RSAConference2019
San Francisco | March 4–8 | Moscone Center

SESSION ID: CSV-W10

Treating Cloud-Specific Threats with Automatic Remediation

Jay Ball
Head of Application Security
Zocdoc

Josh Barons
Head of Information Security
Zocdoc

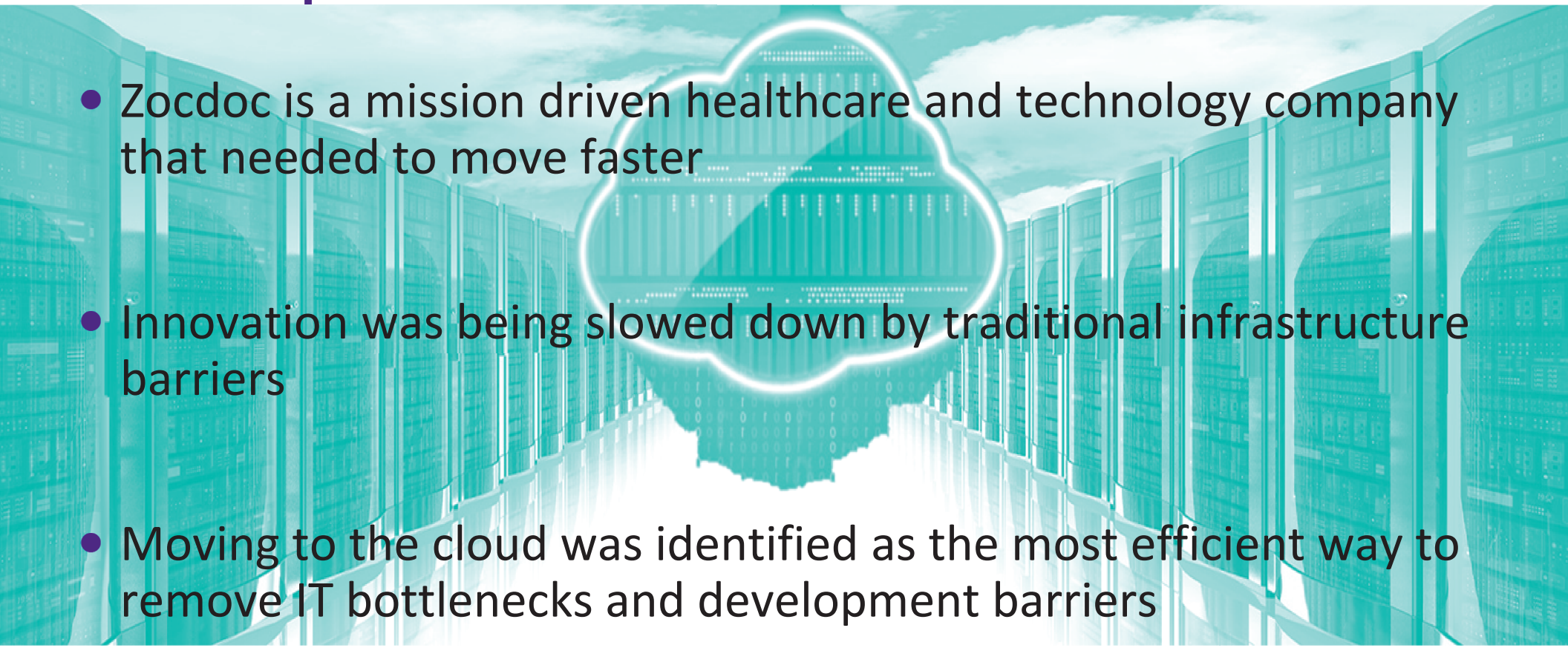
Agenda & Goals

- Zocdoc's transformation to the cloud
- Provide examples of threats and issues we encountered both external and internal
  - AWS
    - S3 Bucket Encryption
    - VPC Port Inspection and Closure
    - Ownership
    - Instance Termination
  - GitHub
    - Public Repos
    - Repo ownership
- Offer solutions
- Get buy-in

RSAConference2019

Zocdoc's move to the Cloud

Zocdoc's path to the cloud



- Zocdoc is a mission driven healthcare and technology company that needed to move faster
- Innovation was being slowed down by traditional infrastructure barriers
- Moving to the cloud was identified as the most efficient way to remove IT bottlenecks and development barriers

RSAConference2019

The Cloud

Cloud

- Cloud is here - It's already been here and probably in your organization also
- Who is responsible - Providers have a shared responsibility model
- Visibility - How can you see into your environments
- Problems at scale - Minor issues easily become major at scale
- Internal ownership - Who owns the relationship?
- Cloud specific threats - Public exposure - Unencrypted data - Open Security groups - Compromised Instance - Visibility and auditing

RSAConference2019

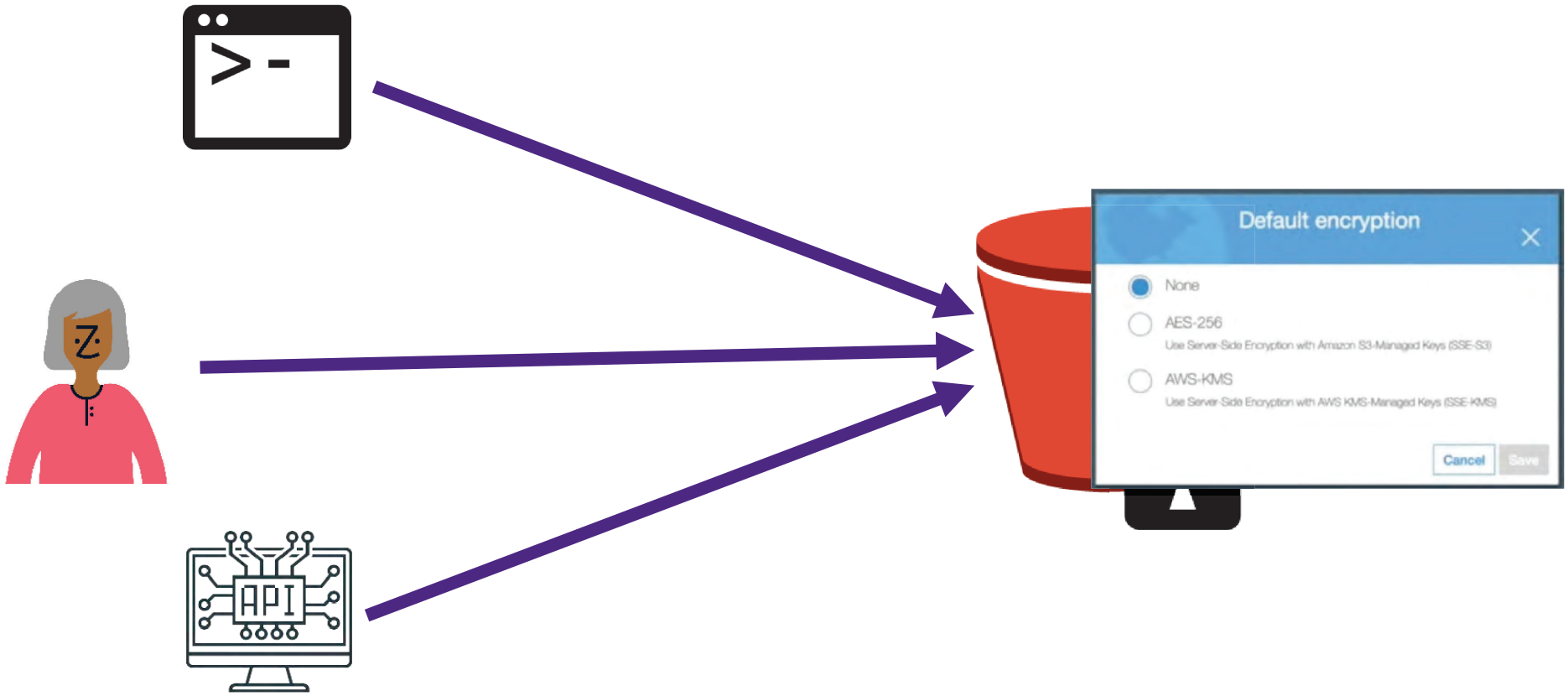
Encryption at Rest

Automated S3 Bucket Encryption

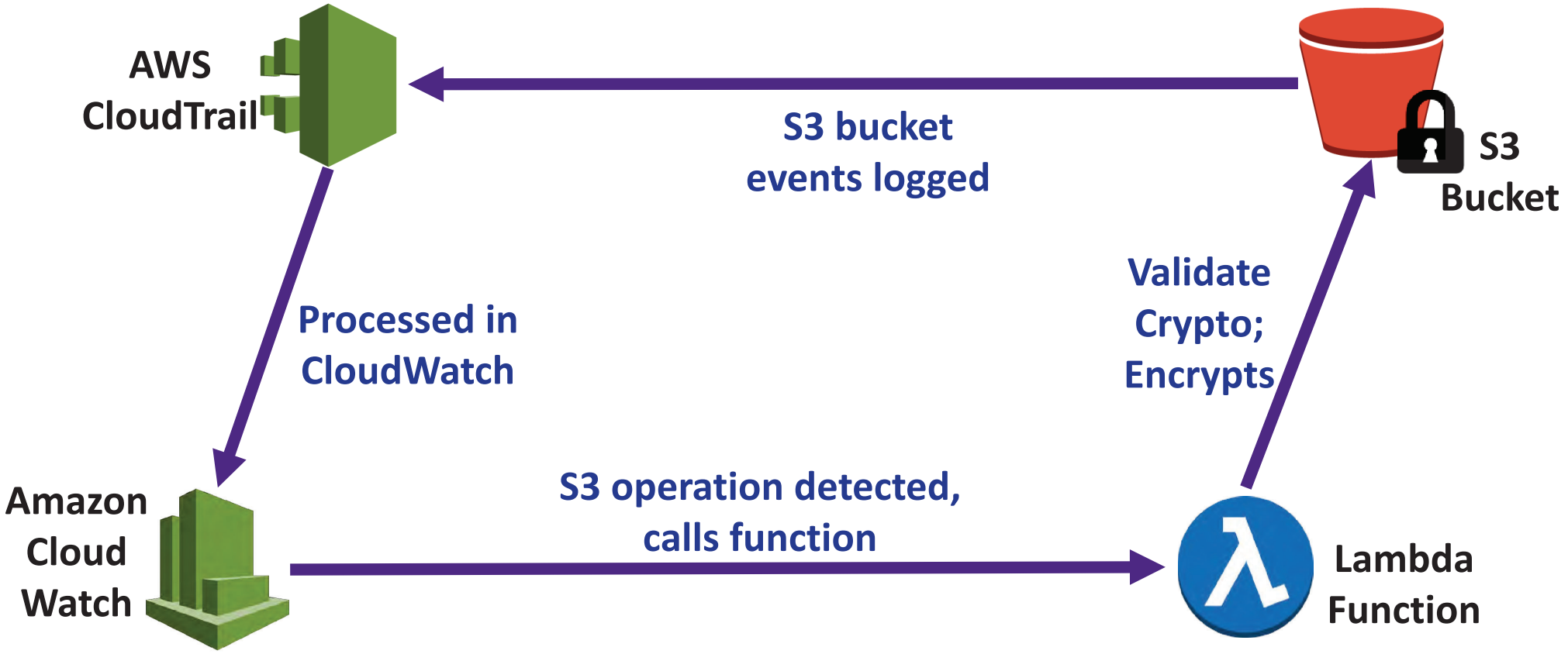
Threat: S3 Bucket Data Exposure

- Pre-cloud, policies and processes enforce encryption at rest
  - We had system build processes
  - Hardened machine images
  - Transparent encryption on RAIDs, SANs, and Servers
- In-cloud, we have these also but data can still be stored unencrypted by default and processes are harder to enforce
- Problem in Cloud:
  - How do we make sure S3 buckets are encrypted
- Solution in Cloud:
  - Automate the bucket encryption when it's lacking

Unencryption Scenarios on S3 Buckets



Automated Encryption of AWS S3 Buckets



RSAConference2019

Recorded Demo

Automated S3 Bucket Encryption

CloudWatch Configuration

```
{
  "source":        [ "aws.s3"      ],
  "detail-type":   [ "AWS API Call via CloudTrail" ],
  "detail": {
    "eventSource": [ "s3.amazonaws.com" ],
    "eventName": [
      "CreateBucket",
      "PutBucketAcl",
      "PutBucketPolicy",
      "PutBucketEncryption",
      "DeleteBucketEncryption"
    ]
  }
}
```

Source Code Highlights

```
s3  = boto3.client('s3')
sns = boto3.client('sns')

def lambda_handler(event, context):
  bucket_name    = event['detail']['requestParameters']['bucketName']
  bucket_creator = event['detail']['userIdentity']['principalId']
  bucket_user    = bucket_creator.split(":",1)

  try:
    currEncrypt = s3.get_bucket_encryption (Bucket=bucket_name)
```

Source Code Highlights

```
except ClientError as e:
      toEncrypt = s3.put_bucket_encryption (
           Bucket=bucket_name,
           ServerSideEncryptionConfiguration = {
               'Rules':[ {
                  'ApplyServerSideEncryptionByDefault':
                        { 'SSEAlgorithm': 'AES256' }
                }]
           } )
      response = sns.publish(
          TopicArn= 'arn:aws:sns:us-east-1:   …
                     000012345678:unEncryptedS3BucketCreated',
          Message= 'Encryption Removed by '+ bucket_user[1]    )
else:
      return currEncrypt
```

Rinse & Repeat; Embrace & Extend

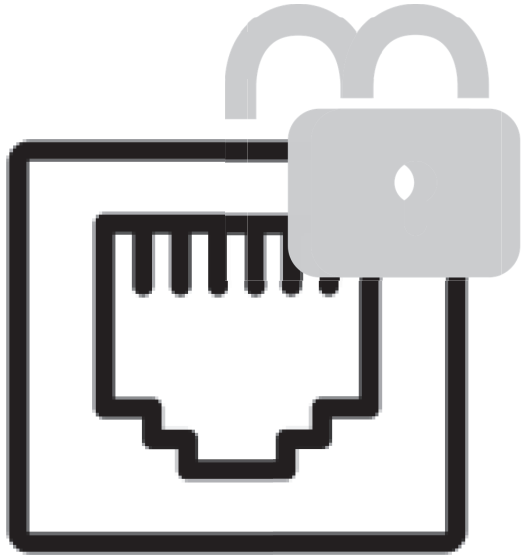
- Design pattern can be replicated for other S3 concerns
- Also implemented “Automated Make Buckets Private”
  - An event causes a bucket to become public
  - Our lambda makes it private if not in the whitelist
  - Amazon finally does this now
- File scanners
  - New file; scan for data leakage compliance, steganography, etc.
- Intracloud duplication
  - Copy all uploads to another cloud provider

RSAConference2019

Sensitive Ports open to Internet

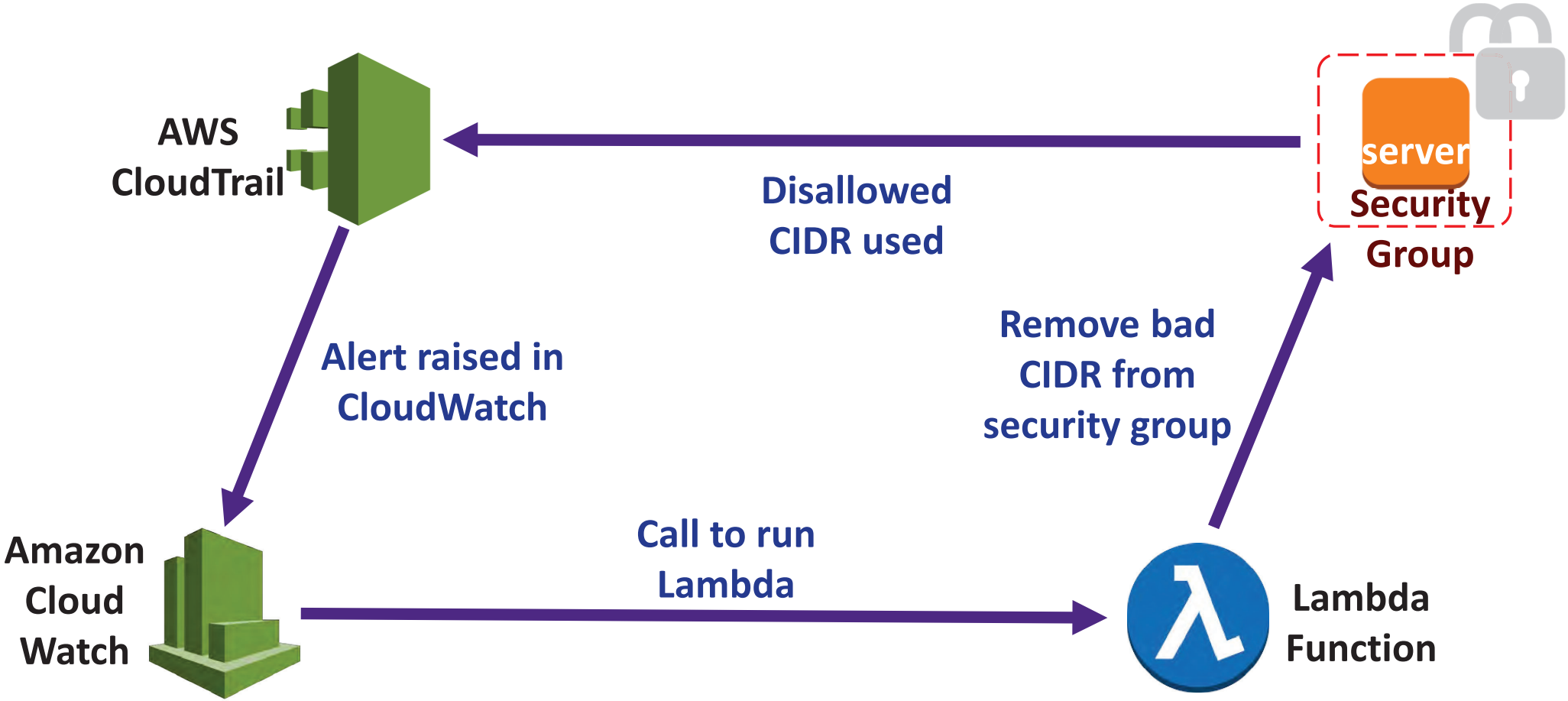
Automated AWS Security Group/VPC Updates

Threat : Ports open to the Internet



- Pre-cloud, we relied on Firewalls and Network security.
  - Deny by default
  - Network change management
- In-cloud, anyone can spin up a system
  - And chose a new or existing security group
- Problems in Cloud:
  - Someone opens sensitive ports to the entire Internet
- Solutions in Cloud:
  - Lock the ports down by auto-correcting the security group

Automated Port Updates in AWS



RSAConference2019

Recorded Demo

Automated Ports, No More -1

CloudWatch Configuration

```
{
  "source":       [ "aws.cloudtrail"  ],
  "detail-type":  [ "AWS API Call via CloudTrail” ],
  "detail": {
    "eventSource":[ "ec2.amazonaws.com"  ],
    "eventName":  [
      "CreateSecurityGroup",
      "AuthorizeSecurityGroupIngress"
    ]
  }
}
```

Code: Initial Sanity Checks

```
def lambda_handler(event, context):
    source = event['detail']['eventSource']
    if source != "ec2.amazonaws.com":
        return    # wrong caller, just silently return

    allowed_event_list =  [
        'CreateSecurityGroup',
        'AuthorizeSecurityGroupIngress'
    ]

    event_name = event['detail']['eventName']

    if not(event_name in allowed_event_list):
        return    # wrong caller, just silently return
```

Code: GetSecurityGroupID

```
resp = event['detail']['responseElements']

if (resp["_return"] != True):
    return    # event not successful, so ignore it

SG_id = 'invalid'
if event_name == 'CreateSecurityGroup':
    SG_id = resp['groupId']
elif event_name == 'AuthorizeSecurityGroupIngress':
    SG_id = event['detail']['requestParameters']['groupId']
```

Code: Group Sensitivity Training

```
ec2 = boto3.resource('ec2')
security_group = ec2.SecurityGroup(SG_id)
sensitive_ports = [ 22, 3389, 54321 ]
ingress_list = security_group.ip_permissions

for each permission in ingress_list:
    if permission.port isa sensitive_ports
       or permission.portrange contains sensitive_ports
       or permission.protocol == -1   ;# aka "all ports"
    then
        sensitive = True
```

Code: Test for Zero CIDR

```
if sensitive:
    for r in perm['IpRanges']:
        if r['CidrIp'] == "0.0.0.0/0":
            print("Removing Ingress Rule: ", json.dumps(perm))
            try:
                security_group.revoke_ingress(
                    CidrIp     = r['CidrIp'],
                    IpProtocol = perm['IpProtocol'],
                    FromPort   = fromport,
                    ToPort     = toport      )
            except ClientError as e:
                print("Error: ", e)
                raise
```

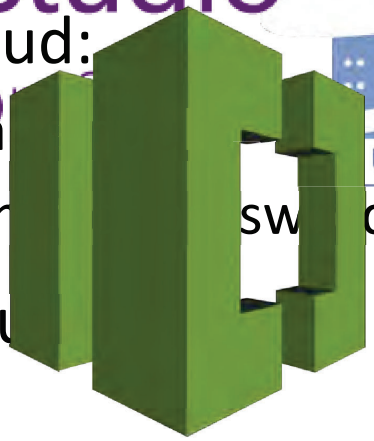
RSAConference2019

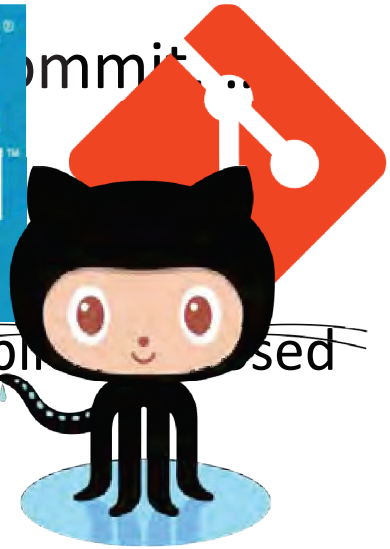
Source Code Repositories Exposed on Internet

Automate repo protection on GitHub

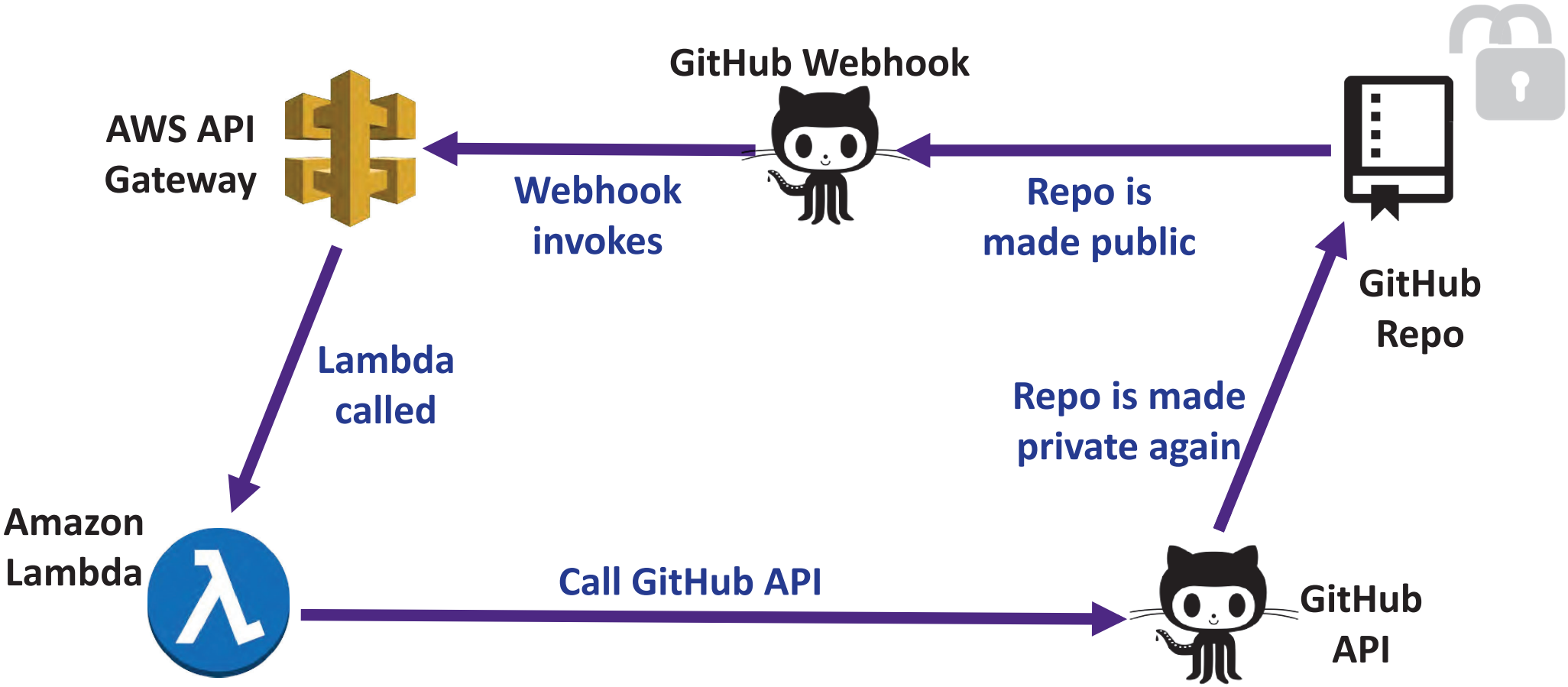
Threat: Public Code Repositories

- Pre-cloud, on-prem code repos, such as TFS, Clearcase, git, …
- Pre-cloud, usage of GitLab, B… Hu… …mmit…
- Problems in Cloud:
  - Acci… be ma…
  - Acci…al comm… sw… ds an… w publ… sed
- Solutions in Clou…
  - We wrote tools



Public to Private Flow



RSAConference2019

Live Demo (with Recorded Backup)

Automate repo protection on GitHub

GitHub WebHook Configuration

```
async function add_hook(organization) {
    var config = {
        url:    `https://${awshost}/stagename/webhook`,
        content_type: "json",
        secret:       decrypted_secret_key
    };
    const hook = await octokit.orgs.createHook( {
        org:    organization,
        name:   'web',
        config,
        events: ['push','public','repository']  });
}
```

Code Highlights: Spellbound Invocations

```
const Octokit = require('@octokit/rest’);

async function list_topic_and_action(){
    const topics =
      await octokit.repos.listTopics({
          owner:   `${body.organization.login}`,
          repo:    `${body.repository.name}`,
          headers: {
              accept: 'app/vnd.github…json'}
          }
      );
```

Code Highlights: Some Privacy Please

```
if ( body.repository.private === false &&
     topics.data.names.includes('open-source')
        === false
) {
   await octokit.repos.update({
     owner: `${body.organization.login}`,
     repo:  `${body.repository.name}`,
     name:  `${body.repository.name}`,
     private: true }  )
```

Code Highlights: Complain to Management

```
var params = {
  Message: `Repo: ${body.repository.full_name} exposed
            by user:     ${body.sender.login}
            at time:     ${body.repository.updated_at}
            Private was: ${body.repository.private}
            Other Repo Topics (none if blank):
                  ${topics.data.names}

            This was corrected, repo was made private.`,
  TopicArn: `arn:aws:sns:reg:012345:github-repo-monitor`,
};
AWS.SNS({apiVersion: '2010-03-31'}).publish(params);
```

RSAConference2019

Repository Ownership Auditing

Automate repo ownership reporting on GitHub

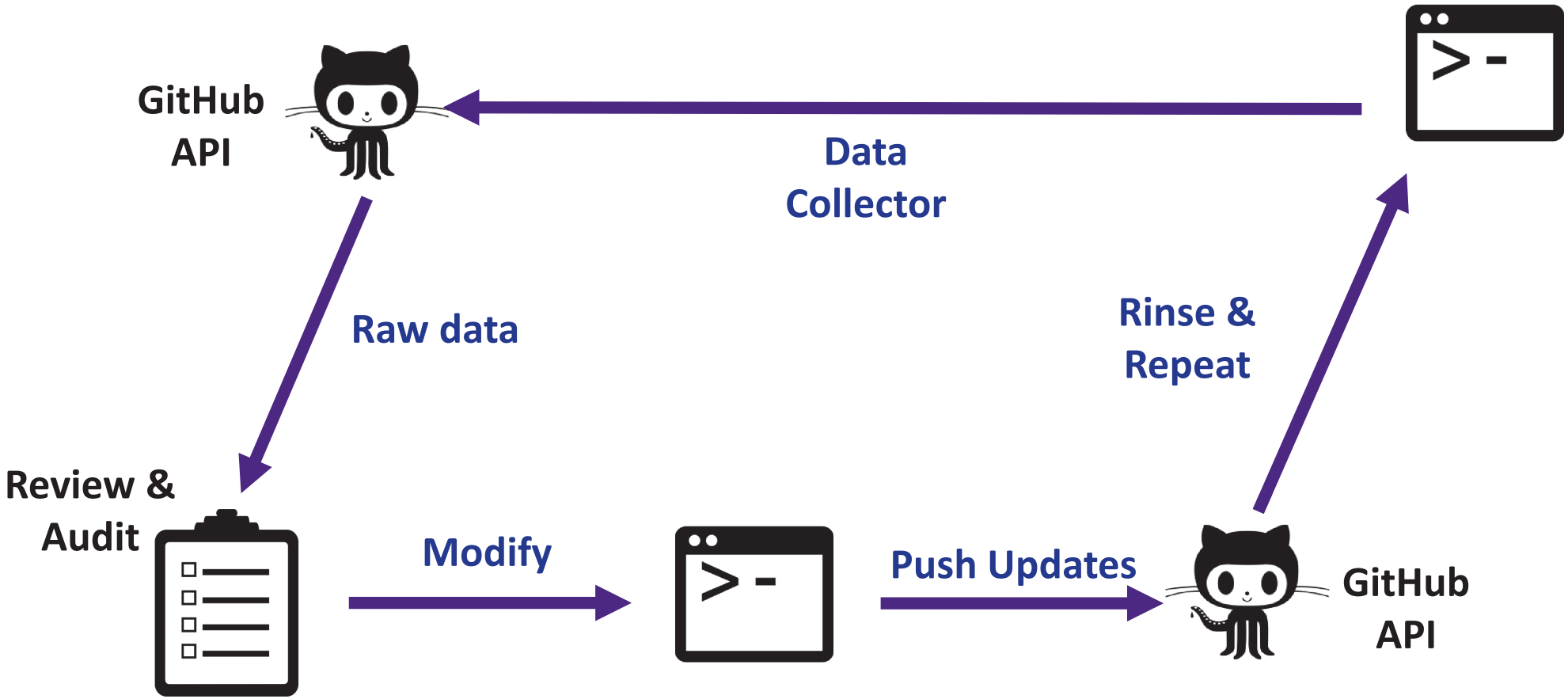
Repository Visibility and Auditing

- Pre-cloud, on-prem code repos with built-in auditing, such as Subversion
- In-cloud, we use other things
- Problems in Cloud:
  - Repos are easy to create by anybody, we pay for it
  - Management visibility it hard
- Solutions in Cloud:
  - We wrote tools

Repo Ownership & Responsibility

- Repos at Zocdoc are required to have responsible team
  - Allows both Infosec and Engineering to assign tickets
- We do not use GitHub's team mechanism for repo access control
  - Nearly all repos are owned by the Engineering Organization
  - Developers move between teams and work on many projects
  - Defining ownership and AC using GitHub's mechanism adds overhead
- The ownership is tracked by overloading GitHub topics
  - Topics named "team-NAME" show ownership

Repo Ownership Auditing in GitHub



Excel at Security with Audits

node_id	Name	Description	Archive	Private	Languages	Tea..
146348261	ZocSec.GitHub	Repo with GitHub security auditin	FALSE	TRUE	Python JavaScript Shell	infosec/a
135461720	ZocSec.SecurityAsCo	AWS code from the Zocdoc Inform	FALSE	FALSE	Python	info

- Basic Info: node_id, Name, Description
- Repo Status: Archived, Private
- Languages

	Teams	Topics	Topic_add	Topic_del	Last Comitte	Last Comit Date
ot Shell	infosec/admin				jay-ball-zocdoc	2019-02-18 16:02:57
	infosec/admin	team-infosec open-source			web-flow	2018-10-26 19:02:53

- Teams – GitHub's Teams and Access Control Permissions
- Topics – All of the topics, including our "teams via topics"
- Update Columns– Topic_Add and Topic_Del
- Last Committer, Last Commit Date

RSAConference2019

Recorded Demo

Automate repo ownership

Calling Convention

```
$ python3 github_tasks.py  --help
usage: github_tasks.py [-h] -t TOKEN [-a] [-u UPDATE]

This tool allow user to audit Enterprise Repos.

optional arguments:
  -h, --help            show this help message and exit
  -t TOKEN, --token TOKEN
                        GitHub User Private Token
  -a, --audit           Audit github Repo
  -u UPDATE, --update UPDATE
                        Import Topics to Repo
```

Code Highlights: Startup

```
from github import Github
from openpyxl import Workbook

if __name__ == "__main__":
    parser = argparse.ArgumentParser( … )
    args = vars(parser.parse_args())
    user = Github(args["token"])
```

Code Highlights: We Excel at This

```
if args["audit"]:   # audit mode
    wb = Workbook()
    ws = wb.active
    ws.title = "Github Inventory"
    …
    wb.create_sheet('Repos Deploy keys')
    deploy_keys_sheet = wb["Repos Deploy keys"]
    wb.create_sheet('Summary')
    summary_sheet = wb["Summary"]

    org = user.get_organization("Zocdoc")
```

Code Highlights: Getting Loopy

```
for repo in user.get_user().get_repos():
   languages = ""; teamlist = ""; topics = ""

   for lang in repo.get_languages():
      languages = languages + lang + " "
   for team in repo.get_teams():
      teamlist = teamlist + team.name + "/" +
          team.permission + " "
   for topic in repo.get_topics():
      topics = topics + " "+ topic
```

Code Highlights: Put a Ring on It

```
try:
   for commits in repo.get_commits():
      if commits.committer :
         github_inventory['k%d' % row] =
                  commits.committer.login
         if …endswith("-zocdoc"):
            human += 1
      break
except GithubException as e:
   github_inventory['k%d' % row] =
                  e.args[1]['message']
```

Code Highlights: More Excel-lent Things

```
github_inventory['a%d' % row] = repo.id
github_inventory['b%d' % row] = repo.name
github_inventory['c%d' % row] = repo.description

github_inventory['d%d' % row] = repo.archived
if repo.archived == True:
    github_inventory['d%d' % row].fill =grassFill

github_inventory['e%d' % row] = repo.private
if repo.private == False:
    github_inventory['e%d' % row].fill = redFill

github_inventory['f%d' % row] = languages
github_inventory['g%d' % row] = teamlist
github_inventory['l%d' % row] = repo.pushed_at
```

Code Highlights: Unloved Repo

```
if topics.find("team-") > 0:
   github_inventory['h%d' % row] = topics
else:
   num_of_topic += 1
   github_inventory['h%d' % row].fill = redFill
   github_inventory['h%d' % row] = topics
   github_inventory['i%d' % row].fill = redFill
```

Code Highlights: Keys and Stats

```
    for key in repo.get_keys():
        deploy_keys_sheet[…] = repo.full_name
        deploy_keys_sheet[…] = str(key)
# end of main loop

# Repos with a team & percentage
summary_sheet['a2'] = num_of_topic
summary_sheet['b2'] = num_of_topic/count*100
wb.save(filepath)  ;# And save…
```

Code Highlights: Transdermal Topical Applications

```
if args["update"]:
   for each spreadsheet row:
      if row.topic_add != "":
         topics = row.topics.split()
         topics.extend(row.topic_add)
         user.get_repo("Zocdoc/%s" % row.repo_name)
            .replace_topics(topics)
      if row.topic_del != "":
         # … etc.
```

RSAConference2019

Ownership & Responsibility of AWS Resources

Determine AWS Resource ownership gaps leveraging the billing statements

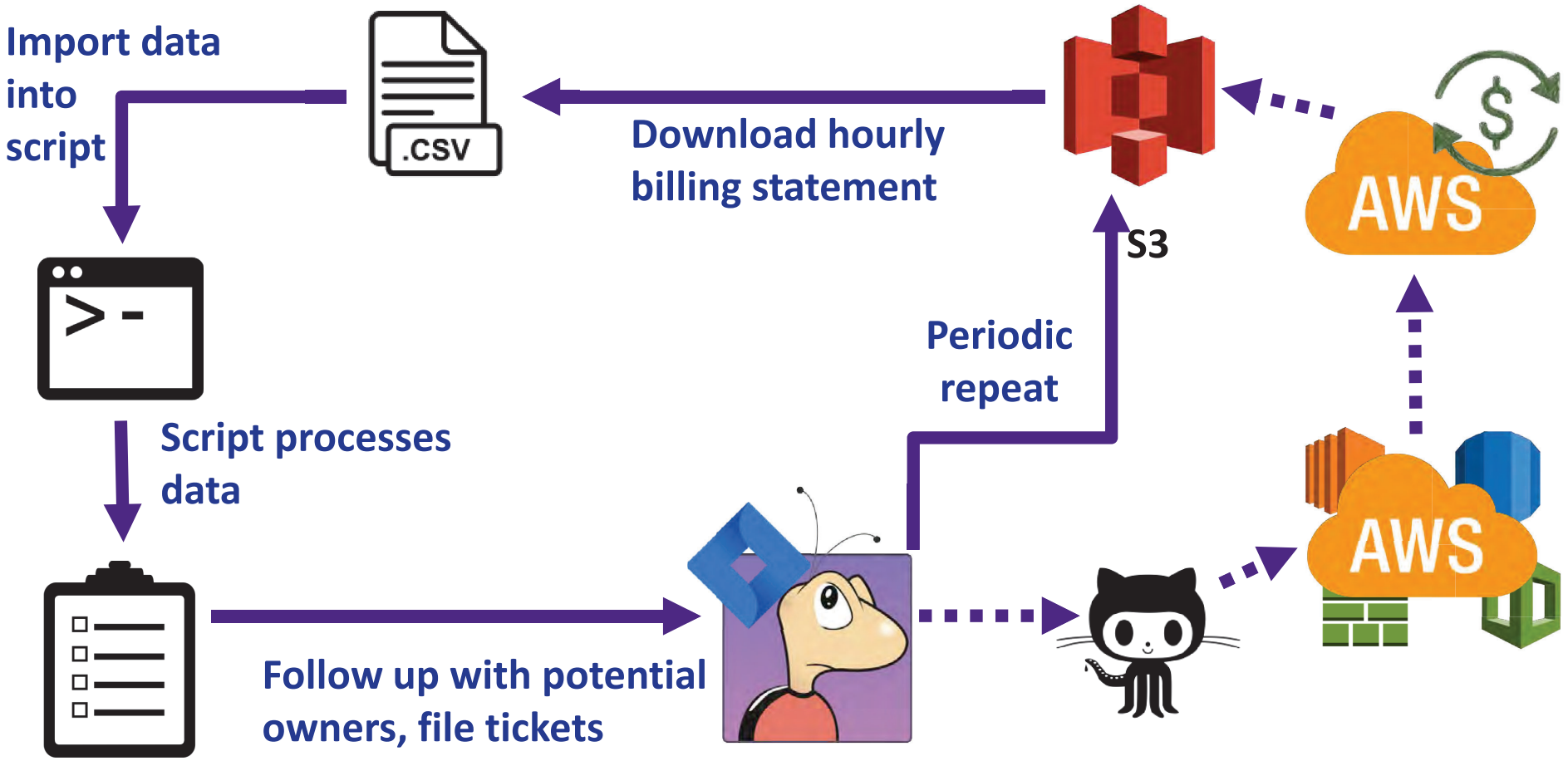
Services in the Cloud

- Pre-cloud, you know who owns servers and storage
  - Since you need hardware, it's easy for IT or Datacenter ops to track
- In-cloud, one can spin up a server or allocate storage without telling anybody
- Problems in Cloud:
  - We're paying for all of this
  - Infosec needs to know ownership during an incident
- Solutions in Cloud:
  - We wrote tools

Assigning Ownership

- Departmental ownership is tracked via AWS tags
- Most deployment is done via Ansible scripts
  - Tags are embedded in script or passed via environment variables
- Infrastructure spun up and torn down as need dictates
- Unlike GitHub, we cannot write easy auto-correction software
  - No point in fixing a tag for an interstitial item
- Owner updates via issue tracker and approved code pulls

AWS Billing and Ownership Flow



RSAConference2019

Recorded Demo

Determine AWS Resource ownership gaps via billing statements

Source Code Highlights: I sed a pun, I'm Bourne to do it

```
#!/bin/bash
input=$1

# convert csv titles to valid sql column names
sed -e '1 s/[:\/-]/_/g' -e '1 s/$/2/g' $1 \
        > temp.csv

sqlite3 < processing.sql
```

Source Code Highlights: Pun Wars II, the Sequel

```
.open cloudhealth-db.sqlite3
.mode csv
.import temp.csv cloudhealthtable

.mode tabs
.header on
.output s3_bucket_owners.txt

select distinct
    lineItem_UsageAccountId,
    lineItem_ResourceId,
    resourceTags_user_Owner
from cloudhealthtable
where
    lineItem_ProductCode = 'AmazonS3'  ;
```

Source Code Highlights: Pun Wars III, this is getting silly

```
.output ec2_owners.txt
select distinct
    lineItem_UsageAccountId,
    lineItem_ResourceId,
    resourceTags_user_Owner,
    resourceTags_user_Creator,
    resourceTags_user_Name,
    resourceTags_user_Project
        --- more useful columns...
from cloudhealthtable
where
    product_productFamily = 'Compute Instance' ;
```

RSAConference2019

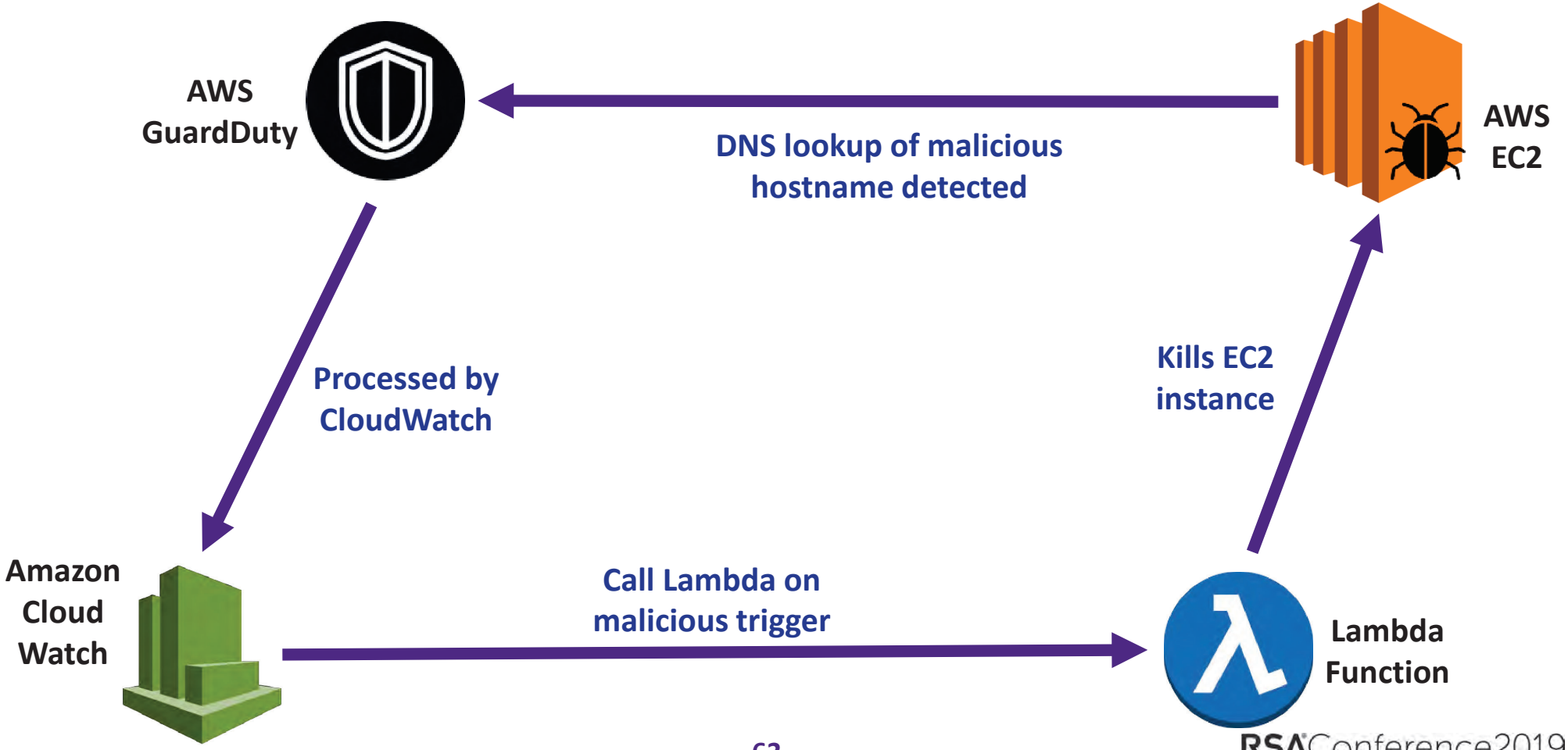
Compromised instance

Just kill it

Compromised systems in the Cloud

- Pre-cloud, leveraged network and security tools or manual processes
- In-cloud, visibility and tools are different
- Problems in Cloud:
  - Visibility
  - Time
  - Scale
- Solutions in Cloud:
  - We wrote a tool

Automated Detection & Response



RSAConference2019

Recorded Demo

***Et tu,* ec2?**

Cloud Watch Configuration

```
{
  "source":      [ "aws.guardduty"      ],
  "detail-type": [ "GuardDuty Finding"  ],
  "detail": {
    "type": [
              "Backdoor:EC2/C&CActivity.B!DNS"
            ]
  }
}
```

Source Code Highlights: Insanity Checks

```
def lambda_handler(event, context):
    source = event['source']
    if source != 'aws.guardduty':
        return ;# wrong caller, silently return

    detail_type = event['detail-type']
    if detail_type != 'GuardDuty Finding':
        return ;# wrong caller, silently return

    event_type = event['detail']['type']
    bad_event_list = [ 'Backdoor:EC2/C&CActivity.B!DNS' ]
    if not(event_type in bad_event_list):
        return ;# wrong event, silently return
```

Source Code Highlights: More Insanity

```
instance_id =
  event['detail']['resource']['instanceDetails']['instanceId']

# only shutdown certain tags:
taglist =
     event['detail']['resource']['instanceDetails']['tags']

for nvp in taglist:
    if "security_guillotine" == nvp['key'].lower():
        runme=True

if not(runme):
    # We are not part of auto-shutdown EC2 group
    return
```

Source Code Highlights: Euthanize It

```
print("Shutting down instance ", instance_id)

ec2 = boto3.client('ec2')
try:
    response = ec2.stop_instances(
          InstanceIds=[ instance_id ]  )

except ClientError as e:
    print('Error', e)
```

Eventful Situations

- GuardDuty alerts on many other events types
  - from mundane to pwned
- We can kill the instance in these critical situations
- Update CloudWatch config and `bad_event_list` variable

```
'Backdoor:EC2/XORDDOS',
'Backdoor:EC2/Spambot',
'Backdoor:EC2/C&CActivity.B!DNS',
'CryptoCurrency:EC2/BitcoinTool.B!DNS',
'Trojan:EC2/BlackholeTraffic',
'Trojan:EC2/DropPoint',
'Trojan:EC2/BlackholeTraffic!DNS',
'Trojan:EC2/DriveBySourceTraffic!DNS',
'Trojan:EC2/DropPoint!DNS',
'Trojan:EC2/DGADomainRequest.B',
'Trojan:EC2/DGADomainRequest.C!DNS',
'Trojan:EC2/DNSDataExfiltration',
'Trojan:EC2/PhishingDomainRequest!DNS'
```

RSAConference2019

Getting buy-in

How do you convince management

Cases to build buy-in

- S3 Bucket Encryption – Compliance / Fear
- VPC Port Inspection and Closure – Parity to non-cloud model
- Public Repos in GitHub – Data leakage and exposure
- GitHub Repo ownership – Auditability and closing vulns
- AWS Ownership – Auditability, billing, and monitoring
- Instance Termination – Incident Response, Containment, and Eradication

Apply What You Have Learned Today

- Determine where there is less cloud control versus on-prem
- How can you close this gap
- We give some tools, you can write others
- Small process improvements add up especially at scale

RSAConference2019

Code Available

Bonus: GitHub remediation code drops today

github.com/Zocdoc/ZocSec.SecurityAsCode.AWS

github.com/Zocdoc/ZocSec.SecurityAsCode.GitHub

RSAConference2019

Special thanks to

Mary Gu

Information Security Program Manager

Linked-In: mary-gu-ua

Gary Tsai

Application Security Engineer

Linked-In: garymalaysia
GitHub: garymalaysia

RSAConference2019

San Francisco | March 4–8 | Moscone Center

SESSION ID: CSV-W10

Thank you!

Jay Ball

Head of Application Security
Zocdoc
Linked-In: veggiespam
Twitter: @veggiespam
Security Blog: veggiespam.com
GitHub: veggiespam

Josh Barons

Head of Information Security
Zocdoc
Linked-In: jbarons
GitHub: CactusSec