

FishMonger's arsenal upgraded: SprySOCKS for Windows

 welivesecurity.com/en/eset-research/fishmongers-arsenal-upgraded-sprysocks-windows

Bundled Files (7) ⓘ			
Scanned	Detections	File type	Name
✓ 2024-07-08	41 / 74	Win32 DLL	klelam00007/tpsvcloc.dll
✓ 2024-04-09	0 / 71	Win32 EXE	klelam00007/ApphostRagistreationVerifier.exe
✓ 2024-04-04	0 / 60	Text	klelam00007/klelam00007.bat
✓ 2024-04-09	0 / 71	Win32 DLL	klelam00007/tpsvc.dll
✓ ?	?	?	klelam00007/KW1B5206BDC1743FP.dat
✓ ?	?	?	klelam00007/KX1B5206BDC1743DD.dat
✓ ?	?	?	klelam00007/X1B5206BDC1743DD.dat

ESET researchers have discovered two as-yet undocumented Windows variants of [SprySOCKS](#), a previously Linux-only backdoor [reportedly](#) used by FishMonger, the group believed to be operated by a Chinese contractor named I-SOON. While we initially discovered the malware samples on VirusTotal, ESET telemetry shows real activity between 2023 and 2024, with several victims in Honduras, Taiwan, Thailand, and Pakistan, targeting mostly government organizations.

The Windows variants discovered are internally marked as WIN_DRV and WIN_PLUS. Both come with a hardcoded C&C configuration and support communication over TCP, UDP, and WebSocket protocols. The core backdoor functionality for both includes support for over 30 C&C commands, covering various functionalities including system information collection, process enumeration, as well as service management and file management functions such as listing, creating, deleting, and transferring files.

In addition to the core backdoor functionality, the WIN_DRV version utilizes kernel drivers to hide the malware's network connections, processes, files, and registry keys, and enables TCP traffic diversion allowing the malware operators to send commands to the backdoor through a random TCP port on the victim's device without exposing the backdoor's real listening port in the network traffic.

Based on ESET telemetry, there are limited indications that some SprySOCKS attack scenarios may involve a UEFI bootkit component, possibly exploiting CVE-2023-24932.

The analysis provided in this report leads us to attribute these new, Windows variants to FishMonger with high confidence.

Key points of this blogpost:

- We discovered two previously undocumented Windows variants of FishMonger's [SprySOCKS](#) backdoor.
- ESET telemetry shows activity between 2023 and 2024, primarily targeting government organizations in Honduras, Taiwan, Thailand, and Pakistan.
- Both Windows variants support communication over TCP, UDP, and WebSocket protocols, and implement over 30 commands.
- The WIN_DRV variant creates a stealthy passive TCP backdoor, relying on a kernel driver to redirect traffic to the backdoor's hidden TCP port whenever specially crafted data is detected inside a received TCP packet.

FishMonger profile

FishMonger – believed to be operated by a Chinese contractor named I-SOON (see our [Q4 2023–Q1 2024 APT Activity Report](#)) – is a cyberespionage group that falls under the Winnti Group umbrella and is most likely operating out of China, from the city of Chengdu. It is also known as Earth Lusca, TAG-22, Aquatic Panda, or Red Dev 10. We [published](#) an analysis of FishMonger in early 2020 when it heavily targeted universities in Hong Kong during the civic protests that started in June 2019. The group is also known to operate watering-hole attacks, as reported by [Trend Micro](#). FishMonger's toolset includes ShadowPad, Spyder, Cobalt Strike, FunnySwitch, SprySOCKS, and the BIOPASS RAT.

Technical analysis

In this section, we provide a technical analysis of these new, Windows variants of FishMonger's SprySOCKS backdoor.

The archive that led us to this discovery was uploaded to VirusTotal in April 2024 under the name klelam00007.zip; its contents are shown in Figure 1.

Bundled Files (7) ⓘ				
Scanned	Detections	File type	Name	
✓ 2024-07-08	41 / 74	Win32 DLL	klelam00007/tpsvclloc.dll	
✓ 2024-04-09	0 / 71	Win32 EXE	klelam00007/ApphostRagistreationVerifier.exe	
✓ 2024-04-04	0 / 60	Text	klelam00007/klelam00007.bat	
✓ 2024-04-09	0 / 71	Win32 DLL	klelam00007/tpsvc.dll	
✓ ?	?	?	klelam00007/KW1B5206BDC1743FP.dat	
✓ ?	?	?	klelam00007/KX1B5206BDC1743DD.dat	
✓ ?	?	?	klelam00007/X1B5206BDC1743DD.dat	

Figure 1. Contents of klelam00007.zip as displayed on VirusTotal

This archive contains various files, including legitimate ones used to host DLL side-loading, and three suspicious-looking, encrypted files with .dat extensions. Our subsequent analysis revealed that these encrypted files contain a new, previously undocumented Windows variant of FishMonger’s SprySOCKS backdoor, labeled WIN_DRV by its developers. Further investigation revealed an additional backdoor version, labeled WIN_PLUS, in ESET Telemetry.

Initial access

FishMonger has been known for targeting the public-facing servers of its victims, often exploiting server-based N-day vulnerabilities, to gain initial access. While we were not able to confirm the exact way FishMonger got into its victims’ systems in this campaign, the presence of a server operating system on some of the victim devices along with FishMonger’s typical modus operandi suggest that the attackers may well have got in through misconfigured or unpatched public-facing applications.

SprySOCKS for Windows

In September 2023, Trend Micro published a [report](#) about a new FishMonger Linux backdoor that its analysts named SprySOCKS. The code of the backdoor is based on an open-source Windows remote access trojan (RAT) named [Trochilus](#), and shares several common characteristics with the [RedLeaves backdoor](#); nevertheless, it was extended and modified enough to be considered a new backdoor. In this report, we analyze two as yet undisclosed Windows variants of v1.8 of SprySOCKS:

- One has been named WIN_DRV by its developers and uses a kernel driver for advanced stealth.
- Another, without the driver, is named WIN_PLUS.

As shown in Figure 2, the backdoor version type and number are hardcoded in the binary.

<pre> aWinDrv: ; DATA XREF: (; CollectInfo aVersiontype: text "UTF-16LE", 'WIN_DRV',0 ; DATA XREF: (; CollectInfo a18: text "UTF-16LE", 'versionType',0 ; DATA XREF: (; CollectInfo aVersionnumber: text "UTF-16LE", '1.8',0 ; DATA XREF: (; CollectInfo text "UTF-16LE", 'versionNumber',0 </pre>	<pre> aWinPlus: ; DATA XREF: (; CollectInfo text "UTF-16LE", 'WIN_PLUS',0 align 8 aVersiontype: ; DATA XREF: (; CollectInfo text "UTF-16LE", 'versionType',0 ; DATA XREF: (; CollectInfo a18: text "UTF-16LE", '1.8',0 ; DATA XREF: (; CollectInfo aVersionnumber: text "UTF-16LE", '1.8',0 ; DATA XREF: (; CollectInfo text "UTF-16LE", 'versionNumber',0 </pre>
--	--

Figure 2. Version type and number hardcoded in WIN_DRV (left) and WIN_PLUS (right) Windows SprySOCKS backdoor variants

The vast majority of artifacts and functionality present in the Linux version of the SprySOCKS backdoor introduced in Trend Micro’s [report](#) can also be found in the newly discovered Windows SprySOCKS variants described in this report. These include:

- the same C&C message format,
- very similar C&C commands (plus some additional ones),
- the same encryption keys and algorithms, and
- the use of the same statically linked networking library ([HP-Socket](#)).

For both of these new SprySOCKS variants, the core backdoor functionality involving C&C communication and available commands is very similar. The most notable differences can be spotted in the way the final backdoor is loaded, in the improved stealthiness, and in the component names and paths used.

In the following subsections, we first analyze components involved in the execution chain of individual SprySOCKS variants, and then we describe the backdoor component, which is mostly the same for both variants.

WIN_DRV components

In an archive uploaded to VirusTotal, we discovered the WIN_DRV version of SprySOCKS, which comes with an empty C&C configuration. As a result, this version does not actively contact any remote addresses; however, it is still capable of launching a TCP server on a random port on the victim’s device, thus acting as a passive backdoor. Interestingly, the attackers don’t need to know this server’s TCP port number because, as explained later, the RawWNPF driver used by the WIN_DRV version allows silent diversion – to the backdoor itself – of TCP traffic received on any open port (more in the section).

As shown in Figure 1, the archive containing the WIN_DRV version of SprySOCKS contains several files:

klelam00007.bat – a batch script responsible for persisting the backdoor. As shown in Figure 3, it:

- copies all files from the current working directory into the %SystemRoot%\Fonts directory (to function properly, the batch file needs to be deployed in the same directory as the rest of the files from the archive),
- creates a scheduled task named ApphostRagistreationVerifier, configured to execute ApphostRagistreationVerifier.exe (which is a legitimate, validly signed executable, renamed by the attackers to mimic the legitimate Microsoft-signed AppHostRegistrationVerifier.exe) with NT AUTHORITY\SYSTEM privileges on every system start. The attackers use the well-known [DLL side-loading](#) technique, taking advantage of the way Windows loads DLLs, to load their own malicious DLL (in this case tpsvcloc.dll) by using a legitimate, signed application. To be specific, in this case the attackers use [Malware Sideloaded via MFC Satellite DLLs](#) technique (note the loc string in the tpsvcloc.dll filename),
 - ApphostRagistreationVerifier.exe – a legitimate, ThinPrint' AutoConnect printer creation service signed executable (SHA-1: FFC3AA7909D4E72C360D65A1F45260DFFE5C99B7) that loads the tpsvc.dll library,
 - tpsvc.dll – a legitimate, signed library that loads the tpsvcloc.dll library,
 - tpsvcloc.dll – the SprySOCKS backdoor loader,
 - X1B5206BDC1743DD.dat – an encrypted container comprising the SprySOCKS backdoor and copies of the next two files,
 - KX1B5206BDC1743DD.dat – DriverLoader, an encrypted kernel driver responsible for loading another kernel driver from KW1B5206BDC1743FP.dat, and
 - KW1B5206BDC1743FP.dat – RawWNPF, an encrypted kernel driver responsible for hiding the backdoor's files and network activity.

```
for /f "tokens=*" %%i in ("%~0") do pushd "%~dpi"
copy /y ".\*.*" "%SystemRoot%\Fonts"
schtasks /create /tn "ApphostRagistreationVerifier"
           /tr "%SystemRoot%\Fonts\ApphostRagistreationVerifier.exe"
           /sc ONSTART /ru "NT AUTHORITY\SYSTEM" /f
schtasks /run /tn "ApphostRagistreationVerifier"
```

Figure 3. klelam00007.bat setting up persistence for the SprySOCKS backdoor (newlines added for readability)

Figure 4 depicts the execution chain of the SprySOCKS WIN_DRV variant.

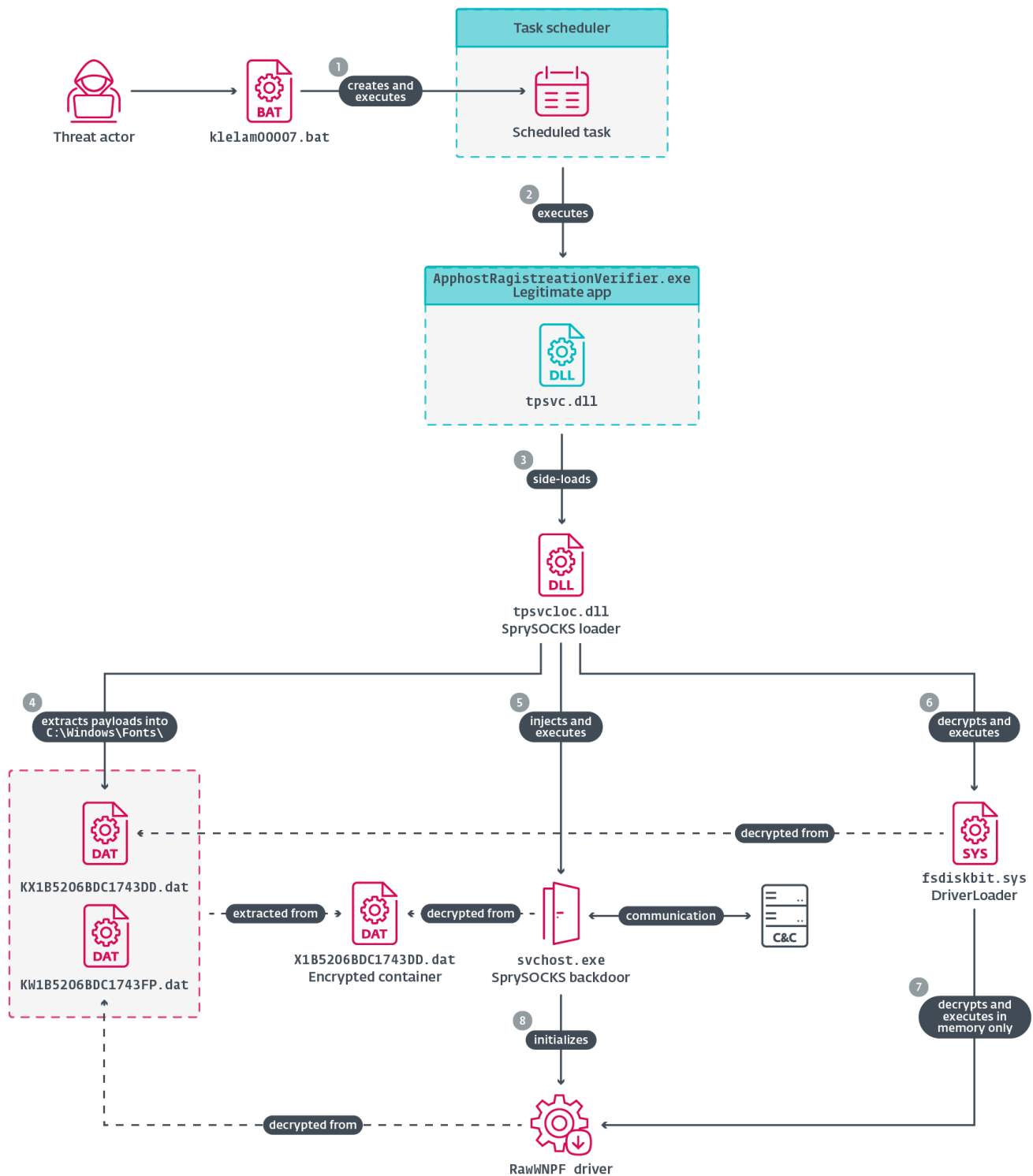


Figure 4. Execution chain of the SpySOCKS WIN_DRV variant

The following three subsections provide technical analyses of the aforementioned components: SpySOCKS loader, DriverLoader driver, and RawWNPF driver.

SpySOCKS loader

The loader starts with initial checks for the presence of a virtual environment and a few security products. It looks for specific libraries (namely: snxhk.dll, SxWrapper.dll, SxIn.dll,

SXIn64.dll, and SbieDll.dll) in the loader's process, and exits if it finds any of them.

As the next step, it verifies whether persistence was set successfully by the klelam00007.bat script, from Figure 3. To do so, it checks whether the current loader's image was loaded from the %SystemRoot%\Fonts\ directory, and tries to access the %SystemRoot%\Fonts\X1B5206BDC1743DD.dat, %SystemRoot%\Fonts\tpsvc.dll, and %SystemRoot%\Fonts\tpsvcloc.dll files. If it finds that any of these files are not where they are supposed to be, it sets up persistence on its own by:

- copying X1B5206BDC1743DD.dat, tpsvc.dll, tpsvcloc.dll, and ApphostRagistreationVerifier.exe from the current working directory into the %SystemRoot%\Fonts\ directory,
- registering the %SystemRoot%\Fonts\ApphostRagistreationVerifier.exe application as a debugger for vds.exe (a Virtual Disk Service that can be automatically executed on system start) by writing the application's path into the registry value HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\vds.exe\debugger, and
- dropping the affair-build.bat file into the %SystemRoot%\Fonts\ directory and then executing it via cmd.exe. This script, shown in Figure 5, clears traces of this process by removing files from the deployment directory and executing the malware again (now from %SystemRoot%\Fonts\) by restarting the vds service.

```
@echo off
ping 127.0.0.1 -n 4
del "<CurrentWorkDir>\ApphostRagistreationVerifier.exe"
del "<CurrentWorkDir>\tpsvcloc.dll"
del "<CurrentWorkDir>\tpsvc.dll"
sc stop vds
sc start vds
del %0
```

Figure 5. affair-build.bat executed by the SprySOCKS loader

When persistence is set, the loader continues with loading payloads from an encrypted container located at %SystemRoot%\Fonts\X1B5206BDC1743DD.dat. The decryption algorithm and key: 128-bit AES in ECB mode with the hardcoded key uXQLESMXGaRMs6BL.

This produces shellcode generated by the [DllToShellCode](#) open-source tool. Before executing the shellcode, it extracts the rest of the encrypted payloads from the container into separate files:

- %SystemRoot%\Fonts\KX1B5206BDC1743DD.dat
- %SystemRoot%\Fonts\KW1B5206BDC1743FP.dat

When done, the loader spawns a new svchost.exe process using CreateProcessAsUserW with a token obtained from spoolsv.exe, and injects the backdoor's shellcode into the process by using the [process doppelganging](#) technique. During the injection process, the shellcode is dropped into a temporary file, using the prefix TH in its filename, within the %TEMP% directory.

As the last step, the loader proceeds to decrypt and execute DriverLoader, a kernel driver hidden inside the previously dropped KX1B5206BDC1743DD.dat file. DriverLoader is first decrypted, then the decrypted contents are saved to C:\Windows\System32\drivers\fsdiskbit.sys. To execute it, the loader installs this driver as a [minifilter](#) driver by manually creating a new service registry key named msdiskserver with an ImagePath value pointing to the dropped driver (as shown in Figure 6) and invokes the NtLoadDriver Windows API function with the registry key as the parameter to load it. If no errors are detected, the loader deletes both the msdiskserver registry key and the fsdiskbit.sys file. After this, the loader is done and exits.

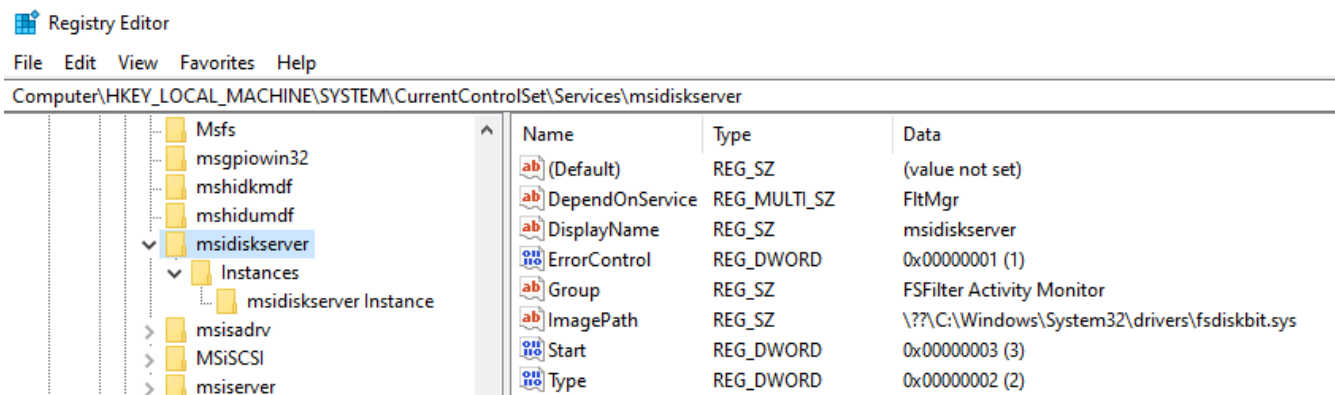


Figure 6. Service registry key created by the SprySOCKS WIN_DRV loader

DriverLoader driver

Before jumping to DriverLoader's functionality, one important note: with the release of Windows Vista, Microsoft introduced driver signature enforcement (DSE), a feature ensuring that only validly signed kernel-mode components are allowed to be executed in the Windows kernel. This means that to execute the fsdiskbit.sys driver (DriverLoader), attackers need to sign it with a trusted certificate.

To make the driver work on at least some outdated or misconfigured systems, the attackers used a leaked certificate available on GitHub in the [PastDSE](#) project repository, and signed the fsdiskbit.sys driver with it. Information about the certificate used can be found in Figure 7.

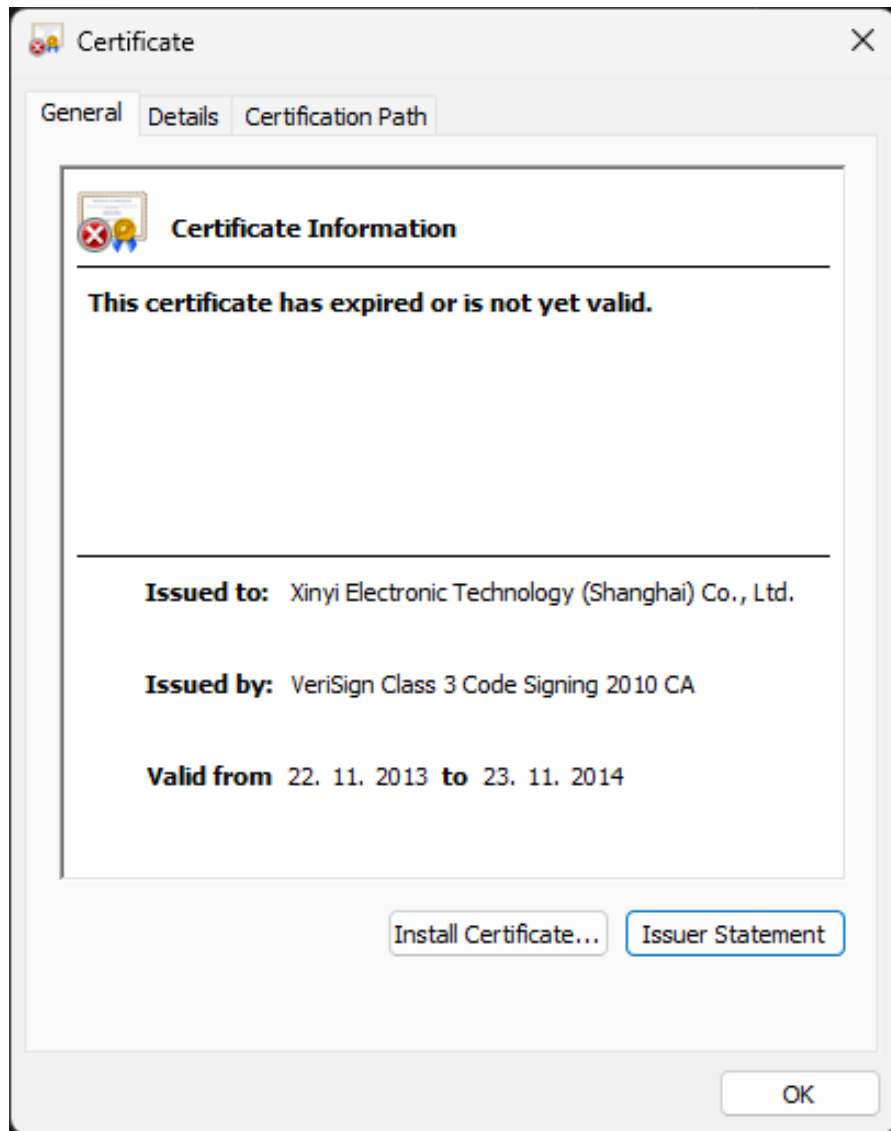


Figure 7. DriverLoader's code-signing certificate

Now to the functionality. The purpose of this component is quite straightforward: to load another driver, this time in memory only. First, it reads and decrypts the contents of the C:\Windows\Fonts\KW1B5206BDC1743FP.dat file, previously created by the loader. It uses the same algorithm and key as used by the loader: 128-bit AES in ECB mode with the key uXQLESMXGaRMs6BL. The decrypted data contains a native PE binary (described in the section), which is then manually mapped and its entry point executed.

There is the PDB path embedded in the DriverLoader binary:

C:\Users\xdd\Desktop\今天\2023-4-11\2023-04-10__注册表驱动加载功能__集成到内测3中-未完成\DriverMemoryLoadDriver\x64\Release\DriverMemoryLoadDriver.pdb

The parts in simplified Chinese machine translate as:

- 今天: Today

- 注册表驱动加载功能__集成到内测3中-未完成: Registry driver loading function__is integrated into internal beta 3-not completed

As we can see in the symbols path, this component seems to have been in development at least since April 2023, which aligns with DriverLoader's compilation timestamp. Similarly, strings in the path suggest that the project this driver is part of was likely still in development when the driver was compiled.

RawWNPF driver

The RawWNPF driver is the component that makes the WIN_DRV version of the SprySOCKS backdoor much stealthier when compared to the WIN_PLUS variant. It allows hiding the backdoor's malicious activity on the compromised system, and can be configured by invoking the driver's custom I/O control codes (IOCTLs). The driver creates a device driver named \Device\RawWNPF; a list of the available IOCTLs, with short descriptions, is shown in Table 1.

Table 1. List of IOCTLs handled by the RawWNPF driver

IOCTL	Description
0x220200	Configure the driver to hide active network connections to and from the specified local TCP port.
0x220300	Unhide the network connections configured with 0x220200.
0x220340	Insert an entry into the hidden connections list.
0x220344	Remove an entry from the hidden connections list.
0x220348	Wipe the whole hidden connections list.
0x22034C	Read the hidden connections list.
0x220350	Insert a process with a specified PID into the hidden processes list.
0x220354	Remove a process with a specified PID from the hidden processes list.
0x220358	Wipe the whole hidden processes list.
0x22035C	Read the hidden processes list.

IOCTL	Description
0x222000	Initialize the driver's main functions (hiding network connections, hiding processes, hiding malware components, network filters, persistence protection). After this initialization, other IOCTLs can be used to configure what exactly should be hidden.
0x222004	Returns two hardcoded DWORD values: 1 and 2. This possibly could be the driver's version.
0x222008	Delete the driver's binary (if it exists).

Hiding specified processes

The RawWNPF driver can be configured to hide processes based on their process IDs, and a list of hidden processes can be managed by invoking the driver's IOCTLs 0x220358, 0x22035C, 0x220354, and 0x220350. To hide a process, the driver hooks execution of the NtQuerySystemInformation system call and modifies its output if information about running processes is being retrieved (i.e., if SystemProcessInformation is passed to the SystemInformationClass parameter). If any of the processes retrieved by this API function match a process from the driver's list of hidden processes, the driver removes this process from the function's output. The way the kernel driver hooks the NtQuerySystemInformation system call seems to be heavily based on source code from the [InfinityHookPro](#) project.

Hiding network activity

The driver can be configured to hide specific active connections (with a specified IP, port, or combination of both) so that they won't be listed in the output of common network administration tools such as netstat.exe. This is achieved by a well-known technique (e.g., [1], [2], [3], ...), where attackers hook IoCompletionRoutine for IOCTL 0x12001B inside the DeviceIoControl function of the nsiproxy.sys Windows kernel driver. The code inside nsiproxy's 0x12001B IOCTL handler is responsible for retrieving the list of active connections, and hooking its IoCompletionRoutine allows attackers to walk through the retrieved list, check for the presence of specific ports, addresses, or both, and hide the specific connection in the list if a match is found. Figure 8 shows the hook function responsible for hiding network connections.

```

NsiData = Irp->UserBuffer;
if ( MmIsAddressValid(NsiData->Entries) )
{
    if ( NsiData->EntrySize == 0x38 )
    {
        pConnEntries = NsiData->Entries;
        nItemCnt = NsiData->ConnCount;
        v17 = nItemCnt;
        KeStackAttachProcess(Context->currProcess, &ApcState);
        for ( i = 0LL; i < nItemCnt; ++i )
        {
            v12 = i;
            RemoteIP = pConnEntries[i].remoteEntry.dwIP;
            RemotePort = __ROL2__(pConnEntries[i].remoteEntry.Port, 8);
            LocalPort = __ROL2__(pConnEntries[i].localEntry.Port, 8);
            if ( LocalPort == gHiddenLocalServerNetPort
                || RemotePort == gHiddenLocalServerNetPort && RemoteIP == 0x100007F
                || CheckConnHidingPolicy(
                    _byteswap_ulong(RemoteIP),
                    RemotePort,
                    _byteswap_ulong(pConnEntries[i].localEntry.dwIP),
                    LocalPort) )
            {
                // remove entry
                *pConnEntries[v12].localEntry.bytesfill0 = 0LL;
                *&pConnEntries[v12].localEntry.bytesfill[8] = 0LL;
                *&pConnEntries[v12].remoteEntry.dwIP = 0LL;
                *&pConnEntries[v12].remoteEntry.bytesfill[12] = 0LL;
            }
            nItemCnt = v17;
        }
    }
}

```

Figure 8. Hex-Rays decompilation of nsiproxy's IoCompletionRoutine hook responsible for hiding network connections

In addition to the hiding of active network connections, the driver contains an interesting functionality allowing it to divert TCP packets received on any open TCP port, to the specified TCP port configured by the IOCTL 0x220200 (it's actually the port of the SprySOCKS backdoor's TCP server), but only in the case that the TCP data received contains specially crafted data. To achieve this, the driver registers its own packet filter objects using Windows Filtering Platform (WFP) API functions, manually parses contents of transferred IPv4 packets (both inbound and outbound traffic is inspected), and proceeds to divert the traffic if the specially crafted data is detected inside a received TCP packet data. The purpose of this feature seems to be mainly a capability to contact the malicious backdoor without the need to embed a C&C address inside the binary. Additionally, even though such diverted traffic can be inspected using tools such as Wireshark, the real port (the one the traffic is diverted to) is not revealed; thus it can be difficult to investigate the real destination for this malicious traffic.

Installed packet filters, along with their identifying information, are listed in Table 2.

Table 2. WFP filter objects registered by the RawWNPf driver

Filter layer name	Filter object name and GUID	Filter object callout name and GUID
Inbound IP Packet v4 Layer	Delivery Optimization (TCP-In) {E980088D-BE44-4057-8E5C-C7FDF8968795}	COInbound {DE0D7F67-94ED-4DDB-8215-9C028B54661B}
Outbound IP Packer v4 Layer	Delivery Optimization (TCP-Out) {33F76397-DBCB-445E-8EC3-AA51ED302D15}	COOutbound {8280DDF3-7489-4402-B9D8-96B50912346B}
ALE Connect v4 Layer	Delivery Optimization (TCP-In) {5746AF70-2917-4861-97E6-D5E4DD569F2D}	COAuthConnect {A33E1AA8-9B0F-44A3-B24A-AEB04CA54C3B}
ALE Listen v4 Layer	Delivery Optimization (TCP-In) {7CB4DFB4-0D20-402D-A49D-BA9660D026E6}	COAuthListen {40045FAF-6BAE-4B48-9119-31B48FFEA629}
ALE Receive/Accept v4 Layer	Delivery Optimization (TCP-In) {2C1AB6EF-0B65-4634-8666-BCB2CF9C72E9}	COAuthAccept {DDFE5189-389F-437F-9B92-59495ED2181A}
ALE ResourceAssignment v4 Layer	Delivery Optimization (TCP-In) {B4AE248F-98D5-446F-88EB-14CF605AE722}	COAuthResAssignment {FE570356-A1A9-413C-94CC-BD6C448E9969}

Hiding the backdoor's files

The driver hides/protects the SprySOCKS backdoor's files by registering itself as a minifilter driver, and installing the following callbacks:

- pre-operation callback triggered on every IRP_MJ_CREATE I/O request and responsible for returning STATUS_NO_SUCH_FILE on every attempt to create or open a file or a directory from the driver's list of hidden/protected files,
- pre-operation callback triggered on every IRP_MJ_DIRECTORY_CONTROL I/O request and responsible for filtering out non-directory-enumeration related requests, so that only the ones related to directory enumeration are passed to the post-operation callback, and

- post-operation callback triggered on IRP_MJ_DIRECTORY_CONTROL I/O requests that passed pre-operation callback checks. This callback is responsible for removing entries of hidden/protected files from any directory listing attempts.

The following hardcoded list of filenames are protected by the driver:

- \SystemRoot\Fonts\tpsvc.dll
- \SystemRoot\Fonts\tpsvclloc.dll
- \SystemRoot\Fonts\ApghostRagistreationVerifier.exe
- \SystemRoot\Fonts\X1B5206BDC1743DD.dat
- \SystemRoot\Fonts\KX1B5206BDC1743DD.dat
- \SystemRoot\Fonts\KW1B5206BDC1743FP.dat

Protecting persistence

The driver calls CmRegisterCallbackEx to install a RegistryCallback routine responsible for hiding the registry key used for the SprySOCKS loader's persistence:

HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution

Options\vds.exe. As a result, all attempts to open or enumerate the key are filtered out by the driver.

WIN_PLUS components

In the SprySOCKS WIN_PLUS version, we first discovered the malicious encrypted container in our telemetry, with the first hit dating back to July 2024 found on the device of a victim in Pakistan. It contained the SprySOCKS backdoor and the SprySOCKS loader. The C&C configuration was present and is shown in Figure 9.

32 30 37 2E 31 34 38 2E 37 38 2E 33 36 00 00 00	207.148.78.36 ..
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00 00 00 00 00 00 00 00 34 34 33 00 00 00 00 00	443.....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
32 30 37 2E 31 34 38 2E 37 38 2E 33 36 00 00 00	207.148.78.36 ..
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00 00 00 00 00 00 00 00 35 33 00 00 00 00 00 00	53.....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
32 30 37 2E 31 34 38 2E 37 38 2E 33 36 00 00 00	207.148.78.36 ..
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00 00 00 00 00 00 00 00 38 30 00 00 00 00 00 00	80.....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00 00 00 00 00 00 00 00 35 33 37 38 31 00 00 00	53781..

- TCP Client configuration (IP + port)
- UDP Client configuration (IP + port)
- HTTP Client configuration (IP + port)
- TCP Server configuration (listening port)

Figure 9. C&C configuration from the WIN_PLUS version of SprySOCKS

The encrypted container was located at the following path on the compromised system:

C:\Windows\System32\spool\drivers\color\config.dat

When decrypted, the container contains a SprySOCKS loader and the SprySOCKS backdoor itself. Further analysis of the SprySOCKS backdoor from the container showed that, in this case, there seemed to be an additional component responsible for loading the SprySOCKS loader from the encrypted container. This component – referenced to as the first-stage loader in this analysis – should be installed as a [print processor](#) under the following registry key:

HKLM\SYSTEM\ControlSet001\Control\Print\Environments\Windows x64\Print Processors\VSPMsg

Interestingly, when we searched our telemetry for anything related to this VSPMsg string, we discovered a file deployed on two different victim devices from Honduras at C:\Windows\System32\spool\prtprocs\x64\VSPMsg.dll. This file turned out to be the first-stage loader responsible for executing the SprySOCKS loader from the aforementioned config.dat file.

An execution diagram of the SprySOCKS WIN_PLUS variant is illustrated in Figure 10.

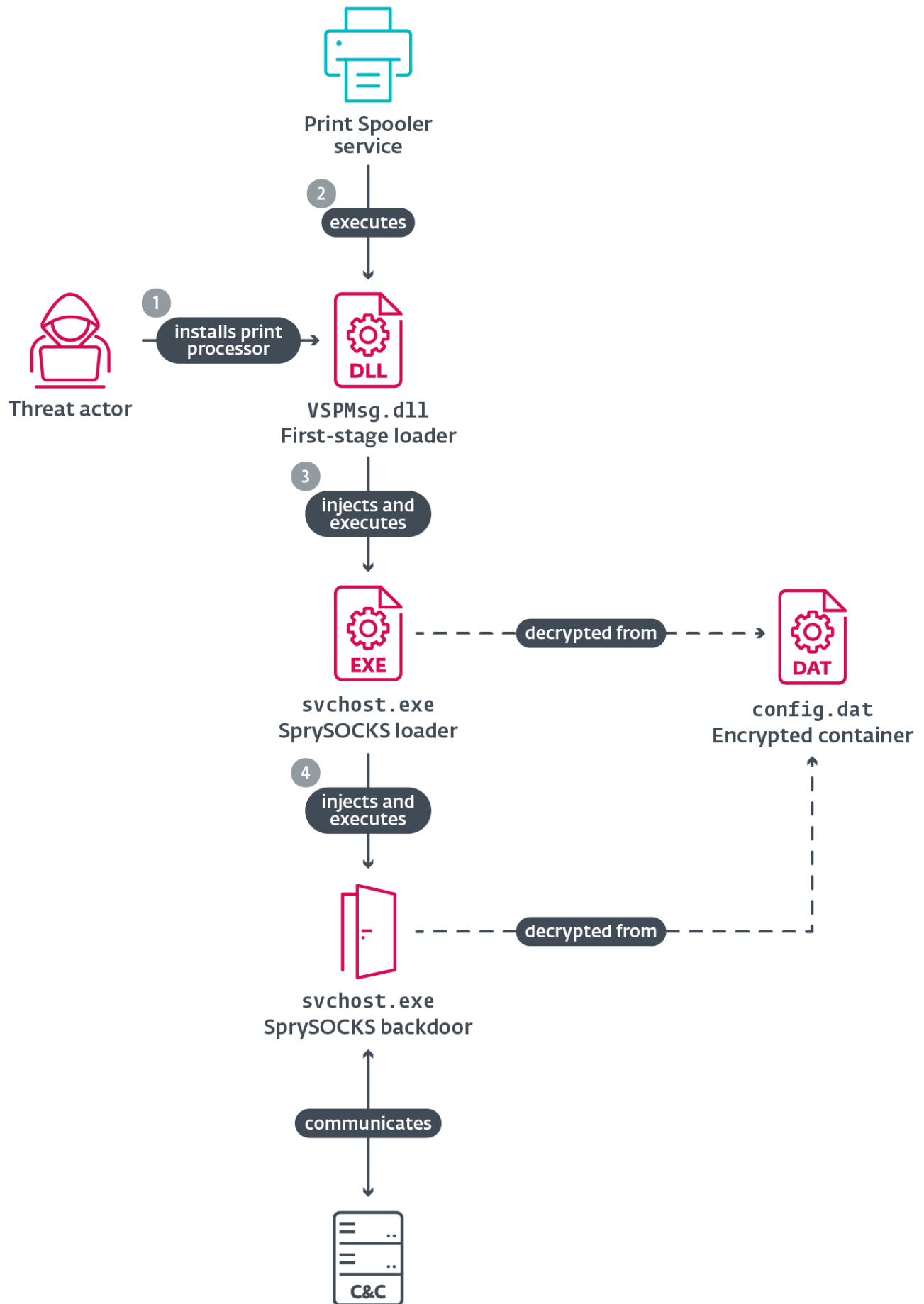


Figure 10. SprySOCKS WIN_PLUS variant execution scheme

First-stage loader

This loader starts by checking whether it was executed by spoolsv.exe, and exits if not; this hides its behavior from automated malware analysis sandboxes, as the loader is intended to be run as a print processor. It continues decrypting the SprySOCKS loader from the encrypted container C:\Windows\System32\spool\drivers\color\config.dat. First it 128-bit AES-ECB decrypts the loader with the hardcoded key uXQLESMXGaRMs6BL, then injects it into the newly created svchost.exe process via [process doppelg nging](#). Meanwhile, the SprySOCKS loader is dropped into a temporary file, with a filename prefix of TH, within the %TEMP% directory.

The sample exports two functions:

- GetErrorMessageModule
- SetErrorMessageModule

While the SetErrorMessageModule function does not do anything, the GetErrorMessageModule function is meant to be used to set persistence for the loader itself. When executed, it registers the loader as a print processor by creating the HKLM\SYSTEM\ControlSet001\Control\Print\Environments\Windows x64\Print Processors\VSPMsg registry key, setting the Driver registry value to VSPMsg.dll, and copying the hardcoded C:\ProgramData\Microsoft Event\PFs\VSPMsg.dll to the C:\Windows\System32\spool\prtprocs\x64\ directory. As the next step, it copies the encrypted container from C:\ProgramData\Microsoft Event\PFs\config.dat to C:\Windows\System32\spool\drivers\color\config.dat and, when done, it generates and drops the affair-build.bat batch script into the C:\Windows\System32\spool\drivers\color\ directory and executes it. As shown in Figure 11, this script's purpose is to cover the loader's tracks by removing the files in the original deployment directory, and triggering execution of the newly installed print processor by restarting the print spooler service.

```
@echo off
sc stop spooler
del "C:\ProgramData\Microsoft Event\PFs\<unknown_loader_process>"
ping 127.0.0.1 -n 66
sc start spooler
netsh int tcp set global ecn=disabled
del C:\ProgramData\Microsoft Event\PFs\VSPMsg.dll
del %0
```

Figure 11. affair-build.bat batch script used by the first-stage SprySOCKS WIN_PLUS loader

SprySOCKS loader

This loader starts by creating a mutex with the hardcoded name fqwhi2d1qaz2, and then proceeds to loading the SprySOCKS backdoor from the encrypted container located at C:\Windows\System32\spool\drivers\color\config.dat. It 128-bit AES-ECB decrypts the backdoor with the hardcoded key uXQLESMXGaRMs6BL, then injects it into the newly created svchost.exe process via [process doppelganging](#). Meanwhile, the SprySOCKS loader is dropped into a temporary file, with a filename prefix of TH, within the %TEMP% directory.

SprySOCKS backdoor

Finally, we proceed to our analysis of the SprySOCKS backdoor itself. In both variants, WIN_DRV and WIN_PLUS, the backdoor functionality is almost the same, and the differences are only in the specific file paths used, registry keys used, and as already mentioned, the WIN_PLUS version does not use the RawWNPF driver for advanced stealthiness.

Both variants analyzed in this report are DLLs with the original name PrcsServer.dll, exporting a function named Stop. They create a mutex named prcs-server-run in the beginning and right after that proceed to the initialization of the backdoor's main functionality, which includes initialization and launching of C&C communication channels (based on the hardcoded configuration) and setting up the keylogger. In addition to these actions, the WIN_DRV backdoor version initializes the RawWNPF driver by invoking its 0x222000 IOCTL handler, and then hides its own process by invoking the driver's 0x220350 IOCTL.

Keylogging is activated only if there is an existing INI file at %appdata%\Microsoft\Vault\lgf.dat that contains a config section with a property named key that is set to 1. If these conditions are met, both backdoors create a mutex named Global{DCAA7ED8-521B-4EAB-BE21-65254CF59239} and periodically log clipboard data along with the active window title and keystrokes into the file %appdata%\Microsoft\Vault\lgf.dat. The data in the file is encrypted using a single-byte XOR cipher with the key 0x44.

C&C communication

The backdoor supports three protocols for communication with the C&C – TCP, UDP, and WebSocket – and can act as both client and server. The networking-related functionality is heavily based on the [HP-Socket](#) networking framework, and some cryptography functions were implemented using the [Crypto++](#) library.

The C&C configuration is embedded in the backdoor, and can contain:

- up to three IP addresses and associated ports, each specifying a C&C IP address and its port for one of the communication channels (TCP, UDP, or WebSocket), and

- up to three port numbers, each specifying a port the backdoor should listen on for new connections. One is used for a TCP server, one for a UDP server, and one for a WebSocket server.

An example configuration from the WIN_PLUS version is shown in Figure 9 and it contains:

- The C&C address and port for the TCP communication channel: 207.148.78[.]36:443.
- The C&C address and port for the UDP communication channel: 207.148.78[.]36:53.
- The C&C address and port for the WebSocket communication channel: 207.148.78[.]36:80.
- The backdoor's TCP server listening port: 53781.

Before initiating any connections or starting a server, the SprySOCKS WIN_DRV version hides any connections from/to the addresses or ports from the configuration by invoking the RawWNPF driver's IOCTLs 0x220340 and 0x220200. As a result, these connections won't be listed in output of tools such as netstat.exe, despite being active. In addition, both backdoor versions execute the netsh.exe utility twice:

```
netsh.exe netsh advfirewall firewall delete rule name="Core Networking - Packet Too Big(ICMPv6 - In)"
```

```
netsh advfirewall firewall add rule name="Core Networking - Packet Too Big(ICMPv6 - In)" dir=in action=allow protocol=tcp localport=53781
```

The first command deletes a specified firewall rule, and the second adds a new firewall rule of the same name as the one just deleted, allowing all inbound TCP traffic sent to the backdoor's TCP server port specified in the configuration.

If the C&C configuration is empty (as in the case of the WIN_DRV version we discovered on VirusTotal), the backdoor starts a TCP server that listens on a random port on the compromised machine and also hides this port by invoking the RawWNPF driver's IOCTL 0x220200. This invocation not only hides the TCP server from being listed in standard networking tools' output, but also activates the TCP-diverting feature provided by the RawWNPF driver. This feature allows attackers to send commands to the backdoor without knowing the real port the backdoor listens on, simply by sending specially crafted TCP data to any open TCP port on the victim's machine.

For the TCP communication channel, the C&C protocol seems to remain the same as in the Linux version analyzed in Trend Micro's [report](#). Each time before sending the actual backdoor's data, it sends a 12-byte header containing the 32-bit CRC of the rest of the header, a DWORD magic value 0xACACBCBC, and a DWORD specifying the size of the data that follows the header.

For the UDP and WebSocket channels, the magic values are different, and so are the message header format and size. For the UDP channel, the magic value is 0xACACBFBC and it's located at offset 0x1C in a 36-byte header, followed by a DWORD specifying the size of the data that follows. In the WebSocket channel, the magic value 0x1BDCCBAA is used as a Masking-Key in the WebSocket [header](#). Figure 12 shows a network traffic capture with the magic values for each of the communication channels.

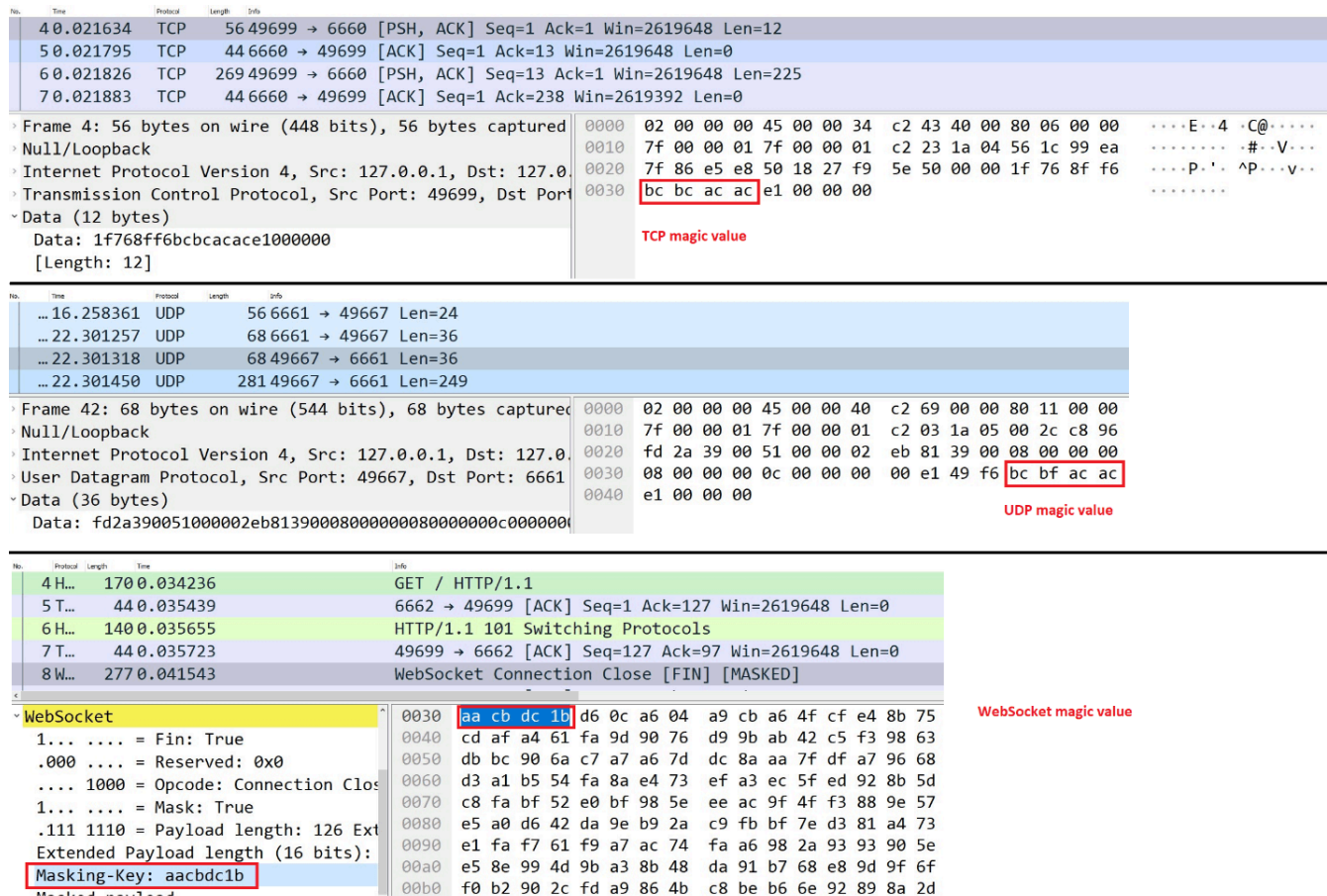


Figure 12. SprySOCKS network-traffic capture showing the magic values used in TCP, UDP, and WebSocket (from top to bottom, respectively) C&C communication channels

Following the header is, again, a 32-bit CRC, then the WORD value 0x0003 (likely indicating the encryption method), followed by 128-bit AES-ECB mode encrypted data (using the hardcoded key QFTHEYjzX3RBOMgZ) that has been base64 encoded.

An example of a C&C message before and after decoding and decryption is shown in Figure 13.

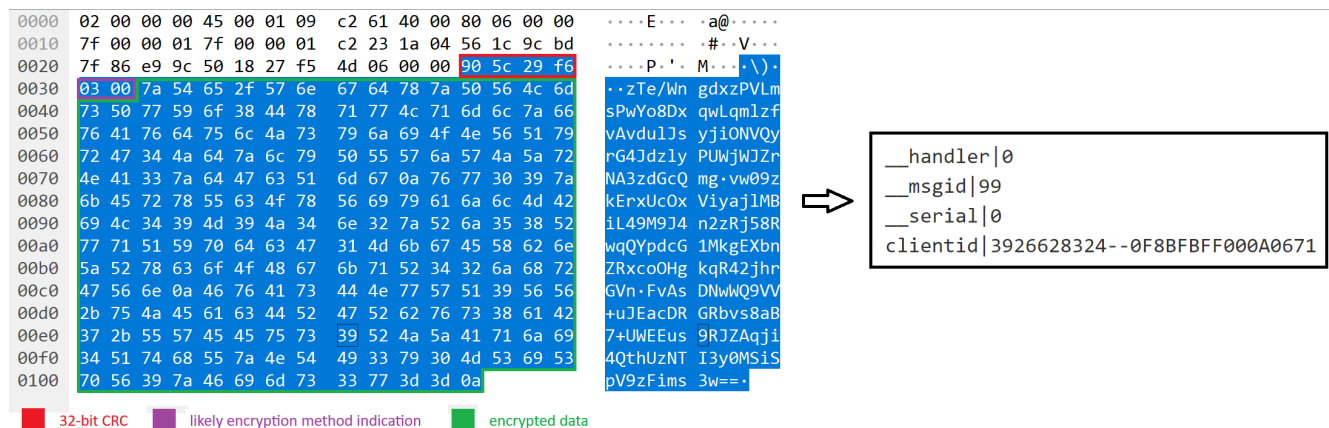


Figure 13. Example SprySOCKS C&C message as seen in Wireshark (left), and its contents after decoding and decryption (right)

The __msgid value in the decrypted C&C message is used to specify a command, identified by a message ID, that should be executed by the backdoor. The list of message IDs supported by the backdoor, along with their description, can be found in Table 3. Note that we haven't analyzed all these commands in depth; therefore, some descriptions are just a rough overview of the part of the code/functionality the message ID is related to.

Table 3. SprySOCKS C&C commands; descriptions marked with * are tentative assessments

Message ID	Description
0x09	Collect client (victim) system information, including: computer name, OS version, network adapter information, information about memory, CPU information, current privileges, system language and version, current time, and the backdoor version (1.8) and version type (WIN_DRV or WIN_PLUS).
0x0A	Start an interactive console.
0x0B	Write into the interactive console.
0x0D	Stop the interactive console.
0x0E	Specify an additional communication channel (do not start the channel). Likely to specify an additional backup C&C.
0x0F	Send C&C message to a different target.*
0x11	Enumerate all processes.
0x12	Enumerate modules of a process specified by a PID.

Message ID	Description
0x13	Terminate a process specified by a PID.
0x14	Close all connections.
0x16	Get current communication channel information.
0x17	Specify additional communication channels (TCP, UDP, or WebSocket) and start them.
0x19	Uninstall the backdoor and exit.
0x1E	Enumerate all services.
0x1F	Configure StartType for a specified service.
0x20	Start services with a specified name.
0x21	Invoke the ControlService function with a specified dwControl parameter.
0x22	Delete a specified service from the service manager. This does not stop the service if it's running.
0x23	Initialize SOCKS proxy.
0x24	Terminate SOCKS proxy.*
0x25	Send data through SOCKS proxy.
0x26	SOCKS proxy-related command.*
0x2A	Upload a specified file.*
0x2B	File-transfer-related helper command.*
0x2C	Download a specified file.*
0x2D	File-transfer-related helper command.*
0x3C	Enumerate free disk space.
0x3D	List files in the specified directory.

Message ID	Description
0x3E	Delete a specified file.
0x3F	Create a specified directory.
0x40	Rename a specified file.
0x41	Execute an existing file.
0x42	Copy a specified file.
0x43	List files from the Recent Windows directories for the logged-in user: %APPDATA%\Microsoft\Windows\Recent\ %APPDATA%\Microsoft\Office\Recent\

Network infrastructure

Only one C&C address has been discovered in this campaign: 207.148.78[.]36, hardcoded in the configuration (shown in Figure 9) of the WIN_PLUS variant of the SprySOCKS backdoor.

Ports from the configuration that should be used by the backdoor to communicate with the C&C:

- TCP: 443
- UDP: 53
- WebSocket: 80

As mentioned in Trend Micro's report, the IP address 207.148.75[.]122, from the same IP range 207.148.64.0/20 as the C&C above, was used by FishMonger operators as a SprySOCKS delivery server in June 2023. This IP range belongs to the Vultr cloud hosting provider.

Conclusion

The discovery of a Windows variant of SprySOCKS, previously known as Linux-only backdoor, represents a meaningful expansion of FishMonger's cross-platform capabilities. Our analysis shows that the Windows port retains most of the core architecture of its Linux predecessor – including the C&C protocol, encryption used, and overall command handling logic – while substituting Windows-native mechanisms where required and improving the stealthiness of the backdoor by bringing the kernel drivers to the game. Considering the limited indications of

possible UEFI bootkit involvement, we advise everyone to keep a close eye on the group's activities.

For any inquiries about our research published on WeLiveSecurity, please contact us at threatintel@eset.com.

ESET Research offers private APT intelligence reports and data feeds. For any inquiries about this service, visit the [ESET Threat Intelligence](#) page.

IoCs

Files

SHA-1	Filename	Detection	Description
955BFC3DCC867256F9F46A606DEB0779FA3416D8	KX1B5206BDC1743DD.dat	Win64/SprySOCKS.A	Encrypted SprySOCKS DriverLoader driver.
44DC4A08C5EB0972C8E18B0E01284E06F09006BB	bthcam.sys	Win64/Agent.ESB	SprySOCKS DriverLoader driver.
AB87B29B6F79487C75CA08D102E79001E536F083	KW1B5206BDC1743FP.dat	Win64/SprySOCKS.A	Encrypted SprySOCKS RawWNPf driver.
6490B8E4AADE25A3EE2DA9A47F312DB2122470BC	X1B5206BDC1743DD.dat	Win64/SprySOCKS.A	Encrypted container of the encrypted WIN_DRV variant of SprySOCKS backdoor, encrypted SprySOCKS RawWNPf and SprySOCKS DriverLoader drivers.

SHA-1	Filename	Detection	Description
E7484C24B88A1A2407A8 F09D734F9A993670285B	klelam00007 .zip	Win64/Agent.CXZ Win64/SprySOCKS.A BAT/Runner.KS	ZIP archive from VirusTotal containing the WIN_DRV variant of SprySOCKS, along with all the backdoor's components; clean binaries used for side-loading are included.
621D1952839BE4B0A1B0 E66E87BCE5062CA368ED	tpsvclloc.dll	Win64/Agent.CXZ	SprySOCKS loader.
2457EED2AB28E37741F1 0914EF929DAD2C8079D4	VSPMsg.dll	Win64/Agent.CXZ	First-stage loader responsible for launching the SprySOCKS loader.
D2C706B1EAF662BF0CE1 24B5032F73ED84BDA24A	N/A	Win64/SprySOCKS.A	WIN_PLUS variant of the SprySOCKS backdoor.
5F3B87CEF56683D9A9E1 9186E0FD0D8019B559C4	N/A	Win64/Agent.CXZ	SprySOCKS loader.
C793CA31E3F6628B5C89 86146953BF66232E9A30	config.dat	Win64/SprySOCKS.A	Encrypted container of the WIN_PLUS variant of the SprySOCKS backdoor and its loader.

SHA-1	Filename	Detection	Description
037DB2445F3D72388CB2 CF8510563148E5A184BE	N/A	BAT/Runner.KS	Batch script that persists the WIN_DRV variant of SprySOCKS.

Network

IP	Domain	Hosting provider	First seen	Details
207.148.78[.]36	N/A	IRT-CHOOPALLC-AP	N/A	C&C IP hardcoded in the SprySOCKS backdoor (WIN_PLUS variant).

MITRE ATT&CK techniques

This table was built using [version 19](#) of the MITRE ATT&CK framework.

Tactic	ID	Name	Description
Reconnaissance	T1592.004	Gather Victim Host Information: Client Configurations	SprySOCKS can collect information about the compromised device, including: computer name, OS version, information about memory and CPU, current privileges, system language and version, current time, and more.
	T1590.005	Gather Victim Network Information: IP Addresses	SprySOCKS can collect information about the compromised device, including information about network interfaces and assigned IP addresses.
Resource Development	T1587.001	Develop Capabilities: Malware	FishMonger has developed custom malware for its operations, including the SprySOCKS backdoor.

Tactic	ID	Name	Description
Execution	T1059.003	Command and Scripting Interpreter: Windows Command Shell	SprySOCKS can launch an interactive cmd.exe command shell, which allows the attackers to execute commands remotely on the compromised machine.
	T1053.005	Scheduled Task/Job: Scheduled Task	SprySOCKS uses a scheduled task to execute its loader on system start.
	T1569.002	System Services: Service Execution	SprySOCKS abuses system services for both one-time and persistent execution.
	T1106	Native API	FishMonger has used Windows APIs to execute code within a victim's system.
Persistence	T1547.012	Boot or Logon Autostart Execution: Print Processors	To achieve persistence, FishMonger installs its malicious loader as a print processor.
Privilege Escalation	T1546.012	Event Triggered Execution: Image File Execution Options Injection	SprySOCKS can install itself as a debugger for the Virtual Disk Service by modifying HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\vds.exe\debugger.

Tactic	ID	Name	Description
Stealth	T1205.002	Traffic Signaling: Socket Filters	SprySOCKS uses the RawWNPF kernel driver to install packet filters capable of redirecting any inbound TCP traffic to the configured local port if a special magic value is detected in the packet.
	T1134.002	Access Token Manipulation: Create Process with Token	FishMonger uses CreateProcessAsUser to execute a new process with a token obtained from the print spooler service.
	T1622	Debugger Evasion	SprySOCK's RawWNPF driver uses the KdDisableDebugger function to disable the kernel debugger, if active.
	T1140	Deobfuscate/Decode Files or Information	SprySOCKS loader decrypts the SprySOCKS backdoor from an encrypted file. Additionally, most of the strings in the SprySOCKS components are encrypted.
	T1070.004	Indicator Removal: File Deletion	The SprySOCKS loader removes original files from the deployment directory after copying them and setting up persistence.
	T1070.009	Indicator Removal: Clear Persistence	SprySOCKS loader removes a service registry value associated with the previously installed malicious minifilter driver after executing the driver.
	T1027.007	Obfuscated Files or Information: Dynamic API Resolution	SprySOCKS components use dynamic API resolution.

Tactic	ID	Name	Description
	T1027.013	Obfuscated Files or Information: Encrypted/Encoded File	SprySOCKS components are stored in an AES-encrypted file on the victim's drive.
	T1055.013	Process Injection: Process Doppelganging	The SprySOCKS loader uses process doppelganging to inject the backdoor into the svchost.exe process.
	T1014	Rootkit	FishMonger uses the RawWNPF kernel driver, which serves as a rootkit responsible for hiding the SprySOCKS malicious activity.
	T1497	Virtualization/Sandbox Evasion	SprySOCKS uses several anti-emulation techniques to prevent automated analysis by emulators or sandboxes.
	T1574.002	Hijack Execution Flow: DLL Side-Loading	FishMonger uses DLL side-loading to execute the SprySOCKS backdoor.
Defense Impairment	T1562.004	Disable or Modify System Firewall	SprySOCKS adds a firewall rule allowing any inbound traffic sent to the backdoor's listening port.

Tactic	ID	Name	Description
Discovery	T1010	Application Window Discovery	SprySOCKS retrieves the active foreground window name as a part of its keylogging functionality.
	T1083	File and Directory Discovery	SprySOCKS can obtain file and directory listings from the compromised system.
	T1518.001	Software Discovery: Security Software Discovery	SprySOCKS components check for the presence of security and sandboxing product libraries (snxhk.dll, SxWrapper.dll, SxIn.dll, SXIn64.dll, SbiefDll.dll, and cmdvrt32.dll) in their own processes.
	T1082	System Information Discovery	SprySOCKS can collect information about the compromised device, including: computer name, OS version, information about memory and CPU, current privileges, system language and version, current time, and more.
	T1614.001	System Location Discovery: System Language Discovery	SprySOCKS can collect information about the compromised device, including system language.
	T1007	System Service Discovery	SprySOCKS can enumerate all services on the system.
	T1124	System Time Discovery	SprySOCKS can collect information about the compromised device, including current system time.
Collection	T1056.001	Input Capture: Keylogging	SprySOCKS implements a keylogger.
	T1115	Clipboard Data	SprySOCKS logs clipboard data, along with the captured keystrokes, as a part of its keylogging functionality.

Tactic	ID	Name	Description
Command and Control	T1132.001	Data Encoding: Standard Encoding	SprySOCKS uses base64 encoding in its custom C&C communication protocol.
	T1573.001	Encrypted Channel: Symmetric Cryptography	SprySOCKS encrypts data sent to, and decrypts data received from, the C&C with 128-bit AES.
	T1008	Fallback Channels	In addition to the TCP communication channel, SprySOCKS can contact its C&C using UDP and WebSocket channels.
	T1665	Hide Infrastructure	SprySOCKS's RawWNPF driver hides the backdoor's active connections from being enumerated when using network tools such as netstat.exe.
	T1571	Non-Standard Port	SprySOCKS uses nonstandard ports to communicate with the C&C.
	T1095	Non-Application Layer Protocol	SprySOCKS uses nonstandard protocols to communicate with the C&C.
Exfiltration	T1041	Exfiltration Over C2 Channel	SprySOCKS can upload various files from the compromised system to the C&C.

