

# Thor vs. Silver Fox – Uncovering and Defeating a Sophisticated ValleyRat Campaign

**N** [nexttron-systems.com/2025/11/28/thor-vs-silver-fox-uncovering-and-defeating-a-sophisticated-valleyrat-campaign](https://nexttron-systems.com/2025/11/28/thor-vs-silver-fox-uncovering-and-defeating-a-sophisticated-valleyrat-campaign)

Maurice Fielenbach

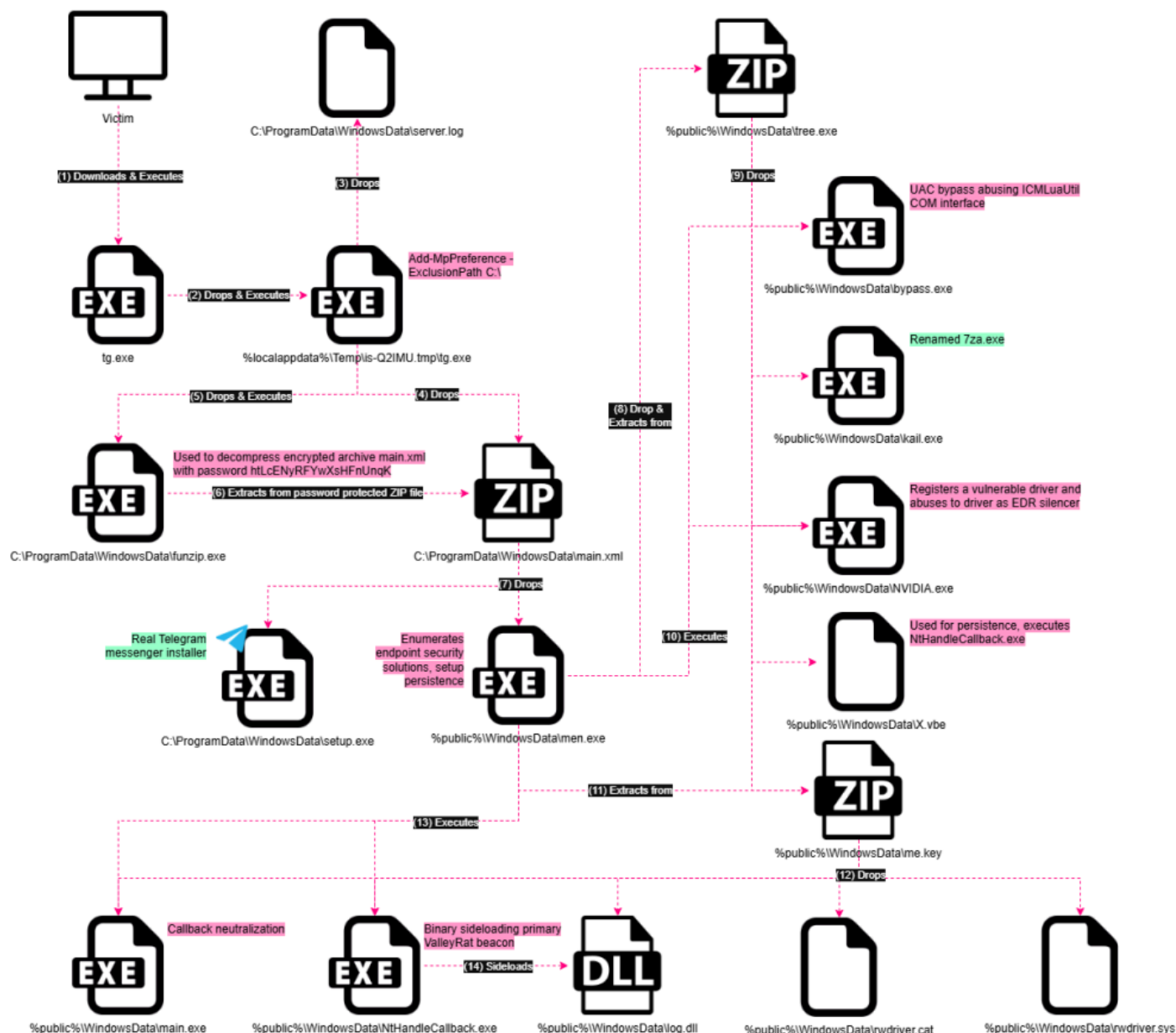
November 28, 2025



Recently, we investigated a highly sophisticated malware campaign that combines multiple layers of obfuscation, endpoint security tampering, and kernel-level tricks. The operators hide behind repackaged installers for popular tools such as Telegram, WinSCP, Google Chrome, and Microsoft Teams. On the surface, victims see a normal installer. In the background, the malware stages files, deploys drivers, tampers with defenses, and finally launches a ValleyRat beacon that keeps long-term access to the system.

Our analysis attributes the described operation to the China-aligned APT commonly referred to as Silver Fox, active since at least 2022. The campaign matches previous Silver Fox activity in its use of archive-based staging, DLL sideloading, abuse of Chinese security products, and Bring Your Own Vulnerable Driver (BYOVD) techniques.

This post follows the full infection chain from the first malicious installer to the final ValleyRat payload and then focuses on detection and hunting strategies. THOR's role is highlighted throughout, as it detects several key stages out of the box and provides strong leads for incident responders.



Silver Fox Telegram installer infection chain

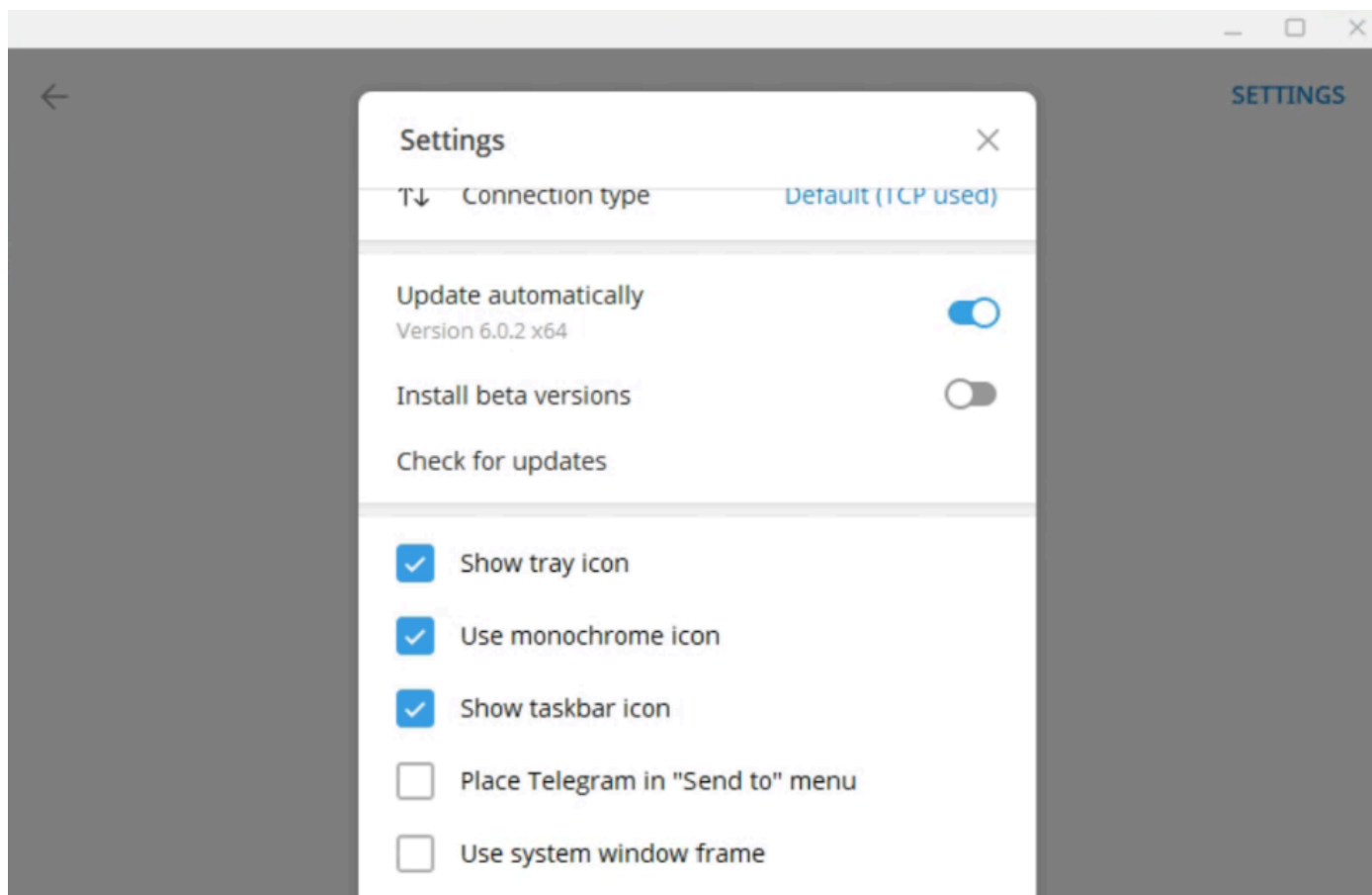
## Infection Chain Overview

At a high level, the infection chain proceeds through a series of well-defined phases. The victim first executes a trojanized Telegram installer. This installer sets a dangerous Microsoft Defender exclusion, stages a password-protected archive together with a renamed 7-Zip binary, and then extracts a second-stage executable. That second-stage orchestrator, `men.exe`, deploys additional components into a folder under the public user profile, manipulates file permissions to resist cleanup, and sets up persistence through a scheduled task that runs an encoded VBE script. This script in turn launches a vulnerable driver loader and a signed binary that sideloads the ValleyRat DLL.

The following sections take each stage in turn and examine what happens on the system.

## A Telegram Installer That Isn't

The campaign begins with a Windows installer named `tg.exe`, with SHA-256 hash `9ede6da5986d8c0df3367c395b0b3924fffb12206939f33b01610c1ae955630d1`. The binary is unsigned, approximately 49.7 MB in size, and uses a Telegram icon. When executed, it displays a genuine-looking Telegram Desktop installer interface, and its settings identify it as Telegram Desktop 6.0.2 (x64), the then-current version.



Telegram UI showing Version 6.0.2

Closer analysis reveals some inconsistencies. The PE header timestamp is 2019-02-11 18:13:16 UTC, whereas the first VirusTotal submission occurred on 2025-08-29 08:13:04 UTC. A six-year gap between compilation and first public sighting is atypical for a mainstream, frequently updated application. At the time of analysis, only a small number of antivirus engines on VirusTotal flagged the file as malicious.

5

/ 71

Community Score

5/71 security vendors flagged this file as malicious

Follow

Reanalyze

Download

Similar

More

9ede6da5986d8c0df3367c395b0b3924ffb12206939f33b01610c1ae955630d1

Size

49.72 MB

Last Analysis Date

21 hours ago

EXE

tg.exe

peexe

overlay

malware

DETECTION

DETAILS

RELATIONS

BEHAVIOR

CONTENT

TELEMETRY

COMMUNITY

Crowdsourced Sigma Rules

Crowdsourced IDS rules

Security vendors' analysis on 2025-08-29T08:13:04 UTC

|                |                         |                     |                         |
|----------------|-------------------------|---------------------|-------------------------|
| Avast          | Other:Malware-gen [Trj] | AVG                 | Other:Malware-gen [Trj] |
| Bkav Pro       | W32.AIDetectMalware     | Malwarebytes        | Malware.AI.160653840    |
| McAfee Scanner | TII9EDE6DA5986D         | Acronis (Static ML) | Undetected              |
| AhnLab-V3      | Undetected              | Alibaba             | Undetected              |

VirusTotal overview of tg.exe

Distribution is achieved via spear-phishing and malvertising. Victims are pushed toward download links or attachments that promise the Telegram installer.

## Staging in ProgramData and Silencing Defender

Once the victim executes `tg.exe`, the installer immediately begins to prepare the environment for further payloads. First, it creates the directory `C:\ProgramData\WindowsData\` and drops several files there. Among those files are `funzip.exe`, which is a renamed 7-Zip command-line binary, and `main.xml`. The latter name suggests a configuration file, but the file's magic bytes clearly identify it as a password-protected ZIP archive. Its high entropy further supports that classification.

Second, the installer uses PowerShell to add a Microsoft Defender exclusion for the entire `C:\` drive.

EIP

76AED51D

CC

int3

76AED51E

CC

int3

76AED51F

CC

int3

76AED520

8BFF

mov edi,edi

76AED521

55

push ebp

76AED522

8BEC

mov ebp,esp

76AED523

5D

pop ebp

76AED524

FF25 C418B576

jmp dword ptr ds:[<CreateProcess>]

76AED525

CC

int3

76AED526

CC

int3

76AED527

CC

int3

76AED528

CC

int3

76AED529

CC

int3

76AED52A

CC

int3

76AED52B

CC

int3

76AED52C

CC

int3

76AED52D

CC

int3

76AED52E

CC

int3

76AED52F

CC

int3

76AED530

CC

int3

76AED531

CC

int3

76AED532

CC

int3

76AED533

CC

int3

76AED534

CC

int3

76AED535

CC

int3

76AED536

CC

int3

76AED537

CC

int3

76AED538

CC

int3

76AED539

CC

int3

76AED53A

CC

int3

76AED53B

CC

int3

76AED53C

CC

int3

76AED53D

CC

int3

76AED53E

CC

int3

76AED53F

CC

int3

76AED540

8BFF

mov edi,edi

76AED541

55

push ebp

76AED542

8BEC

mov ebp,esp

76AED543

56

push esi

76AED544

8B35 A80FB876

mov esi,dword ptr ds:[76B80FA8]

76AED545

85F6

test esi,esi

76AED546

0F85 C0F50000

jne kernel132.76AFCB14

76AED547

33C0

xor eax,eax

76AED548

5C

pop esi

76AED549

5D

pop ebp

76AED54A

C2 0800

ret 8

76AED54B

CC

int3

76AED54C

CC

int3

76AED54D

CC

int3

76AED54E

CC

int3

76AED54F

CC

int3

76AED550

CC

int3

76AED551

CC

int3

CreateProcessW

JMP.<CreateProcessW>

TermrvDeleteValue

EAX

00000000

EBX

00000000

ECX

0255E0C0

EDX

0019FA9C

EBP

0019FAA8

ESP

0019FA54

ESI

00000000

EDI

0255E0C0

EIP

76AED520

<kernel132.CreateProcess>

EFLAGS

00000344

ZF

1

PF

1

AF

0

OF

0

SF

0

DF

0

CF

0

TF

1

IF

1

LastError

00000000 (ERROR\_SUCCESS)

LastStatus

C0150008 (STATUS\_SXS\_KEY\_NOT\_FOUND)

GS

002B

FS

0053

ES

002B

DS

002B

CS

0023

SS

002B

ST(0)

FFFF0000000000000000

x87r0

Empty

Invalid

ST(1)

4034B8000017000C800

x87r1

Empty

12947848934719538.00

ST(2)

4034B800000088019400

x87r2

Empty

9007199256869317.000

ST(3)

4035F000CA005C000800

x87r3

Empty

33774309999990380.00

ST(4)

4035D800CA000000E600

x87r4

Empty

30399731283263603.00

ST(5)

4034CC01840194010000

x87r5

Empty

14355691970297972.00

ST(6)

4034C800CC01B4019400

x87r6

Empty

14073967886008421.00

ST(7)

4035AE00880074008600

x87r7

Empty

24488718114619459.00

x87TagWord

FFFF

x87TW\_0

3 (Empty)

x87TW\_1

3 (Empty)

x87TW\_2

3 (Empty)

x87TW\_3

3 (Empty)

x87TW\_4

3 (Empty)

x87TW\_5

3 (Empty)

x87TW\_6

3 (Empty)

x87TW\_7

3 (Empty)

PowerShell command adding a Defender exclusion for the entire C:\ drive

It then extracts files from the ZIP archive `main.xml` by executing

```
"C:\ProgramData\WindowsData\funzip.exe" x -y -phtLcENyRFYwXsHFnUnqK -o"C:\ProgramData\WindowsData" "C:\ProgramData\WindowsData\main.xml".
```

The arguments mirror 7-Zip-style semantics: `x` extract, `-y` assume yes, `-p<password>`, and `-o<output>`. One artifact dropped out particularly: the password `htLcENyRFYwXsHFnUnqK`. Extraction results in two further files written to `C:\ProgramData\WindowsData\` which are `men.exe` and `setup.exe`, a legitimate Telegram version 6.0.2 installer. Both are finally executed.

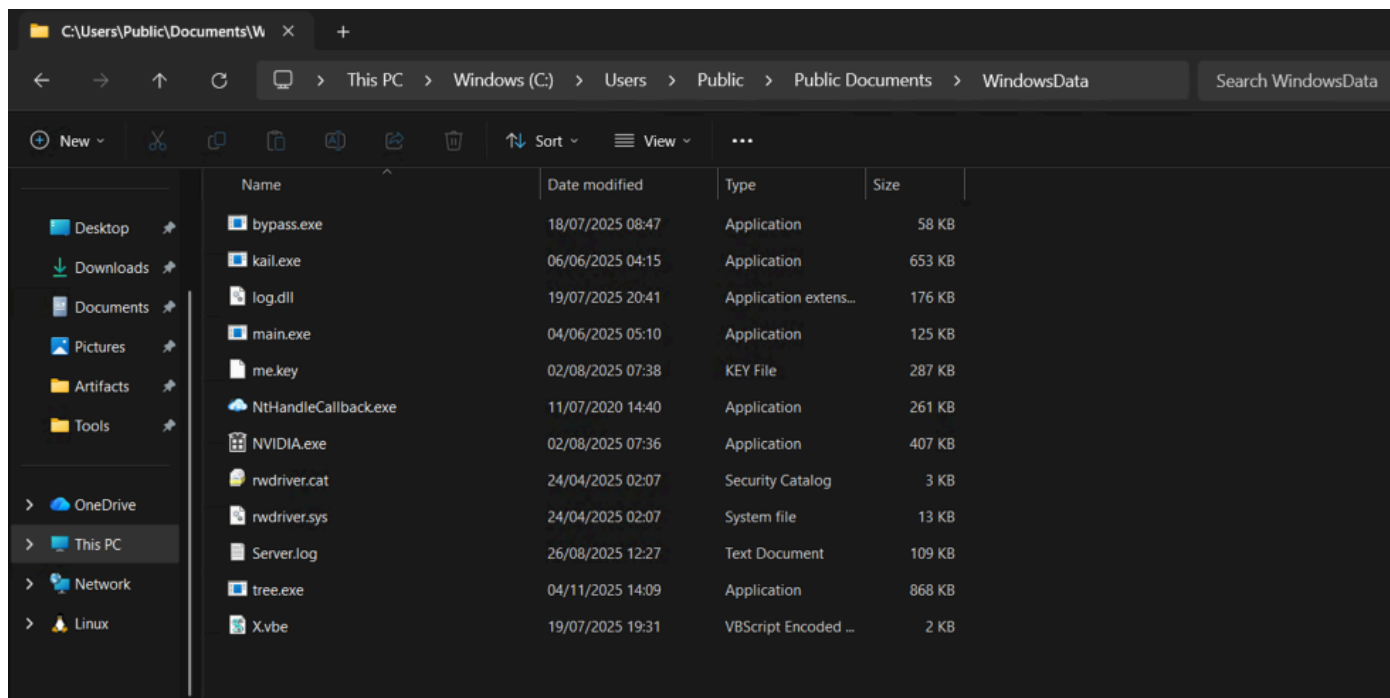
## men.exe Orchestrating the Infection

---

`men.exe` is the main control component for this campaign. Once running, it performs environmental reconnaissance, deploys additional files, establishes persistence, and coordinates driver loading and ValleyRat execution.

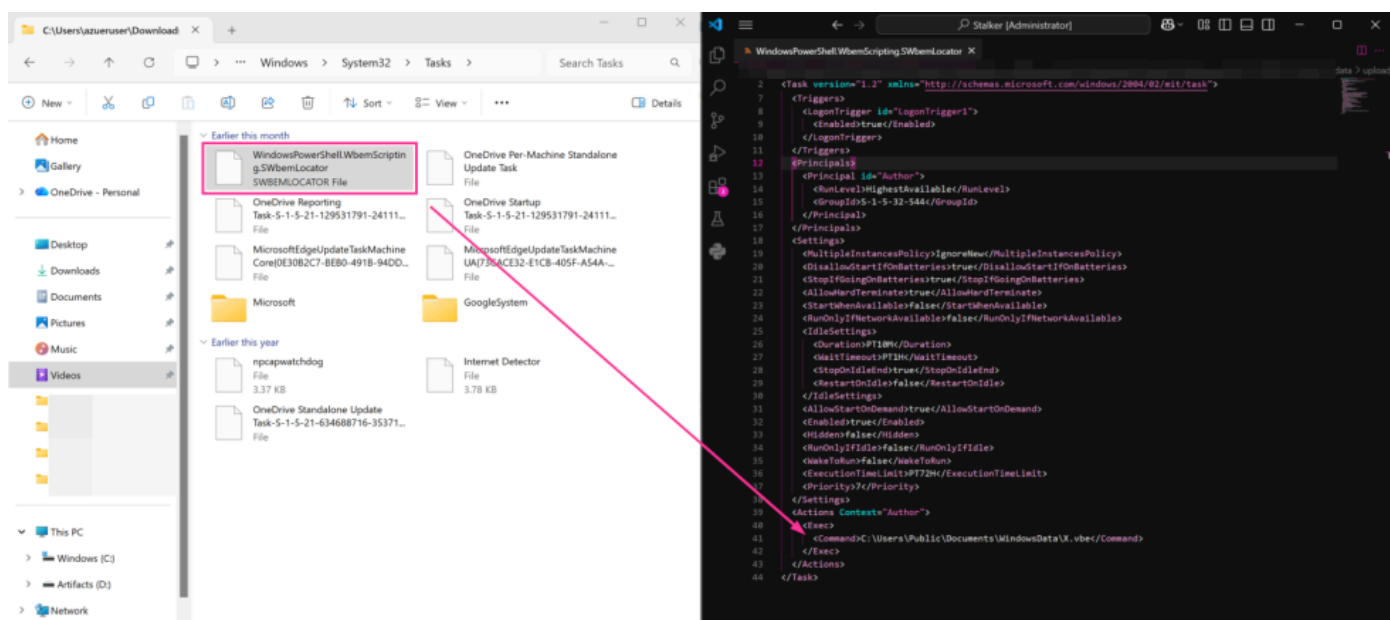
As an initial step, `men.exe` enumerates running processes to identify endpoint security related processes. This includes Microsoft Defender's `MsMpEng.exe` and processes belonging to Chinese security products such as `ZhuDongFangYu.exe` and `360tray.exe`. This behaviour confirms both awareness of installed defenses and a focus on regions where these products are prevalent. Similar campaigns and strategies have been observed in the past as well.

`men.exe` then drops additional password-protected archives with misleading file extensions `tree.exe` and `me.key` into `%PUBLIC%\Documents\WindowsData\` and decrypts them at runtime using embedded passphrases (`Server8888` and `killstartup`). After extraction, this directory contains a collection of binaries and data files including both encrypted archives as well: `tree.exe`, `bypass.exe`, `kail.exe`, `me.key`, `X.vbe`, `NVIDIA.exe`, `log.dll`, `main.exe`, `NtHandleCallback.exe`, `rwdriver.sys`, and `rwdriver.cat`. Each file serves a particular role in the chain, ranging from further decompression and privilege escalation to driver deployment and loader functionality.



Public directory contents after second stage execution

**men.exe** also prepares long-term persistence and makes remediation harder by creating a scheduled task named **WindowsPowerShell.WbemScripting.SWbemLocator**, a name carefully chosen to resemble legitimate Windows components. The task triggers at user login, runs with the Administrator principal and highest available privileges, and executes the encoded script file **X.vbe** using the configured default VBE handler, in our case **WScript.exe**. At the same time, **men.exe** modifies the access control list on **%PUBLIC%\Documents\WindowsData\** to include a deny entry that prevents even administrators from deleting files and directories under that path.



Created scheduled task WindowsPowerShell.WbemScripting.SWbemLocator executing X.vbe

## Components and Capabilities

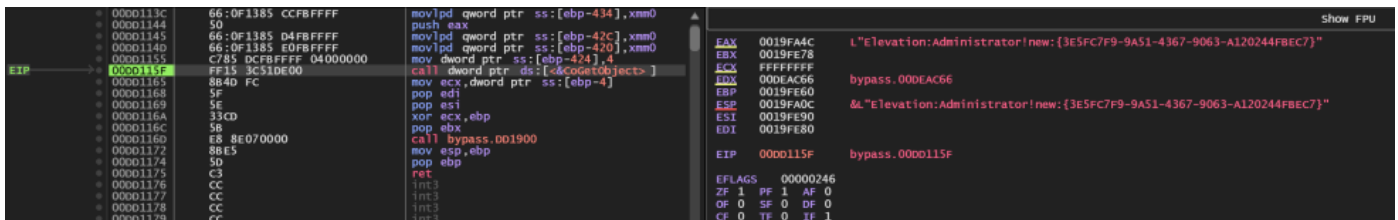
The components dropped by `men.exe` implement the key capabilities required for this campaign. The most relevant ones for defenders are summarized here.

### UAC Bypass

`bypass.exe` provides UAC bypass and privilege escalation. It performs runtime unpacking by overwriting one of its sections in memory, complicating static analysis, and abuses the `ICMLuaUtil` elevated COM interface, part of the long-standing `CMSTPLUA` UAC COM class bypass family. The code uses elevation monikers referencing CLSIDs such as

- `3E5FC7F9-9A51-4367-9063-A120244FBEC7`,
- `6EDD6D74-C007-4E75-B76A-E5740995E24C`,
- `D2E7041B-2927-42FB-8E9F-7CE93B6DC937`,
- `0A16D195-6F47-4964-9287-9F4BAB6D9827`,

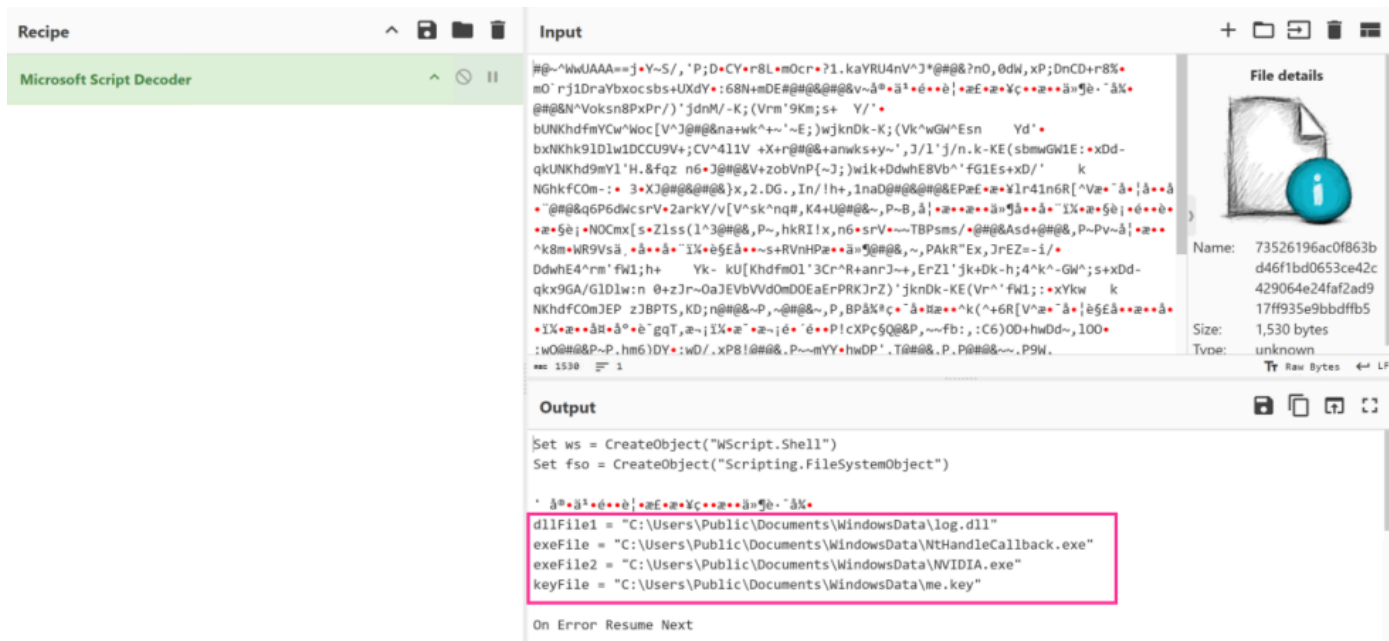
and ultimately launches `C:\Windows\SysWOW64\DllHost.exe /Processid:{<CLSID>}`.



Elevation moniker creation and execution

### Persistence via Scheduled Task (X.vbe)

`X.vbe` is an encoded VBScript and forms the main persistence mechanism. The scheduled task `WindowsPowerShell.WbemScripting.SwbemLocator` executes this script at logon via the configured default VBE handler. After decoding the file contents, the script contains numerous Chinese-language strings and comments and is responsible for launching both `NVIDIA.exe` and `NtHandleCallback.exe`.



X.vbe decoded with CyberChef showing references to NVIDIA.exe and NtHandleCallback.exe

## Callback Manipulation and Logging Interference

`main.exe` manages the custom driver `rwdriver.sys` (with catalog `rwdriver.cat`). The driver is used to circumvent user-mode hooks employed by EDR products and to interfere with certain logging mechanisms, for example by overloading log forwarding pipelines so that events never reach a SIEM. It also ensures that disabled network interfaces are re-enabled when needed, supporting both command-and-control and lateral movement. To install the driver, `main.exe` issues a command equivalent to `sc create rwdriver binPath="C:\Users\Public\Documents\WindowsData\rwdriver.sys" type=kernel start=demand`.

## BYOVD Resulting in Security Process Termination

`NVIDIA.exe` is a loader for the vulnerable but signed driver `NSecKrn164.sys`. It drops the driver into `%LOCALAPPDATA%\Temp` under a random name and loads it as a kernel-mode driver, granting `ring 0` access. This driver exposes an IOCTL interface, using code `0x2248E0`, that allows arbitrary process termination, which the malware uses to repeatedly kill security solution processes.

```

40 while ( 1 )
41 {
42     i = 0;
43     pProcessArray = processList;
44     do
45     {
46         currentProcess = *pProcessArray;
47         th32ProcessID = 0;
48         memset(&pe.cntUsage, 0, 300);
49         pe.dwSize = 304;
50         Toolhelp32Snapshot = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
51         Toolhelp32SnapshotReturn = Toolhelp32Snapshot;
52         if ( Toolhelp32Snapshot != (HANDLE)-1LL )
53         {
54             if ( Process32First(Toolhelp32Snapshot, &pe) )
55             {
56                 do
57                 {
58                     th32ProcessID = pe.th32ProcessID;
59                     if ( !(unsigned int)CompareExeName((unsigned __int8 *)pe.szExeFile, (unsigned __int8 *)currentProcess, v9) )// On match, captures PID and closes the snapshot
60                         goto processTermination;
61                 }
62                 while ( Process32Next(Toolhelp32SnapshotReturn, &pe) );
63                 CloseHandle(Toolhelp32SnapshotReturn);
64             }
65             else
66             {
67 processTermination:
68                 CloseHandle(Toolhelp32SnapshotReturn);
69                 if ( th32ProcessID )
70                 {
71                     InBuffer = th32ProcessID;
72                     DeviceIoControl(g_DriverHandle, 0x2248E0u, &InBuffer, 4u, 0, 0, (LPDWORD)&BytesReturned, 0);// Terminate process
73                 }
74             }
75             ++i;
76             ++pProcessArray; // Next image name in list
77         }
78     } while ( i < 20 ); // Iterates a fixed list of 20 target image names
79     Sleep(1000u); // Sleeps 1000 ms between full passes.
80 }
81 }

```

Process Termination loop in NVIDIA.exe using IOCTL 0x2248E0

The driver is then registered under `HKLM\SYSTEM\CurrentControlSet\Services\<RandomName>`.

Finally, `NtHandleCallback.exe` is a legitimately signed executable abused for DLL sideloading. Placed together with a malicious `log.dll` in the same directory, and executed without an explicit DLL path, it loads `log.dll` from its working directory in line with the Windows DLL search order. In this campaign, `log.dll` implements the ValleyRat beacon and establishes command and control.

At that point, ValleyRat is running with persistence backed by the VBE script and with two drivers working to weaken defenses and logging.

## Detection and Prevention

This malware infection chain is complex, multi-layered and still active. We continue to observe infected hosts, not only via trojanized Telegram installers but also via other repackaged productivity tools. The good news is that even highly advanced campaigns like this one offer numerous, reliable detection opportunities throughout the entire chain.

In this chapter, we walk step by step through the infection chain and map concrete detection options to each phase. Where useful, we include Sigma-style examples. Throughout the chapter, you will see that this campaign can be uncovered with relatively simple custom rules and is also comprehensively covered by THOR.

## Malvertising and Phishing Leading to Infection

---

The infection chain starts with very traditional initial access vectors: phishing and malvertising. Users are lured to download and execute fake installers for Telegram, Chrome, WinSCP, Microsoft Teams and other popular tools.

At this stage, detection is mostly about hygiene and policy rather than specific signatures. Ideally, local administrator rights are tightly restricted so that users cannot install arbitrary software in the first place. Application control technologies such as AppLocker or equivalent solutions should block unknown or unsigned installers by default.

From a hunting perspective, EDR or inventory systems can be queried for unsigned or oddly signed installers whose product name or description claims to be Telegram, Teams or similar but whose publisher does not match the vendor. THOR can support this by flagging suspicious PE metadata and unsigned installers in high-value locations.

## First-Stage Execution

---

The first clearly observable host-side indicator is the creation and use of the directory `C:\ProgramData\WindowsData\`. Under normal circumstances, this folder does not exist. Its creation, combined with executable content inside it, is therefore a valuable early hunting lead.

By default, writing into `C:\ProgramData` requires administrative privileges, which already tells you something about the user context in which the initial dropper runs. In this campaign, the trojanized installer `tg.exe` stages the orchestrator `men.exe` there and later deletes that executable. Detection based purely on the presence of `men.exe` is therefore unreliable. However, the existence of `C:\ProgramData\WindowsData\` containing only a single “legitimate” installer such as Telegram, Chrome, WinSCP or Microsoft Teams is a strong anomaly that should trigger further investigation.

To capture these events, file system activity must be logged. This can be achieved with Sysmon event ID 11 (file creation) or via native file system auditing combined with an appropriate SACL on `C:\ProgramData\WindowsData\` and usage of a detection rule such as the following example:

```
title: File Creation In WindowsData ProgramData Folder
description: Detects file creation events in C:\ProgramData\WindowsData
detection:
  selection:
    EventID: 11
    TargetFilename|startswith: 'C:\\ProgramData\\WindowsData\\'
  condition: selection
```

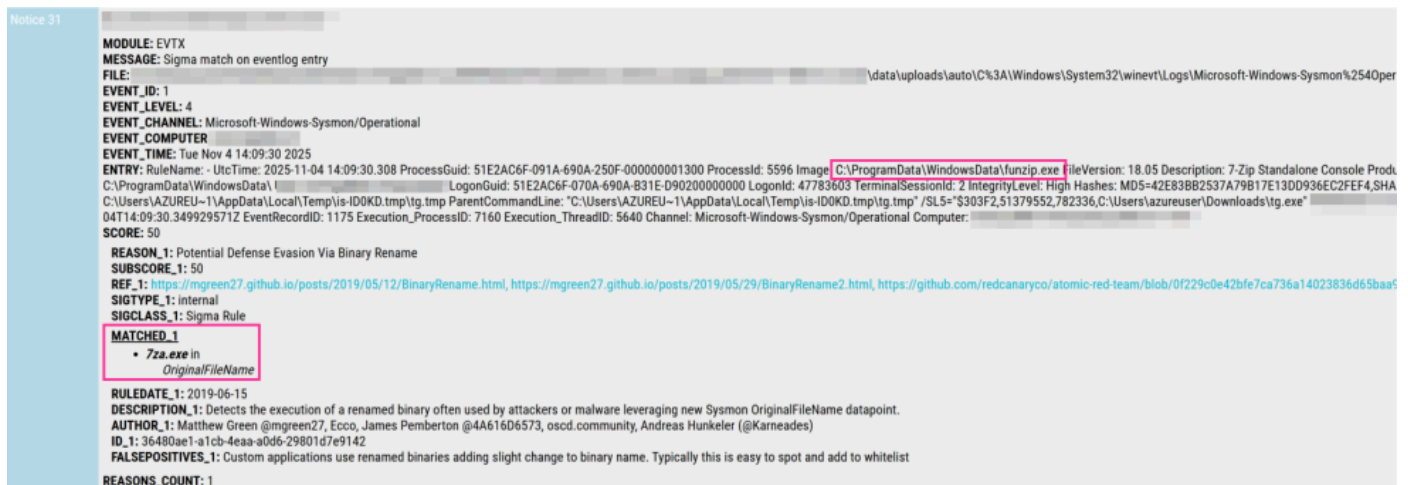
THOR uses this kind of telemetry to flag suspicious content in that directory and to correlate it back to its parent processes.

## Second-Stage Extraction

Before dropping the second stage files from the encrypted `main.xml` archive, `tg.exe` modifies Microsoft Defender for Antivirus configuration and adds a path exclusion for the entire system drive `c:\`. This change is logged as Microsoft Defender Operational event ID 5007. While there are legitimate use cases for exclusions, broad exclusions such as a full drive are extremely suspicious and should be closely monitored. THOR detects this kind of Defender configuration tampering out of the box and will highlight it as a high severity finding. SigmaHQ maintains a public rule that also covers this scenario:

[https://github.com/SigmaHQ/sigma/blob/43b6fae2a001d3af9a53984167dd39d2319fbb40/rules/windows/builtin/windefend/win\\_defender\\_config\\_change\\_exclusion\\_added.yml](https://github.com/SigmaHQ/sigma/blob/43b6fae2a001d3af9a53984167dd39d2319fbb40/rules/windows/builtin/windefend/win_defender_config_change_exclusion_added.yml).

Once the exclusion is in place, the initial installer extracts the second stage from `main.xml` using the password `htLcENyRFYwXsHFUnqK`. This string is a strong hunting artifact and can be searched in process creation logs such as Sysmon event ID 1 or Security event ID 4688. In addition, THOR heuristics detect `funzip.exe` as a renamed 7-Zip component based on the original file name and metadata.



THOR detecting the renamed 7za.exe including the whole command line

A generic Sigma rule to detect password-protected archive extraction via 7-Zip utilities could look like this:

title: Password Protected Compressed File Extraction Via 7-Zip  
description: Detects usage of 7zip utilities (7z.exe, 7za.exe and 7zr.exe) to extract password protected zip files.  
logsource:

category: process\_creation  
product: windows

detection:

selection\_img:  
- Description|contains: '7-Zip'  
- OriginalFileName:  
- '7za.exe'

selection\_password:  
CommandLine|contains|all:  
- ' -p'  
- ' x '  
- ' -o'  
- ' -y'

condition: all of selection\_\*

## Public Folder Staging (%PUBLIC%\Documents\WindowsData)

---

The use of %PUBLIC%\Documents\WindowsData\ as a staging directory is one of the strongest anomalies in this campaign. The public user profile is intended for shared user data, not for executables and drivers. Under normal circumstances, one should not observe .exe, .dll, .sys or .vbe files created and executed from this location.

Tracking file creation under %PUBLIC%\Documents\WindowsData\ using Sysmon event ID 11 or file system auditing SACLs provides high-value hunting leads. THOR detects this stage of the infection chain in several ways. It first flags the types of files located in %PUBLIC%\Documents\WindowsData\, then detects the files themselves using YARA and heuristic rules that cover suspicious PE signatures, known malicious or suspicious signers, typical password-protected ZIP artefacts and encoded VBE script content. Among others, the following THOR rule families are triggered in our testing:

- EXT\_SUSP\_PE\_Signed\_by\_Suspicious\_Entity\_Mar23
- [MAL\\_CobaltStrike\\_Apr18\\_1\\_CSharp\\_Powershell\\_Combo](#)
- SUSP\_ZIP\_PasswordProtected\_Content\_Phishing\_Sep22
- EXT\_Winnti\_Rootkit
- [SUSP\\_Encoded\\_VBE](#)



```

Warning Suspicious scheduled task found ENTRY: "NAME: WindowsPowerShell.WbemScripting.SwbemLocator PATH: \
\WindowsPowerShell.WbemScripting.SwbemLocator COMMAND: C:\\Users\\Public\\Documents\\WindowsData\\X.vbe" S
CORE: 60 NAME: WindowsPowerShell.WbemScripting.SwbemLocator PATH: \WindowsPowerShell.WbemScripting.SwbemLo
cator COMMAND: C:\\Users\\Public\\Documents\\WindowsData\\X.vbe ENABLED: true LASTRUN: Fri Nov 7 21:14:34 2025
NEXTRUN: Mon Jan 1 00:00:00 0001
REASON_1: Filename IOC \\[A-Za-z0-9]\\.(exe|com|dll|bat|scr|vbs|vbe|hta|lnk|iso|ps1|psm1|VBS|VBE|EXE|DLL)$
SUBSCORE_1: 60 REF_1: Typical file names used by lazy developers or threat actors 2024-09-30 Internal Rese
arch (SCRIPT, T1014) SIGTYPE_1: internal SIGCLASS_1: Filename IOC MATCHED_1: \\X.vbe
REASONS_COUNT: 1
FILE_1: C:\\Users\\Public\\Documents\\WindowsData\\X.vbe EXISTS_1: yes TYPE_1: VBE Encoded
SIZE_1: 1530
MD5_1: 04264646287bb028ad5280cf4da39358
SHA1_1: 9d779f88247b1a20194e1df02b2903a52eef32d6
SHA256_1: 73526196ac0f863bd46f1bd0653ce42c429064e24faf2ad917ff935e9bbdfffb5 FIRSTBYTES_1: 23407e5e577755414
1413d3d6a7f597e532f2c27 / #@~^WwUAAA==jY~S/,
CREATED_1: Sat Jul 19 19:32:11.000 2025 OWNER_1: BUILTIN\Administrators
Info Finished module DURATION: 0 hours 0 mins 27 secs SCANNED_ELEMENTS: 190

```

Persistence detection based on a one letter script file name

Creation of the scheduled task generates Security event ID 4698 and contains the full XML task definition. Building on this telemetry, you can use a dedicated correlation rule such as:

```

title: Scheduled Task Creation With Command In Users Public Directory
description: Detects creation of a scheduled task whose action command
points to \\Users\\Public
detection:
    selection:
        EventID: 4698
        TaskContent|contains:
            - '\\Users\\Public\\'
            - '\\\\Users\\\\Public\\\\'
    condition: selection

```

In a strictly controlled environment, any scheduled task creation not explicitly whitelisted can serve as an early investigation lead. The rule above can be easily extended with more suspicious directories if needed.

## UAC Bypass with ICMLuaUtil

The UAC bypass used in this campaign can be detected very reliably by its COM interface usage and CLSIDs. The technique abuses the `ICMLuaUtil` elevated COM interface, resulting in command lines like `C:\Windows\SysWOW64\DllHost.exe /Processid:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}`.

Both Security event ID 4688 and Sysmon event ID 1 can be used to detect such invocations by focusing on `dllhost.exe` combined with specific CLSIDs. The following Sigma rule provides an example:

```
title: UAC Bypass via ICMLuaUtil
id: 49f2f17b-b4c8-4172-a68b-d5bf95d05130
status: test
description: Detects the pattern of UAC Bypass using ICMLuaUtil Elevated COM interface
references:
  - https://www.elastic.co/guide/en/security/current/uac-bypass-via-icmluautil-elevated-com-interface.html
author: Florian Roth (Nextron Systems), Elastic (idea), Maurice Fielenbach (Hexastrike Cybersecurity)
date: 2022-09-13
modified: 2025-10-01
tags:
  - attack.defense-evasion
  - attack.privilege-escalation
  - attack.t1548.002
logsource:
  category: process_creation
  product: windows
detection:
  selection_proc:
    - ParentImage|endswith: '\dllhost.exe'
    - ParentProcessName|endswith: '\svchost.exe'
  selection_child_cmd:
    CommandLine|contains:
      - '/Processid:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}'
      - '/Processid:{D2E7041B-2927-42FB-8E9F-7CE93B6DC937}'
  selection_parent_cmd:
    ParentCommandLine|contains:
      - '/Processid:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}'
      - '/Processid:{D2E7041B-2927-42FB-8E9F-7CE93B6DC937}'
  filter_werfault:
    NewProcessName|endswith: '\WerFault.exe'
  condition: selection_proc and (selection_child_cmd or selection_parent_cmd) and
not filter_werfault
falsepositives:
  - Unknown
level: high
```

THOR detects this behaviour through its process creation analysis and specific heuristics for UAC bypass techniques, including **CMSTPLUA/ICMLuaUtil**.

## Driver Registration and Loading

---

Driver loading activities are relatively rare and therefore stand out when they deviate from expected paths. New drivers are normally registered from

**SystemRoot\System32\drivers\Name.sys** (expandable form) or **\??**

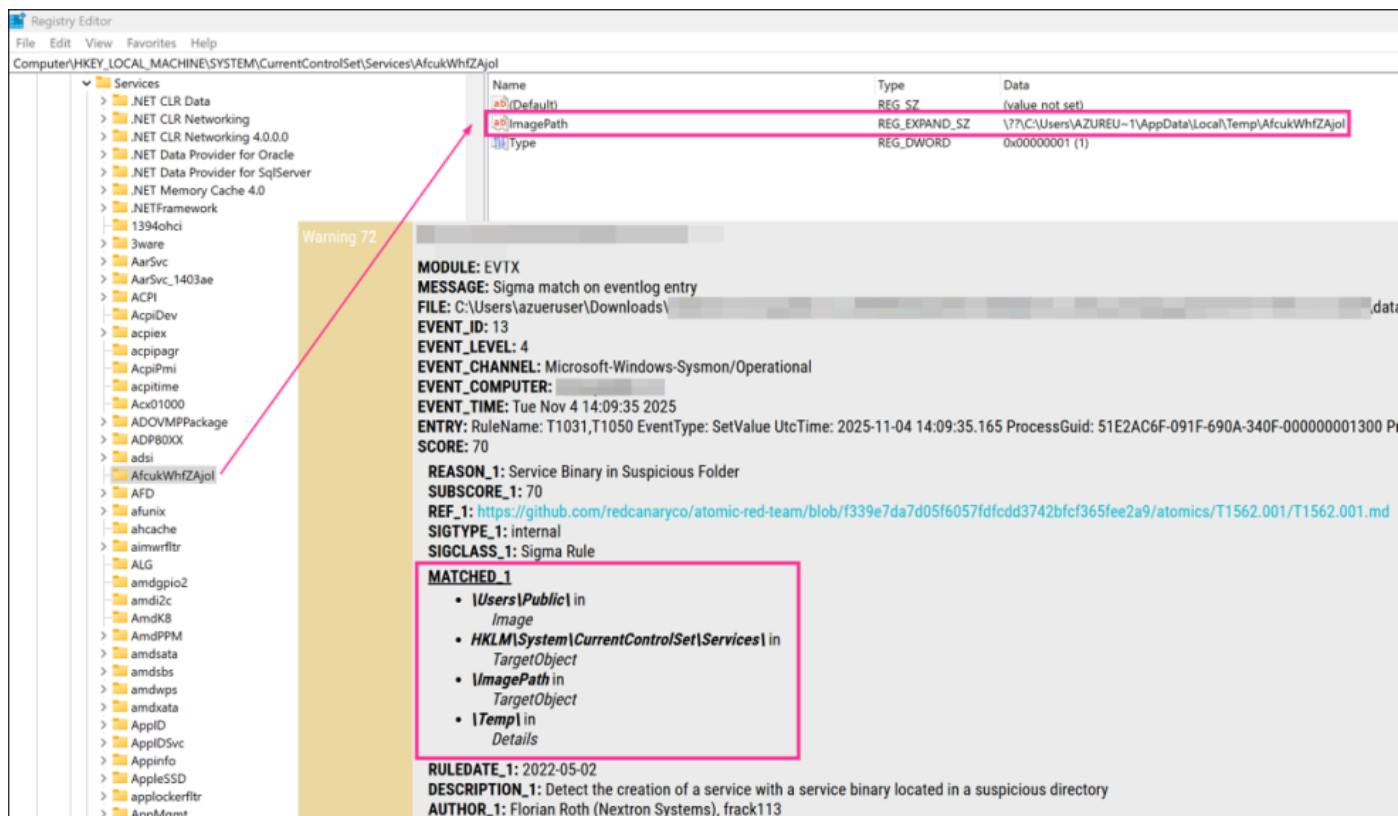
**\C:\Windows\System32\drivers\Name.sys** (native form). Any driver path outside these locations is suspicious. Loading a driver from **%PUBLIC%** or temporary directories is particularly indicative of malicious activity.

During the infection chain, multiple drivers are used and registered. One of them, `C:\Users\Public\Documents\WindowsData\rwdriver.sys`, is installed as a kernel service and generates Security event ID 4697, “A service was installed in the system”. Service installations from suspicious locations can be detected with a simple Sigma rule such as:

```
title: Service Creation In Users Public Directory
description: Detects creation of a service running from \Users\Public
detection:
  selection:
    EventID: 4697
    ServiceFileName|contains:
      - '\\Users\\Public\\'
      - '\\\\Users\\\\Public\\\\'
  condition: selection
```

Beyond service creation events, registry modifications under `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\` provide additional detection opportunities. Sysmon event ID 13 (registry value set), or native registry auditing combined with a SACL on that key, generates Security events 4663 followed by 4657. By correlating these events with suspicious file paths in `%PUBLIC%` or `%LOCALAPPDATA%\Temp`, defenders can quickly uncover driver-based infection stages.

THOR detects these drivers via YARA signatures (for example, `EXT_Winnti_Rootkit` in related tooling), driver location checks and generic service analysis, and clearly flags kernel services loaded from user-writable directories.



Driver registration showing up as service in the Registry

## C2 Activity

The final stage of the campaign is the ValleyRat command-and-control activity. In our analysis, the C2 endpoint used was the domain `keepmasterr[.]com` with traffic observed to `161[.]248[.]15[.]144[:]9000`. Connections can be inspected with tools such as `netstat -naob`, but more importantly, THOR's `ProcessCheck` module highlights the responsible process. In this case, THOR points directly to `NtHandleCallback.exe`, which is signed but suspicious and detected by the YARA rule `EXT_SUSP_PE_Signed_by_Suspicious_Entitiy_Mar23`, based on the file signature. From there, analysts can pivot into memory analysis or process dumping to recover the sideloaded beacon `log.dll`.

```

> 1/1 > Running module 'ProcessCheck'
Info Starting module
Warning Possibly Dangerous file found SCORE: 60
FILE: C:\Users\Public\Documents\WindowsData\NtHandleCallback.exe EXT: .exe TYPE: EXE
SIZE: 266704
MD5: bcec4279d95d0b09ebcf4388cb9a4f64
SHA1: fbd8966926b4597049c115f5d5eb89659a5df213
SHA256: c027cf868757babab33686bf4c41192339e04fa89ad868409a5cd4ed90a1f71e FIRSTBYTES:
4d5a90000300000004000000ffff0000b8000000 / MZ
CREATED: Thu Jun 19 20:03:20.000 2025 MODIFIED: Sat Jul 11 14:40:26.000 2020 ACCESSE
D: Fri Nov 7 21:35:03.746 2025 PERMISSIONS: BUILTIN\Administrators:F / NT AUTHORITY
\BATCH:C / NT AUTHORITY\INTERACTIVE:C / NT AUTHORITY\SERVICE:C / NT AUTHORITY\SYSTEM
:F OWNER: BUILTIN\Administrators COMPANY: 杭州顺网科技股份有限公司 DESC: APlus Modul
e LEGAL_COPYRIGHT: Copyright (C) 2015 PRODUCT: 网维大师客户端 ORIGINAL_NAME: startbc
.exe INTERNAL_NAME: APlus IMPHASH: a5319769a542e803ada192c4e7434bda
REASON_1: YARA rule EXT_SUSP_PE_Signed_by_Suspicious_Entitiy_Mar23 / Find driver sig
ned by suspicious company (see references) SUBSCORE_1: 60 REF_1: https://www.mandian
t.com/resources/blog/hunting-attestation-signed-malware SIGTYPE_1: internal SIGCLASS
_1: YARA Rule MATCHED_1: Str1: "Hangzhou Shunwang Technology Co.,Ltd" in "N1\x110\x0
f\x06\x03U\x04\x08\x13\x08Zhejiang1\x110\x0f\x06\x03U\x04\x07\x13\x08Hangzhou1-0+\x0
6\x03U\x04\x0a\x13$Hangzhou Shunwang Technology Co.,Ltd1-0+\x06\x03U\x04\x03\x13$Han
gzhou Shunwang Technology Co.,Ltd0\x82\x01" at 0x3e83c RULEDATE_1: 1970-01-01 TAGS_1
: FILE, SUSP, VENDOR RULENAME_1: EXT_SUSP_PE_Signed_by_Suspicious_Entitiy_Mar23 AUTH
OR_1: Arnim Rupp (https://github.com/ruppde)
REASONS_COUNT: 1

```

ProcessCheck C2 detection by THOR pointing to the ValleyRat beaconing

DNS logging provides another reliable angle. Sysmon event ID 22 records DNS queries, including those issued by `C:\Users\Public\Documents\WindowsData\NtHandleCallback.exe`. Detection rules that combine unusual process paths under `%PUBLIC%` with DNS requests to previously unseen or suspicious domains offer a powerful way to surface C2 communications, even if IP-based indicators change over time.

THOR can be used both to detect the suspicious process and to dump it for further offline analysis, which ultimately leads to the discovery of the ValleyRat beacon and a complete understanding of the C2 protocol and infrastructure.

# Appendix

---

## Indicators

---

### Droppers

00CA4CC46E7E6FC5351CC211D6D23A2310EA743C7A17A79723CCDCFA08291D3  
E774704B4E45FAFCD0F8344F61DFD67177EF14DE664D5610E7047D8E42621845  
9EDE6DA5986D8C0DF3367C395B0B3924FFB12206939F33B01610C1AE955630D1  
E122FF1369A2A229B8342C5688E49F627C29708353F8851DDA5787F51B64E3C0  
3C5354F43B9D8BDF26B613B947BFAC4E6598BCDAD3063449551017A5C0939FA1  
2EF1C67DE60E77B61219E7B653F3C5AA838912138196CF049A1D7A07FBBBA3A4  
789A0DA3F438C01A8CBACADDA3977F9CD878B4EE8B0F88D317AB3AF85CA166AA  
6DD391BB10499681F406744962230BF176454A073955913A6278058E41BC02B2  
7E42401B9983CD3B090AE97A9C723A46C9ECEE04BBB5F55D0BDA60226F043131  
ABAE8AD8577E2BCB7389769BE60D60F6D230E5827A9FF7ADFE92C62A38E99D64  
FCCA326A15C217825A055D95AC672900286F3889E27AF91FE6789430BFC08A45  
F1BDE45AC4A34B8EC885FB5FB07F5E47F89F97B257CC38A1EB37FC0C308C4A04  
6B75596830DA05B93CD2F514DA7369A7766C3C8A9F42729E9D87E483ED52613B  
A8C5B9FD1A97C0057834E3E3C6FC7A218A10212883CE4E8947C06F3ACFB282A7  
1ABBE524C98A788301FC7AC6DD401FFA79CA2E24C2FFC112954C3B0463ABDA8E  
7D193400C53F895672577CFBD6B21AE20A204FA0C07BB68A8207BD4DF47AE02A  
6D72056A797E39D234432F98E651DE996B09E829AE389221ACD993239600F13E  
B70434F49FBD5E87C23030711647BB77FBC59986656DBBFD9481A3C376217772  
7B5B05580C96E9AA8DD7A7410DA67CEC720BA8ACB5CA073473D46C2C01248583

## Second Stage

funzip.exe:

1EB33D7A653E4A05B1575357029A826661429DE77F8EAE5CBBBD8EC6C6AD22FB1

main.xml:

B67A14305B802ED9F9039CE5C8F49B2BC3075CA3EFD7F6BB63E0AAAB993E594B

men.exe:

9D3BAF43AC52740895E50E3D5E4C8D6618AC1DDB4B54955AD7493366C63DEEB2

setup.exe:

DA1056565881DA22BA1E87CE776184A300727A918519111F975331CBDD136ADE

bypass.exe:

F330E21CF670DA67160937525DD5CA80B1F26C9B3DDDD34ADB2F175D85C485F1

kail.exe:

72C33F24FB5853D2EF70ADECE5C7CACEDD8E568A9025F7A82FD5EF5C2F9967C5

me.key:

4EB84EE6B19B7AC124063A5AFD2B21BA3A41B6EF13348E2A2EE0D94433D82A87

X.vbe:

73526196AC0F863BD46F1BD0653CE42C429064E24FAF2AD917FF935E9BBDFFB5

NVIDIA.exe:

B4AC2E473C5D6C5E1B8430A87EF4F33B53B9BA0F585D3173365E437DE4C816B2

log.dll:

567760CCA4E5B0590D7ED5C7CA61F75ADD6FFAB199B351CE93E7429DF988D046

main.exe:

FB249BFF9449BBD715D936E6BCE4CE2354434DC9EB305E352FFADBC82562252F

NtHandleCallback.exe:

C027CF868757BABAB33686BF4C41192339E04FA89AD868409A5CD4ED90A1F71E

rwdriver.cat:

E90B505E3B31E15E608F2F9FB1C0FABDFF29B91988EB6A61A73556E05E182D4C

rwdriver.sys:

1C763AF41B74C7502D70093763723939A8025199E0AC7E39C04B5CF992F9E273

NSecKrn164.sys:

206F27AE820783B7755BCA89F83A0FE096DBB510018DD65B63FC80BD20C03261

## C2

ydbao8[.]cyou

zbj22[.]zbj888uu1[.]com

keepmasterr[.]com

knoeyyrt[.]com

ax[.]iuu8888[.]top