Attacking Storage Services:
Lynchpin of Cloud Services

Anant Shrivastava

Technical Director, NotSoSecure Global Services

https://redteamvillage.org/

RED TEAM VILLAGE @
HITB+CyberWeek UAE

Virtual Edition, 18-19 November 2020

Agenda

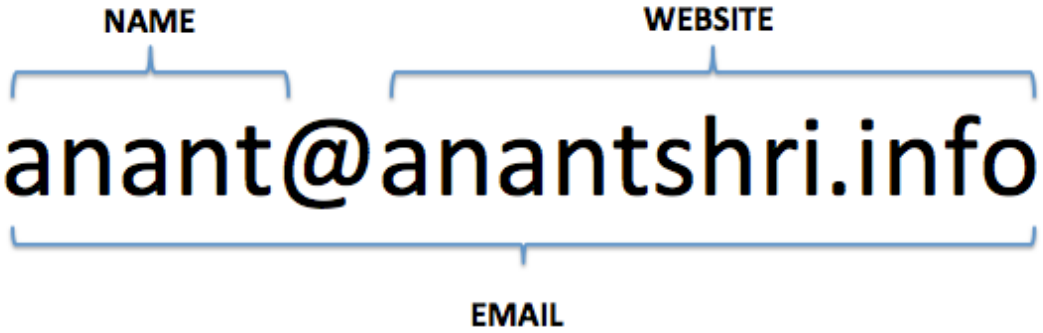
How I am going to bore you for next 30 minutes

- Quick Overview of Cloud Storage (so we all know what I am talking about)
- Examples of real world attack scenario (coz I don't want a demo fail)
- Attack methodology that we can follow (rehash of all things cloud storage attack)
- How to prevent (coz giving out gyaan is what this talk is all about)
- Question and Answer (As if I have not bored you enough)

About Myself: Anant Shrivastava

a.k.a Why should you listen to me

- Director NotSoSecure Global Services
- Sysadmin / Development / Security : all shades of IT
- Project Owner: HackingArchivesofIndia, AndroidTamer, CodeVigilant
- Contributor : null, G4H and many more
- @anantshri on social platforms



Cloud Storage

What are we going to talk about

- Storage service examples are:
  - AWS S3 buckets
  - Azure Storage
  - GCP Storage
  - Digital Ocean Spaces
  - Sharepoint
  - One Drive
  - Dropbox
  - Google Drive Storage
  - Code commit
  - Code repositories: github, bitbucket

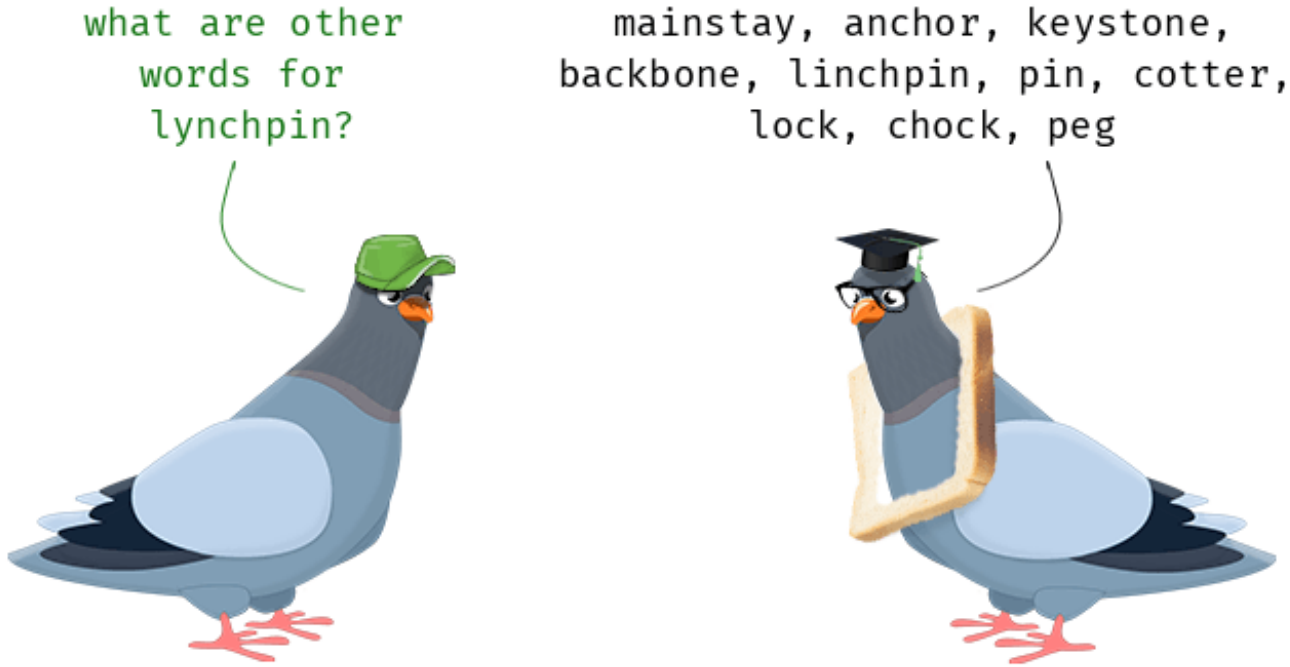
Generally used for storage of objects:

- Files
- Documents
- Source Code
- Transient objects
- Keys
- Secrets

Why Cloud Storage

Why target the storage

- Cloud Storage is a Lynchpin for cloud services



Cloud Storage: Why Attack

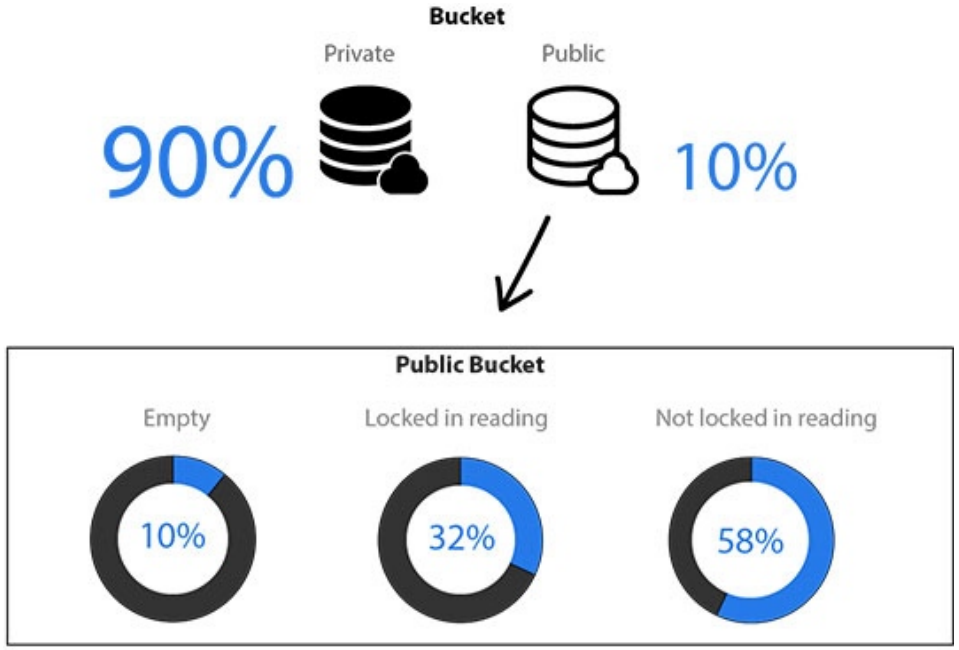
Why target the storage

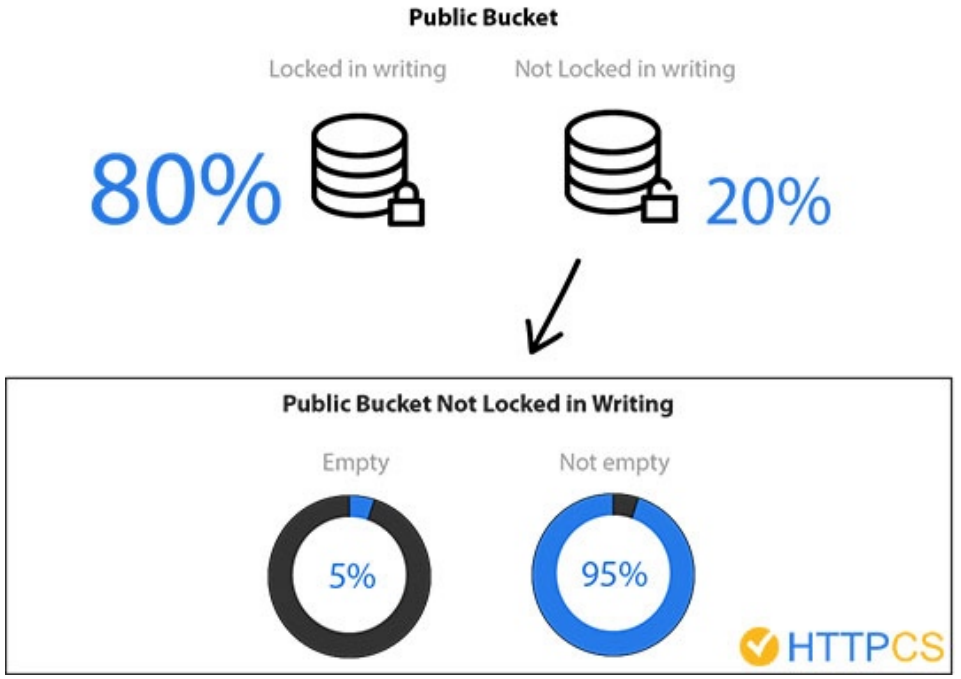
- Consider it like an external disk keeping data in it
- A big part of cloud is able to operate as the data is decoupled from it.
- Such services rely heavily on the Cloud Storage to provide the data backend.
- Besides the data almost all cloud offerings need storage services for:
  - Platform as a Service: Source code for Application
  - Function as a Service: Source code for Function
  - Secret Storage: unbelievable number of people assume storage is good to store passwords or keys or api secrets or confidential data
  - to name a few....

bleepingcomputer.com/news/security/7-percent-of-all-amazon-s3-servers-are-exposed-expl...

◊ Top defense contractor Booz Allen Hamilton leaks 60,000 files, including employee security credentials and passwords to a US government system.
◊ Verizon partner leaks personal records of over 14 million Verizon customers, including names, addresses, account details, and for some victims — account PINs.
◊ An AWS S3 server leaked the personal details of WWE fans who registered on the company's sites. 3,065,805 users were exposed.
◊ Another AWS S3 bucket leaked the personal details of over 198 million American voters. The database contained information from three data mining companies known to be associated with the Republican Party.
◊ Another S3 database left exposed only leaked the personal details of job applications that had Top Secret government clearance.
◊ Dow Jones, the parent company of the Wall Street Journal, leaked the personal details of 2.2 million customers.
◊ Omaha-based voting machine firm Election Systems & Software (ES&S) left a database exposed online that contained the personal records of 1.8 million Chicago voters.
◊ Security researchers discovered a Verizon AWS S3 bucket containing over 100 MB of data about the company's internal system named Distributed Vision Services (DVS), used for billing operations.
◊ An auto-tracking company leaked over a half of a million records with logins/passwords, emails, VIN (vehicle identification number), IMEI numbers of GPS devices and other data that is collected on their devices, customers and auto dealerships.

Writable Public Storages





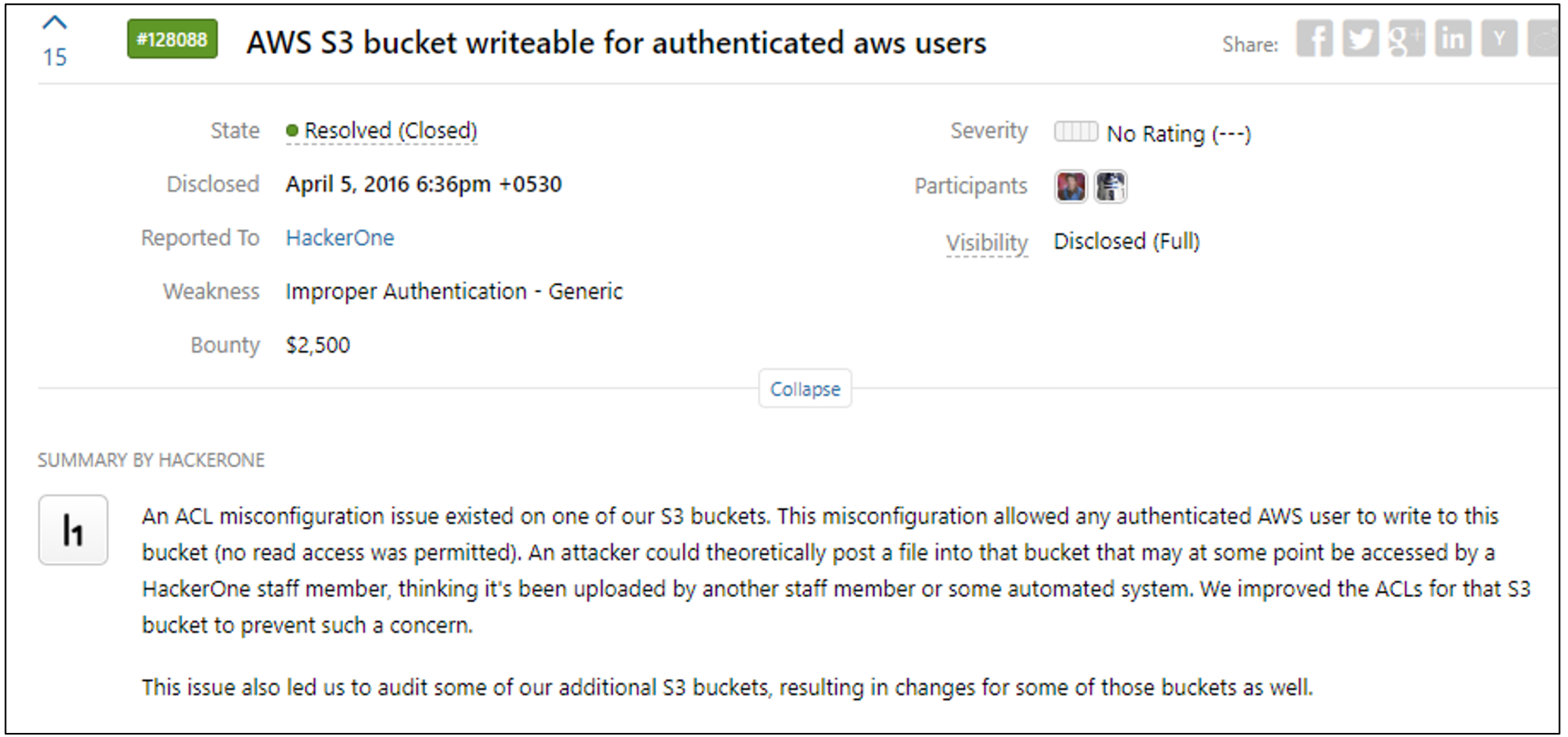
https://www.bleepingcomputer.com/news/security/2-percent-of-amazon-s3-public-buckets-arent-write-protected-exposed-to-ransom-attacks/
https://blog.httpcs.com/etude-dimpact-configuration-aws-buckets-amazon-s3/


SO WHAT'S
THE BIG PROBLEM?!

Case Studies
(best way to show impact)

Case Study 1: Authenticated User Access

- Looks can be deceiving



15

#128088 AWS S3 bucket writeable for authenticated aws users

Share:

State ● Resolved (Closed)

Disclosed April 5, 2016 6:36pm +0530

Reported To HackerOne

Weakness Improper Authentication - Generic

Bounty $2,500

Severity No Rating (---)

Participants

Visibility Disclosed (Full)

Collapse

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.

https://hackerone.com/reports/128088

Case Study 2: Rocket.chat Installer

What's in the name

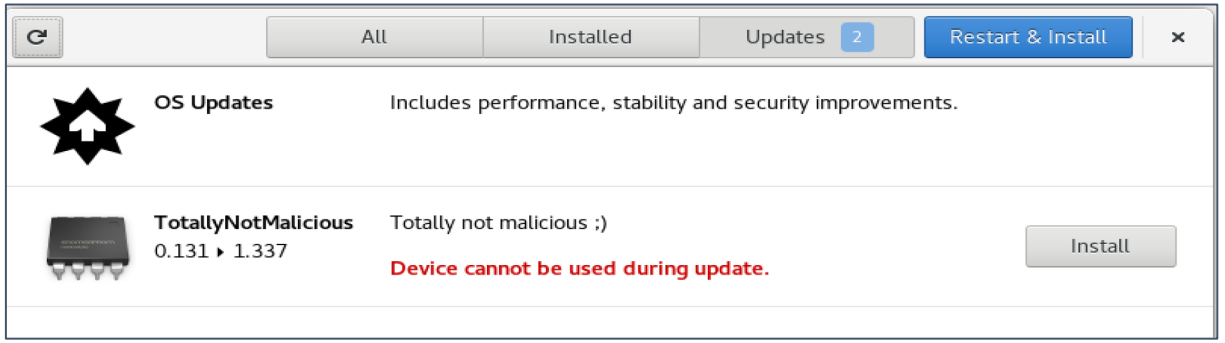
- Unclaimed S3 bucket referenced inside Rocket.chat installer
- Reference https://hackerone.com/reports/399166

```
~ λ ROOTPATH=/var/www/rocket.chat
~ λ cd $ROOTPATH
/var/www/rocket.chat λ tar zxf rocket.chat.tgz  &&  rm rocket.chat.tgz
rm: remove regular file 'rocket.chat.tgz'? y
/var/www/rocket.chat λ cd $ROOTPATH/bundle/programs/server
/var/www/rocket.chat/bundle/programs/server λ npm install
npm notice created a lockfile as package-lock.json. You should commit this file.
up to date in 1.179s
found 0 vulnerabilities

/var/www/rocket.chat/bundle/programs/server λ npm test

> frog-backdoor@1.0.0 test /var/www/rocket.chat/bundle/programs/server
> node backdoor.js

You have been hacked by the frog army.
We do not frogive.
We do not froget.
```

Case Study 3: Fwupd CVE-2020-10759

Updates can be tricky

- Unclaimed S3 Bucket in LVFS (Linux Vendor Firmware Service)
- A PGP Signature bypass in fwupd binary
- In **26 days** : **2.5 million** update requests, **500,000** unique IP



Reference

- https://github.com/justinsteven/advisories/blob/master/2020_fwupd_dangling_s3_bucket_and_CVE-2020-10759_signature_verification_bypass.md
- https://github.com/fwupd/fwupd/blob/1.3.9/src/fu-keyring-gpg.c#L255-L316


YEAH, BUT...
WHAT'S IN IT FOR ME?

Pizza as a Service 2.0

http://www.paulkerrison.co.uk

Tradition On-Premises (legacy)	Infrastructure as a Service (IaaS)	Containers as a Service (CaaS)	Platform as a Service (PaaS)	Function as a Service (FaaS)	Software as a Service (SaaS)
Conversation	Conversation	Conversation	Conversation	Conversation	Conversation
Friends	Friends	Friends	Friends	Friends	Friends
Beer	Beer	Beer	Beer	Beer	Beer
Pizza	Pizza	Pizza	Pizza	Pizza	Pizza
Fire	Fire	Fire	Fire	Fire	Fire
Oven	Oven	Oven	Oven	Oven	Oven
Electric / Gas	Electric / Gas	Electric / Gas	Electric / Gas	Electric / Gas	Electric / Gas
Homemade	Communal Kitchen	Bring Your Own	Takeaway	Restaurant	Party

- Configuration
- Functions
- Scaling...
- Runtime
- OS
- Virtualisation
- Hardware

You Manage — Vendor Manages

Attacking Cloud Storage : Methodology

a.k.a Why more than half of you are here

- In all the case studies storage played an important part
- So now we have established that storage attacks can be catastrophic
- Methodology we will follow

1. Enumerate and Gather information
2. Identify vulnerabilities
3. Exploit vulnerabilities
4. Post exploitation and pivoting
5. Circle back to step 1

Attack: Enumeration

Let's hunt in the dark

- Storage names are unique in the whole of service provider
- A lot of the are predictable for an organization
- Storages are hosted solutions hence port scanning is out of door
- Enumerate / bruteforce various possible bucket name
- No point listing all tools of trade as they keep on changing

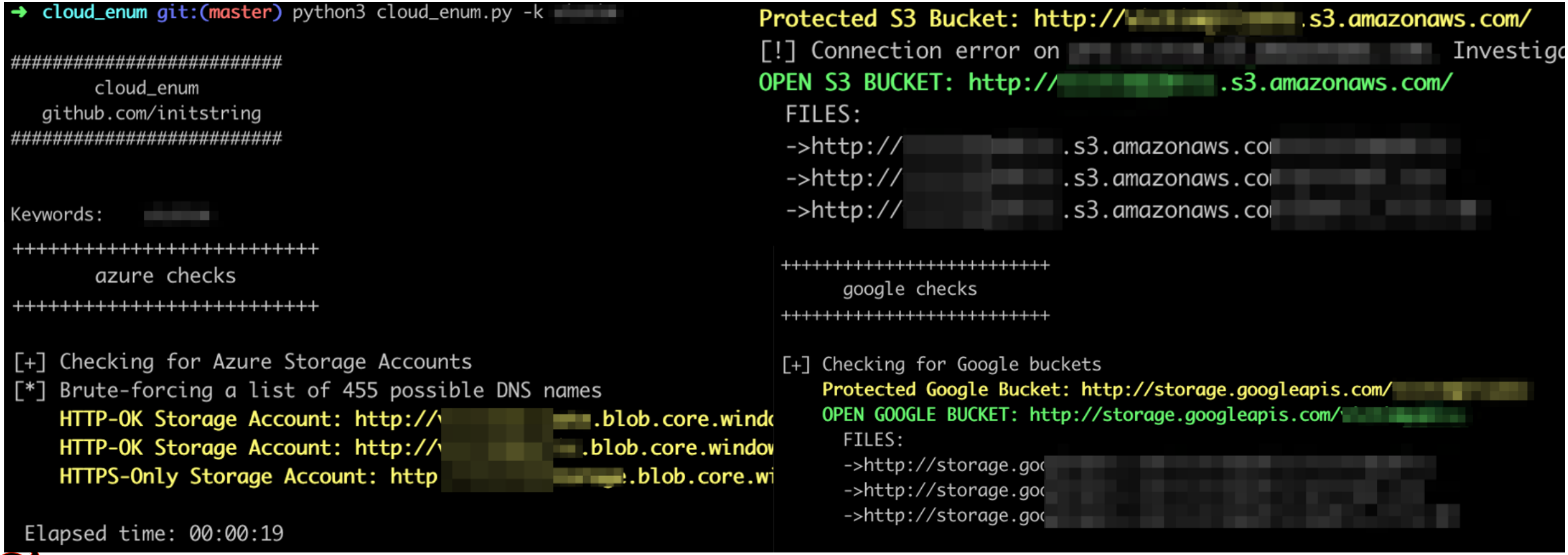
Folks @ Redhuntlabs are maintaining this list:

https://github.com/redhuntlabs/Awesome-Asset-Discovery#cloud-infrastructure-discovery

https://github.com/redhuntlabs/Awesome-Asset-Discovery#domain--subdomain-discovery

Cloud Bucket Enumeration Tools: cloud_enum

- Cloud_enum: https://github.com/initstring/cloud_enum



AWS Storage buckets

- Access AWS buckets
  - `https://s3.amazonaws.com/bucket_name`
  - `https://<bucketname>.s3.amazonaws.com`
- Bucket Enumeration possible via difference in error messages

  `https://s3.amazonaws.com/bucket_name/`
- For REST style URL we now need region tagged

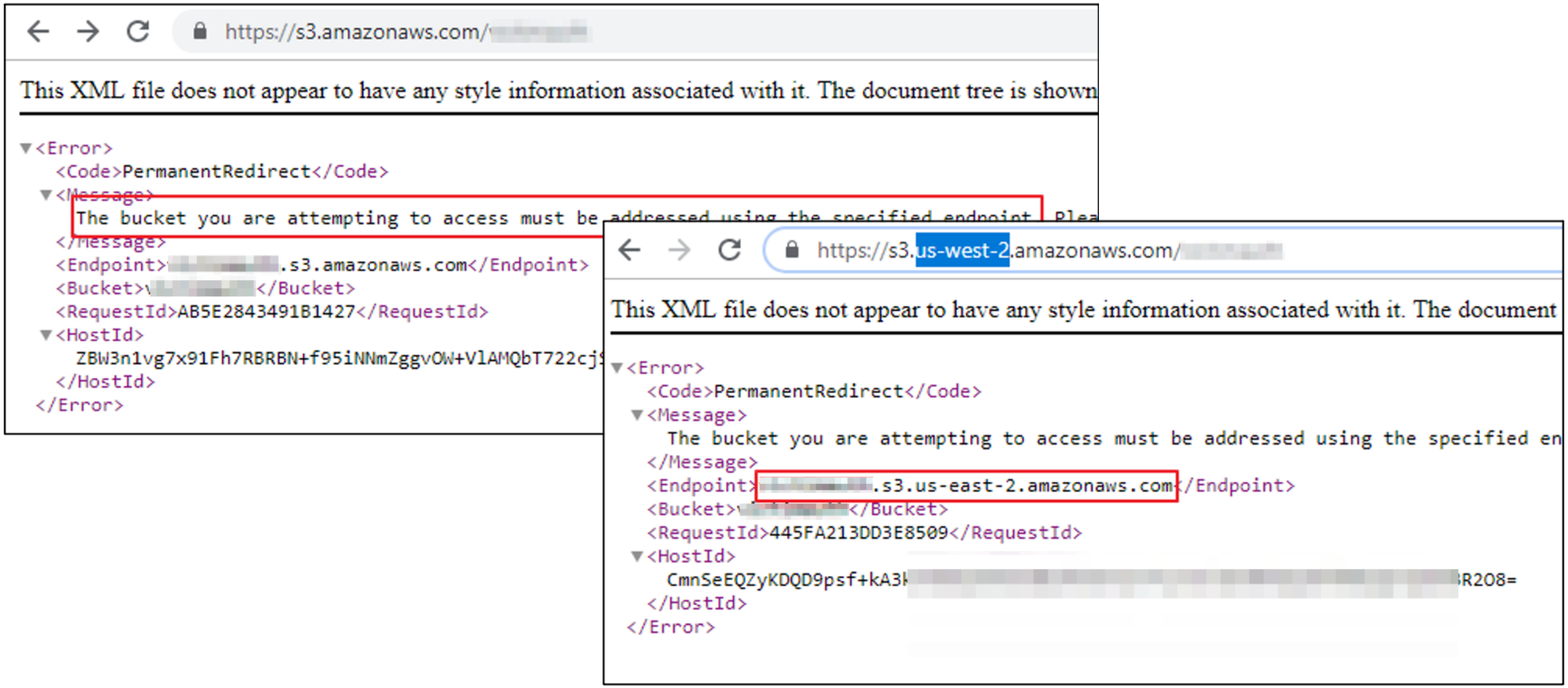
  `https://s3.<region>.amazonaws.com/<bucket_name>/`

Identifying region of Bucket

- Request to any random region url will reveal correct URL

  `https://s3.<anyregion>.amazonaws.com/bucket_name`

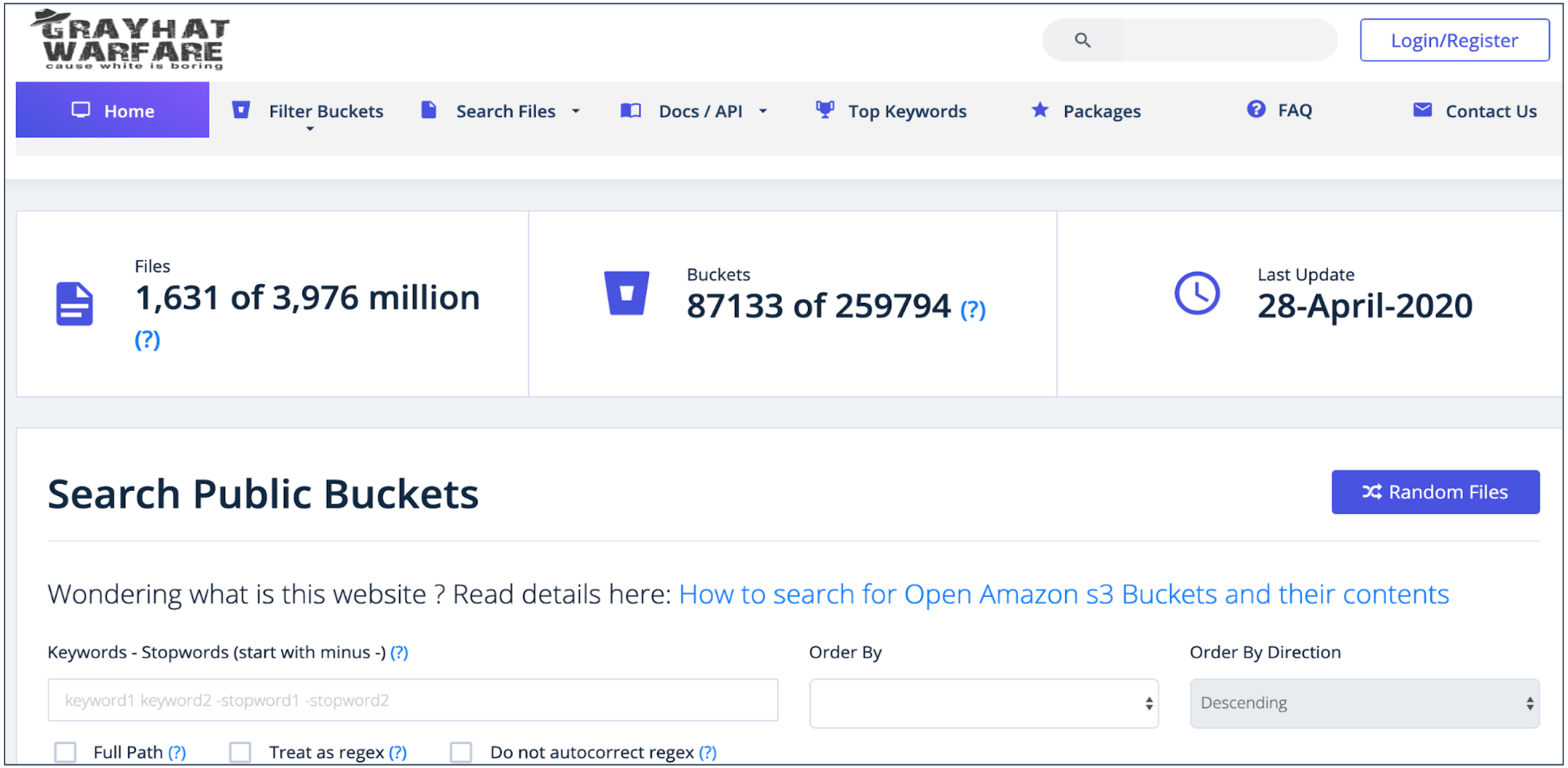
AWS S3 Buckets Enumeration



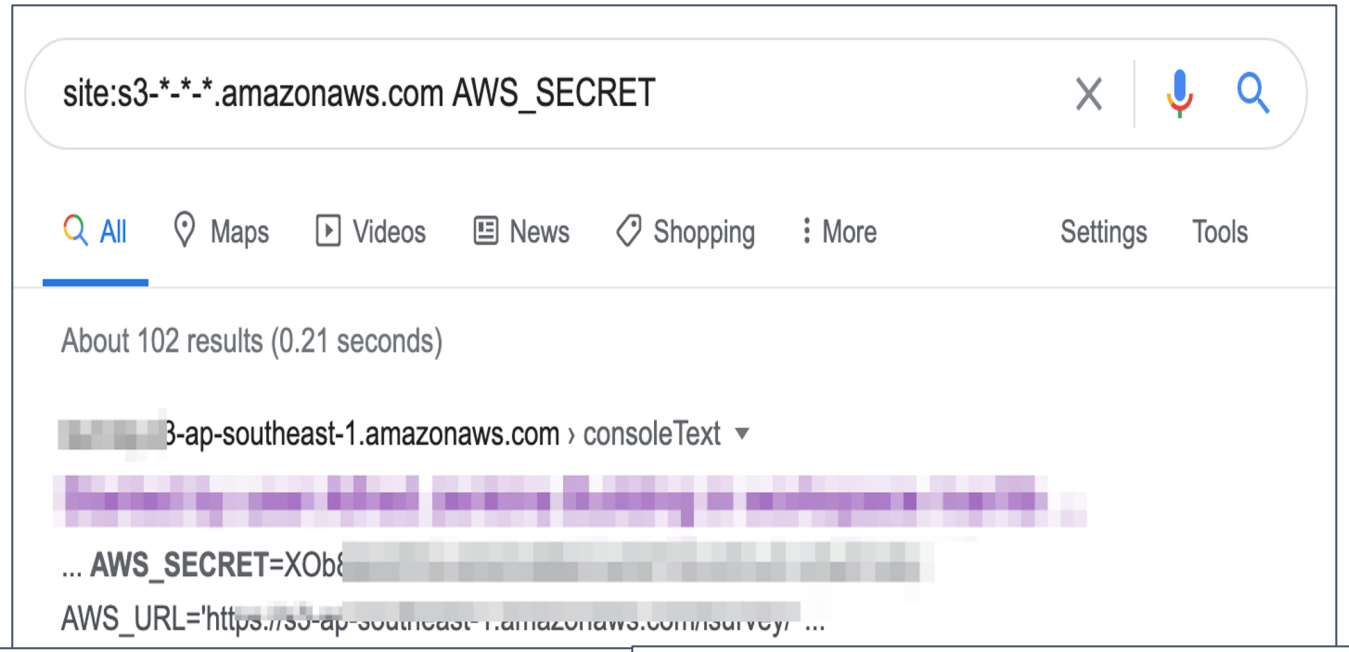
Cloud Bucket URL Scraper

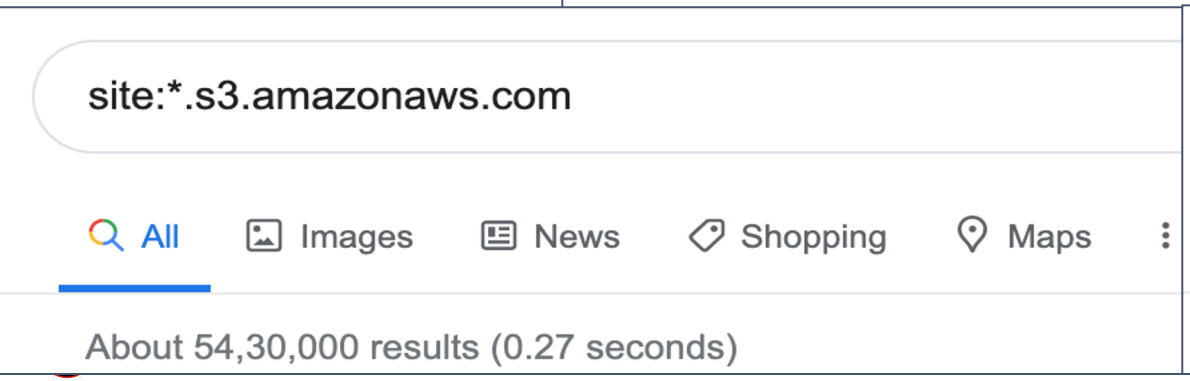
- Cloud Scraper
- Extracts out cloud URLs from HTML source of the website
- Project: https://github.com/jordanpotti/CloudScraper

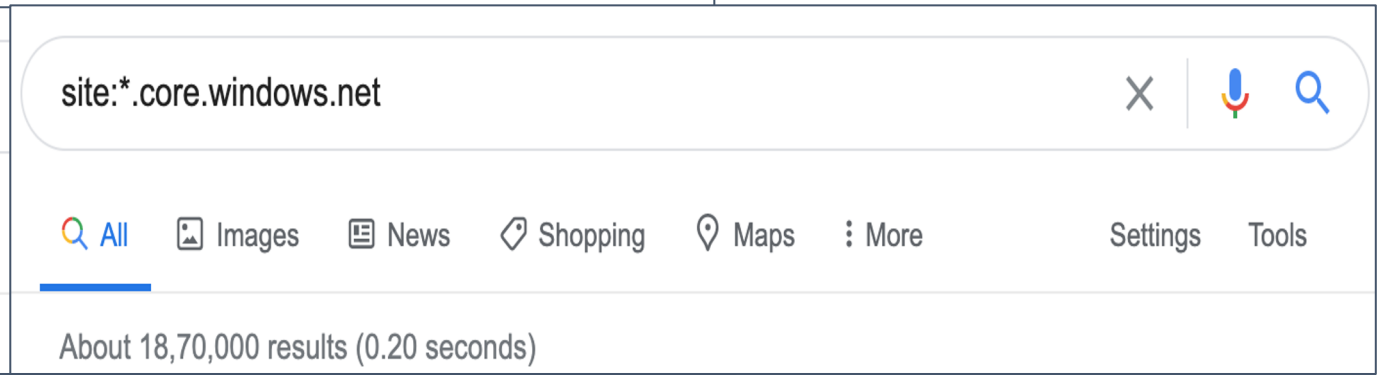
AWS Cloud Bucket Search Engine



Google Dork in Action







Attack: Identification and Exploitation

Primary Focus on permissions:

- Anonymous access granted on bucket
- Misconfigured write access for a resource
- Restricted to auth user (any authenticated user)
- Lax IAM Rules/Policies giving access to data

Azure SAS URL's

- Azure allows creation of URLs with specific access to storage accounts
- These URL's are pretty popular amongst Developers

Example URL

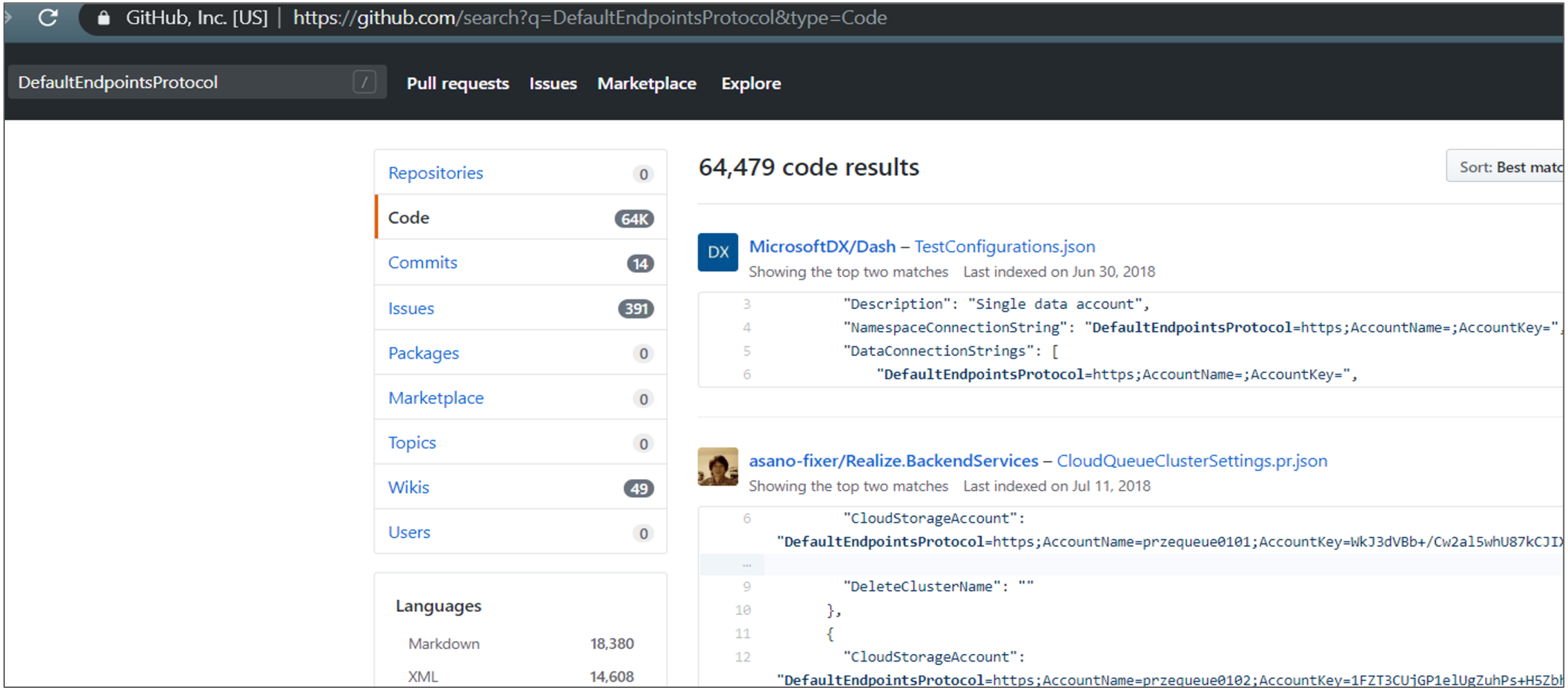
```
https://<accountname>.<service>.core.windows.net/?sv=2018-03-
28&ss=bfqt&srt=sco&sp=rwdlacup&se=2019-09-30T17:13:23Z&st=2019-
09-
30T09:13:23Z&sip=88.208.222.83&spr=https&sig=LCoN4d%2B%2BZSzPtPO7
1fMS34k%2FhLf2Wjen9pzhlAGFfPU%3D
```

Storage Attacks: Azure

Parameter	Description
sv	**Optional**. Specifies the storage service version
ss	**Required**. Specifies the services accessible , Possible values include: Blob (b), Queue (q), Table (t), File (f)
srt	**Required**. Specifies the signed resource types that are accessible with the account SAS. - Service (s): Access to service-level APIs - Container (c): Access to container-level APIs - Object (o): Access to object-level APIs for blobs, queue messages, table entities, and files
sp	**Required**. Permissions for the account - Read (r): Permits read operations - Write (w): Permits write operations - Delete (d): Valid for Container & Object types, except for queue messages. - List (l): Valid for Service and Container resource types only. - Add (a): Valid only for: queue messages, table entities, & append blobs. - Create (c): Valid for the following Object resource types only: blobs and files. Users can create new blobs or files, but may not overwrite existing blobs or files. - Update (u): Valid for the following Object resource types only: queue messages and table entities. - Process (p): Valid for the following Object resource type only: queue messages.
se	**Required**. Expiry Date.
st	**Optional**. Validity Start Date. If omitted, it is assumed to be the time when the storage service receives the request.
sip	**Optional**. IP address or a range of IP addresses allowed
spr	**Optional**. Permitted protocol. Possible values are HTTP (https, http) or HTTPS only (https).
sig	**Required**. The signature part of the URI is used to authorize the request made with the shared access signature.

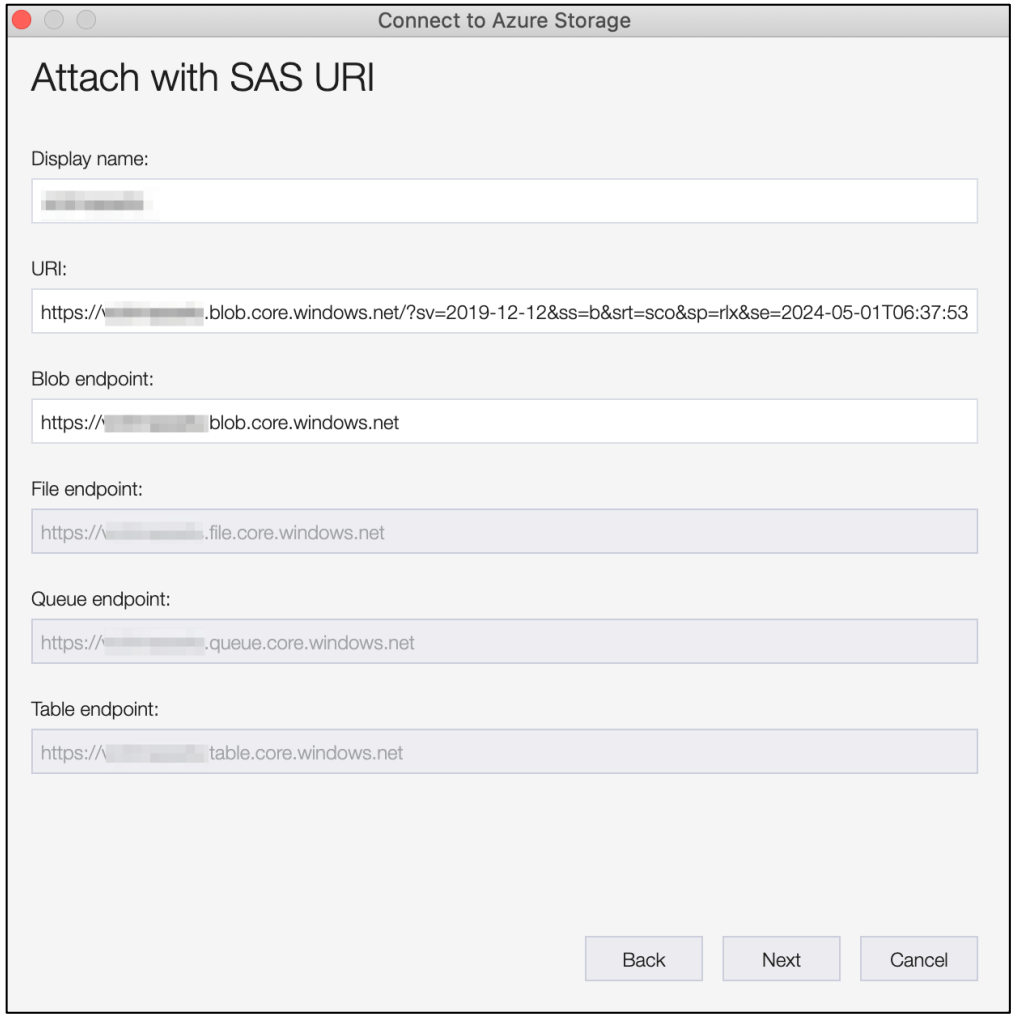
https://notsosecure.com/identifying-exploiting-leaked-azure-storage-keys/

Leaked Storage Account Keys



https://github.com/search?q=DefaultEndpointsProtocol&type=Code

Connecting to Azure Storage



Azure Storage

- Azure storage can be accessed by

  `https://<storagename>.blob.core.windows.net/<container>`
- Container Content can be listed at

  `https://<storagename>.blob.core.windows.net/<container>?restype=container&comp=list`
- Container content can be directly read via web url

  `https://<storagename>.blob.core.windows.net/<container>/<file>`

Case Study: Azure Storage

Careful with that URL

Starting point: Overly Privileged Azure Storage SAS URL is exposed

Exploitation Process:

1. Obtain an Azure Storage SAS URL
2. Load the URL in Azure Storage explorer or similar
3. Identify various assets available in the storage
4. Access the source code of the Azure function
5. Plant a backdoor, next invocation gets the backdoor running
6. Hide the backdoor

Ref: https://www.notsosecure.com/identifying-exploiting-leaked-azure-storage-keys/

Attack: Post Exploitation

You got data but what next

- Once access to storage accounts is obtained it opens various path
- Example:
  - Files such as office documents can reveal softwares and usernames
  - FaaS / PaaS code can lead to RCE in application allowing post exploitation and pivot
  - Sensitive data in buckets can be worth many times over the company valuation
  - Secrets such as passwords or keys can lead to pivot in the network

Credential Harvesting

Hunting for the username

- Office Documents generally Embed Username details
- Downloadable content on website
- Google Dorking
  - Filetype: xlsx / doc / docx
- Extracting Metadata
  - Specialized tools like FOCA (https://github.com/ElevenPaths/FOCA)
  - Homemade automation

```
exiftool *.docx | egrep "Author|Last Modified By|Creator" | sort -u
exiftool *.xlsx | egrep "Author|Last Modified By|Creator" | sort -u
exiftool *.pdf 2>&1 | egrep "Author" | sort -u
```

Case Study: SSRF to EC2 takeover

Don’t leave keys to the kingdom in bucket

- Starting point: SSRF on a Web Application
- Obtained Metadata details (account id, region, security-credentials)
- Using credentials enumerate all s3 buckets
- One s3 bucket contained pem files for all ec2 boxes
- Enumerate instances to identify higher power roles
- Obtained access to those instances via pem files
- Backdooring the AWS account by creating new id with iam:* capabilities
- Reference: https://www.threatstack.com/cloud-attack (not exact but similar)

Case Study: PaaS: Elastic Beanstalk

From Beans to RCE

Starting point: SSRF on an application hosted in AWS Elastic Beanstalk

Exploitation Process:

1. Obtained Metadata details (account id, region, security-credentials)
2. No direct access to read S3 bucket list
3. Enumerated bucket name using the account id and region
4. Access source code of the application via AWS S3 CLI
5. CI/CD in place hence a backdoor pushed to S3 bucket will result in shell deployed on the official website
6. Summitroute did extra research & identified more such naming patterns

Ref: https://www.notsosecure.com/exploiting-ssrf-in-aws-elastic-beanstalk/
https://summitroute.com/blog/2019/02/10/aws_resource_naming_patterns/
https://gist.github.com/0xdabbad00/645837c1fcd043876d13a56819188227

Case Study: AWS Cognito Analysis

- AWS Temp Credentials can be obtained if identity pool is known
- Leveraged crowd sourcing via commoncrawl, decompiling android apk
- Collected a total of 2504 identity pool identifiers
- Explored permissions on each pool identifier
  - more than **1 in 5** AWS Cognito configurations **are insecure**
  - **906 S3 buckets** which contained **sensitive** information
  - identified **1572 lambda** functions, exposing at least **78 sensitive env variables**

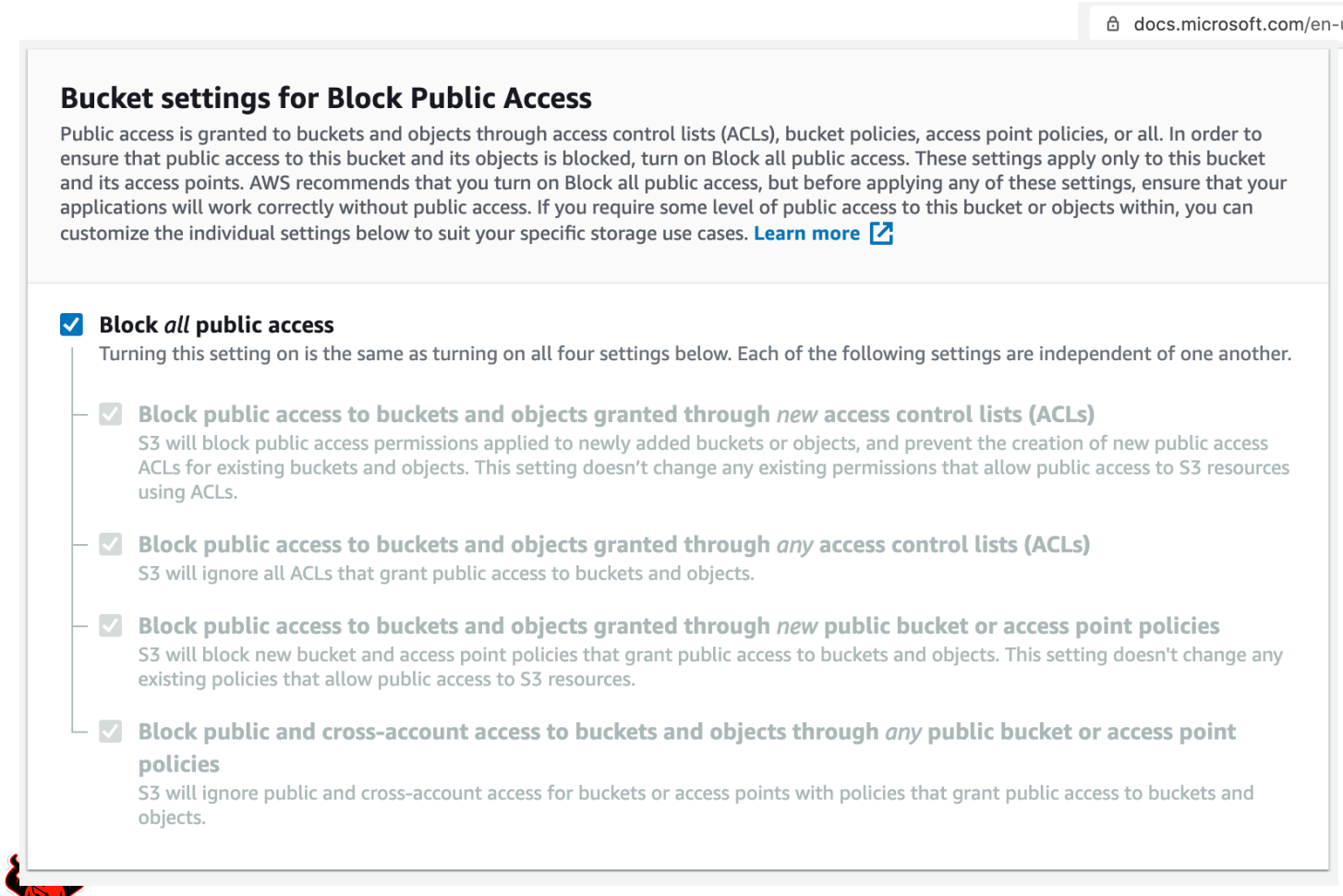
References: https://andresriancho.com/internet-scale-analysis-of-aws-cognito-security/

Defenses

a.k.a things everyone should do but no one cares

- Cloud Vendors
  - Provide warnings on possible problematic scenario's
  - Automation for common problem detection and solution
- Tenant
  - Identity and Access management is key
  - No unauth /public access unless required
  - Authenticated user is most definitely not the setting you want
  - Periodically run scans to identify variations
  - Ensure you are aware of your own setup limitations

Vendor Warnings



docs.microsoft.com/en-us/rest/api/storageservices/create-account-sas

to:

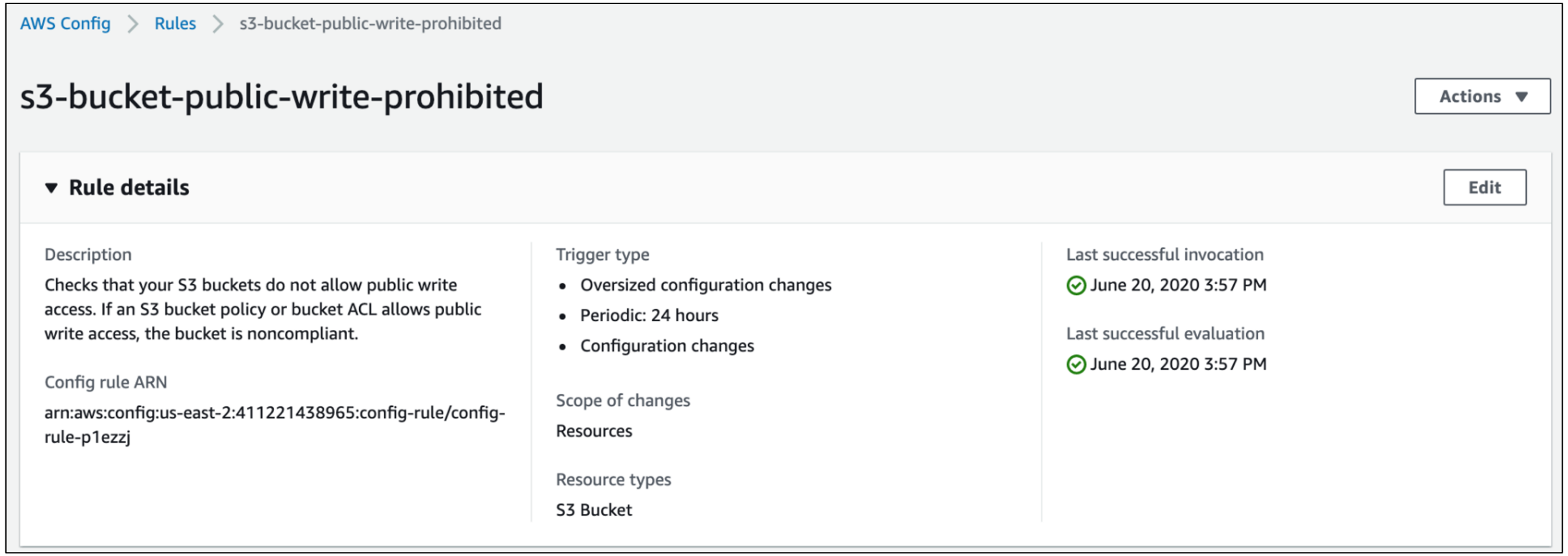
- Delegate access to service-level operations that are not currently available with a service-specific SAS, such as the `Get/Set Service Properties` and `Get Service Stats` operations.
- Delegate access to more than one service in a storage account at a time. For example, you can delegate access to resources in both the Blob and File services with an account SAS.
- Delegate access to write and delete operations for containers, queues, tables, and file shares, which are not available with an object-specific SAS.
- Specify an IP address or range of IP addresses from which to accept requests.
- Specify the HTTP protocol from which to accept requests (either HTTPS or HTTP/HTTPS).

Stored access policies are currently not supported for account SAS.

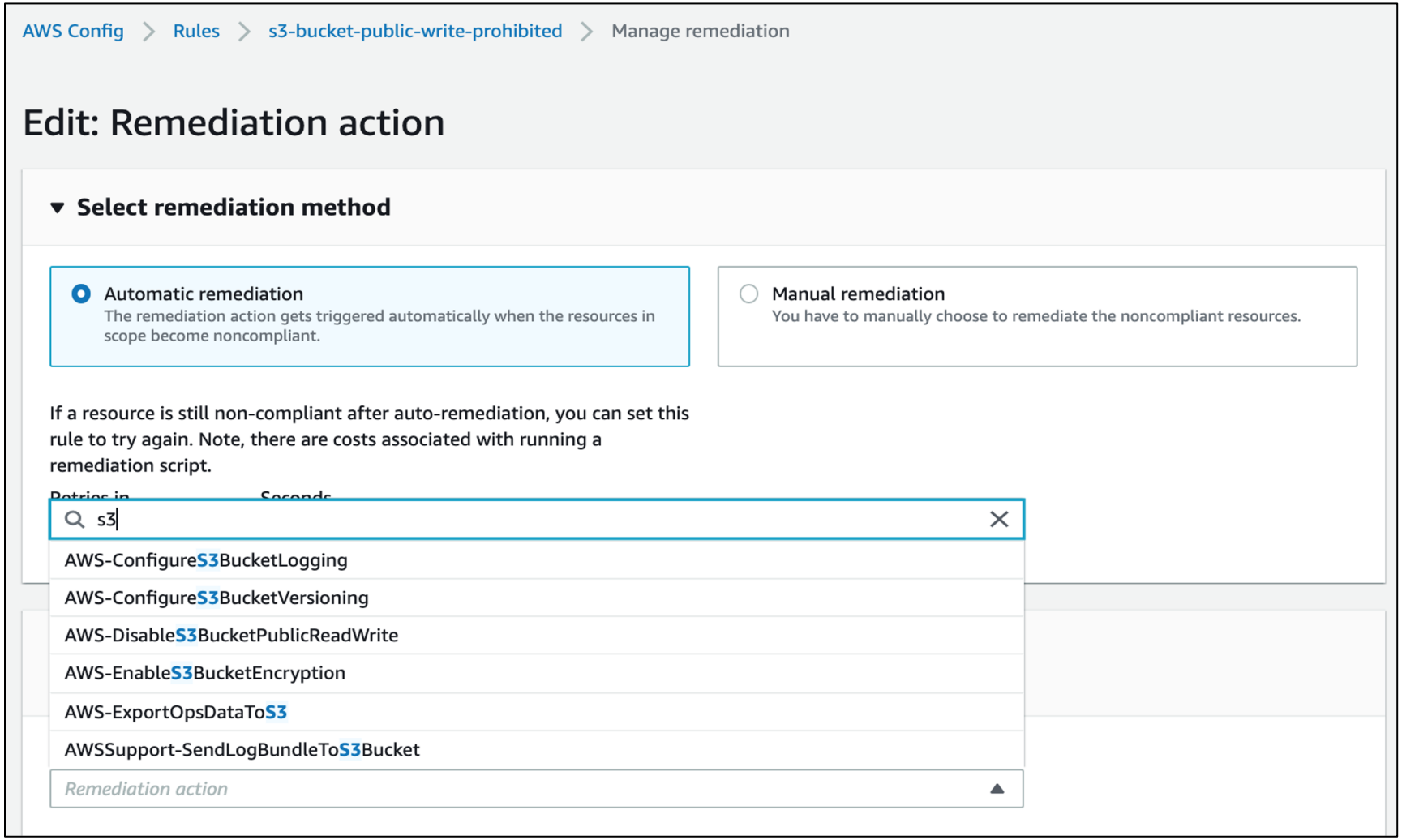
⊗ Caution

Shared access signature are keys that grant permissions to storage resources, and should be protected in the same manner as an account key. It's important to protect a SAS from malicious or unintended use. Use discretion in distributing a SAS, and have a plan in place for revoking a compromised SAS. Operations that use shared access signatures should be performed only over an HTTPS connection, and shared access signature URIs should only be distributed on a secure connection such as HTTPS.

Vendor : AWS Config

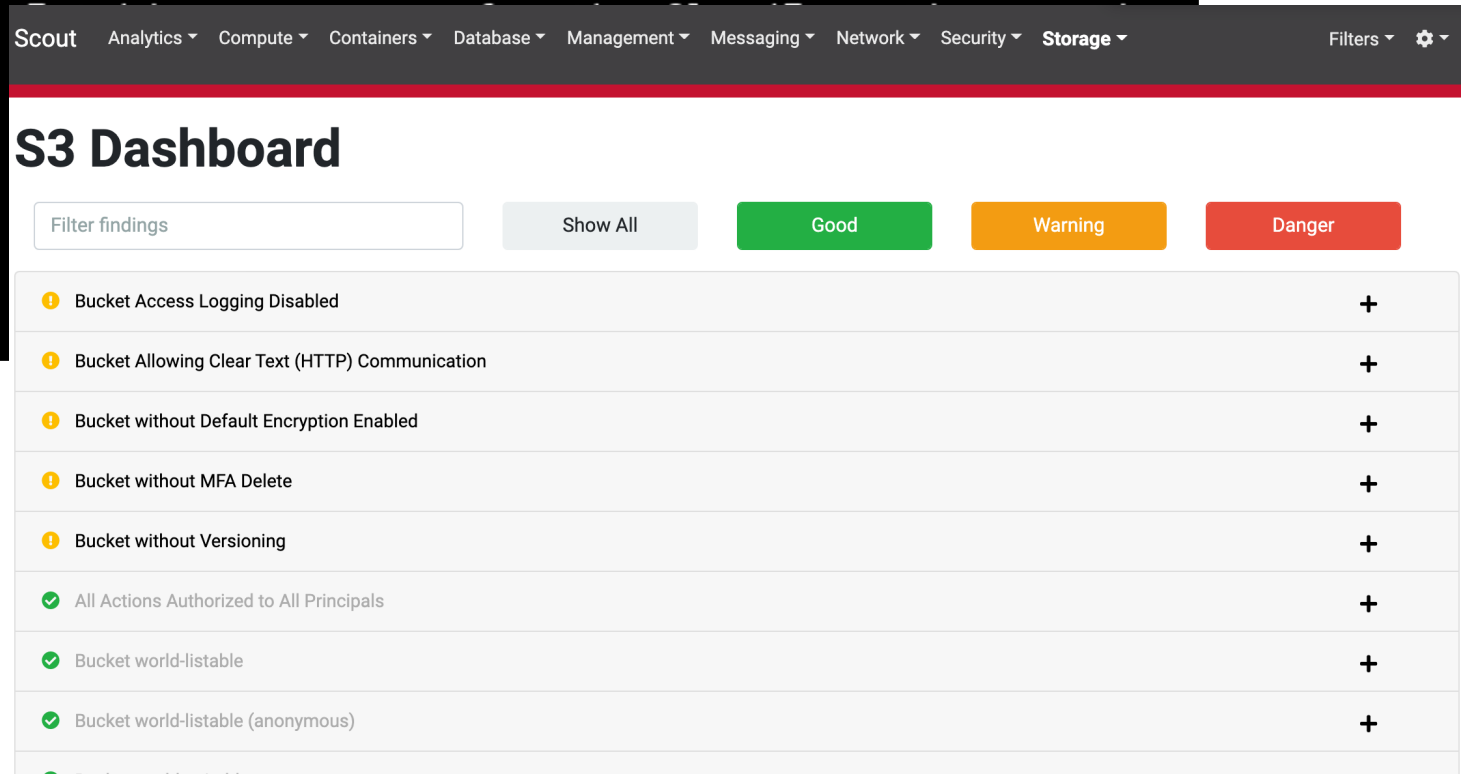


AWS Config: Auto Remediation



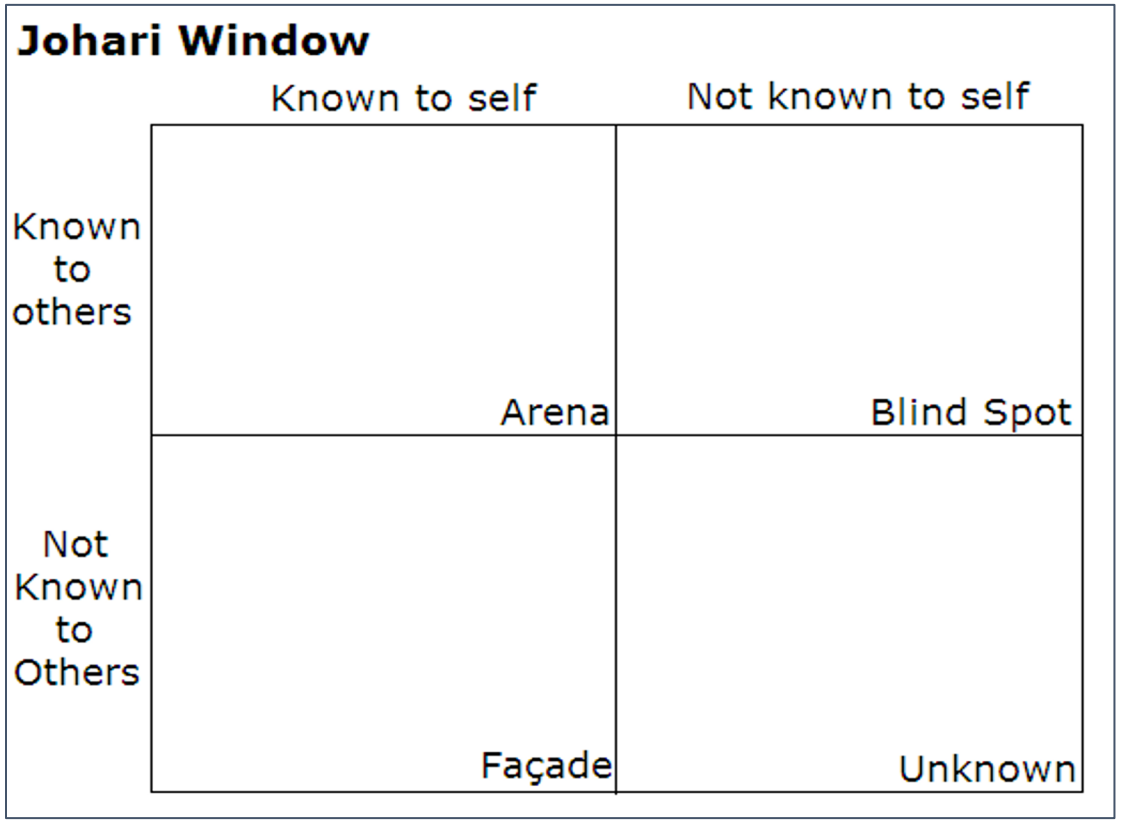
Tenant: Periodic Scan: Scout Suite

```
(venv) root@kali:~/cloud_tools/ScoutSuite# python scout.py aws
2019-05-25 16:27:23 kali scout[2286] INFO Launching Scout
2019-05-25 16:27:23 kali scout[2286] INFO Authenticating to cloud provider
2019-05-25 16:27:33 kali scout[2286] INFO Gathering data from APIs
2019-05-25 16:27:33 kali scout[2286] INFO Fetching resources for the Lambda service
2019-05-25 16:27:34 kali scout[2286] INFO
2019-05-25 16:27:36 kali scout[2286] INFO
2019-05-25 16:27:37 kali scout[2286] INFO
2019-05-25 16:27:39 kali scout[2286] INFO
2019-05-25 16:27:40 kali scout[2286] INFO
2019-05-25 16:27:41 kali scout[2286] INFO
2019-05-25 16:27:43 kali scout[2286] INFO
2019-05-25 16:27:44 kali scout[2286] INFO
```



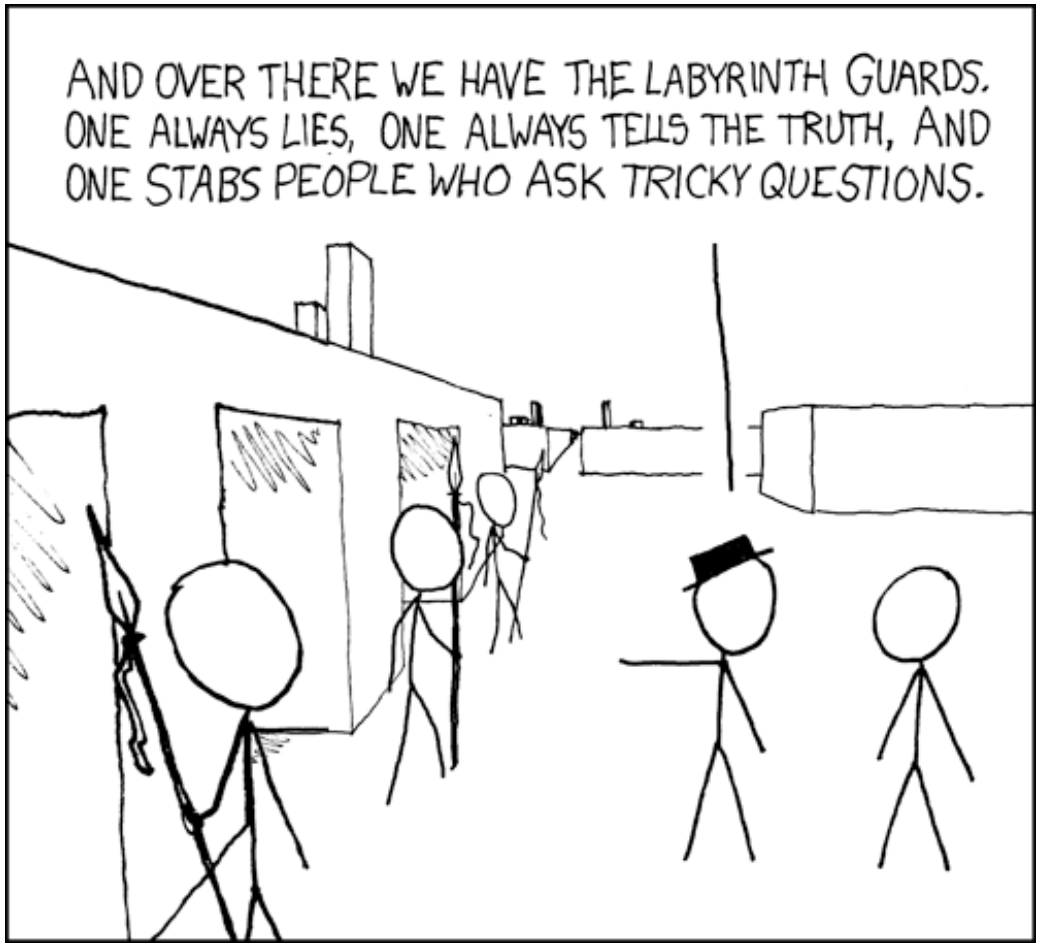
Tenant: Prepare for Disaster

- Be prepared for the inevitable
- Validate your Setup
  - Simulate hacks
  - Observe reactions
  - Finetune reactions
  - Revalidate



Question and Answers

I will now look at the chat and answer questions.



https://xkcd.com/246/

Additional Reference Material

- Jason Haddix's awesome The bug hunter's Methodology Series
- https://github.com/jhaddix/tbhm
- https://cloudsecwiki.com/

Thank You

Anant Shrivastava, anant@anantshri.info

See you at HITB's Discord channel for questions & answers!

https://redteamvillage.org/

RED TEAM VILLAGE @
HITB+CyberWeek UAE
Virtual Edition, 18-19 November 2020