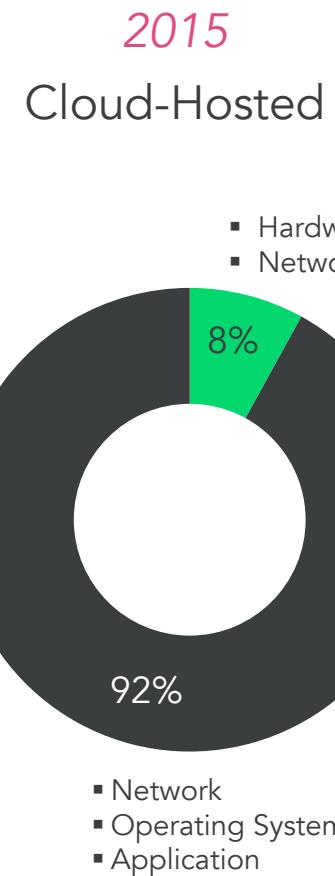
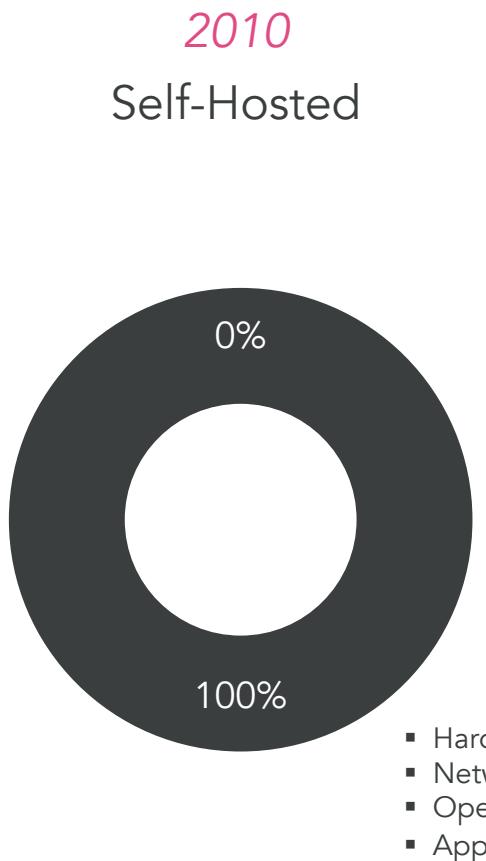
SECURITY IN THE CLOUD WHO IS RESPONSIBLE?



BROUGHT TO YOU BY
CODESHIELD.IO

02

„Shared Responsibility“ Security Model



■ Software Provider's Responsibility

■ Cloud Provider's Responsibility

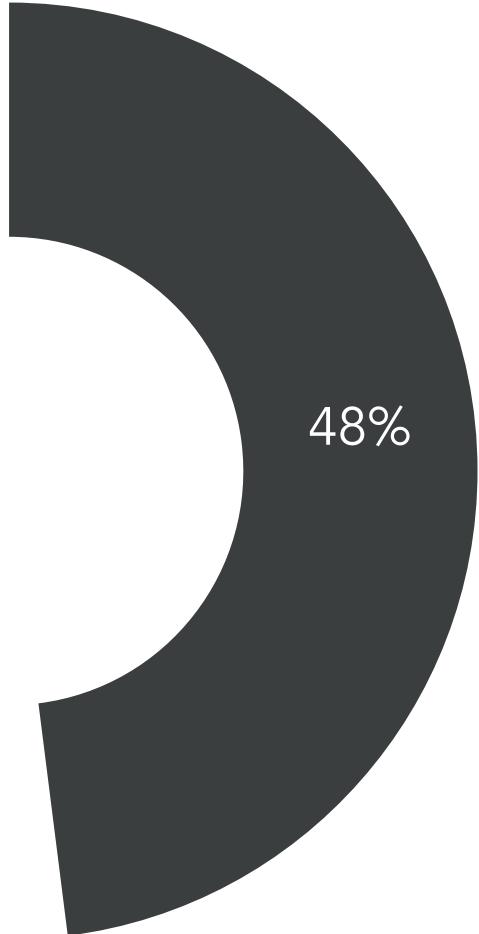
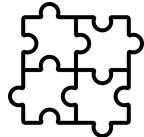


BROUGHT TO YOU BY
CODESHIELD.IO

<https://securityboulevard.com/2019/01/serverless-and-the-evolution-in-cloud-security-how-faas-differs-from-iaas/>

02

Cloud-Native Application



Infrastructure-as-Code

*Detect **data-flows** in architecture and
misconfiguration in infrastructure*



Open-Source Dependencies

*Detect **known** CVEs
in **unknown** Code*



Application / Lambda /
“Function” Code

*Detect **unknown** vulnerabilities
in **own** code*



BROUGHT TO YOU BY
CODESHIELD.IO

03

SECURITY IN THE CLOUD WHAT SHOULD WE DO?



BROUGHT TO YOU BY
CODESHIELD.IO

2010

Self-Hosted

Monolith

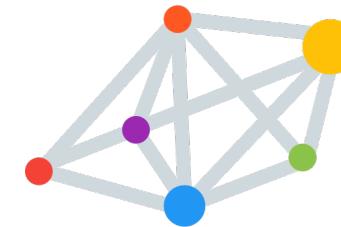


1-3 Components

2020

Cloud-Native

Everything-as-Function



200+ Cloud-Resources

03

Security Audit

2010

Self-Hosted

Monolith



1-3 Components

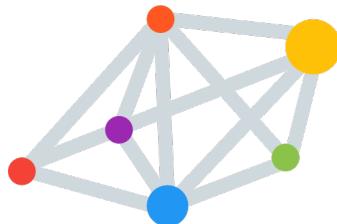
- 01** Is the software documented?
- 02** What is the general architecture of the software?
- 03** Which assets does your software process?
- 04** Which sensitive data does your software handle?
- 05** Which security mechanisms are in place?
- 06** Is the infrastructure correctly configured?
- 07** Does the code contain existing vulnerabilities (CVEs)?
- 08** Is the custom code written in a secure manner?



BROUGHT TO YOU BY
CODESHIELD.IO

03 Security Audit

2020
Cloud-Native
Everything-as-Function



200+ Cloud-Resources

- 01** Is the software documented?
- 02** What is the general architecture of the software?
- 03** Which assets does your software process?
- 04** Which sensitive data does your software handle?
- 05** Which security mechanisms are in place?
- 06** Is the infrastructure correctly configured?
- 07** Does the code contain existing vulnerabilities (CVEs)?
- 08** Is the custom code written in a secure manner?



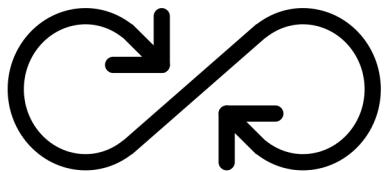
04

SECURITY IN THE CLOUD WHAT CAN WE DO?



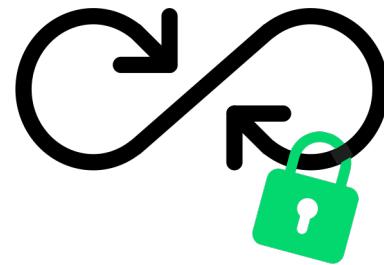
BROUGHT TO YOU BY
CODESHIELD.IO

DevOps

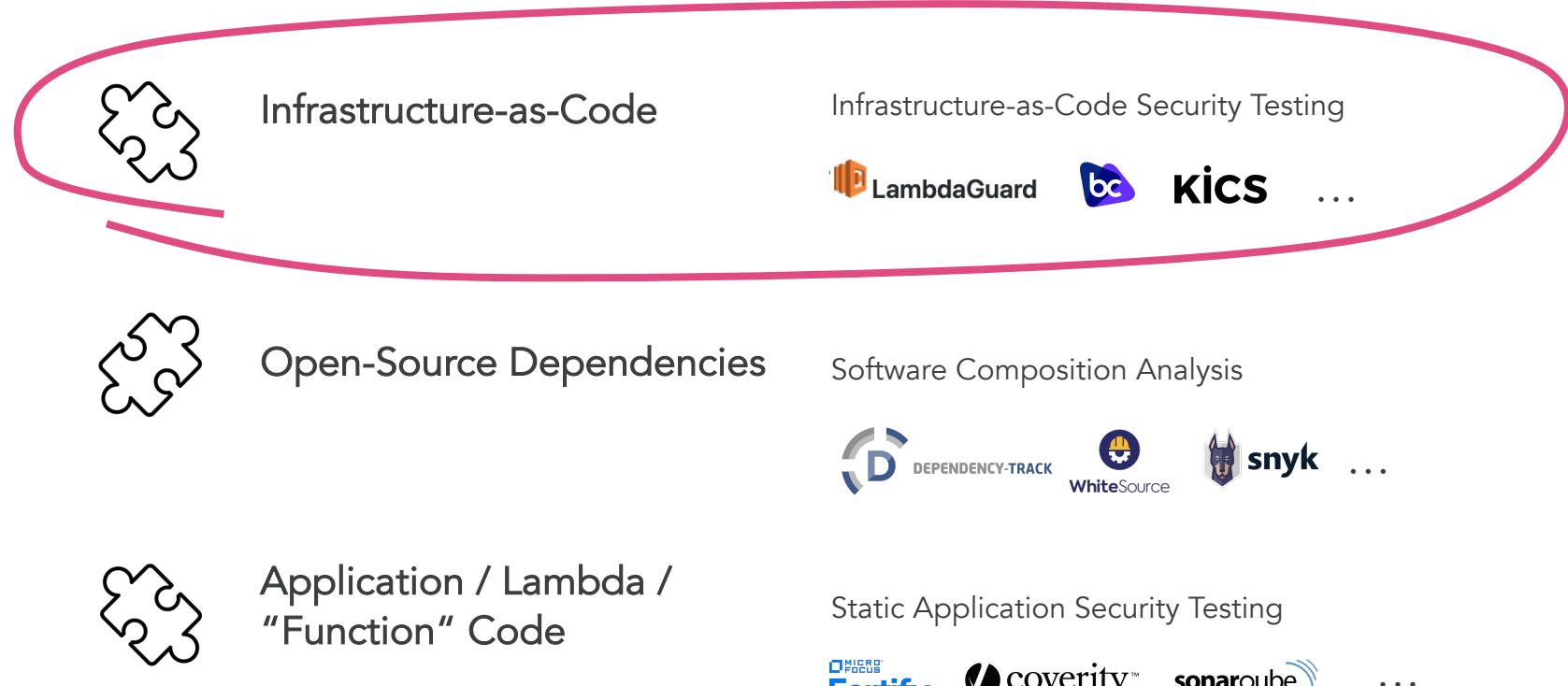
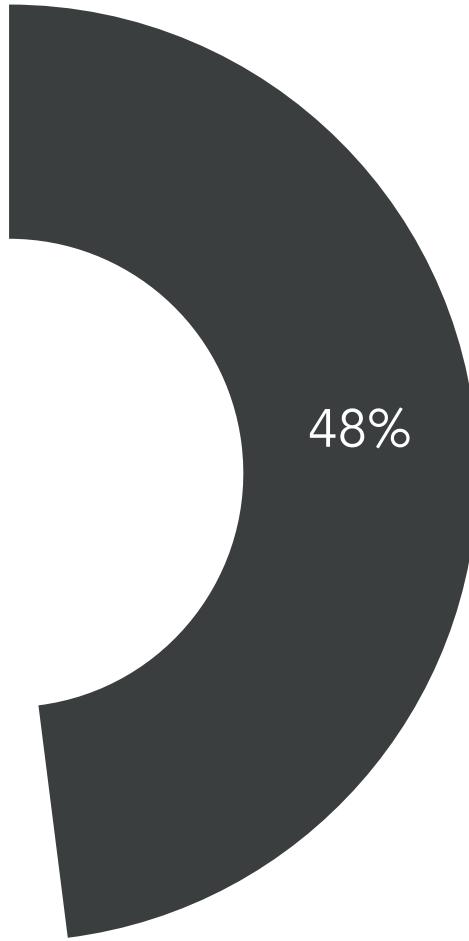


- Continuous Testing
- Continuous Delivery
- Continuous Deployment

DevSecOps



- Continuous Security
 - Static Application Security Testing
 - Software Composition Analysis
 - Infrastructure-as-Code Security Testing



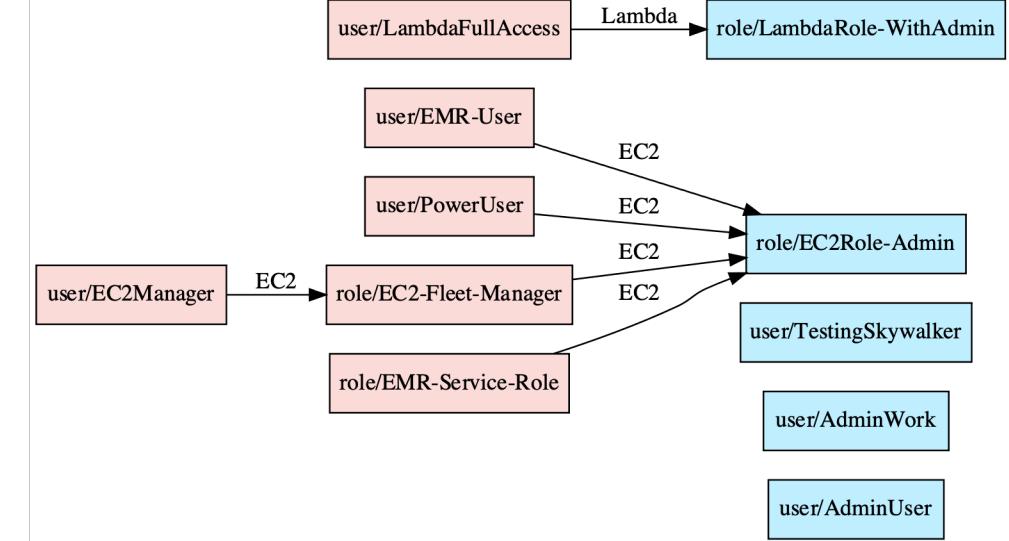
04 Principal Mapper

Benefits

- Visualizes IAM and AWS Identities as Graph
- Detects Privilege Escalation
- Detects Identity Misconfiguration

Shortcomings

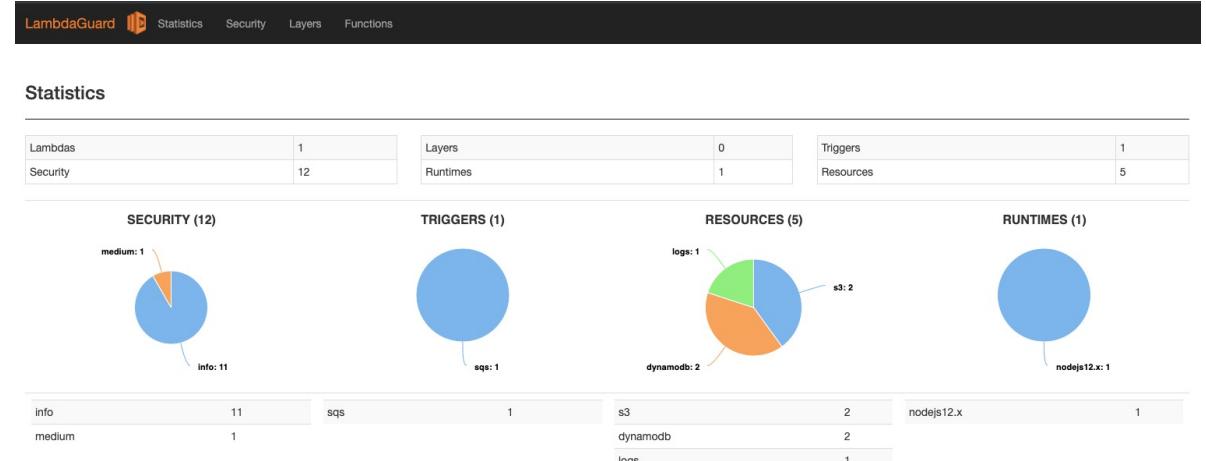
- No Fix Recommendations
- No Code-Level Analysis



<https://github.com/nccgroup/PMapper>

Benefits

- Inspects CloudFormation Templates
- Displays triggers & resources
- Potential security issues
(based on CloudFormation Template)



Shortcomings

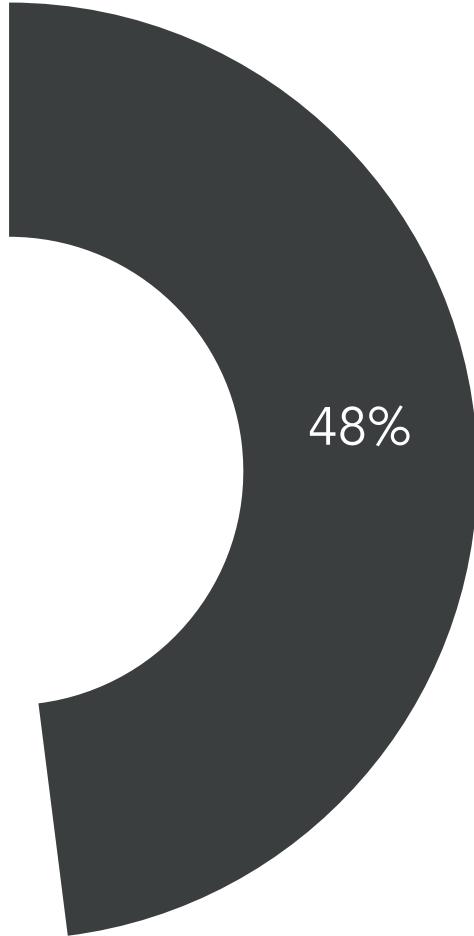
- Views Resources individually
- „Big Picture“ Missing
- No Code Analysis



BROUGHT TO YOU BY
CODESHIELD.IO



<https://github.com/Skyscanner/LambdaGuard>



Infrastructure-as-Code

Infrastructure-as-Code Security Testing



KiCS

...



Open-Source Dependencies

Software Composition Analysis



DEPENDENCY-TRACK



snyk

...



Application / Lambda /
“Function” Code

Static Application Security Testing



coverity™

sonarqube

...

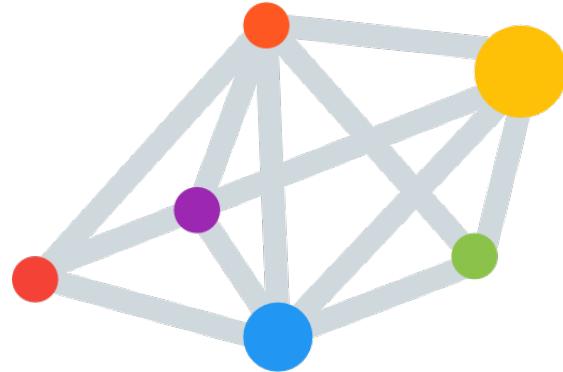
Tools **miss the big picture** and produce irrelevant results (False Positives / False Negatives)

04

Provides the „Big Picture“, dives down to Details



- Connect your Git Repositories
- Connect your AWS Account
- Visualizes infrastructure data-flows based on CloudFormation Template
- Detects security vulnerabilities
 - Least privilege (IaC + SAST)
 - SCA results with prioritization



BROUGHT TO YOU BY
CODESHIELD.IO

Task for
Security



Infrastructure-as-Code

Ease of Attack: Easy

Detect **data-flows** and **misconfiguration** in architecture



Graph Visualization + IaC-Analysis



Manuel Benz
M.Sc. IT-Security
4+ yrs research

Cloud-Native Application

Open-Source Code

Ease of Attack: Medium

Detect **known** CVEs in **unknown** Code



Bytecode Analysis + "Fingerprinting"



Andreas Dann
M.Sc. Computer Science
4+ yrs research

Application Code

Ease of Attack: Difficult

Detect **unknown** vulnerabilities in **known** code



Precise & Efficient Data-Flow Engine



Johannes Späth
PhD Computer Science
5+ yrs research



04 Architecture Overview

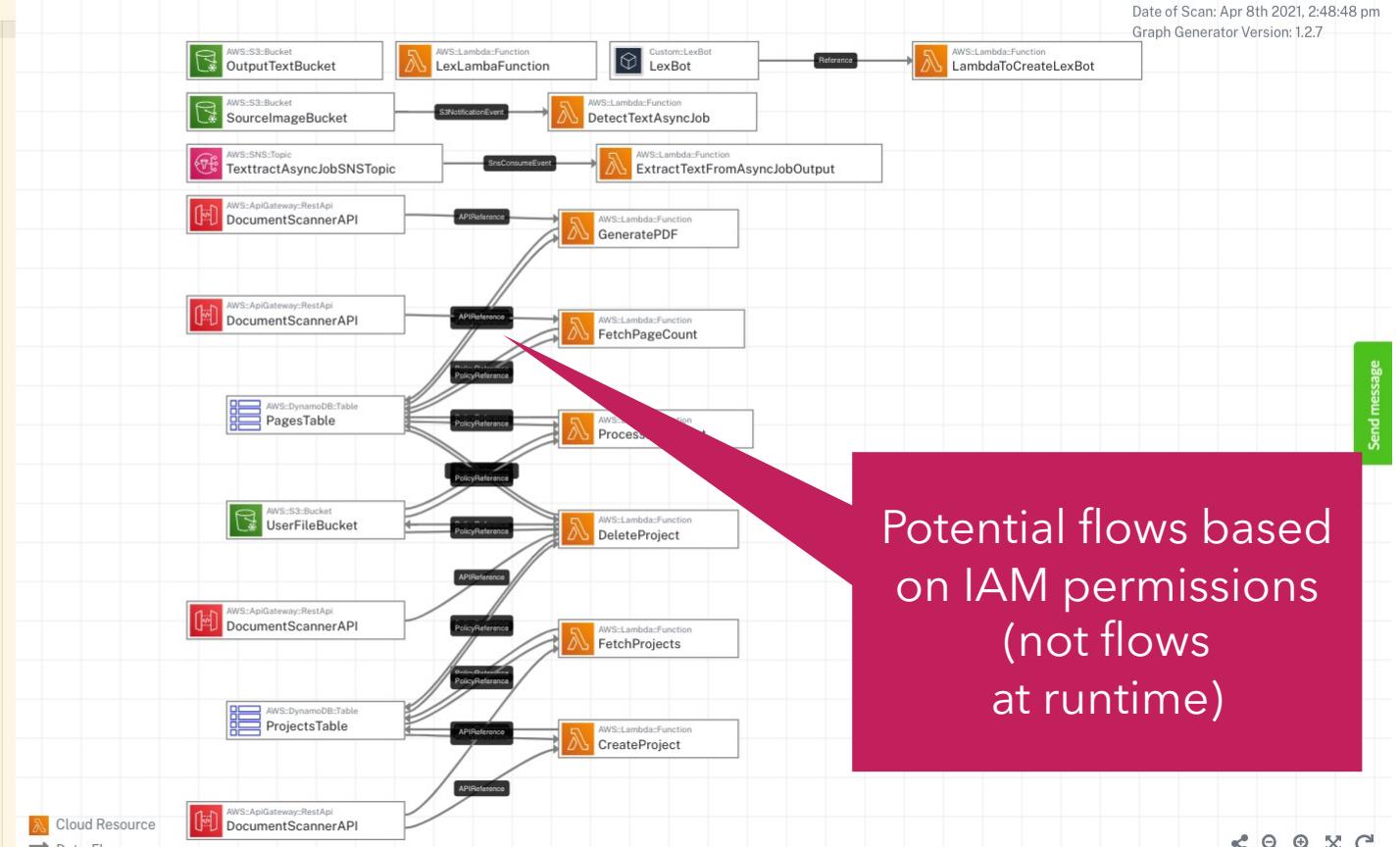


Repositories / Kush-1411/Aws-Doctor / 0847faab62e0126b14eed6005528034f07829710

```
cfntemplate.yml
1 AWSTemplateFormatVersion: 2010-09-09
2 Transform:
3 - AWS::Serverless-2016-10-31
4 Parameters:
5 BotName:
6   Type: String
7   Description: Prefix to add to Lex resource names
8 Default: InvoiceBot
9 MinLength: 3
10 MaxLength: 32
11 AllowedPattern: ^[a-zA-Z\.\_]+$  
  ConstraintDescription: "Must conform with the permitted Lex Bot name pattern.\n12 \\\n"
13 Resources:
14 DetectTextAsyncJob:
15   Type: AWS::Serverless::Function
16   Properties:
17     Handler: lambda:create_textextract_detect_text_async_job.handler
18     Runtime: python3.7
19     CodeUri: s3://aws-codestar-us-east-1-820570838999-meaningfulconve-pipe/
c4f60558e37b987d55576305b05d010a
20       Description: ''
21       MemorySize: 512
22       Timeout: 30
23 Environment:
24   Variables:
25     SNS_TOPIC_ARN:
26       Ref: TextextractAsyncJobSNSTopic
27     SNS_ROLE_ARN:
28       Fn::GetAtt:
29         - TextextractSNSTopicRole
30         - Arn
31 Role:
32   Fn::GetAtt:
33     - DetectTextAsyncJobRole
34     - Arn
35 Events:
36   BucketEvent1:
37     Type: S3
38     Properties:
39       Bucket:
40         Ref: SourceImageBucket
41       Events:
42         - s3:ObjectCreated:*
43 DetectTextAsyncJobRole:
44   Type: AWS::IAM::Role
45   Properties:
46     AssumeRolePolicyDocument:
```

Docs FAQ Contact

Date of Scan: Apr 8th 2021, 2:48:48 pm
Graph Generator Version: 1.2.7



Potential flows based
on IAM permissions
(not flows
at runtime)



©2021 CodeShield GmbH; all rights reserved.



BROUGHT TO YOU BY
CODESHIELD.IO

Reachability Analysis for Software Composition Analysis

The screenshot shows the CodeShield interface. At the top, there's a dark header with the CodeShield logo. Below it, a section titled "Vulnerable Open-Source Components" displays four categories: Critical (0), High (1), Medium (0), and Low (1). The main area shows a "package.json" file with two vulnerabilities listed:

- pkg:npm/minimist@0.0.8**:
 - Prototype Pollution**: A warning icon. Description: Affected versions of `minimist` are vulnerable to prototype pollution. Arguments are not properly sanitized, allowing an attacker to modify the prototype of `Object`, causing the addition or modification of an existing property that will exist on all objects. Parsing the argument `--__proto__.y=Polluted` adds an uncaught error and crashes the application. This is exploitable if attackers have control over the arguments.
 - CWE-94: Improper Control of Generation of Code ('Code Injection')**: A warning icon. Description: The software constructs all or part of a code segment using externally-influenced input from an upstream component, but it does not correctly neutralizes special elements that could modify the syntax or behavior of the intended code segment.

Three callout boxes highlight features of the tool:

- A pink box points to the findings: "Displays findings with exact location in your infrastructure".
- A pink box points to the attack paths: "Visualize potential attack paths within your infrastructure".
- A pink box points to the security warnings: "Prioritize security warnings".



Combining SAST + IaC: Violation of Least Privilege

CloudFormation



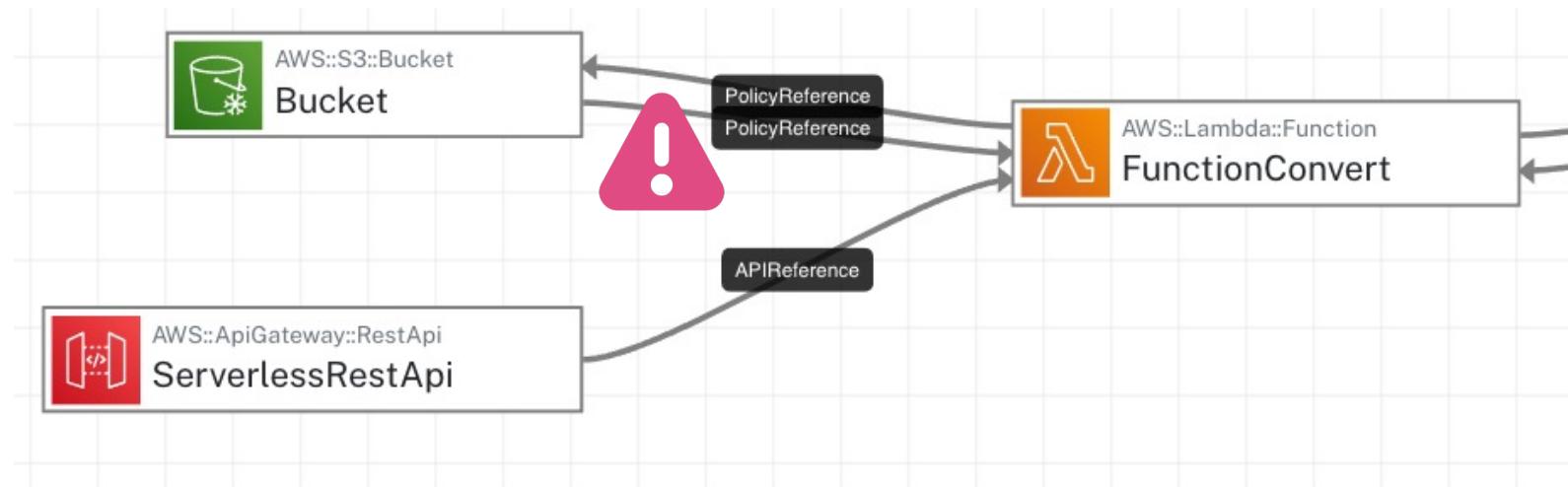
Policies:

```
58   - DynamoDBCrudPolicy:  
59     TableName: !Ref Table  
60   - S3FullAccessPolicy:  
61     BucketName: !Ref Bucket
```

Lambda's Source Code



```
80     s3Client.putObject(  
81       new PutObjectRequest(System.getenv("BUCKET_NAME"), key, txt,  
objectMetadata)  
     .withCannedAcl(CannedAccessControlList.PublicRead));
```





Dr. Johannes Späth

CEO & Co-Founder



johannes.spaeth@codeshield.io



Technologiepark 8, 33100 Paderborn



BROUGHT TO YOU BY
CODESHIELD.IO

