Cloud Security
&
Best Practice in AWS
Resources
I am ..
Audit Script
AWS Best practices
Quick Overview
CRUX..
BREACHES
amazon web services
Security
BEST PRACTICES

I am ..

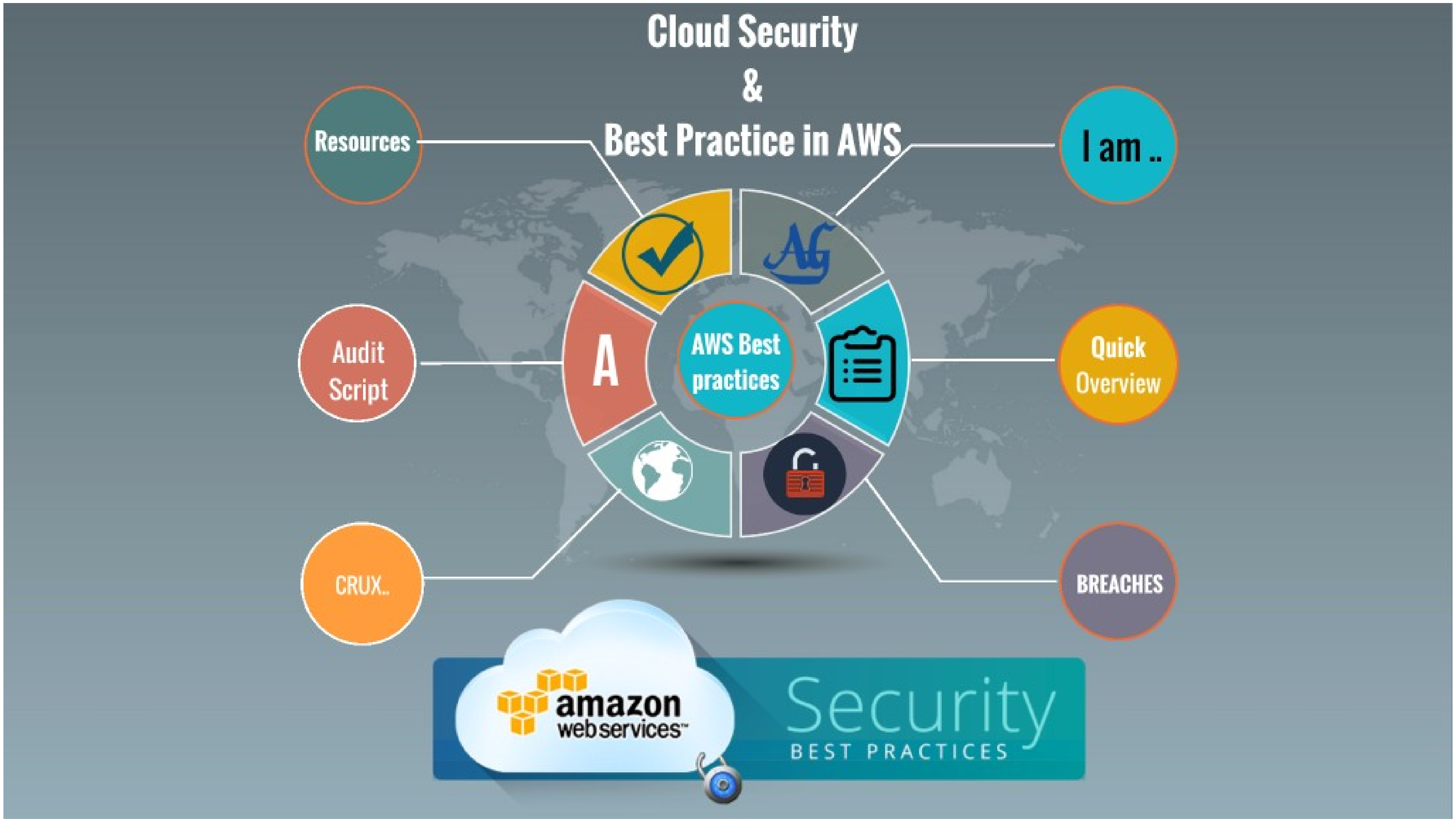
Ankit Giri (@aankitgiri)

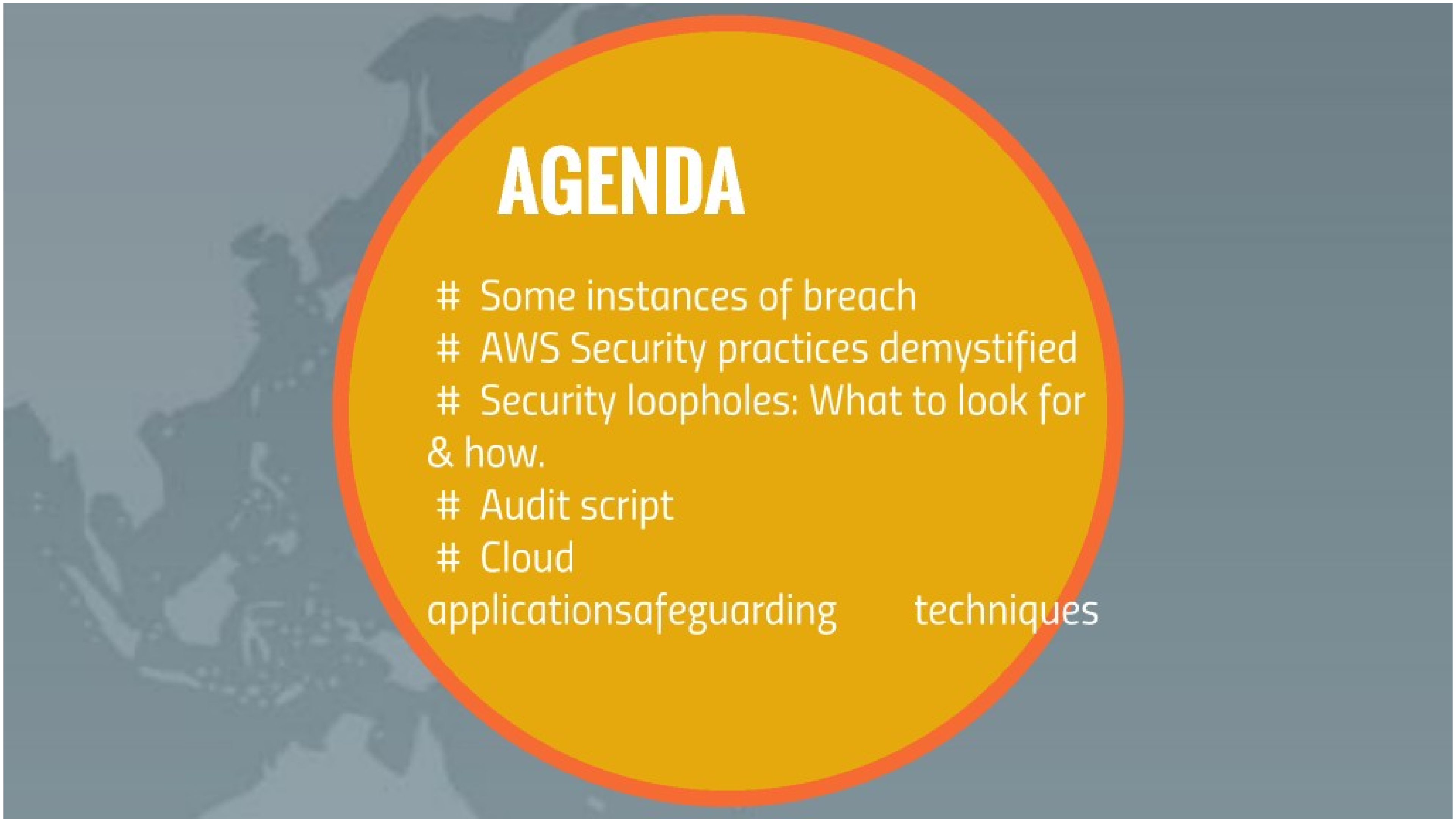
Associate Security Consultant | Security Compass

Web, Network, IoT, and Mobile Application Security Researcher.

Bug Hunter (Hall of Fame: EFF, GM, HTC, Sony, Mobikwik, AT&T, Pagerduty and more)

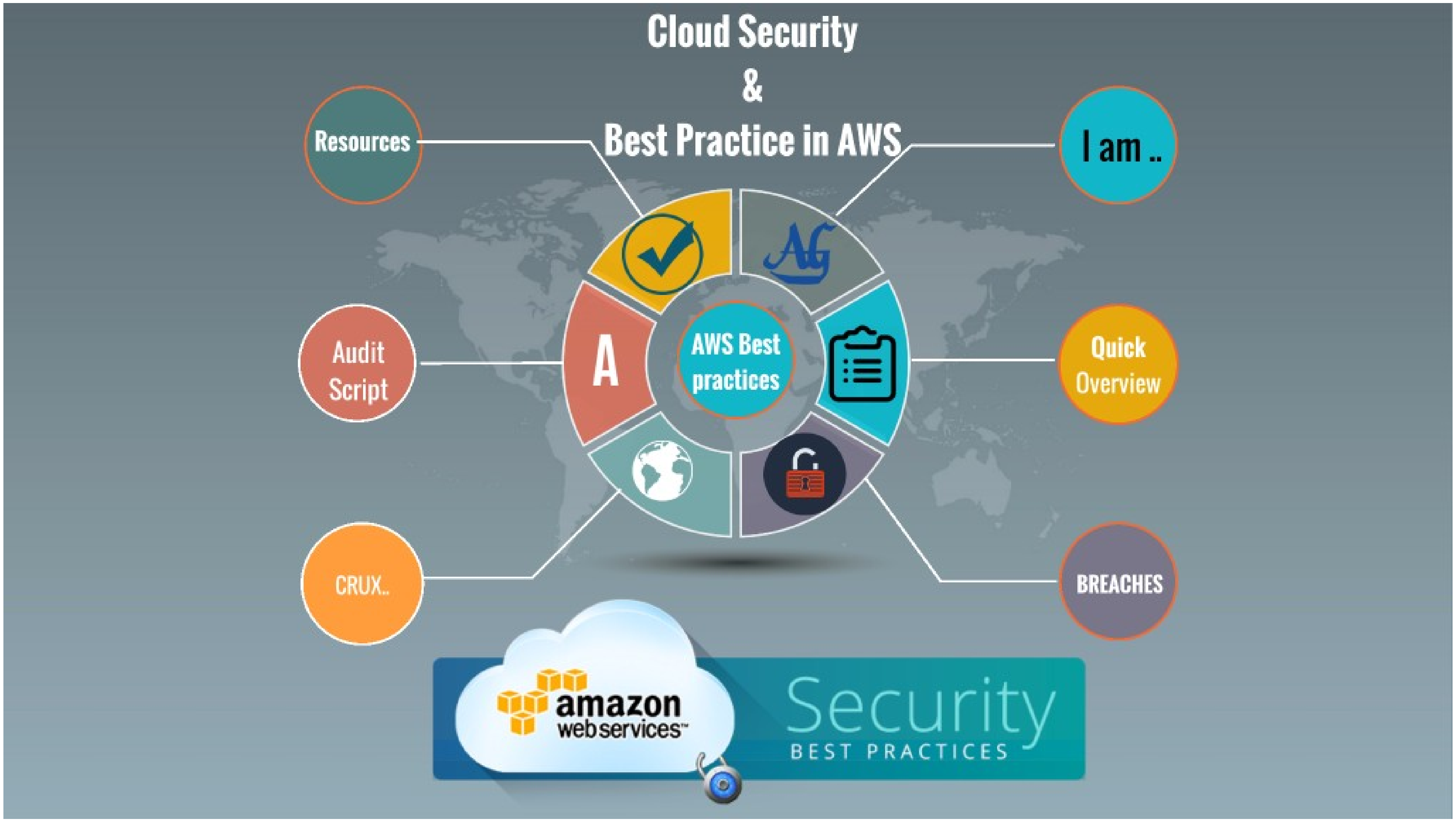
Blogger, Orator, and Active Contributor to OWASP & Null Community
Chapter Leader of Peerlyst Delhi-NCR Chapter


Cloud Security
&
Best Practice in AWS
Resources
I am ..
Audit Script
A
AWS Best practices
Quick Overview
CRUX..
BREACHES
amazon web services
Security
BEST PRACTICES



AGENDA

 # Some instances of breach
 # AWS Security practices demystified
 # Security loopholes: What to look for & how.
 # Audit script
 # Cloud applicationsafeguarding techniques


Cloud Security
&
Best Practice in AWS
Resources
I am ..
Audit Script
Quick Overview
CRUX..
BREACHES
AWS Best practices
A
amazon web services
Security
BEST PRACTICES

Breach Instances

Account compromise via leak of AWS Keys on GitHub

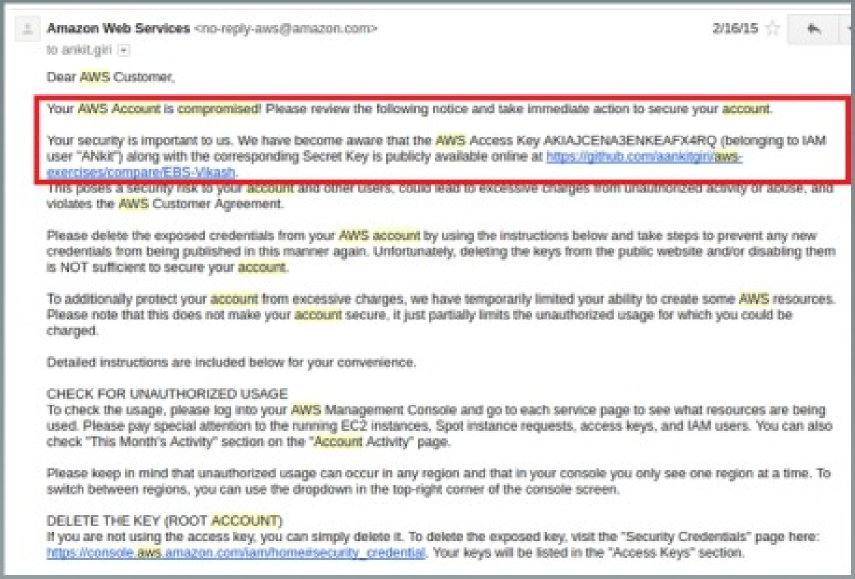
Cloud Infra can be Compromised ?

Publicly accessible S3 Bucket, Folders & Files

Can Jenkins Lead to a Disaster?

Account compromise via leak of AWS Keys on GitHub :

I woke up one morning to find a few emails and a missed phone call from an Amazon support number regarding the compromise of my AWS account.



Amazon Web Services <no-reply-aws@amazon.com> 2/16/15
to ankit.giri

Dear AWS Customer,

Your AWS Account is compromised! Please review the following notice and take immediate action to secure your account.

Your security is important to us. We have become aware that the AWS Access Key AKIAJCENA3ENKEAFX4RQ (belonging to IAM user "ANkit") along with the corresponding Secret Key is publicly available online at https://github.com/aankitgiri/aws-exercises/compare/EBS-Vikash.
This poses a security risk to your account and other users, could lead to excessive charges from unauthorized activity or abuse, and violates the AWS Customer Agreement.

Please delete the exposed credentials from your AWS account by using the instructions below and take steps to prevent any new credentials from being published in this manner again. Unfortunately, deleting the keys from the public website and/or disabling them is NOT sufficient to secure your account.

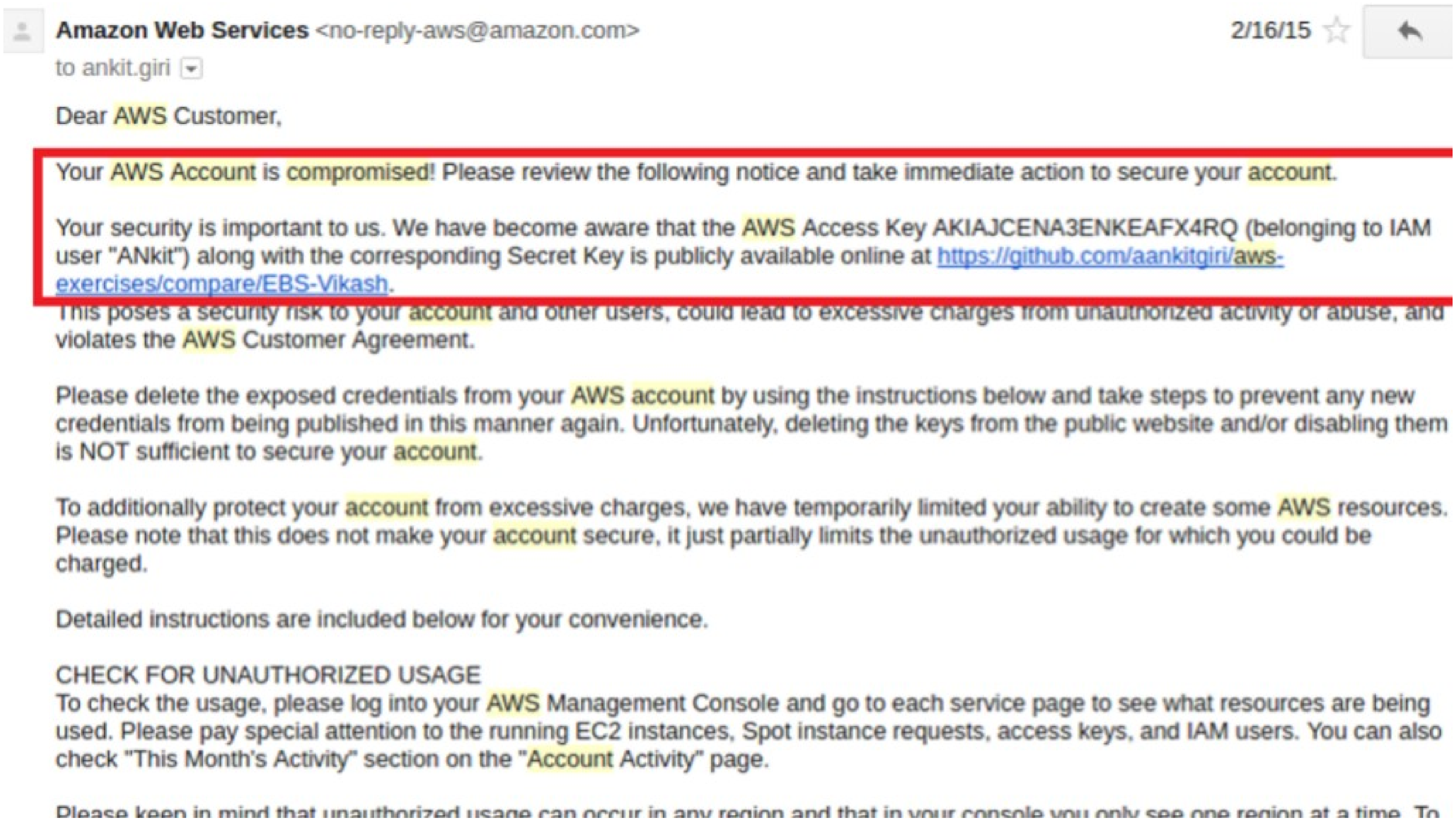
To additionally protect your account from excessive charges, we have temporarily limited your ability to create some AWS resources. Please note that this does not make your account secure, it just partially limits the unauthorized usage for which you could be charged.

Detailed instructions are included below for your convenience.

CHECK FOR UNAUTHORIZED USAGE
To check the usage, please log into your AWS Management Console and go to each service page to see what resources are being used. Please pay special attention to the running EC2 instances, Spot instance requests, access keys, and IAM users. You can also check "This Month's Activity" section on the "Account Activity" page.

Please keep in mind that unauthorized usage can occur in any region and that in your console you only see one region at a time. To switch between regions, you can use the dropdown in the top-right corner of the console screen.

DELETE THE KEY (ROOT ACCOUNT)
If you are not using the access key, you can simply delete it. To delete the exposed key, visit the "Security Credentials" page here: https://console.aws.amazon.com/iam/home#security_credential. Your keys will be listed in the "Access Keys" section.



Amazon Web Services <no-reply-aws@amazon.com> 2/16/15
to ankit.giri

Dear AWS Customer,

Your AWS Account is compromised! Please review the following notice and take immediate action to secure your account.

Your security is important to us. We have become aware that the AWS Access Key AKIAJCENA3ENKEAFX4RQ (belonging to IAM user "ANkit") along with the corresponding Secret Key is publicly available online at https://github.com/aankitgiri/aws-exercises/compare/EBS-Vikash.

This poses a security risk to your account and other users, could lead to excessive charges from unauthorized activity or abuse, and violates the AWS Customer Agreement.

Please delete the exposed credentials from your AWS account by using the instructions below and take steps to prevent any new credentials from being published in this manner again. Unfortunately, deleting the keys from the public website and/or disabling them is NOT sufficient to secure your account.

To additionally protect your account from excessive charges, we have temporarily limited your ability to create some AWS resources. Please note that this does not make your account secure, it just partially limits the unauthorized usage for which you could be charged.

Detailed instructions are included below for your convenience.

CHECK FOR UNAUTHORIZED USAGE
To check the usage, please log into your AWS Management Console and go to each service page to see what resources are being used. Please pay special attention to the running EC2 instances, Spot instance requests, access keys, and IAM users. You can also check "This Month's Activity" section on the "Account Activity" page.

Please keep in mind that unauthorized usage can occur in any region and that in your console you only see one region at a time. To

AWS Key Disclosure

What was my mistake in this case?

- Hardcoded keys within a script that was published to GitHub.
- Attackers are using bots to scrape public GitHub repos for keys.
- With the access and secret keys in hand, an attacker can fully interact with AWS API.
- A common attack is to spin up multiple EC2 instances and farm bitcoins.

The Root cause..

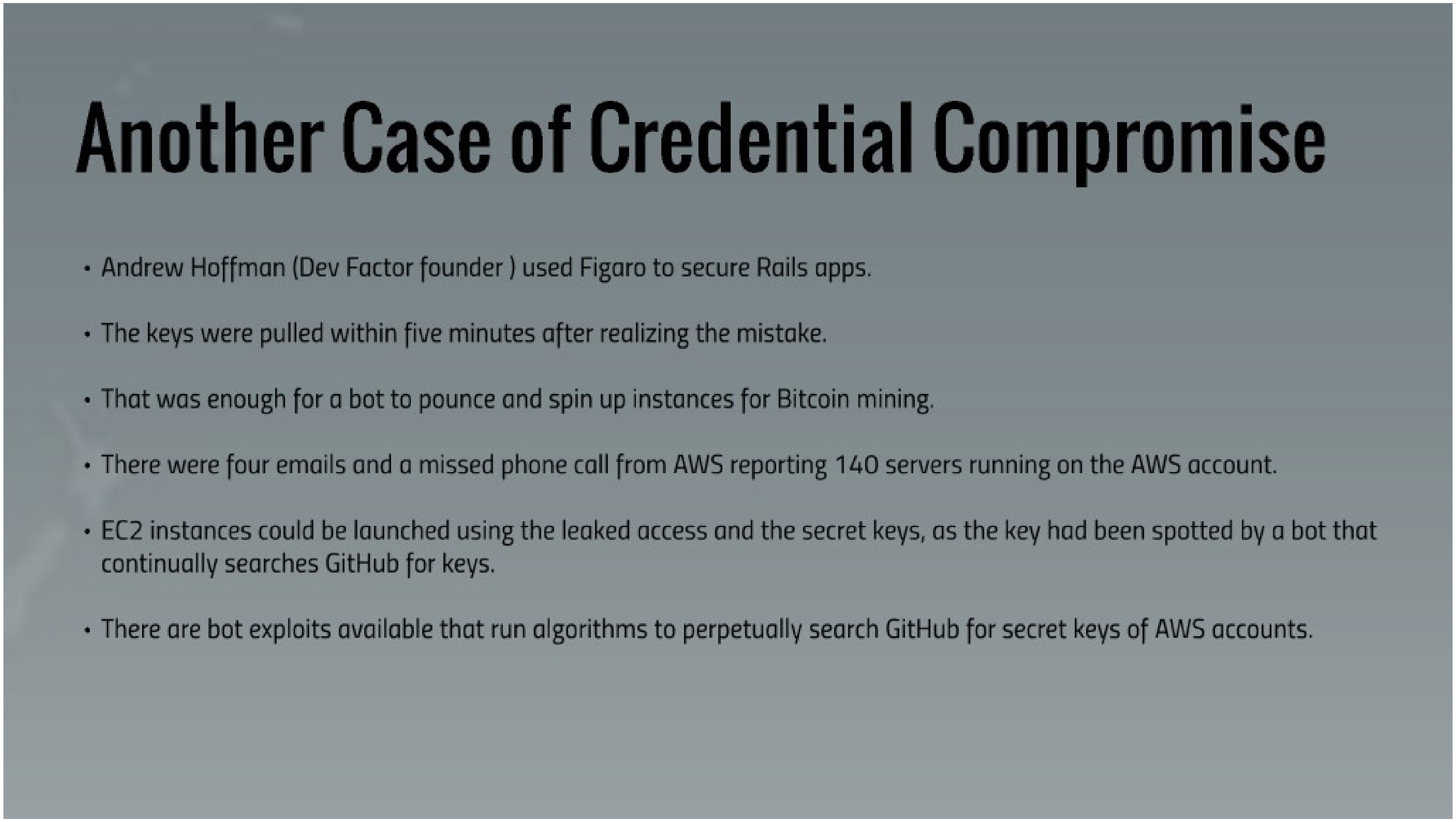
The secret keys are issued by Amazon Web Services when users open an account and provide applications with access to AWS resources.

When opening an account, users are told to "store the keys in a secure location," and are warned that the key needs to remain "confidential in order to protect your account."

AWS reminds subscribers that "anyone who has your access key has the same level of access to your AWS resources that you do. Consequently, we go to significant lengths to protect your access keys, and in keeping with our shared-responsibility model, you should as well."

However, a search on GitHub reveals thousands of results where code containing AWS secret keys can be found in plain text, which means anyone can access those accounts.

You can try searching for secret keys. It's fun.



Another Case of Credential Compromise

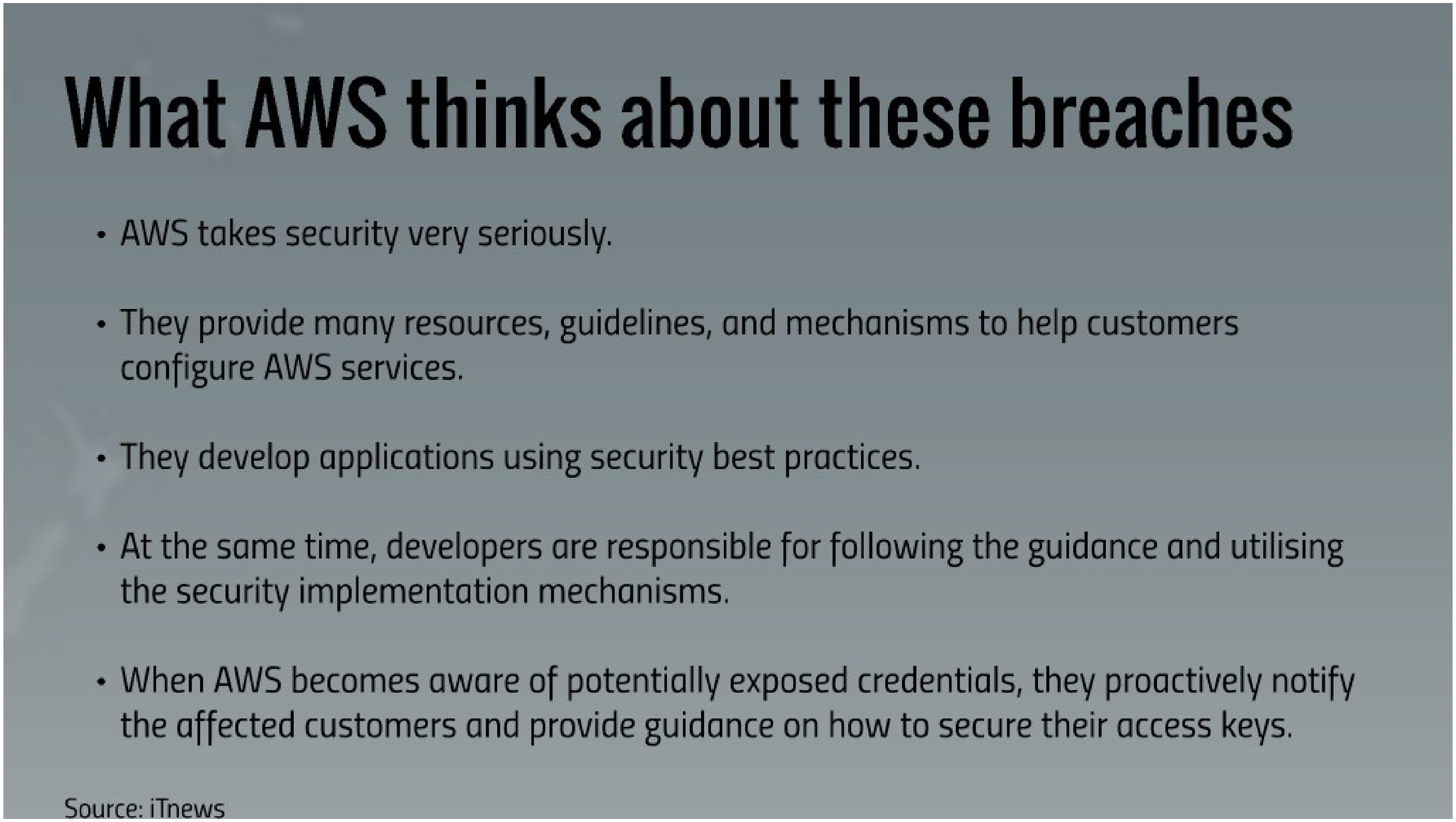
- Andrew Hoffman (Dev Factor founder) used Figaro to secure Rails apps.
- The keys were pulled within five minutes after realizing the mistake.
- That was enough for a bot to pounce and spin up instances for Bitcoin mining.
- There were four emails and a missed phone call from AWS reporting 140 servers running on the AWS account.
- EC2 instances could be launched using the leaked access and the secret keys, as the key had been spotted by a bot that continually searches GitHub for keys.
- There are bot exploits available that run algorithms to perpetually search GitHub for secret keys of AWS accounts.

A Few Instances of Breaches :

Several bloggers have admitted getting a shock after receiving large bills for bandwidth usage they didn't initiate. For example, Luke Chadwick was hit with a US$3493 (A$3842) bill in December because of unauthorised activity. To his relief, this was later refunded by AWS.

Earlier this year, AWS contacted Rich Mogull, analyst and CEO of Securosis, after three days of unusual activity on his account had run up US$500 in charges. In a blog post from January, Mogull said he had mistakenly published his AWS secret key on GitHub.

"I did not completely scrub my code before posting to GitHub. I did not have billing alerts enabled ... This was a real mistake ... I paid the price for complacency," he admitted in his blog."



What AWS thinks about these breaches

- AWS takes security very seriously.
- They provide many resources, guidelines, and mechanisms to help customers configure AWS services.
- They develop applications using security best practices.
- At the same time, developers are responsible for following the guidance and utilising the security implementation mechanisms.
- When AWS becomes aware of potentially exposed credentials, they proactively notify the affected customers and provide guidance on how to secure their access keys.

Source: iTnews

SSRF Attack (Live)

Organization B is complaining about random requests being flooded on their servers from one of Organization A's IPs.

Upon investigating the issue, it was found that the complaint was legitimate.

The servers belonging to Organization A were making random requests to different servers (not belonging to the internal network).
+ Increased the Network Out data from the instances, resulting in high billing on the account.

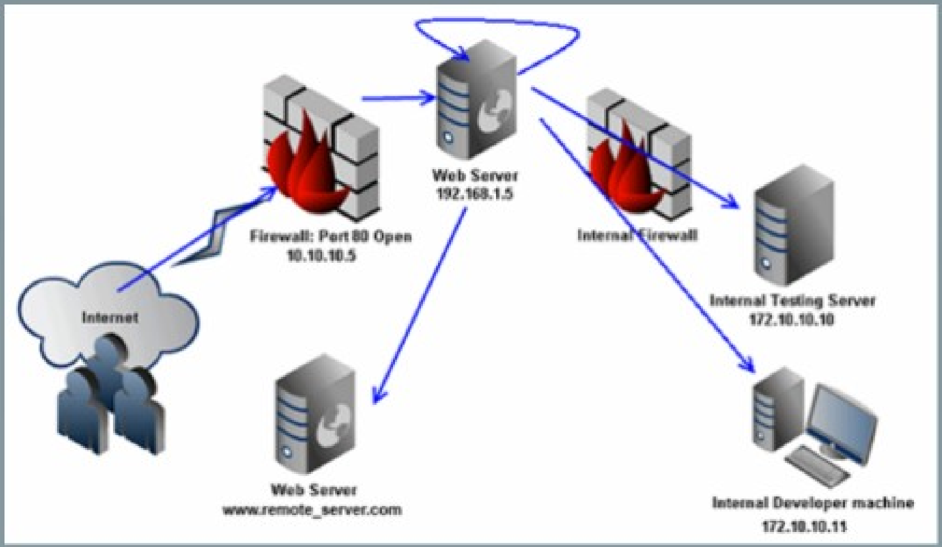
How Cloud Infra can be Compromised Through a Web Application

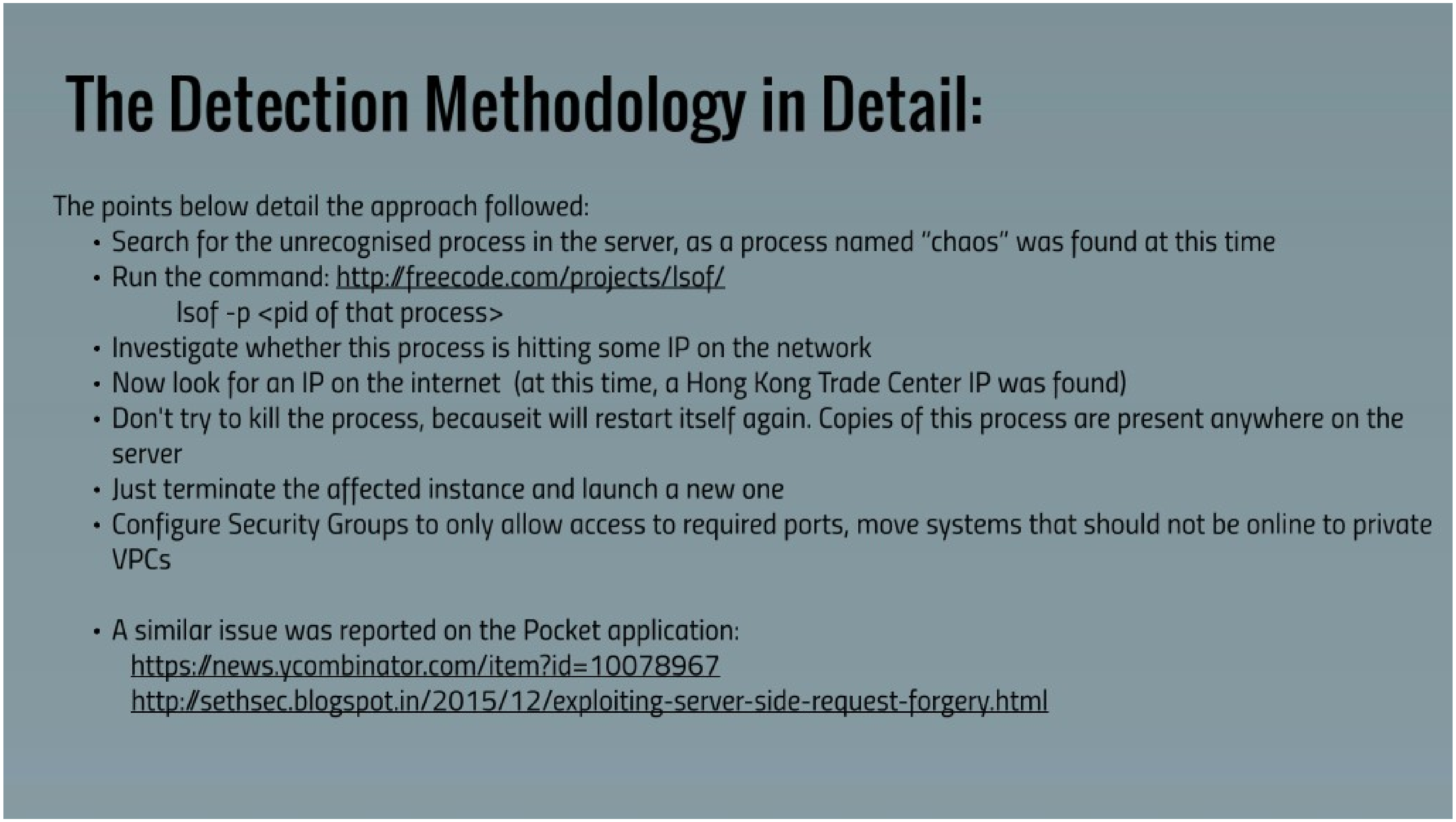
- Server Side Request Forgery (SSRF):

SSRF attacks target internal systems behind the firewall that are normally inaccessible from the outside world.

- With SSRF, the following is possible:

Access services from the same server that is listening on the loopback interface
Enumerate and attack services that are running on these hosts
Exploit host-based authentication services.





The Detection Methodology in Detail:

The points below detail the approach followed:

- Search for the unrecognised process in the server, as a process named "chaos" was found at this time
- Run the command: http://freecode.com/projects/lsof/

  ```
  lsof -p <pid of that process>
  ```
- Investigate whether this process is hitting some IP on the network
- Now look for an IP on the internet (at this time, a Hong Kong Trade Center IP was found)
- Don't try to kill the process, becauseit will restart itself again. Copies of this process are present anywhere on the server
- Just terminate the affected instance and launch a new one
- Configure Security Groups to only allow access to required ports, move systems that should not be online to private VPCs

- A similar issue was reported on the Pocket application:
  https://news.ycombinator.com/item?id=10078967
  http://sethsec.blogspot.in/2015/12/exploiting-server-side-request-forgery.html

There's a Hole in 1,951 Amazon S3 Buckets :

Rapid7 discovered 12 328 unique buckets of the following types:
Public: 1951
Private: 10 377

Approximately 1 in 6 buckets of the 12 328 identified were left open for the perusal of anyone who's interested.

These 12 328 buckets were skewed towards those identified based on the following:
domain name, word list, and use within websites

From the 1951 public buckets that were gathered, a list of over 126 billion files was found.
The sheer number of files made it unrealistic to test the permissions of every single object, so a random sampling was taken instead.
Over 40 000 publicly visible files were reviewed, and many contained sensitive data.

Some specific examples of the data found are listed below:

- Personal photos from a medium-sized social media service
- Sales records and account information for a large car dealership
- Affiliate tracking data, click-through rates, and account information for an ad company's clients
- Employee personal information and member lists across various spreadsheets
- Unprotected database backups containing site data and encrypted passwords
- Video game source code and development tools for a mobile gaming firm
- PHP source code including configuration files containing usernames and passwords
- Sales "battlecards" for a large software vendor

Root Cause :

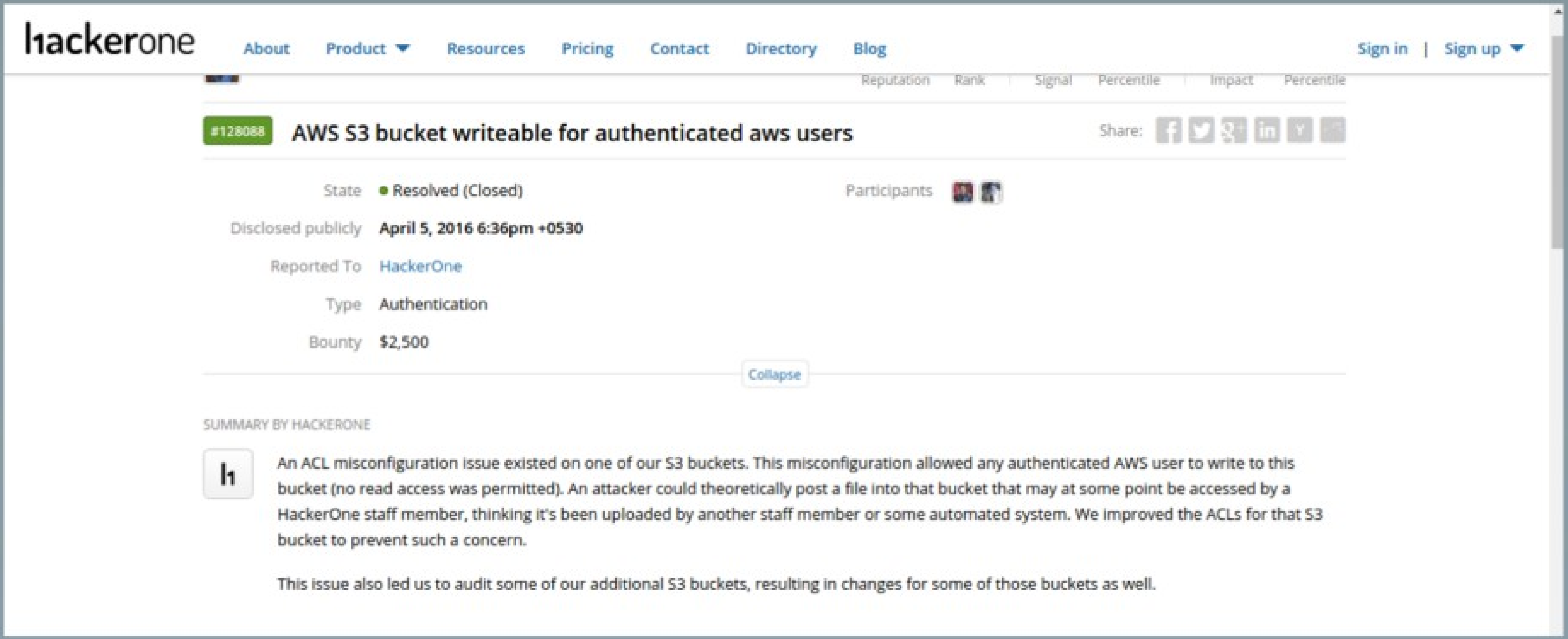
There isn't a security hole in Amazon's storage cloud. Some Amazon S3 account holders fail to set their buckets to private. However, the fact that all S3 buckets have predictable, publicly accessible URLs doesn't help.

What makes it a common vulnerability?
Buckets and objects have their own access control lists (ACLs). However, if a user does lock down files within a public bucket, a data thief could still glean potentially sensitive information from the file names, including customer names or how frequently applications are backed up.

All S3 buckets have a unique, predictable, and have a publicly accessible URL, which makes it easy to track down buckets and determine which are private and which are public. This may contribute to the problem. By default, this URL will be either

http://s3.amazonaws.com/[bucket_name]/ or http://[bucket_name].s3.amazonaws.com/.

Publicly Accessible S3 Bucket, Folders and Files:



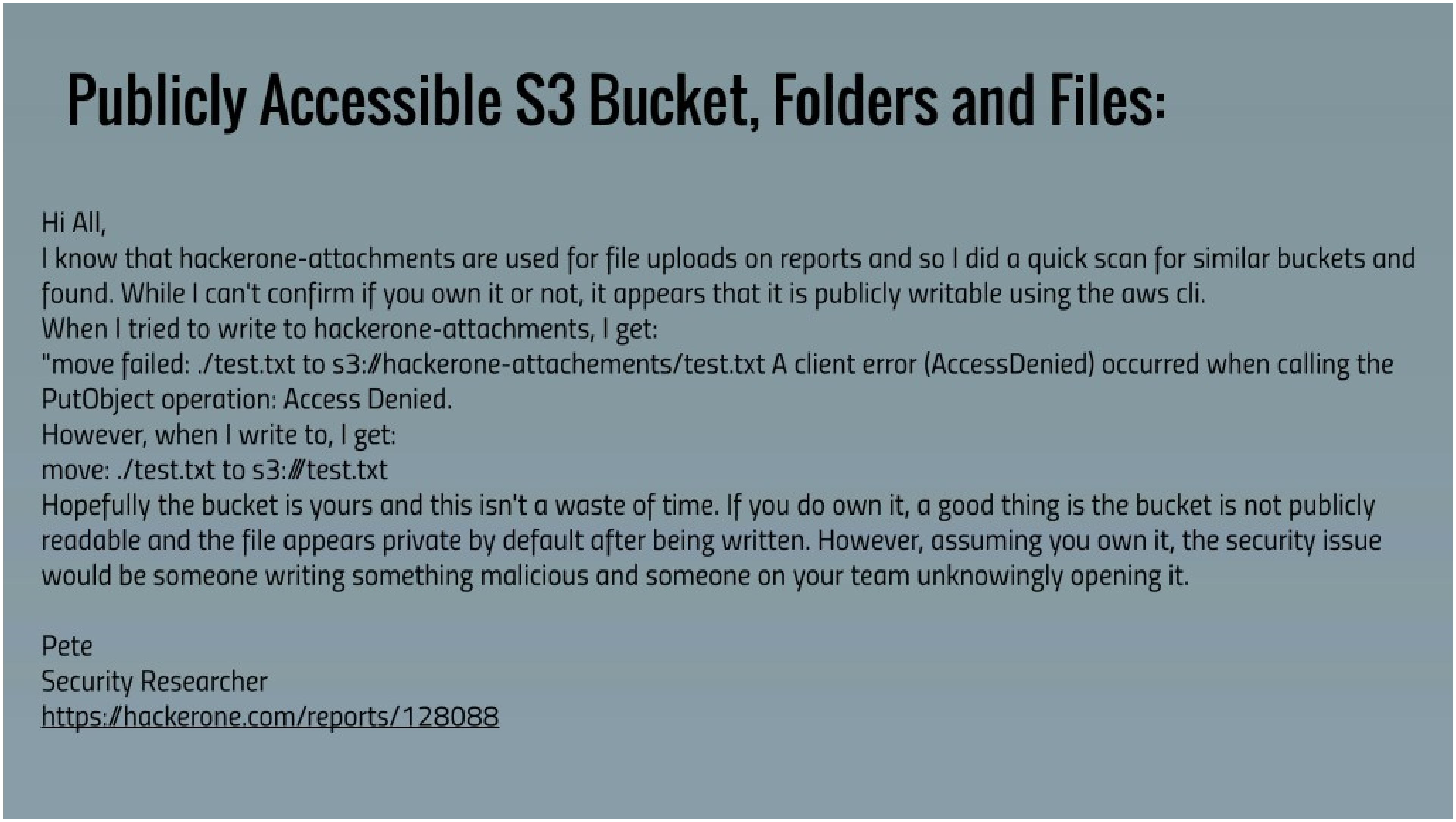
#128088 AWS S3 bucket writeable for authenticated aws users

State	● Resolved (Closed)	Participants
Disclosed publicly	April 5, 2016 6:36pm +0530	
Reported To	HackerOne	
Type	Authentication	
Bounty	$2,500	

SUMMARY BY HACKERONE

An ACL misconfiguration issue existed on one of our S3 buckets. This misconfiguration allowed any authenticated AWS user to write to this bucket (no read access was permitted). An attacker could theoretically post a file into that bucket that may at some point be accessed by a HackerOne staff member, thinking it's been uploaded by another staff member or some automated system. We improved the ACLs for that S3 bucket to prevent such a concern.

This issue also led us to audit some of our additional S3 buckets, resulting in changes for some of those buckets as well.



Publicly Accessible S3 Bucket, Folders and Files:

Hi All,
I know that hackerone-attachments are used for file uploads on reports and so I did a quick scan for similar buckets and found. While I can't confirm if you own it or not, it appears that it is publicly writable using the aws cli.
When I tried to write to hackerone-attachments, I get:
"move failed: ./test.txt to s3://hackerone-attachements/test.txt A client error (AccessDenied) occurred when calling the PutObject operation: Access Denied.
However, when I write to, I get:
move: ./test.txt to s3:///test.txt
Hopefully the bucket is yours and this isn't a waste of time. If you do own it, a good thing is the bucket is not publicly readable and the file appears private by default after being written. However, assuming you own it, the security issue would be someone writing something malicious and someone on your team unknowingly opening it.

Pete
Security Researcher
https://hackerone.com/reports/128088

Publicly Accessible S3 Bucket, Folders and Files:

HackerOne rewarded yaworsk with a $2500 bounty.
4thApril, 2016
Thanks again for the great report, @yaworsk.

It's always great to see researchers thinking outside the normal web app confines and considering other parts of technical infrastructure that could affect a company. While you weren't able to read any contents from that bucket (and even if you would, everything is encrypted using AES-256), an attacker could theoretically post a file into that bucket that may at some point be accessed by a HackerOne staff member, thinking it's been uploaded by another staff member or some automated system.

This issue also led us to audit some of our additional S3 buckets (we have quite a few), and we found some similar issues among several of them. As such, we are increasing the bounty to take those additional mitigations into account, even though they weren't discovered by you.

Look forward to seeing what you find next! Happy hacking. :-)

Another Case where S3 Bucket Data was Leaked:

Recently, an Amazon S3 bucket (http://shopify.com.s3.amazonaws.com/) was unintentionally left with directory listing enabled. Even though the files in the bucket were all publicly accessible, it was not intended for the directory listing to be visible.

https://hackerone.com/reports/57505

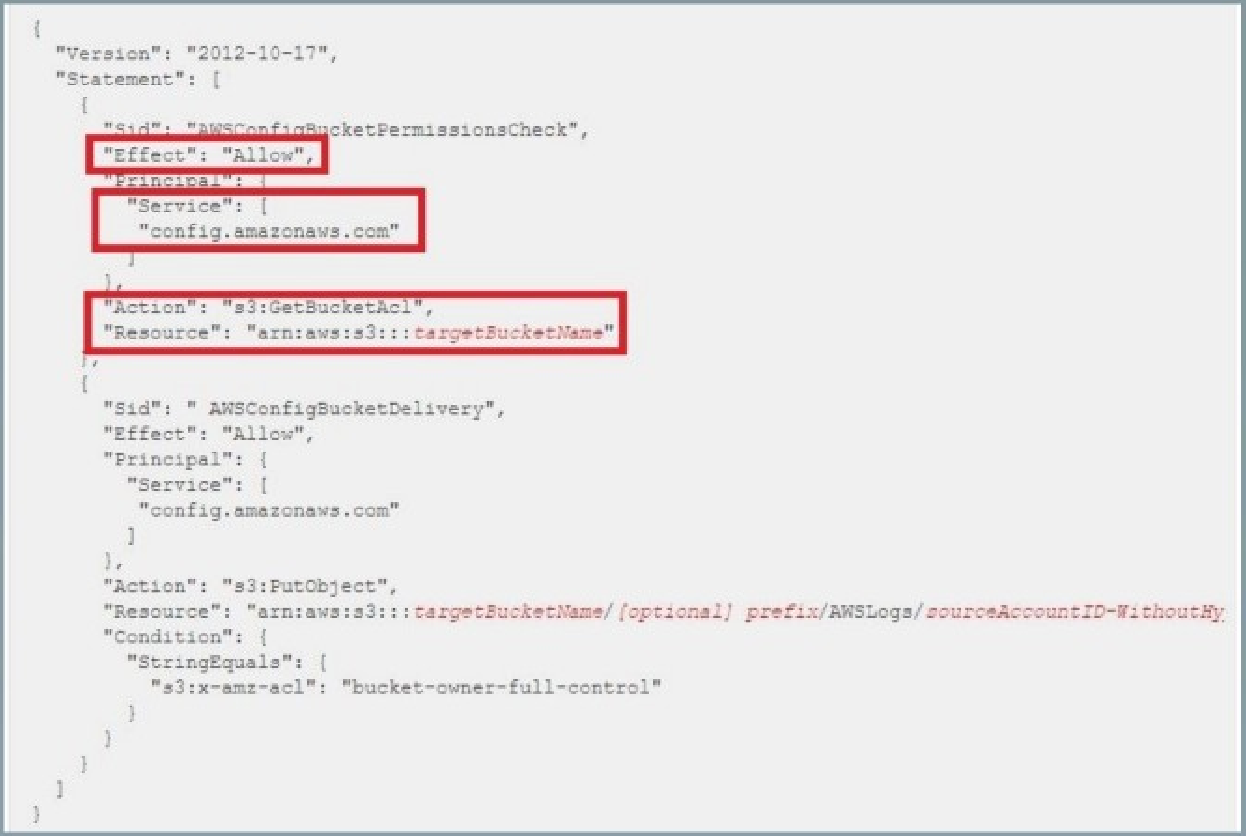
Bucket Finder Script:

How to use and exploit scenario:

http://hackoftheday.securitytube.net/2013/04/finding-publicly-readable-files-in-your.html

https://digi.ninja/projects/bucket_finder.php

How to Get a Hold of AWS S3?



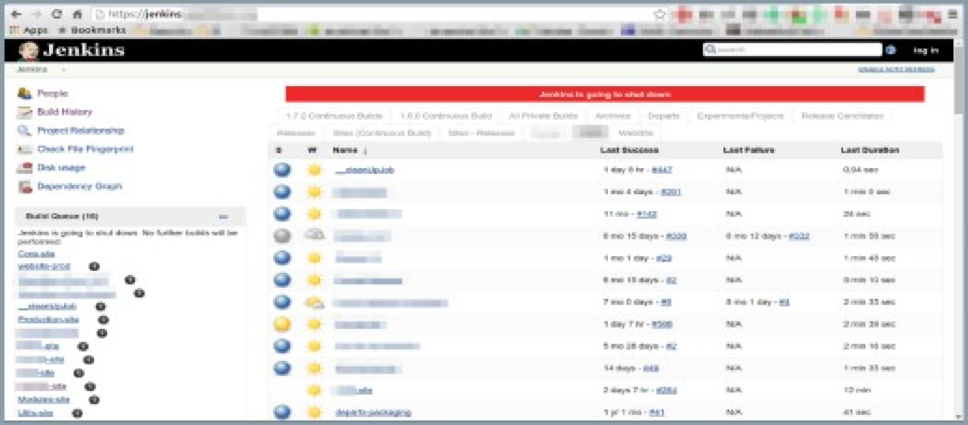
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AWSConfigBucketPermissionsCheck",
      "Effect": "Allow",
      "Principal": {
        "Service": [
         "config.amazonaws.com"
        ]
      },
      "Action": "s3:GetBucketAcl",
      "Resource": "arn:aws:s3:::targetBucketName"
    },
    {
      "Sid": " AWSConfigBucketDelivery",
      "Effect": "Allow",
      "Principal": {
        "Service": [
         "config.amazonaws.com"
        ]
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::targetBucketName/[optional] prefix/AWSLogs/sourceAccountID-WithoutHy
      "Condition": {
        "StringEquals": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    }
  ]
}
```

Can Jenkins Lead to a Disaster?

- To simplify and automate code deployments on AWS, there are multiple tools used
- Atlassian Bamboo, Jenkins, and Hudson.

- During my recent search for discovering admin consoles and critical applications, I came across this instance of what might be a logical vulnerability prevailing across multiple web applications.

- I was searching for a publicly accessible Jenkins console through Google Dorking. My search query listed some of the websites that had Jenkins as a part of their domain name.



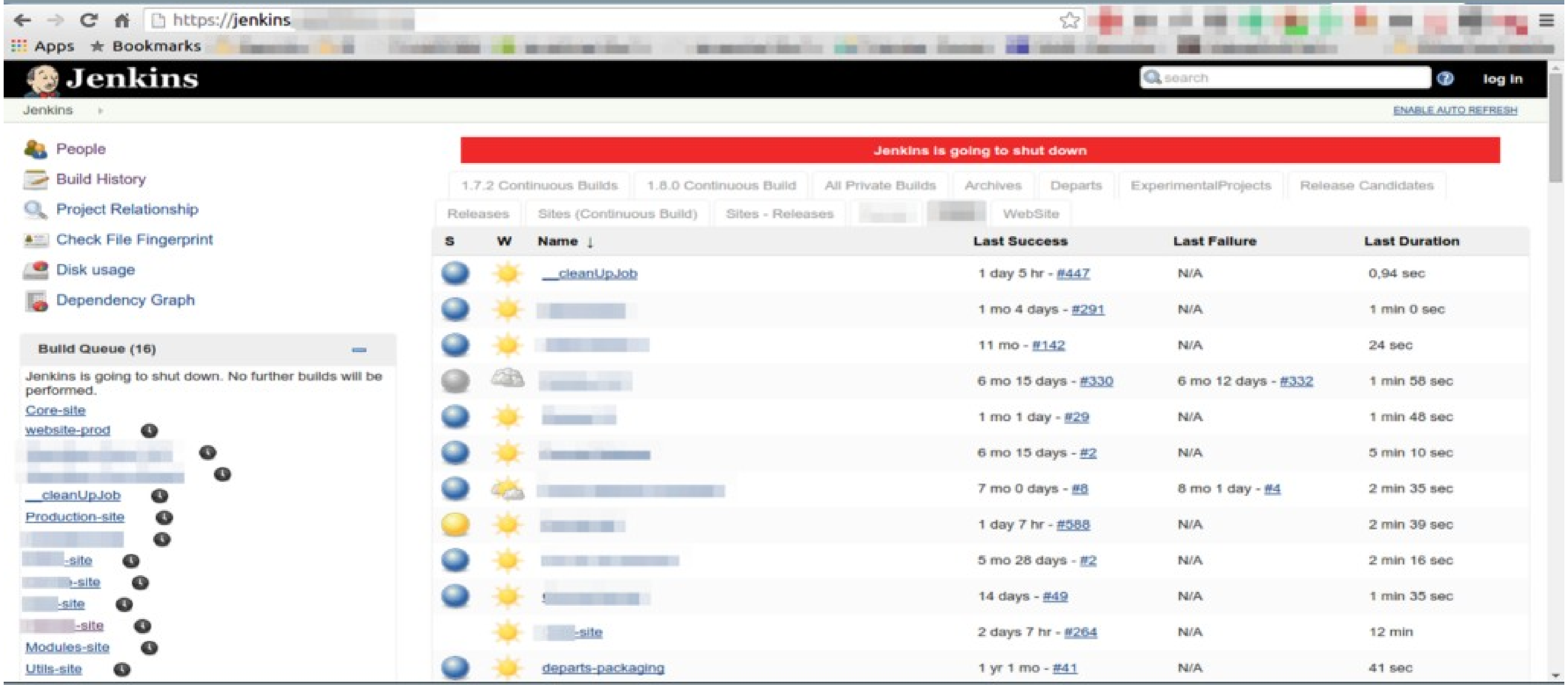
Build Queue (16)

Jenkins is going to shut down. No further builds will be performed.

Core.site
website-prod
[redacted]
__cleanUpJob
Production.site
[redacted]
[redacted].site
[redacted]-site
[redacted].site
[redacted]-site
Modules.site
Ultra.site

S	W	Name ↓	Last Success	Last Failure	Last Duration
		__cleanUpJob	1 day 8 hr - #447	N/A	0.04 sec
		[redacted]	1 mo 4 days - #201	N/A	1 min 0 sec
		[redacted]	11 mo - #142	N/A	24 sec
		[redacted]	6 mo 15 days - #330	6 mo 12 days - #332	1 min 50 sec
		[redacted]	1 mo 1 day - #23	N/A	1 min 45 sec
		[redacted]	6 mo 15 days - #2	N/A	8 min 10 sec
		[redacted]	7 mo 0 days - #5	8 mo 1 day - #4	2 min 33 sec
		[redacted]	1 day 7 hr - #300	N/A	2 min 39 sec
		[redacted]	5 mo 28 days - #2	N/A	2 min 16 sec
		[redacted]	14 days - #48	N/A	1 min 35 sec
		[redacted].site	2 days 7 hr - #284	N/A	12 min
		deploy-packaging	1 yr 1 mo - #41	N/A	41 sec

s that had Jenkins as a part of their domain name.



https://jenkins[redacted]

Apps ★ Bookmarks [redacted]

Jenkins

search · log in

Jenkins ›

ENABLE AUTO REFRESH

- People
- Build History
- Project Relationship
- Check File Fingerprint
- Disk usage
- Dependency Graph

Build Queue (16)

Jenkins is going to shut down. No further builds will be performed.

- Core-site
- website-prod
- [redacted]
- [redacted]
- __cleanUpJob
- Production-site
- [redacted]
- [redacted]-site
- [redacted]-site
- [redacted]-site
- [redacted]-site
- Modules-site
- Utils-site

Jenkins is going to shut down

1.7.2 Continuous Builds | 1.8.0 Continuous Build | All Private Builds | Archives | Departs | ExperimentalProjects | Release Candidates | Releases | Sites (Continuous Build) | Sites - Releases | [redacted] | [redacted] | WebSite

S	W	Name ↓	Last Success	Last Failure	Last Duration
●	☀	__cleanUpJob	1 day 5 hr - #447	N/A	0,94 sec
●	☀	[redacted]	1 mo 4 days - #291	N/A	1 min 0 sec
●	☀	[redacted]	11 mo - #142	N/A	24 sec
●	☁	[redacted]	6 mo 15 days - #330	6 mo 12 days - #332	1 min 58 sec
●	☀	[redacted]	1 mo 1 day - #29	N/A	1 min 48 sec
●	☀	[redacted]	6 mo 15 days - #2	N/A	5 min 10 sec
●	⛅	[redacted]	7 mo 0 days - #8	8 mo 1 day - #4	2 min 35 sec
●	☀	[redacted]	1 day 7 hr - #588	N/A	2 min 39 sec
●	☀	[redacted]	5 mo 28 days - #2	N/A	2 min 16 sec
●	☀	[redacted]	14 days - #49	N/A	1 min 35 sec
	☀	[redacted]-site	2 days 7 hr - #264	N/A	12 min
●	☀	departs-packaging	1 yr 1 mo - #41	N/A	41 sec

Can Jenkins Lead to a Disaster?

What took me by surprise is that I found many Jenkins consoles with no authentication mechanism enabled on them. I can easily open the **console, read the usernames, the build history, logs of builds**, and much more. This made me think about the necessity of having proper security implemented on a Jenkins console. So, I went ahead and wrote a blog post covering the devastating effect of a compromised Jenkins server and how to protect this from happening:

http://www.tothenew.com/blog/why-compromised-jenkins-can-lead-to-a-disaster/

Why Can Compromised Jenkins Lead to a Disaster?

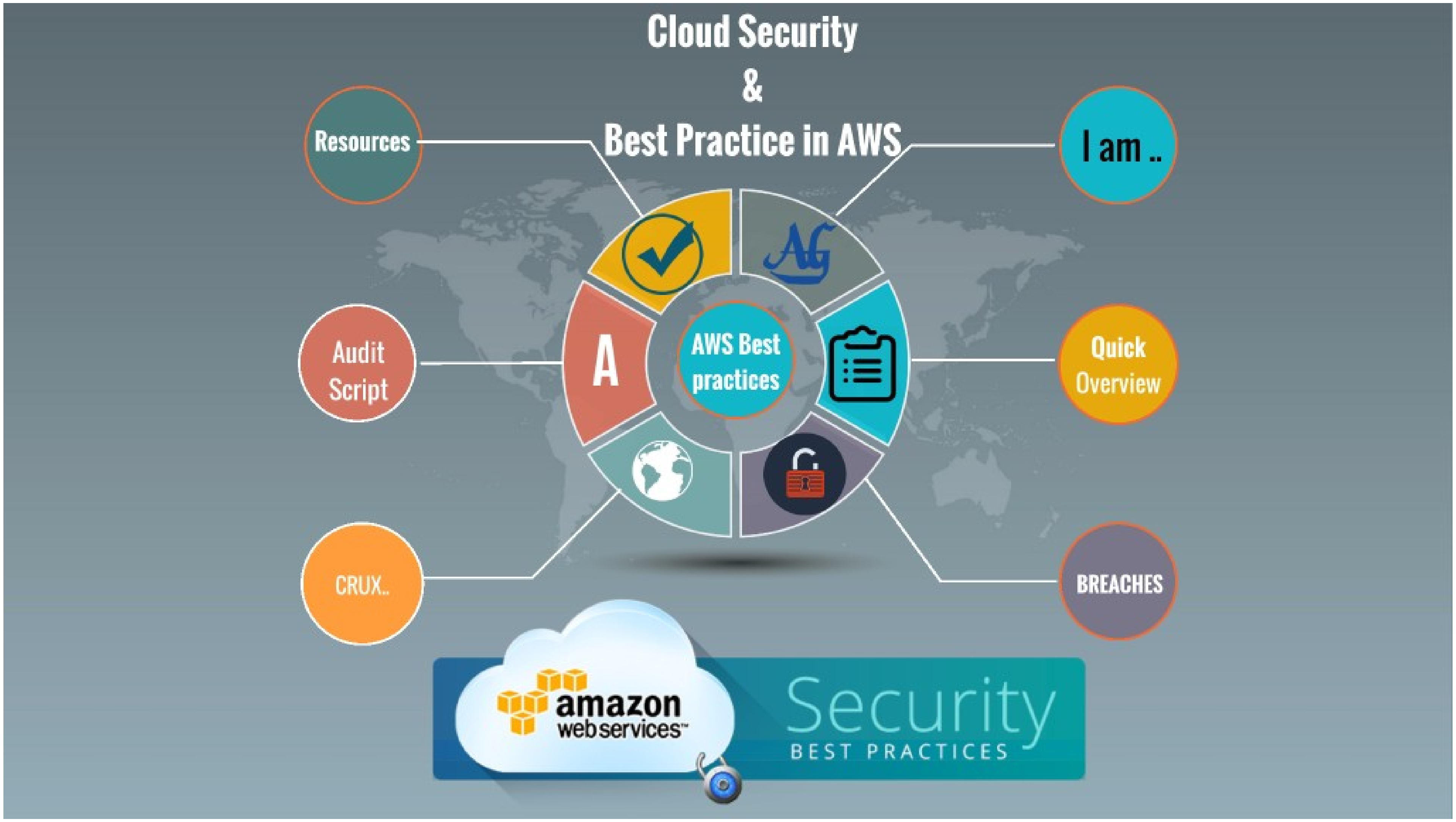
An Exposed Treasure of Credentials such as AWS Access Keys and Secret Key
Impact: Anyone can use the AWS account linked to the Access keys and Secret keys to launch resources, leading to the owner being billed for it. A person possessing these credentials gets full access to the AWS account.

Exposed Server PEM files, IP Addresses, Usernames, Email Address, etc.
Impact: The disclosure of the above mentioned information will lead to hackers logging in to the server (and there may be hundreds of servers accessible from this console), running arbitrary commands on it, getting access to users, their respective password, and whatnot!

GitHub SSH Key
With the Git plugin installed on Jenkins, Git commands can be run from the Jenkins console itself. So, the publicly accessible Jenkins console will enable an attacker to view, modify, and update the code of the production application.

In the worst-case scenario, the attacker can issue a git delete command and delete all the existing code, bringing the application down by deleting the "production" branch containing the application's code.

S3 Bucket
The Jenkins console might have access to private S3 buckets containing content such as images, log files, code backups, and more. This access to essential data can be abused to delete important files and folders.


Cloud Security
&
Best Practice in AWS
Resources
I am ..
Audit Script
Quick Overview
CRUX..
BREACHES
AWS Best practices
A
amazon web services
Security
BEST PRACTICES

Best Practices to Protect AWS Accounts from Unauthorized Access and Usage:

1

2

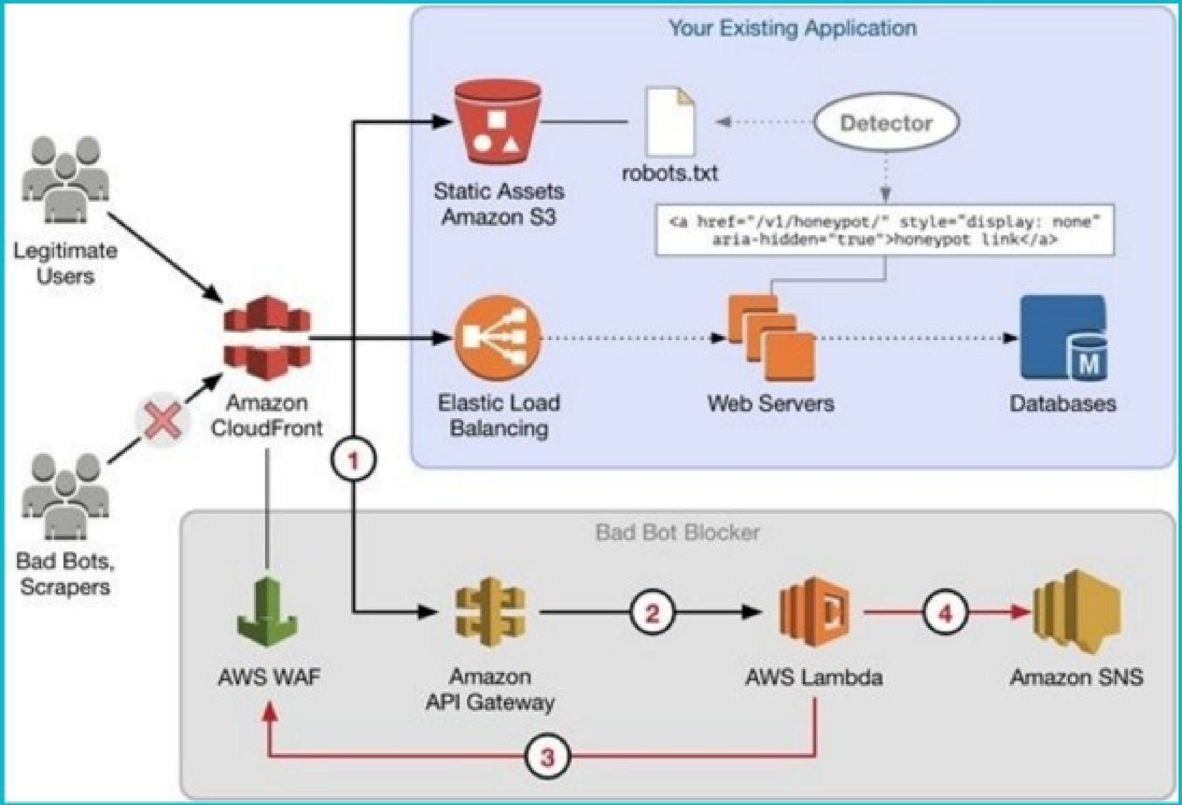
- Disable the root access key and secret key
- Enable use of MFA for each IAM user
- Have minimal IAM users with Admin rights : Provide least privileges and limit IAM entity's actions with strong/explicit policies
- Use Roles for EC2 instances for access permissions wherever possible.
- Use IAM roles with STS AssumeRole wherever possible
- Rotate all the keys on a regular basis (i.e. 60 days)
- Do not allow 0.0.0.0/0 for any EC2/ELB security group
- Apply proper S3 bucket policies with public access to only files instead of sharing a bucket/ folder
- Create all resources within a VPC
- Make sure only required ports are open on EC2 instances

Additional Best Practices to Protect AWS Accounts:

- Ensure all Back-end servers (i.e. database servers) are accessible via web servers only
- Have a separate IAM user for each team member
- Attach all policies on groups and include users in a group as per access requirement
- Avoid making the server_Status page accessible publicly
- Ensure all SSL certificates follow SHA512 encryption
- Take care when migrating to SHA512
- Set alarms on billing using Amazon CloudWatch
- This practice can be very useful to detect DDoS attacks and high data transfer occurrences

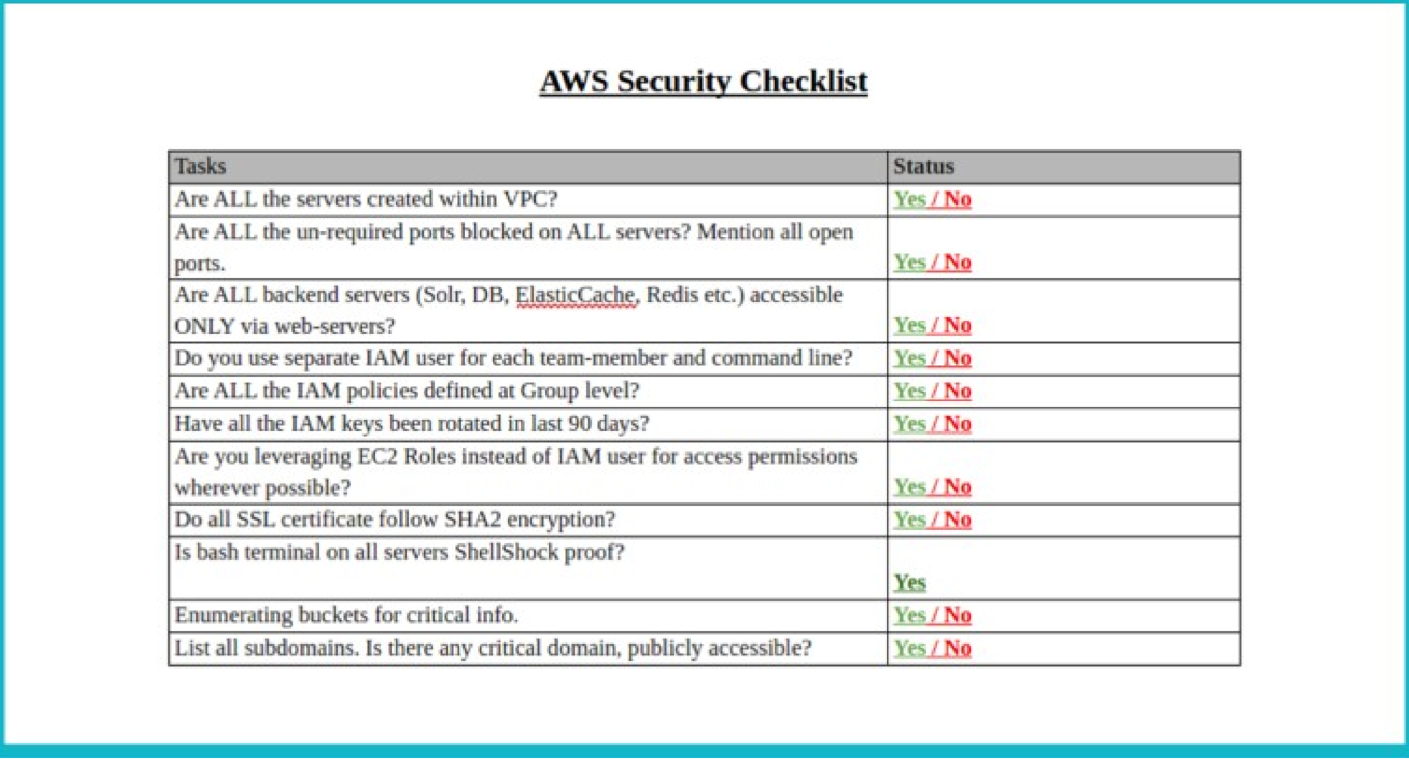
See the following to set alerts on billing: http://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/free-tier-alarms.html

Best Practices to Protect AWS Accounts from Unauthorized Access and Usage:



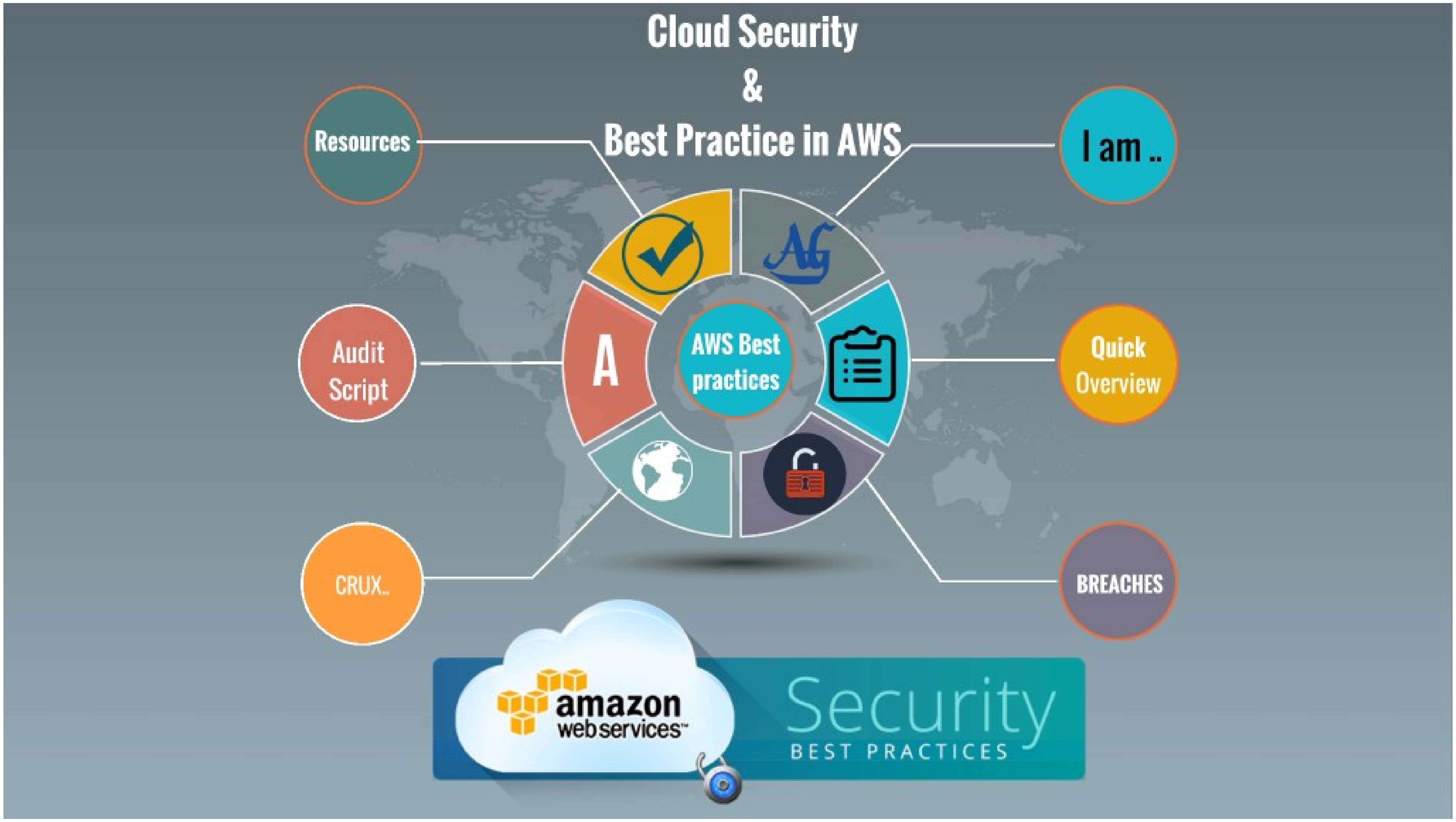
```
<a href="/v1/honeypot/" style="display: none"
aria-hidden="true">honeypot link</a>
```

Best Practices to Protect AWS Accounts from Unauthorized Access and Usage:



AWS Security Checklist

Tasks	Status
Are ALL the servers created within VPC?	Yes / No
Are ALL the un-required ports blocked on ALL servers? Mention all open ports.	Yes / No
Are ALL backend servers (Solr, DB, ElasticCache, Redis etc.) accessible ONLY via web-servers?	Yes / No
Do you use separate IAM user for each team-member and command line?	Yes / No
Are ALL the IAM policies defined at Group level?	Yes / No
Have all the IAM keys been rotated in last 90 days?	Yes / No
Are you leveraging EC2 Roles instead of IAM user for access permissions wherever possible?	Yes / No
Do all SSL certificate follow SHA2 encryption?	Yes / No
Is bash terminal on all servers ShellShock proof?	Yes
Enumerating buckets for critical info.	Yes / No
List all subdomains. Is there any critical domain, publicly accessible?	Yes / No


Cloud Security
&
Best Practice in AWS
Resources
I am ..
Audit Script
Quick Overview
CRUX..
BREACHES
AWS Best practices
A
AG
amazon web services
Security
BEST PRACTICES

How does it all come together and work?

In-house Solutions:

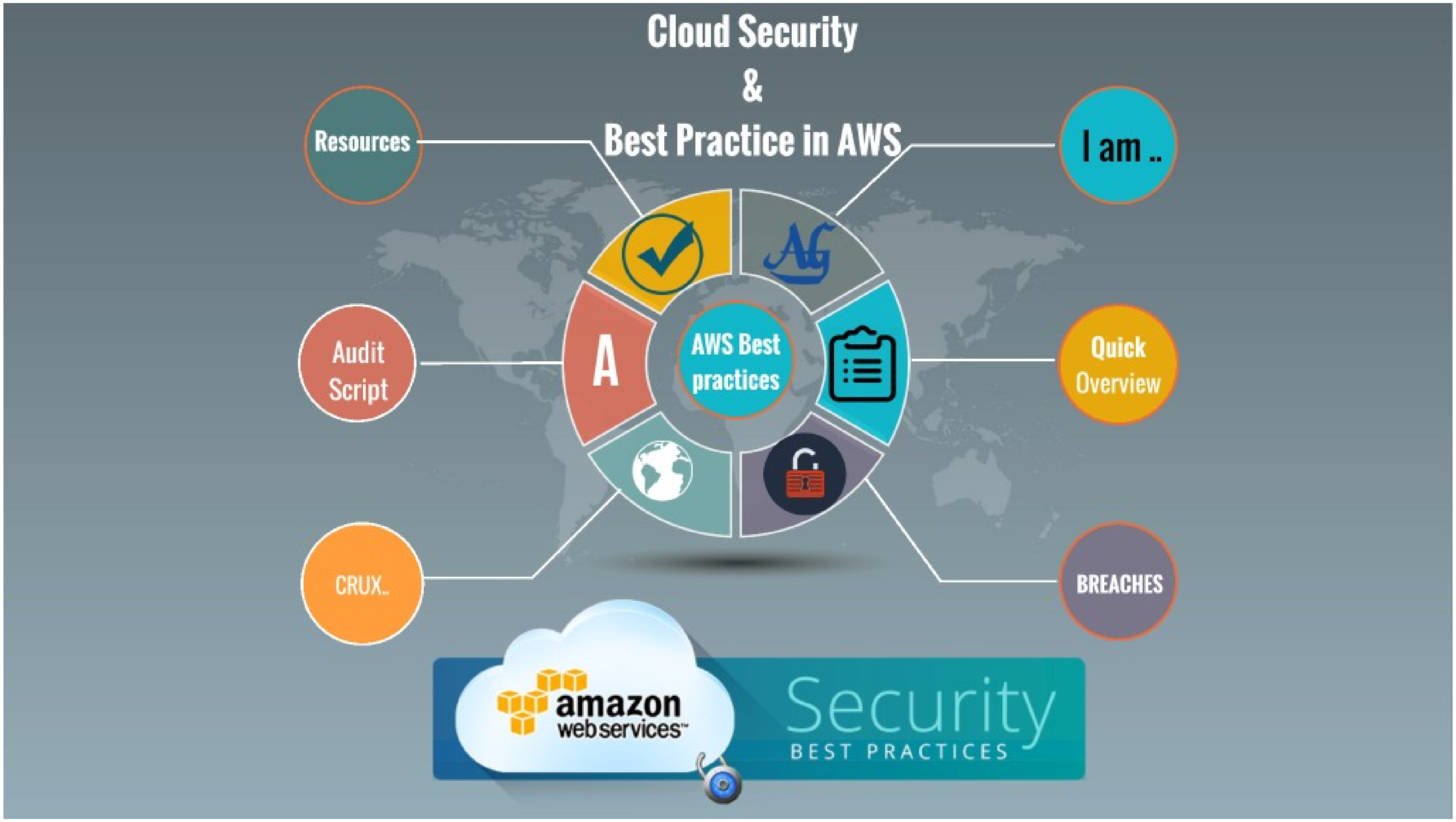
In-house Solutions:

There is an Audit script that scans the AWS infrastructure for Security loopholes: Description and its need

Center for Internet Security compliance and best practices
Hardened AMIs provided by CIS:
https://aws.amazon.com/marketplace/search/results/ref=dtl_navgno_search_box?page=1&searchTerms=center+for+internet+security

AWS Whitepaper: https://d0.awsstatic.com/whitepapers/Security/AWS_Security_Whitepaper.pdf

Top 10 AWS Security Best practices: http://info.evident.io/top-10-best-practices-for-aws-security-ebook.html?utm_medium=partner&utm_source=ADG


Cloud Security
&
Best Practice in AWS
Resources
I am ..
Audit Script
AWS Best practices
A
Quick Overview
CRUX..
BREACHES
amazon web services
Security
BEST PRACTICES

Audit Script
Prowler

Features :

It covers hardening and security best practices for all AWS regions related to:

- Identity and Access Management (24 checks)
- Logging (8 checks)
- Monitoring (15 checks)
- Networking (5 checks)

With Prowler you can:

- get a colourish or monochrome report or a CSV format report for diff
- run specific checks without having to run the entire report
- check multiple AWS accounts in parallel

Prowler

Tool based on AWS-CLI commands for AWS account security assessment and hardening, following guidelines of the CIS Amazon Web Services Foundations Benchmark 1.1

https://github.com/alfresco/prowler

Cloud Security
&
Best Practice in AWS
Resources
I am ..
Audit Script
AWS Best practices
Quick Overview
CRUX..
BREACHES
amazon web services
Security
BEST PRACTICES

What to take home?

got questions

@aankitgiri

Cloud Security
&
Best Practice in AWS
Resources
I am ..
Audit Script
AWS Best practices
Quick Overview
CRUX..
BREACHES
amazon web services
Security
BEST PRACTICES