Internal Investigation - write up

Description :

```
We acquired this disk image without Khalid (our intern)
knowing because we are conducting an internal investigation at Meta Airlines.
We suspect someone has connections with a shady organization that steals aviation technologies,
especially flight simulator software, cockpit interface designs, and advanced navigation systems.
Can you figure out if he is involved with them or not?
```

by 0WL

Solution :

we are given a disk image and we notice its extension is ad1 (AccessData Custom Content Image) which can be viewed using FTK imager , Autopsy doesn't support it which make it a bit difficult to navigate through the disk image without the modules of Autopsy and plugins .

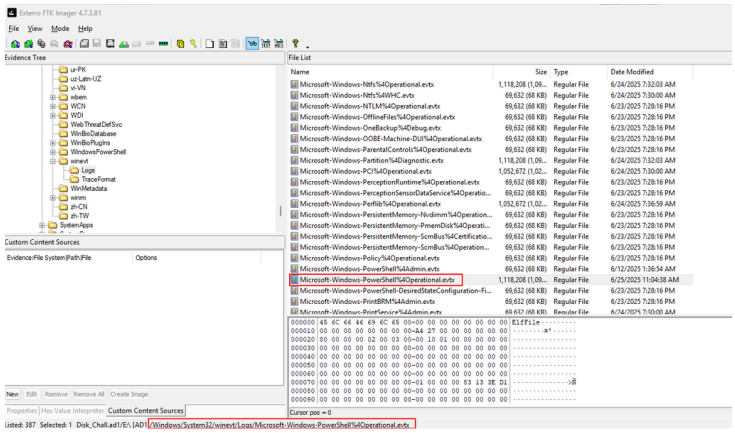
upon loading the image into FTK imager we are overwhelmed by the amount of data and the disk size we cant go through each file in it , we need to have a systematic way and a check list for things that we always need to check in a disk image, for the sake of this challenge will go through the ones that solve it but i will write down everything you usually should check if you don't have any idea where to start .

check list :

- Registry files (Software , System etc..)
- Event log files EVTX
- users directory and their folders like document and desktop etc ...
- Prefetch Files
- PowerShell History
- Executable Files
- Browser History

These are the important things we usually need to look for (The order doesn't matter) , regarding challenge i will go through the Event log files which are located in this path `C:\Windows\System32\winevt\Logs\`

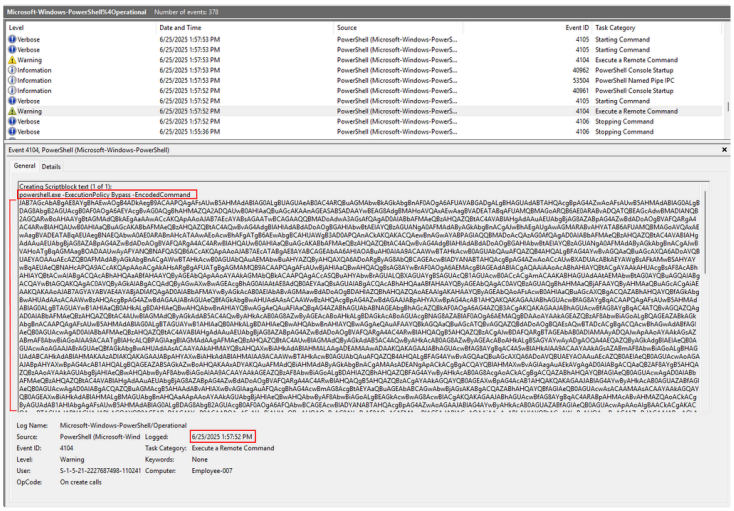
there are a lot of event log files I am interested in the “Microsoft-Windows-PowerShell/Operational.evtx” also “windows powershell.evtx” is useful if available .



Microsoft-Windows-PowerShell/Operational.evtx path

We can export it into our host by right clicking on it then export files.

After exporting we can view it using Event viewer which is a built-in tool in windows , going through the events we can find this suspicious event that was logged at “6/25/2025 1:57:52 PM”



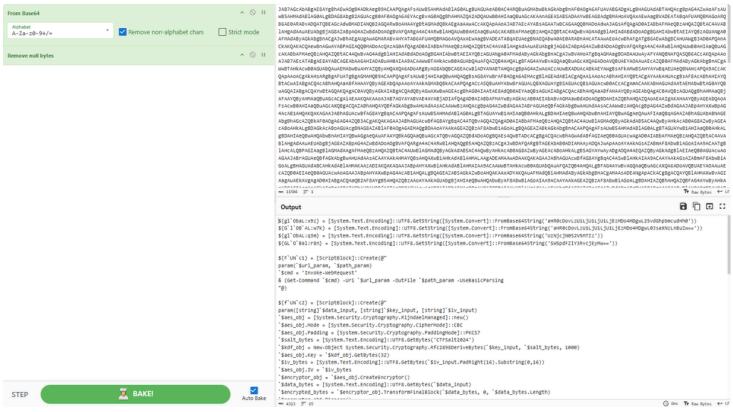
Suspicious Event

We notice there is this command :

```
powershell.exe -ExecutionPolicy Bypass -EncodedCommand <encoded_string>
```

its to execute base64-encoded PowerShell commands

used for Obfuscation and Malicious purposes , we can copy the encoded string into CyberChef



Decoded String

using the recipe shown in the picture above we can view the decoded string which reveal a PowerShell script upon close look into it we notice its Obfuscated you can through it to an AI to understand what it does.

Breakdown of the PowerShell script

The script begins by decoding several base64 strings:

```
aHR0cDovLzU1LjU1LjU1LjEzMDo4MDgwL25vdGhpbmcudHh0  →  http://55.55.55.130:8080/nothing.txt   aHR0cDovLzU1LjU1LjU1LjEzMDo4MDgwL0JsaXNzLnBuZw==  →
http://55.55.55.130:8080/Bliss.png   UzNjcjN0S2V5MTIz                                →  S3cr3tKey123

SW5pdFZlY3RvcjEyMw==                          →  InitVector123
```

there is also functions that does the following :

- `${F UN c1}` – Downloads a file using `Invoke-WebRequest`
- `${f UN c2}` – Encrypts data using AES (CBC mode) with a hardcoded salt (`CTFSalt2024`)
- `${aD s FuNc}` – Writes data to a file or uses `cmd.exe` to echo and execute it

it also downloads files from remote host

```
Invoke-WebRequest http://55.55.55.130:8080/Bliss.png → saved as vacation_photo.png
Invoke-WebRequest http://55.55.55.130:8080/nothing.txt → saved as temp file (then deleted)
```

The vacation_photo.png is saved under the downloads directory .

it then encrypts the content of nothing.txt and embed it in the meta data of vacation_photo.png in the comment field

the content of nothing.txt was Encrypted using AES-256-CBC with:

- Key = `S3cr3tKey123`

- IV = `InitVector123` (padded)
- Salt = `CTFSalt2024`

you can export the vacation_photo.png into your host and view the meta data using exiftool either online or cli :

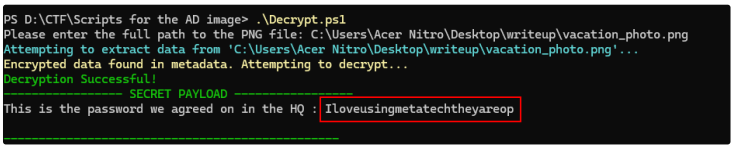
Encrypted data

knowing all of these factors we can easily decrypt using this PowerShell script :

```powershell
$secretKey = "S3cr3tKey123"
$initVector = "InitVector123"

Function Get-ImageMetadataComment {
    param(
        [string]$filePath
    )

    try {
        Add-Type -AssemblyName System.Drawing

        $image = [System.Drawing.Image]::FromFile($filePath)

        $propItem = $image.GetPropertyItem(0x9286)

        $image.Dispose()

        if ($propItem) {
            $commentData = [System.Text.Encoding]::ASCII.GetString($propItem.Value).TrimEnd("`0")
            return $commentData
        } else {
            Write-Warning "No comment metadata (ID 0x9286) found in the image."
            return $null
        }

    } catch {
        Write-Error "Failed to read image metadata. Error: $($_.Exception.Message)"
        return $null
    }
}

Function Decrypt-Data {
    param(
        [string]$encryptedBase64,
        [string]$key,
        [string]$iv
    )

    try {
        $aes = [System.Security.Cryptography.RijndaelManaged]::new()
        $aes.Mode = [System.Security.Cryptography.CipherMode]::CBC
        $aes.Padding = [System.Security.Cryptography.PaddingMode]::PKCS7

        $saltBytes = [System.Text.Encoding]::UTF8.GetBytes('CTFSalt2024')

        $kdf = New-Object System.Security.Cryptography.Rfc2898DeriveBytes($key, $saltBytes, 1000)
        $aes.Key = $kdf.GetBytes(32)

        $ivBytes = [System.Text.Encoding]::UTF8.GetBytes($iv.PadRight(16).Substring(0,16))
        $aes.IV = $ivBytes

        $decryptor = $aes.CreateDecryptor()

        $encryptedBytes = [System.Convert]::FromBase64String($encryptedBase64)
        $decryptedBytes = $decryptor.TransformFinalBlock($encryptedBytes, 0, $encryptedBytes.Length)

        $decryptor.Dispose()
        $aes.Dispose()

        return [System.Text.Encoding]::UTF8.GetString($decryptedBytes)
    } catch {
        Write-Error "Decryption failed. Ensure the key, IV, and salt are correct. Error: $($_.Exception.Message)"
        return $null
    }
}

$imagePath = Read-Host "Please enter the full path to the PNG file"

if (-not (Test-Path $imagePath)) {
    Write-Host "Error: File not found at '$imagePath'" -ForegroundColor Red
    exit
```

```
}

Write-Host "Attempting to extract data from '$imagePath'..." -ForegroundColor Cyan

$encryptedData = Get-ImageMetadataComment -filePath $imagePath

if ($encryptedData) {
    Write-Host "Encrypted data found in metadata. Attempting to decrypt..." -ForegroundColor Yellow

    $decryptedPayload = Decrypt-Data -encryptedBase64 $encryptedData -key $secretKey -iv $initVector

    if ($decryptedPayload) {
        Write-Host "Decryption Successful!" -ForegroundColor Green
        Write-Host "----------------- SECRET PAYLOAD -----------------" -ForegroundColor Green
        Write-Host $decryptedPayload
        Write-Host "--------------------------------------------------" -ForegroundColor Green
    } else {
        Write-Host "Failed to decrypt the payload." -ForegroundColor Red
    }
} else {
    Write-Host "Could not find any hidden data in the image." -ForegroundColor Red
}
```

here are the result of executing it :



Decrypted data

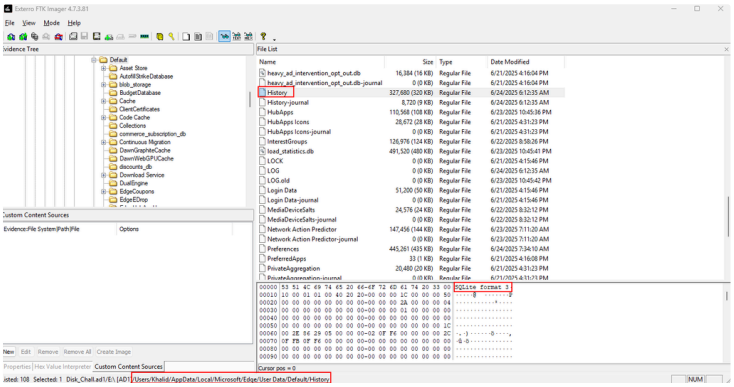
as we can see we got a password “Iloveusingmetatechtheyareop”

hmmm... but where do we use it ??

going back to the check list or by brainstorming where can we put a password ?

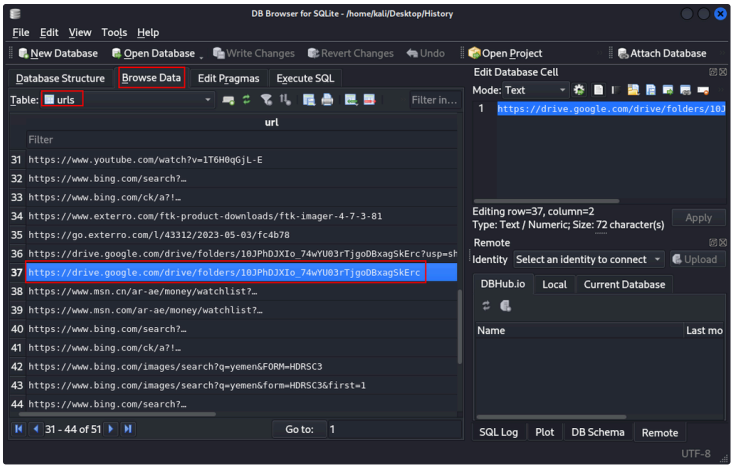
maybe an encrypted db , the internet , zipped file etc ...

lets check the browser history there is no harm on checking it based on the browser the user is using the path can changes but based on the programs that the user have it seems he uses Edge going to this path we can obtain the browser history of Edge `C:\Users\%USERNAME\AppData\Local\Microsoft\Edge\User Data\Default\`



Browser History (1)

we can see that its a DB lets export it and then view it using DB Browser for SQLite or we can always go basics and just use strings



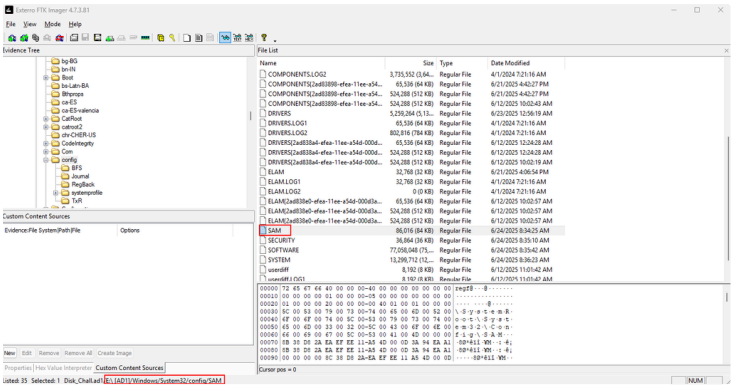
Browser History (2)

we can view the urls that user visited by going to the Browse Data tab then from the Table scroll box we chose urls Table , going through them we notice a drive link upon clicking on it we can find 2 files one is instructions to unzip the other is the zipped file where the plan resides

Instructions to unzip

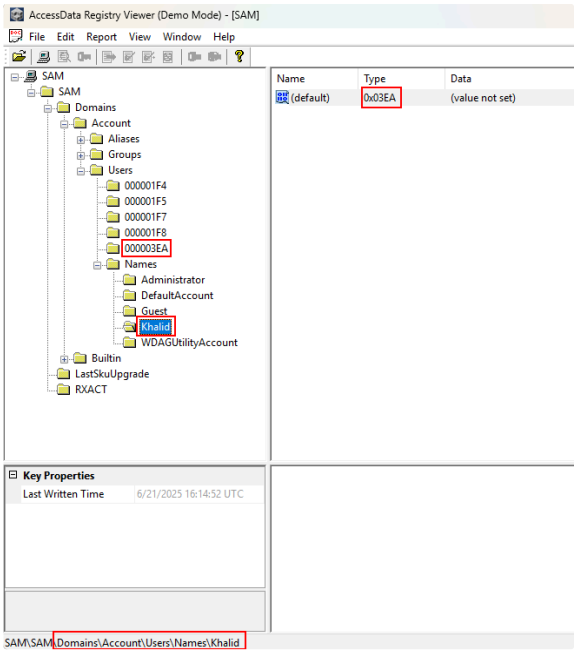
we understand the zipped file is protected using the password we found + the users childhood nickname

doesn't this ring a bell “childhood nickname” its usually a security question that companies uses in case you forgot your password you will need to answer them , so in windows where are the security questions saved ? ; they are saved in the SAM registry file where can be found in this path `C:\Windows\System32\config\`



SAM (1)

we can either export it and then view it using registry explorer or you can scroll down after clicking on it in FTK imager on the bottom panel , lets use registry explorer



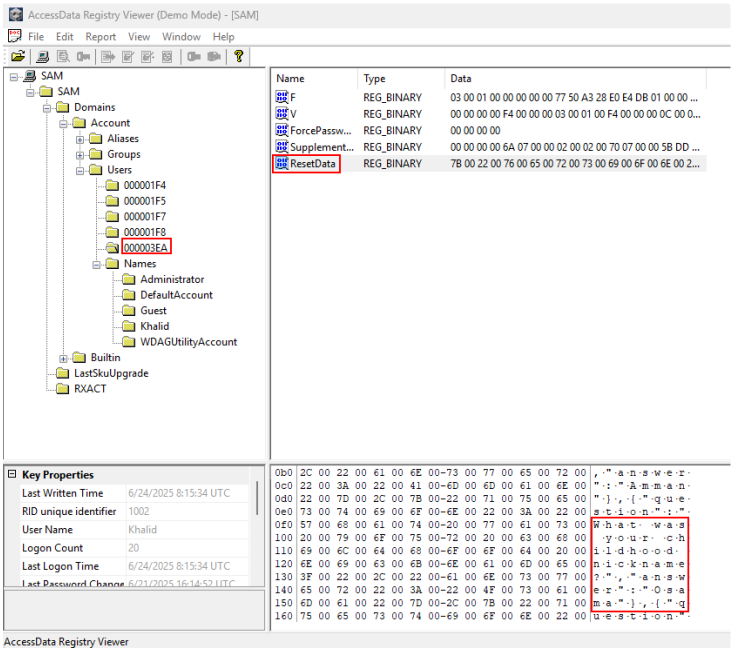
SAM (2)

After opening SAM in registry viewer we can go to this registry path to determine the users RID (Relative Identifier)

`Domains\Account\Users\Names`

we know kahlid RID ends with 3EA now going the registry key of this RID in this path

`Domains\Account\Users\000003EA`



SAM (3)

opening this key we can find 4 values we click on the ResetData which contains the things you will need in case you forgot your password and which to reset it , Scrolling down we can find this 2nd security question out of 3 which says “What was your childhood nickname ?” the answer is “Osama”

So now we got everything we need to unzip the file we found in the drive , based on the instruction provided to unzip we have the password which should be “IloveusingmetatechtheyareopOsama”

It worked !!! 🎇

inside it we can find the PLAN.txt which contain this flag at the end :

`METACTF{m374_41r11n35_73chn0l0g135_1n_y0ur_h4nd5_m155i0n_4cc0mpl15h3d_4g3nt_007}`