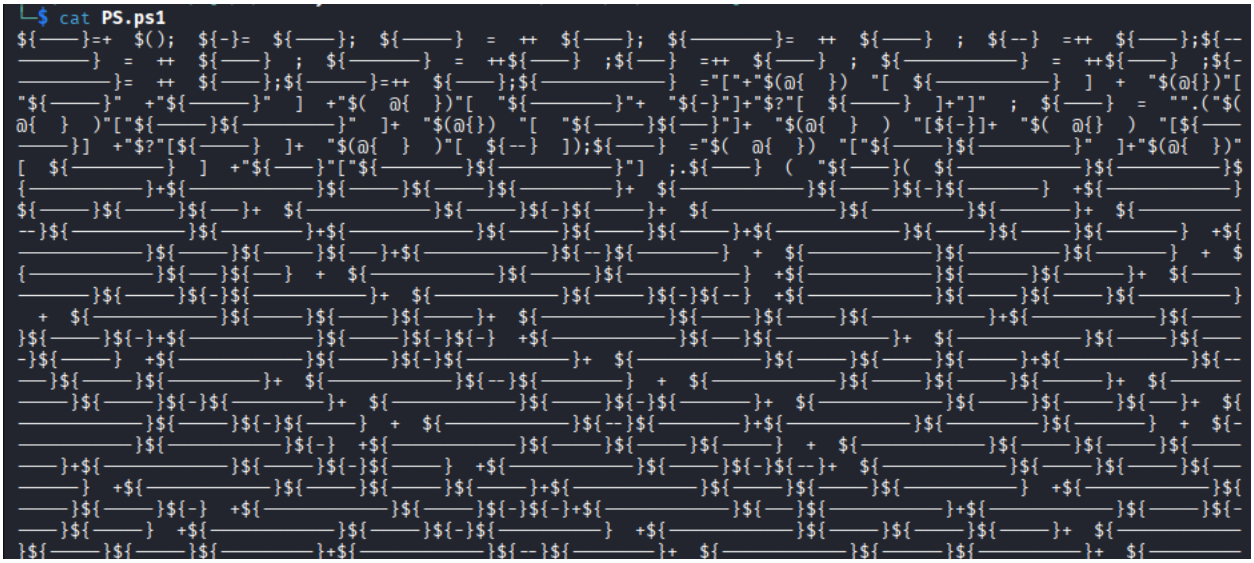
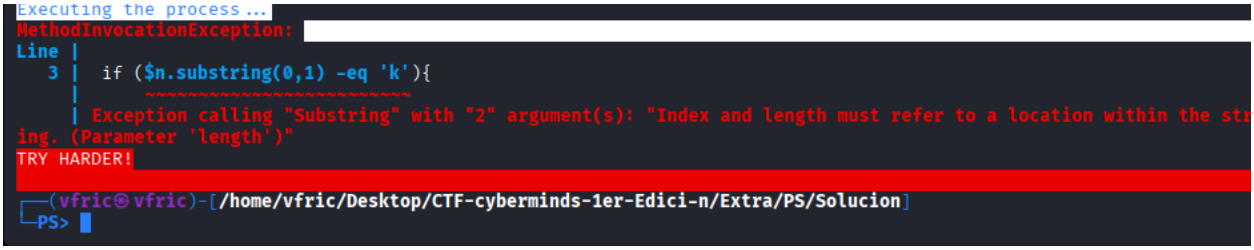
PS

Al descargar el archivo proporcionado, se procede a abrir el script PowerShell para su análisis estático. A primera vista se observa que el contenido se encuentra fuertemente ofuscado, utilizando concatenaciones de caracteres y variables sin sentido aparente, lo que dificulta la lectura directa del código.



A continuación, se ejecuta el script utilizando PowerShell (pwsh) con el objetivo de observar su comportamiento en tiempo de ejecución. Al ejecutarlo, el script únicamente muestra el mensaje “Executing the process...”, sin revelar información adicional relevante

Sin embargo, al ejecutar directamente el código ofuscado en la línea de comandos, se produce una excepción que resulta clave para el análisis.



Con el fin de analizar el comportamiento del script de forma controlada, se abre el archivo en un editor (VS Code) y se establece un breakpoint sobre la variable n utilizando el comando:

```
┌──(vfric㉿vfric)-[/home/vfric/Desktop/CTF-cyberminds-1er-Edici-n/Extra/PS/Solucion]
└─PS> Set-PSBreakpoint -Variable n

 ID Script                 Line Command              Variable                 Action
 -- ------                 ---- -------              --------                 ------
  1                                                  n


┌──(vfric㉿vfric)-[/home/vfric/Desktop/CTF-cyberminds-1er-Edici-n/Extra/PS/Solucion]
└─PS> ./PS.ps1
Executing the process ...
Entering debug mode. Use h or ? for help.

Hit Variable breakpoint on '$n' (Write access)

At line:2 char:1
+ $n = $MyInvocation.MyCommand.Name
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Posteriormente, se vuelve a ejecutar el script. La ejecución se detiene automáticamente cuando la variable n es asignada, permitiendo avanzar paso a paso mediante el depurado.

Durante esta ejecución controlada, el código comienza a mostrarse en texto claro, eliminando la ofuscación inicial.

```
┌──(vfric㉿vfric)-[/home/vfric/Desktop/CTF-cyberminds-1er-Edici-n/Extra/PS/Solucion]
└─PS> l


    1:  Write-Host -BackgroundColor white -ForegroundColor blue "Executing the process ..."
    2:* $n = $MyInvocation.MyCommand.Name
    3:  if ($n.substring(0,1) -eq 'k'){
    4:          #$xorkey = "try brute force!"
    5:          Write-Host "Flag: V15SU05yAlEDUAZpYQJEB0dlWQFfWGplBVFQB0ZFVwZfSQ=="
    6:          Write-Host -BackgroundColor green -ForegroundColor white "CONGRATZ!"
    7:          exit
    8:  }
    9:  Write-Host -BackgroundColor red -ForegroundColor white "TRY HARDER!"
```

Al listar el código durante la depuración, se observa claramente la estructura real del script. En ella se identifica una condición que evalúa el primer carácter del nombre del archivo y, en caso de no cumplirse, muestra el mensaje “TRY HARDER!”.

Dentro de la rama no ejecutada del if, se localizan dos líneas especialmente relevantes:

$xorkey = "try brute force!"

Write-Host "Flag: V15SU05yAlEDUAZpYQJEB0dlWQFfWGplBVFQB0ZFVwZfSQ=="

Estas líneas revelan que la flag se encuentra codificada en Base64 y posteriormente cifrada mediante XOR, utilizando una clave que aparentemente ha sido comentada.

```
import base64

# Cadena cifrada (la misma del reto)
```

```
cipher_b64 = "V15SU05yAlEDUAZpYQJEB0dlWQFfWGplBVFQB0ZFVwZfSQ=="

# Cargar diccionario (ajusta la ruta si es necesario)
with open("/usr/share/seclists/Passwords/Common-Credentials/500-worst-passwords.txt", "r",
encoding="utf-8", errors="ignore") as f:
passwords = [line.strip() for line in f]

def xor_ps_style(b64_string: str, key: str) -> str:
"""
Replica EXACTAMENTE:
[System.Text.Encoding]::UTF8
Base64 -> string UTF-8 -> bytes -> XOR -> string UTF-8
"""
# Base64 -> bytes
decoded_bytes = base64.b64decode(b64_string)

# bytes -> string UTF-8 (como hace PowerShell)
decoded_str = decoded_bytes.decode("utf-8", errors="ignore")

data_bytes = decoded_str.encode("utf-8")
key_bytes = key.encode("utf-8")

out = bytearray()
j = 0
for i in range(len(data_bytes)):
out.append(data_bytes[i] ^ key_bytes[j])
j += 1
if j >= len(key_bytes):
j = 0

return out.decode("utf-8", errors="ignore")

# Fuerza bruta de claves
for pwd in passwords:
result = xor_ps_style(cipher_b64, pwd)
if "flag" in result.lower() or "d3c0d3" in result.lower():
print("[+] CLAVE:", pwd)
print("[+] RESULTADO:", result)
break
```

```
└─$ python3 solucion.py
[+] CLAVE: 123456
[+] RESULTADO: flag{D3c0d3_P0w3rSh3ll_S4cc3ssf4l}
```