Republic of Tunisia
Ministry of Higher Education and Scientific Research

Private Higher School of Technology & Engineering
TEK-UP University

Academic Project Report

Malware Analysis Course

Submitted in partial fulfillment of the requirements for the

National Engineering Degree in Computer Systems and Networks

Specialization: Security of Computer Systems and Networks

STX RAT Malware Analysis Report

Malware Analysis Case Study

Prepared by: Mohamed Yacine Riabi

Academic Supervisor: Mrs. Ameni Ben Khalifa, Professor

Conducted at: TEK-UP University

Academic Year: 2025–2026

May 2026

Declaration and Safety Statement

This report was prepared for the Malware Analysis course at TEK-UP University as an academic case study. The selected malware family is STX RAT / STXRAT, and the selected sample is the installer-themed executable `HWiNFO_Monitor_Setup.exe`.

> The sample must be treated as malicious. It must not be executed on a personal computer, production system, shared network, or any machine that contains real credentials. Any future execution must take place only in an isolated malware-analysis virtual machine with a clean snapshot, disabled shared folders, restricted clipboard sharing, and controlled network simulation.

This report separates four classes of evidence: direct local static evidence, public sandbox evidence, public reputation evidence, and family-level research drawn from public reporting. A behavior is not described as locally observed unless the workspace contains local screenshots, logs, or output files supporting that claim.

Abstract / Executive Summary

This report analyzes an STX RAT / STXRAT sample associated with the SHA256 hash eefc0f986dd3ea376a4a54f80ce0dc3e6491165aefdd7d5d6005da3892ce248f. The filename theme is `HWiNFO_Monitor_Setup.exe`, an installer-themed Windows executable.

The strongest local evidence is static. The collected artifacts identify the file as a PE32 Windows GUI executable, 4,233,610 bytes, with Inno Setup metadata, setup-loader strings, suspicious entropy in executable sections, an overlay, and imports consistent with installer staging and loader-like behavior. peframe reports features including mutex, anti-debug, packer, crypto, registry, token, and file-operation indicators. capa produced no usable output and is therefore excluded from capability claims.

Local FLARE VM execution did not proceed: the on-host anti-malware engine detected and blocked the sample before any payload behavior could be observed. As a substitute for interactive dynamic analysis, an interactive ANY.RUN session was used, supported by automated execution in Hatching Triage, CAPE, and FileScan. Public reputation evidence from MalwareBazaar, VirusTotal, and Hybrid Analysis supports the malicious classification and STXRAT association.

Key finding: the selected sample is best assessed as an installer-themed STXRAT delivery or loader artifact. The evidence supports packing, setup-wrapper behavior, decoy hardware-monitor UI presentation, and PowerShell-staged execution. Live C2 contact, durable persistence, credential theft, and a recovered final payload remain unconfirmed for this sample.

Contents

List of Figures

List of Tables

1 Introduction and Scope

1.1 Objective

The objective is to produce an evidence-driven academic malware-analysis case study for the STX RAT / STXRAT family, focused on the selected installer-themed executable. The analysis distinguishes what is directly supported by local artifacts from what is reported by public sandboxes, public reputation services, or public family-level research.

1.2 Scope

The scope is limited to the sample identified by SHA256 eefc0f986dd3ea376a4a54f80ce0dc3e6491165aefdd7d5d6005da3892ce248f. The report uses local static outputs under `output/static/`, screenshots under `figures/`, public malware-intelligence pages, and publicly available STX RAT research.

The report does not claim a successful local dynamic execution. The local FLARE VM run was terminated by the host's anti-malware engine before payload behavior could be captured. Interactive dynamic evidence in this report therefore originates from a public ANY.RUN sandbox session, with corroborating automated runs from Hatching Triage, CAPE, and FileScan.

1.3 Evidence Model

Table 1.1: Evidence model used in this report

Evidence class	Meaning	Examples in this report

Direct local evidence	Produced or stored in the project workspace	peframe output, FLOSS output, rabin imports/exports, static screenshots, local pre-execution screenshots
Public sandbox evidence	Generated by an external sandbox execution	ANY.RUN interactive session, Hatching Triage, CAPE, FileScan
Public reputation evidence	Multi-source malicious labeling and YARA matches	MalwareBazaar, VirusTotal, Hybrid Analysis
Family-level research	Behavior described for STX RAT as a malware family in public reporting	C2 cryptography, unpacking routine, staged delivery model
Analyst interpretation	Reasoned assessment based on correlated evidence	Loader hypothesis, ATT&CK confidence levels, defensive recommendations

1.4 Safety and Handling Rules

The sample remained inside a controlled malware-analysis environment. The intended local execution method for the selected EXE was a user-launched installer launch in a dedicated FLARE VM with host-only networking, a clean snapshot, and simulated services. Because the local FLARE VM blocked execution at launch, no live malware ran on the local host. The ANY.RUN session was conducted in the vendor's isolated browser-mediated VM and produced the interactive screenshots used in Chapter 7.

1.5 Report Methodology

The methodology follows six steps:

1. identify the sample and verify hash, type, and metadata;
2. collect public reputation evidence for triage context;
3. inspect local static outputs for packaging, imports, strings, entropy, and loader indicators;
4. attempt local FLARE VM execution; document the AV block and substitute with an ANY.RUN interactive session;

5. correlate the interactive ANY.RUN evidence with automated runs from Triage, CAPE, and FileScan;
6. map supported behaviors to MITRE ATT&CK, produce IOCs, and derive defensive recommendations.

2 STX RAT Background

2.1 Family Overview

STX RAT is a Windows remote access trojan with infostealer capability. Public reporting first documented it on April 8, 2026 following attempted delivery against a finance-sector environment in late February 2026 [3]. The family name derives from the Start of Text byte `0x02`, which is prefixed to command-and-control messages.

This chapter provides family-level research context. It explains how the malware family is documented to operate in public reporting, but it does not convert those behaviors into local findings for the selected `HWiNFO_Monitor_Setup.exe` sample unless later chapters provide matching local or public-sandbox evidence.

2.2 Discovery and Campaign Context

Public reporting describes an initial browser-downloaded VBScript that writes and launches a JScript stage through Windows Script Host. The JScript then downloads archive content, extracts the next stage, and starts a PowerShell loader [3]. Later activity has been linked to trojanized FileZilla installers. This is relevant context for the selected case because the analyzed sample is also installer-themed, but the selected artifact is not the same VBScript described in that public research.

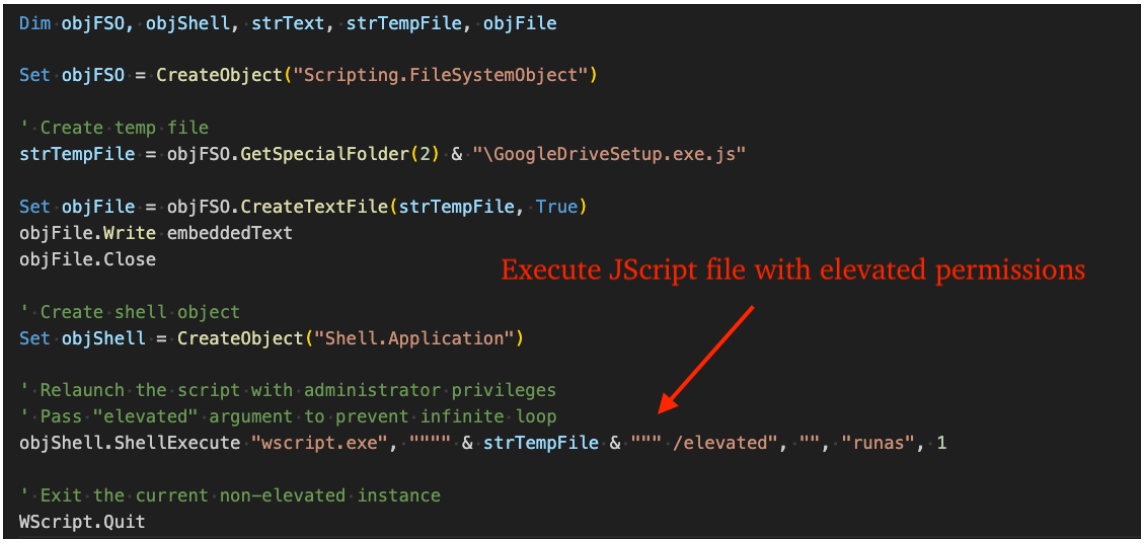


Figure 2.1: Public reporting screenshot illustrating the WScript-to-JScript initial-stage delivery chain. The image was retrieved from the public report using `Invoke-WebRequest` (procedure documented in Appendix C) and is referenced here as family-level research context, not as local evidence for the selected sample [3].

2.3 Delivery Chain

Reported STX RAT activity relies on social engineering and trusted-software themes. A victim is encouraged to run content that appears tied to ordinary software, after which the chain proceeds through script staging, archive extraction, an in-memory PowerShell loader, unpacking, and C2-gated tasking.

Stage	Reported command or action	Evidence basis
Elevated script launch	`"C:\Windows\System32\wscript.exe""C:\Users\<Username>\AppData\Local\Temp\business-structure.xlsx.js"/elevated`	Family research [3]
PowerShell payload execution	`powershell.exe-Command"[Console]::In.ReadToEnd()` Invoke-Expression"|	Family research [3]
Initial-stage download host	`147.45.178[.]61`	Family research [3]

Table 2.1: Family-level command lines and delivery indicators from public reporting. These are not local command-line evidence for the selected EXE sample.

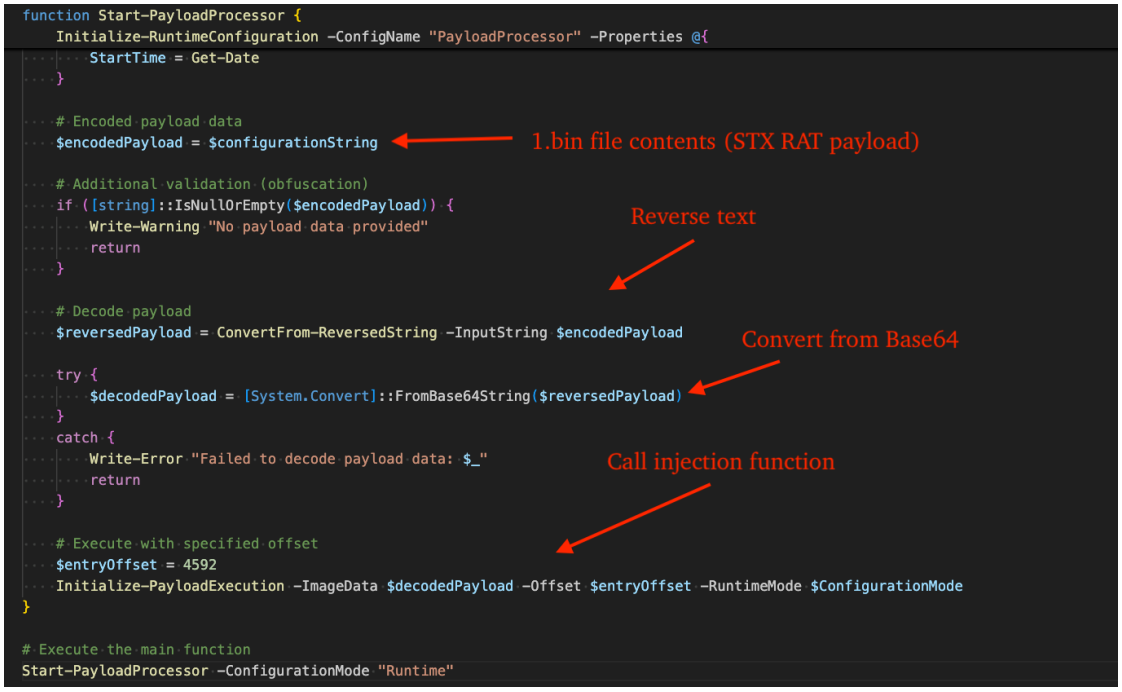


Figure 2.2: Public reporting screenshot showing the PowerShell loader logic that pipes content into `Invoke-Expression` via standard input. The image was retrieved via `Invoke-WebRequest` (Appendix C) and is used to explain family-level in-memory staging [3].

2.4 Loader and Packing Behavior

Public reporting documents a loader/unpacking pattern that uses `init` and `run` exports, XXTEA decryption, Zlib decompression, runtime API resolution, executable memory allocation, and a transfer of execution into unpacked code [3]. This pattern matches the static profile of the selected sample (installer wrapping, overlay data, elevated entropy in executable sections, loader-relevant imports) but the correlation alone does not prove that the same unpacking routine ran during analysis.

Figure 2.3: Public reporting screenshot of the documented STX RAT unpacking routine (XXTEA / Zlib / runtime API resolution). Retrieved via `Invoke-WebRequest` (Appendix C); referenced as research context for interpreting the local packing indicators in Chapter 6 [3].

2.5 Capabilities

Public reporting describes STX RAT as supporting multi-stage unpacking, encrypted strings, custom C2 cryptography, anti-analysis checks, hidden remote desktop functionality, credential and cookie theft, crypto-wallet targeting, FTP credential access, persistence, tunneling, and follow-on payload execution [3]. These are family-level findings and become sample-level findings only when supported by this sample's local static evidence or by clearly labeled public sandbox evidence.

Credential collection is reported to be C2-gated and activated only after operator or backend instruction [3]. This is consistent with the absence of credential-store activity in the available evidence for the selected sample: without a successful C2 handshake, the credential-theft path is never reached. It is not, however, proof that the capability is absent from the binary.

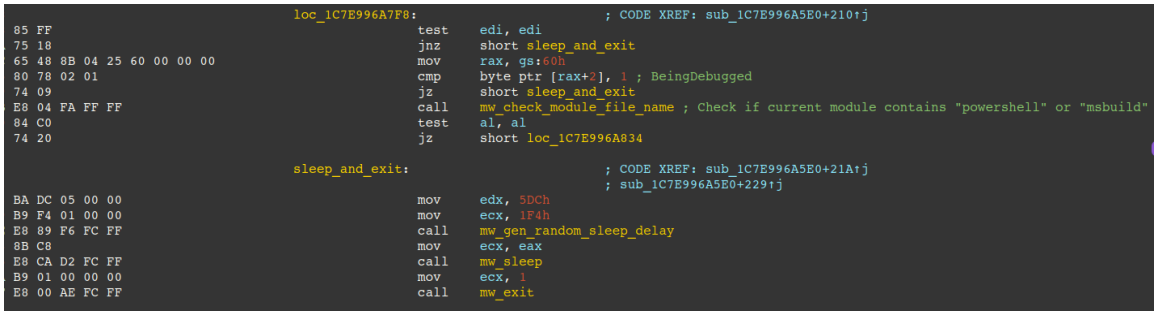


Figure 2.4: Public reporting screenshot of an anti-debug and process-name check used to detect analysis environments. Retrieved via `Invoke-WebRequest` (Appendix C); explains family-level anti-analysis behavior that may contribute to incomplete sandbox or VM execution [3].

2.6 C2 and Operator Control

The documented C2 channel uses a proprietary TCP protocol protected by X25519 key exchange, Ed25519 key validation, and ChaCha20-Poly1305 encryption and authentication [3]. Reported infrastructure includes clearweb and Tor/onion endpoints, with `95.216.51[.]236:31415` and `yu7sbzk2tgm4vv56qgvsq44wnwgct6sven4akbb2n3onp46f42fcstid[.]onion` listed at the family level. These indicators are useful for hunting; this report does not claim that the selected sample contacted them.

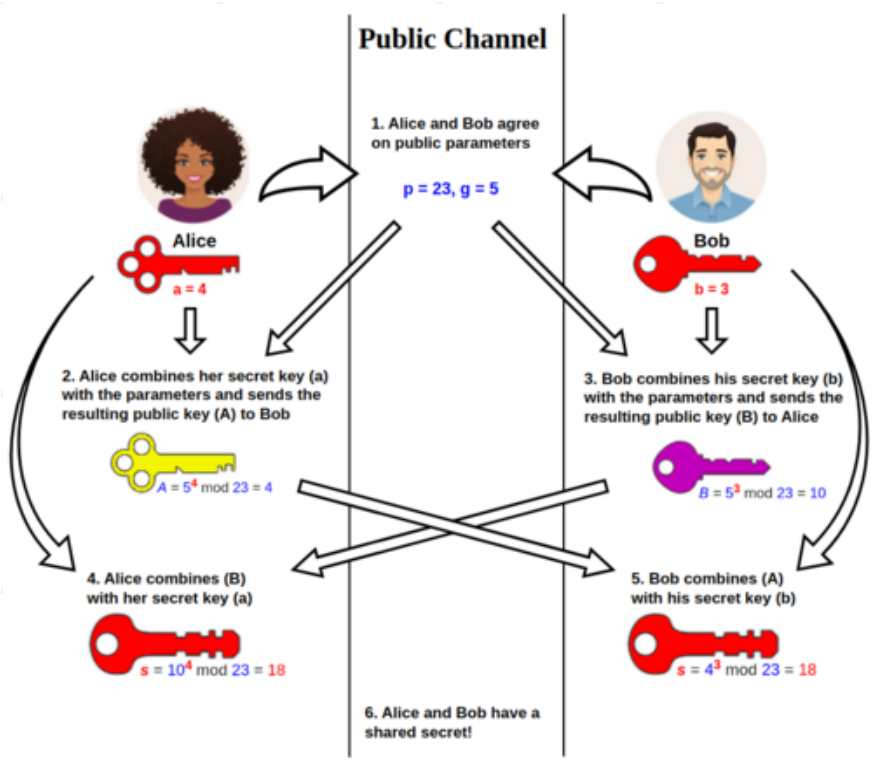


Figure 2.5: Public reporting screenshot illustrating the family-level C2 key-exchange model (X25519 / Ed25519 / ChaCha20-Poly1305). Retrieved via `Invoke-WebRequest` (Appendix C); not local packet-capture evidence [3].

A relevant operator-control behavior is the execution of C2-supplied payloads, including PowerShell content passed through standard input to a child PowerShell process. The defensive implication is that the command line visible on the endpoint can be minimal even when substantial script content is being executed through a redirected pipe.

```
pStartupInfo.StartupInfo.dwFlags |= 0x100u;
pStartupInfo.StartupInfo.cb = 0x68;
pStartupInfo.StartupInfo.hStdError = 0LL;
pStartupInfo.StartupInfo.hStdOutput = 0LL;
pStartupInfo.StartupInfo.hStdInput = hReadPipe;
mw_get_powershell_path(&powershellCommandString, (v7 & 2) != 0);
v8 = 0;
v41 = 29267;
v9 = v39;
v39[0] = xmmword_1C7E99A6698;
v39[1] = xmmword_1C7E99A66A8;
v39[2] = xmmword_1C7E99A66B8;
v40 = 0x1E00071E1F0E1819LL;
do
{
  v10 = 58 * (v8 / 58);
  v11 = v8++;
  *v9 ^= v11 - v10 + 57;
  v9 = (v9 + 1);
}
while ( v8 < 58 );
mw_string_assign(&powershellCommandString, v39);// -Command "[Console]::In.ReadToEnd() | Invoke-Expression"
CreateProcessW = mw_resolve_api_wrapper(0x8Eu);
wszPowershellCommand = 0LL;
v14 = mw_multibyte_to_wide_char_wrapper(&powershellCommandString);
v15 = 0LL;
if ( v14 )
{
  mw_free_heap_mem_0(0LL);
  wszPowershellCommand = v14;
  v15 = v14;
}
v16 = CreateProcessW(
        0LL,
        wszPowershellCommand,              // powershell.exe -Command "[Console]::In.ReadToEnd() | Invoke-Expression"
        0LL,
        0LL,
        1,                                 // bInheritHandles == TRUE
        0x8000000,                         // dwCreationFlags == CREATE_NO_WINDOW
        0LL,
        0LL,
        &pStartupInfo,
        &pProcessInformation);
```

Figure 2.6: Public reporting screenshot showing fileless PowerShell execution through redirected standard input. Retrieved via `Invoke-WebRequest` (Appendix C); included as family-level operator-control context, not as local command-line evidence [3].

2.7 Impact

If fully deployed, STX RAT creates serious operational risk: remote operator control, credential exposure, browser-session theft, wallet theft, FTP credential compromise, hidden desktop interaction, proxying, persistence, and follow-on execution. For defenders, a cleaned endpoint may still require credential rotation and session invalidation if credential access or successful C2 tasking is confirmed by endpoint telemetry.

3 Sample Identification

3.1 Selected Sample

Table 3.1: Selected sample metadata

Field	Value
Filename theme	`HWiNFO_Monitor_Setup.exe`
Local analysis name	`sample.exe`
SHA256	eefc0f986dd3ea376a4a54f80ce0dc3e6491165aefdd7d5d6005da3892ce248f
SHA1	02a53d660332c25af623bbb7df57c2aad1b0b91b
MD5	cdc459a866361463d719bc89622300f3
File size	4,233,610 bytes
File type	PE32 executable (GUI), Intel 80386, Windows, 11 sections
MalwareBazaar signature	`STXRAT`
imphash	`88016fcdef7f227c62171d0afad9aae4`

3.2 Source and Chain of Custody

The sample was obtained from public malware-intelligence sources. MalwareBazaar records the selected hash under the STXRAT signature; the workspace contains corroborating screenshots from MalwareBazaar, VirusTotal, Hybrid Analysis, ANY.RUN, Hatching Triage, CAPE, and FileScan. The local static workflow refers to the file as `sample.exe` (analyst-assigned filename), while the public filename theme is `HWiNFO_Monitor_Setup.exe`.

3.3 Hash Verification

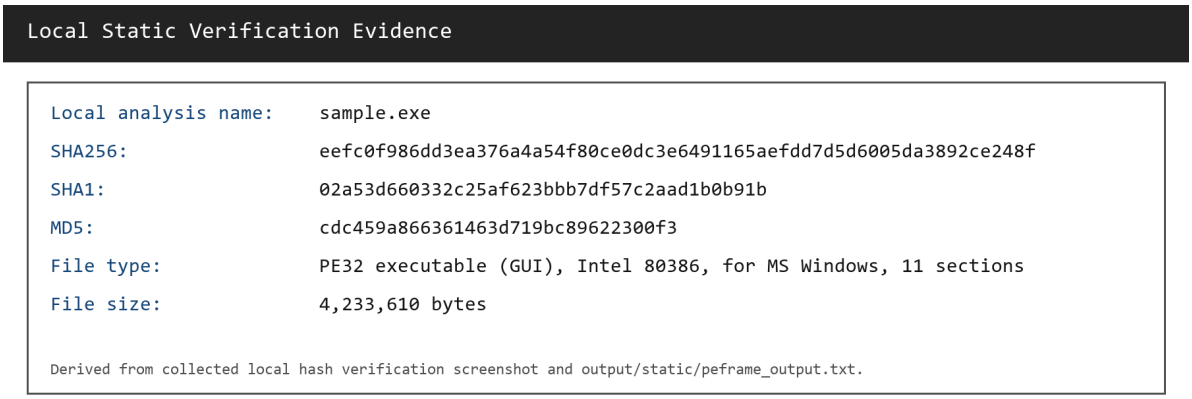


Figure 3.1: Local hash and file-type verification panel produced from `sha256sum`, `sha1sum`, `md5sum`, and `file`. Confirms the analyzed binary identity, PE32 GUI executable classification, and file size against the metadata table above.

The local hash values match the SHA256, SHA1, and MD5 values used in all public-platform correlation. The static workflow therefore operates on the intended sample. Hash verification confirms identity only; it carries no behavioral implication.

3.4 File Type and Metadata

```
remnux@yacine:~/Desktop/malware$ file sample.exe
sample.exe: PE32 executable (GUI) Intel 80386, for MS Windows, 11 sections
remnux@yacine:~/Desktop/malware$ exiftool sample.exe
ExifTool Version Number         : 13.50
File Name                       : sample.exe
Directory                       : .
File Size                       : 4.2 MB
File Modification Date/Time     : 2026:05:18 08:53:28+00:00
File Access Date/Time           : 2026:05:18 10:20:04+00:00
File Inode Change Date/Time     : 2026:05:18 10:19:34+00:00
File Permissions                : -rw-r--r--
File Type                       : Win32 EXE
File Type Extension             : exe
MIME Type                       : application/octet-stream
Machine Type                    : Intel 386 or later, and compatibles
Time Stamp                      : 2026:02:11 11:40:27+00:00
Image File Characteristics      : Executable, 32-bit
PE Type                         : PE32
Linker Version                  : 2.25
Code Size                       : 726016
Initialized Data Size           : 178688
Uninitialized Data Size         : 0
Entry Point                     : 0xb1e60
OS Version                      : 6.1
Image Version                   : 0.0
Subsystem Version               : 6.1
Subsystem                       : Windows GUI
File Version Number             : 0.0.0.0
Product Version Number          : 0.0.0.0
File Flags Mask                 : 0x003f
File Flags                      : (none)
File OS                         : Win32
Object File Type                : Executable application
File Subtype                    : 0
Language Code                   : Neutral
Character Set                   : Unicode
Comments                        : This installation was built with Inno Setup.
Company Name                    : CPUID
File Description                : HWiNFO Monitor Setup
File Version                    :
Legal Copyright                 :
Original File Name              :
Product Name                    : HWiNFO Monitor
Product Version                 : 1.63
```

Figure 3.2: Local PE metadata showing version information consistent with an Inno Setup-built installer. Supports the installer-wrapper interpretation pursued in Chapter 6.

peframe reports a PE32 GUI executable with image base `0x400000`, entry point `0xb1e60`, timestamp 2026-02-11 11:40:27, 11 sections, and `DLL: false`. The version block contains the comment "This installation was built with Inno Setup," the file description "HWiNFO Monitor Setup," and the product name "HWiNFO Monitor." This metadata is consistent with a legitimate-looking hardware-monitor installer theme but can be trivially forged in any PE.

3.5 Initial Classification

The sample is handled as a malicious installer-themed delivery artifact associated with STXRAT. The classification rests on the MalwareBazaar STXRAT signature for this exact hash, the local Inno Setup and packing indicators, and the corroborating multi-engine detections in Chapter 5. AV labels alone are not treated as exact family proof.

3.6 Evidence Boundaries

No evidence in the workspace proves credential theft, durable persistence, recovered final payload bytes, or live C2 contact specifically attributable to this sample. Those remain either family-level research or public-sandbox observations until matching local telemetry is collected.

4 Laboratory Environment

4.1 Lab Architecture

The lab was a two-VM model: a Windows FLARE VM intended for sample execution and instrumented network capture, plus a Linux service-simulation host. The local execution attempt did not yield runtime telemetry because the FLARE VM's on-host anti-malware engine detected the sample at launch and blocked it before any payload code could run. Pre-execution preparation evidence (Process Explorer baseline, Procmon filter setup) is retained as local lab context; interactive dynamic evidence in Chapter 7 originates from an ANY.RUN session conducted as a substitute.

4.2 Isolation Model

The intended isolation model used host-only networking, no production internet routing, no shared folders containing personal files, no real credentials, and a clean snapshot for rollback. The ANY.RUN session was conducted in the vendor's isolated browser-mediated VM, which prevents lateral exposure to the host or local network.

4.3 Tool Inventory

Table 4.1: Tool inventory and evidence status

Tool / source	Purpose	Evidence status
MalwareBazaar	Sample identity, signature, vendor and sandbox links	Screenshots present
VirusTotal	Public multi-engine detection and relations	Screenshots present

Hybrid Analysis	Public verdict and threat-score summary	Screenshot present
ANY.RUN	Interactive sandbox execution (used as substitute for blocked local run)	Screenshots present
Hatching Triage	Automated sandbox execution	Screenshots present
CAPE	Automated sandbox execution	Screenshots present
FileScan	Emulation and indicator summary	Screenshot present
peframe	PE metadata, packer, anti-debug, crypto, imports, entropy	Text output and screenshot present
rabin2	Imports and exports	Text output and screenshot present
FLOSS / strings	Extracted strings and installer artifacts	Text output and screenshot present
capa	Capability triage	Output file present but empty; excluded from analysis
Procmon (local)	Local process, file, registry, thread telemetry	Filter configured; no events captured (AV block)
Process Explorer (local)	Live process tree	Baseline screenshot only (AV block)
FakeNet-NG / INetSim	Simulated network services	Not engaged (AV block prevented execution)
Wireshark, Regshot, Autoruns, Sysmon	Local telemetry	Not collected (AV block)

4.4 Snapshot and Rollback Strategy

A clean Windows VM snapshot was taken before sample transfer. After the AV block, the VM was rolled back to the clean snapshot rather than reused, because even an aborted ex-

ecution can leave AV-quarantine state, residual file-system entries, or AV-recorded telemetry that could distort a re-run.

4.5 Local Execution Outcome

The local FLARE VM was prepared with Procmon, Process Explorer, and Wireshark running; the sample was copied to `C:\T1\textbackslashUsers\T1\textbackslash<user>\T1\textbackslashDownloads\T1\textbackslash` and launched as an interactive user. The endpoint anti-malware engine identified the binary at launch and prevented process creation, with no child processes spawned and no file or registry activity recorded by Procmon. Continuing to attempt local execution would have required disabling the AV engine on the host, which was rejected as a safety regression. The decision was made to substitute an ANY.RUN interactive session for dynamic evidence; that session is the source of the execution-result screenshots in Chapter 7.

> Local dynamic evidence is therefore limited to lab-preparation screenshots. All execution-result behavior (decoy UI, process spawn, PowerShell activity, file installation, installer error) was observed in ANY.RUN. This is public sandbox evidence and is labeled as such throughout Chapter 7.

5 Public Reputation Evidence

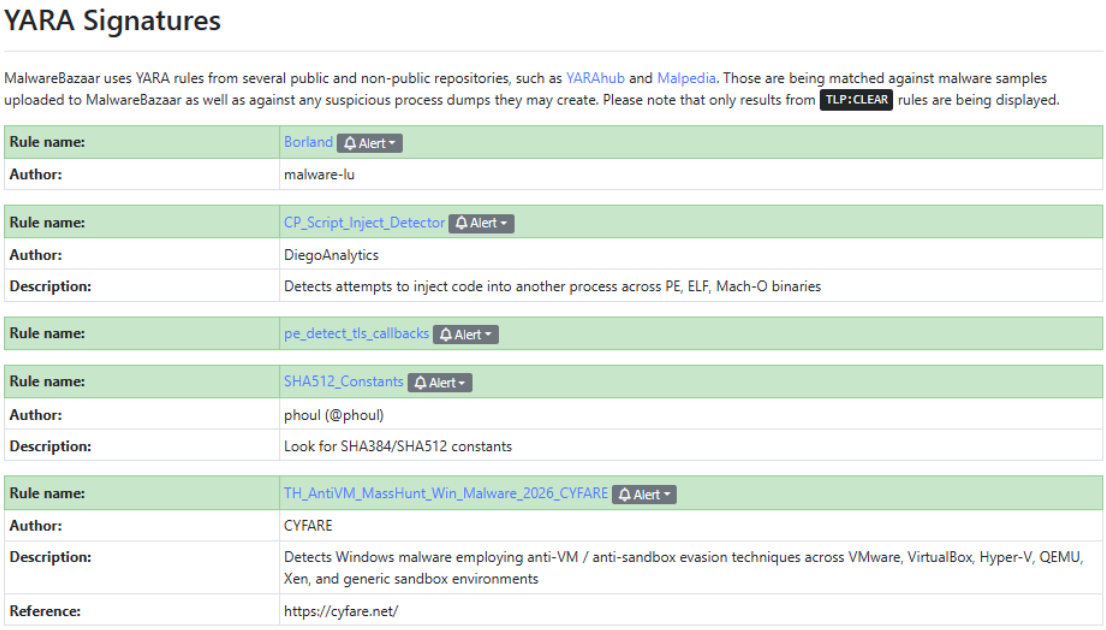
5.1 MalwareBazaar

MalwareBazaar is used here for sample identity, signature context, and platform correlation [1, 2].

SHA256 hash:	eefc0f986dd3ea376a4a54f80ce0dc3e6491165aefdd7d5d6005da3892ce248f
SHA3-384 hash:	e1da71bfd61179b1dff5d5dd0d3d64527a5fe5ccf8985efcfce68af873777f4636774583f94c998893497521348bb480
SHA1 hash:	02a53d660332c25af623bbb7df57c2aad1b0b91b
MD5 hash:	cdc459a866361463d719bc89622300f3
humanhash:	undress-mango-papa-kentucky
File name:	HWiNFO_Monitor_Setup.exe
Download:	download sample
Signature	STXRAT Alert
File size:	4'233'610 bytes
First seen:	2026-04-10 04:42:09 UTC
Last seen:	*Never*
File type:	exe
MIME type:	application/x-dosexec
imphash	88016fcdef7f227c62171d0afad9aae4 (7 x OffLoader, 7 x ValleyRAT, 3 x Gh0stRAT)
ssdeep	49152:Vul2hj6XF18ahT8kRdwlcwcBQSuBP9HqT9LnTiHejJkT6Dt7ON9Vnc:V5Oj6JR8kRdwlHcBIHqxLnmMBJ+c
Threatray	464 similar samples on MalwareBazaar
TLSH	T14516023FB24B653EE06E5A3539B7E130583B6A62A5124C1696F4C88CCF650B11E3F787
TrID	63.8% (.EXE) Inno Setup installer (107240/4/30) 24.7% (.EXE) Win32 EXE PECompact compressed (generic) (41569/9/9) 3.8% (.EXE) Win64 Executable (generic) (6522/11/2) 2.6% (.EXE) Win32 Executable (generic) (4504/4/1) 1.2% (.EXE) Win16/32 Executable Delphi generic (2072/23)
Magika	pebin
dhash icon	8a256575616d0182 (1 x STXRAT)
Reporter	KnownSpotter
Tags:	exe STXRAT

Figure 5.1: MalwareBazaar sample page for the selected hash, showing the STXRAT signature and the `HWiNFO_Monitor_Setup.exe` filename theme. Confirms public sample identity and family association.

MalwareBazaar associates the selected SHA256 with the STXRAT signature and the documented filename theme. This supports the family attribution and lets the static and dynamic findings be correlated against a known sample identity. Reputation labeling is not a substitute for behavioral validation.



YARA Signatures

MalwareBazaar uses YARA rules from several public and non-public repositories, such as YARAhub and Malpedia. Those are being matched against malware samples uploaded to MalwareBazaar as well as against any suspicious process dumps they may create. Please note that only results from TLP:CLEAR rules are being displayed.

Rule name:	Borland Alert
Author:	malware-lu
Rule name:	CP_Script_Inject_Detector Alert
Author:	DiegoAnalytics
Description:	Detects attempts to inject code into another process across PE, ELF, Mach-O binaries
Rule name:	pe_detect_tls_callbacks Alert
Rule name:	SHA512_Constants Alert
Author:	phoul (@phoul)
Description:	Look for SHA384/SHA512 constants
Rule name:	TH_AntiVM_MassHunt_Win_Malware_2026_CYFARE Alert
Author:	CYFARE
Description:	Detects Windows malware employing anti-VM / anti-sandbox evasion techniques across VMware, VirtualBox, Hyper-V, QEMU, Xen, and generic sandbox environments
Reference:	https://cyfare.net/

Figure 5.2: MalwareBazaar YARA rule matches for the selected sample. Indicates that public detection rules already flag features of this binary; useful for hunt-rule cross-reference.

5.2 VirusTotal

VirusTotal provides a multi-vendor detection picture for the selected hash [11].

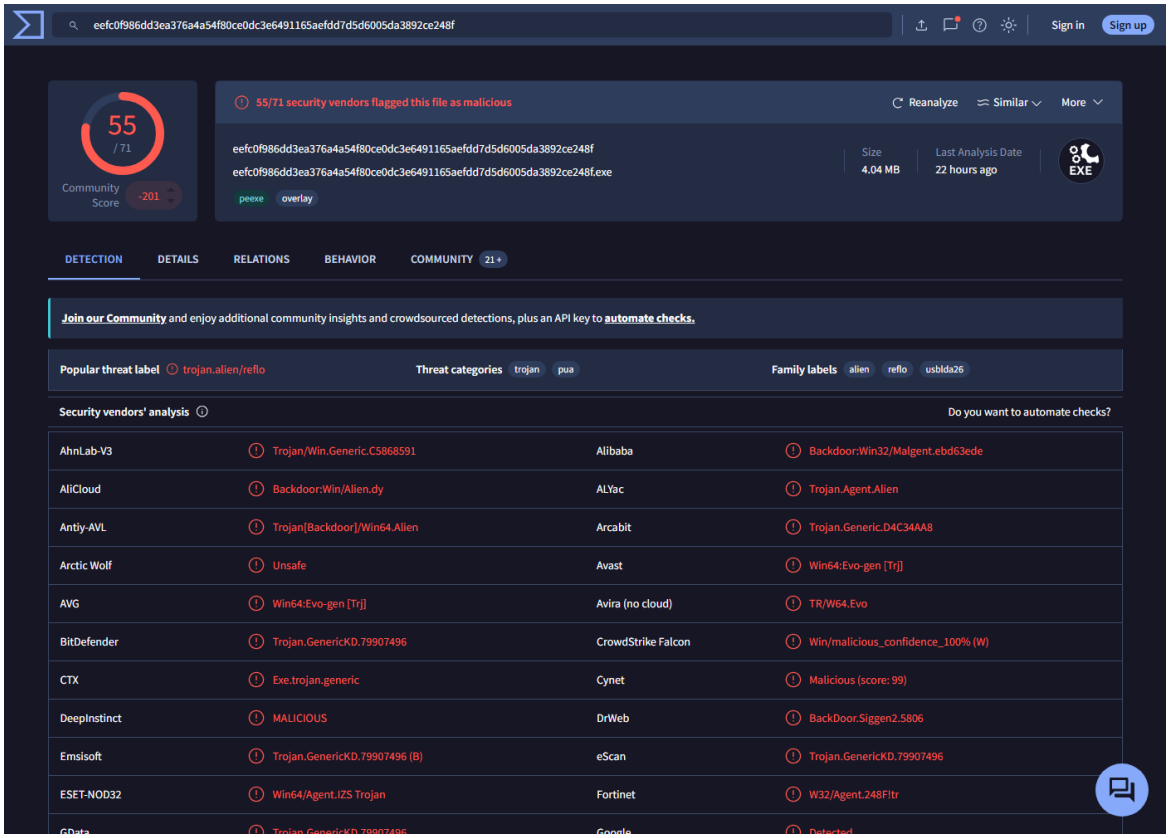


Figure 5.3: VirusTotal detection overview for the selected hash, showing a high malicious-to-total ratio across engines. Supports malicious classification but does not by itself prove specific STX RAT capabilities.

A high detection ratio across independent engines is treated as strong corroboration of the malicious classification. Vendor labels themselves are inconsistent across engines and are not used to make capability claims.

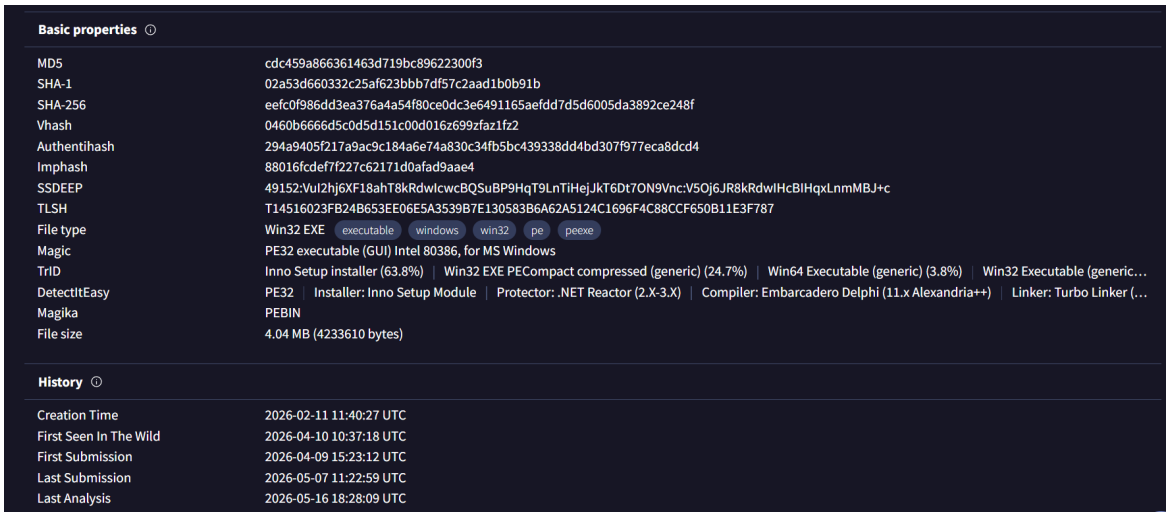


Figure 5.4: VirusTotal properties and file-history view for the selected hash. This supports metadata consistency and timeline context, not direct local execution behavior.

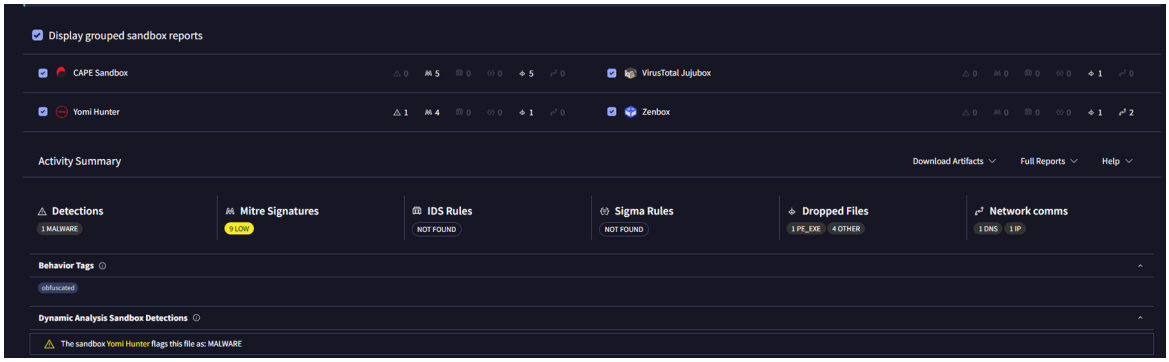


Figure 5.5: VirusTotal behavioral-summary view for the selected sample. Used as public triage context; detailed behavioral claims in this report are based on ANY.RUN, Triage, CAPE, and local static evidence.

5.3 Hybrid Analysis

Hybrid Analysis provides a public verdict and threat-score summary for the selected hash [4].

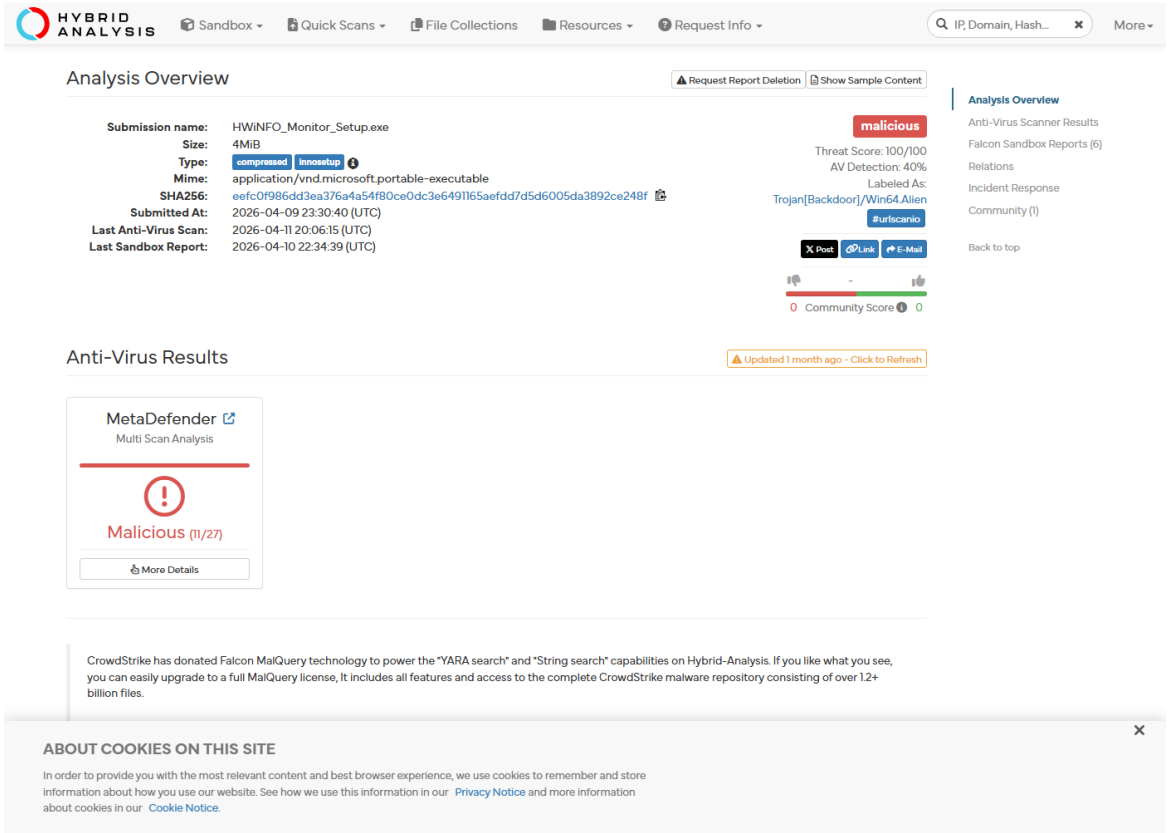


Figure 5.6: Hybrid Analysis overview classifying the sample as malicious with a high threat score. Used here for verdict corroboration; detailed runtime behavior is sourced from the sandbox sections in Chapter 7.

The Hybrid Analysis summary view corroborates the MalwareBazaar and VirusTotal

classification. Only the verdict and score are used at this stage; trace-level behavior is treated separately in Chapter 7.

5.4 Reputation Evidence Correlation

Table 5.1: Public reputation evidence correlation

Source	**Reported observation**	**Supported interpretation**	**Conf.**
MalwareBazaar	STXRAT signature and sample metadata	Sample-level STXRAT family association	High
VirusTotal	Broad cross-engine malicious detections	Malicious classification	High
Hybrid Analysis	Malicious verdict and high threat score	Unsafe installer-themed executable	Medium

6 Static Analysis

6.1 Hash and File-Type Verification

The local artifacts confirm the selected sample identity and PE type. The file is not a DLL: peframe reports `DLL: false`, and the local file-type screenshot identifies a PE32 GUI executable. The DLL-injection execution method is therefore not applicable to this artifact.

6.2 PE Metadata

```
remnux@yacine:~/Desktop/malware$ rabin2 -I sample.exe
arch     x86
baddr    0x400000
binsz    4233610
bintype  pe
bits     32
canary   true
injprot  false
retguard false
class    PE32
cmp.csum 0x004118c0
compiled Wed Feb 11 11:40:27 2026
crypto   false
endian   little
havecode true
hdr.csum 0x00000000
laddr    0x0
lang     c
linenum  false
lsyms    false
machine  i386
nx       true
os       windows
overlay  true
cc       cdecl
pic      true
relocs   false
signed   false
sanitize false
static   false
stripped false
subsys   Windows GUI
va       true
```

Figure 6.1: Local `rabin2` metadata for the sample, showing PE class, subsystem, entry point, and related headers. Confirms the Windows EXE classification.

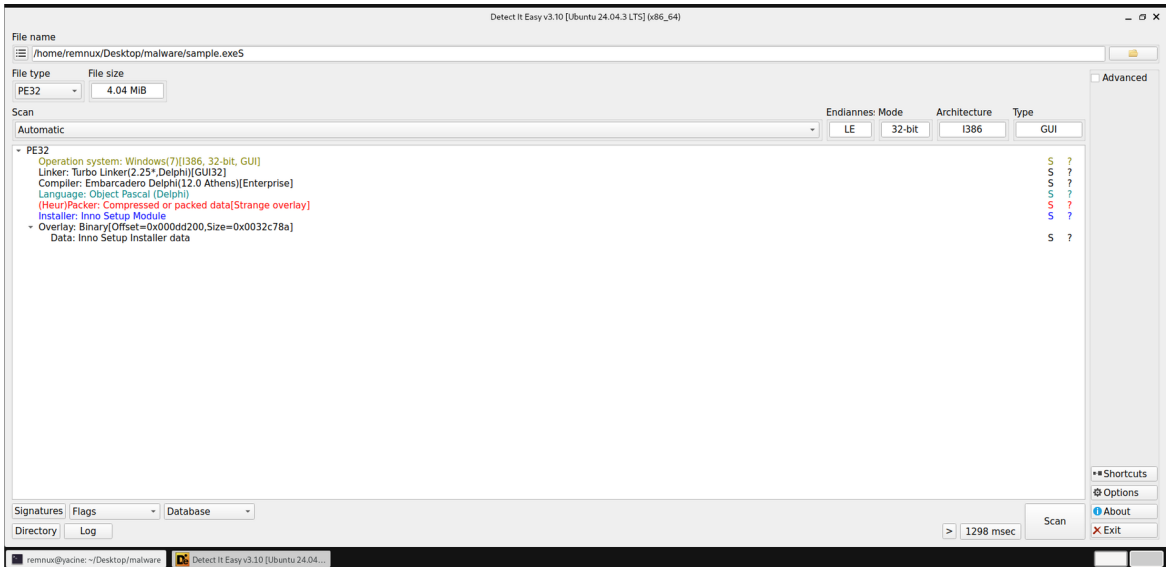


Figure 6.2: Detect It Easy overview showing the sample as a PE32 executable with installer and packing-related signatures. This corroborates the local PE metadata and supports the installer-wrapper interpretation.

peframe reports the file description "HWiNFO Monitor Setup," product name "HWiNFO Monitor," and the "This installation was built with Inno Setup" comment. The binary presents itself as a setup program consistent with the filename theme. Metadata fields are forgeable and do not by themselves justify a benign or malicious conclusion.

6.3 Sections, Overlay, and Entropy

```
remnux@yacine:~/Desktop/malware$ rabin2 -S sample.exe
[Sections]

nth paddr          size vaddr         vsize perm type name
_____________________________________________________________
0   0x00000400  0xafa00 0x00401000  0xb0000 -r-x ---- .text
1   0x000afe00   0x1a00 0x004b1000   0x2000 -r-x ---- .itext
2   0x000b1800   0x4000 0x004b3000   0x4000 -rw- ---- .data
3   0x00000000      0x0 0x004b7000   0x8000 -rw- ---- .bss
4   0x000b5800   0x1200 0x004bf000   0x2000 -rw- ---- .idata
5   0x000b6a00    0x200 0x004c1000   0x1000 -rw- ---- .didata
6   0x000b6c00    0x200 0x004c2000   0x1000 -r-- ---- .edata
7   0x00000000      0x0 0x004c3000   0x1000 -rw- ---- .tls
8   0x000b6e00    0x200 0x004c4000   0x1000 -r-- ---- .rdata
9   0x000b7000  0x12000 0x004c5000  0x12000 -r-- ---- .reloc
10  0x000c9000  0x14200 0x004d7000  0x15000 -r-- ---- .rsrc
```

Figure 6.3: Local section table and per-section entropy. Elevated entropy in executable sections plus a present overlay are consistent with packed or compressed content embedded in an installer wrapper.

peframe flags the file with `IsPacked` and `HasOverlay`, and reports suspicious entropy of 6.38 for `.text`, 6.04 for `.itext`, and 6.70 for `.reloc`. These values are consistent with compressed or packed content but do not identify a specific unpacking routine.

The combination of Inno Setup metadata, `SetupLdr.e32` strings, overlay data, and elevated entropy is significant because installer wrappers commonly embed compressed setup payloads and extract them at runtime. In conjunction with the STXRAT classification from Chapter 5, this combination supports an installer-wrapper-plus-loader interpretation. Static evidence alone does not prove what the unpacked payload does.

6.4 Packer and Loader Indicators

peframe additionally reports packer, crypto, anti-debug, and mutex feature flags. FLOSS and `strings` output include `Inno Setup Setup Data (6.7.0)`, `Inno Setup Messages (6.5.0)`, `SetupLdr.e32`, `InnoSetupLdrWindow`, and the standard setup command-line help strings. These artifacts confirm the wrapper technology used; they do not imply maliciousness on their own. The combined assessment here relies on the wrapper indicators plus the multi-engine STXRAT classification.

6.5 Imports and Exports

```
remnux@yacine:~/Desktop/malware$ rabin2 -E sample.exe
[Exports]
nth paddr      vaddr      bind   type size lib          name                demangled
1   0x0000363c 0x004ba63c GLOBAL FUNC 0    SetupLdr.e32 dbkFCallWrapperAddr
2   0x0000d978 0x0040e578 GLOBAL FUNC 0    SetupLdr.e32 __dbk_fcall_wrapper
```

Figure 6.4: Local imports/exports listing showing setup-loader exports and Windows API imports. Used as the basis for the loader-relevant API discussion below.

The rabin2 export listing contains the `SetupLdr.e32` exports `dbkFCallWrapperAddr` and `__dbk_fcall_wrapper`. The most analytically relevant import groups are process creation and threading APIs, memory-management APIs, temporary-file and write APIs, registry-query APIs, token APIs, and locale/system discovery APIs.

Specific imports are loader-consistent when interpreted in context: `CreateProcessW` can launch an extracted installer component, `GetTempPathW` and `WriteFile` can support temporary extraction, `VirtualAlloc` and `VirtualProtect` can prepare executable memory, and `RegOpenKeyExW` can query installation or environment state. These same APIs are also routine in legitimate installers. They are treated here as available capabilities, not as proof of process injection, persistence, or credential theft.

6.6 Strings and FLOSS

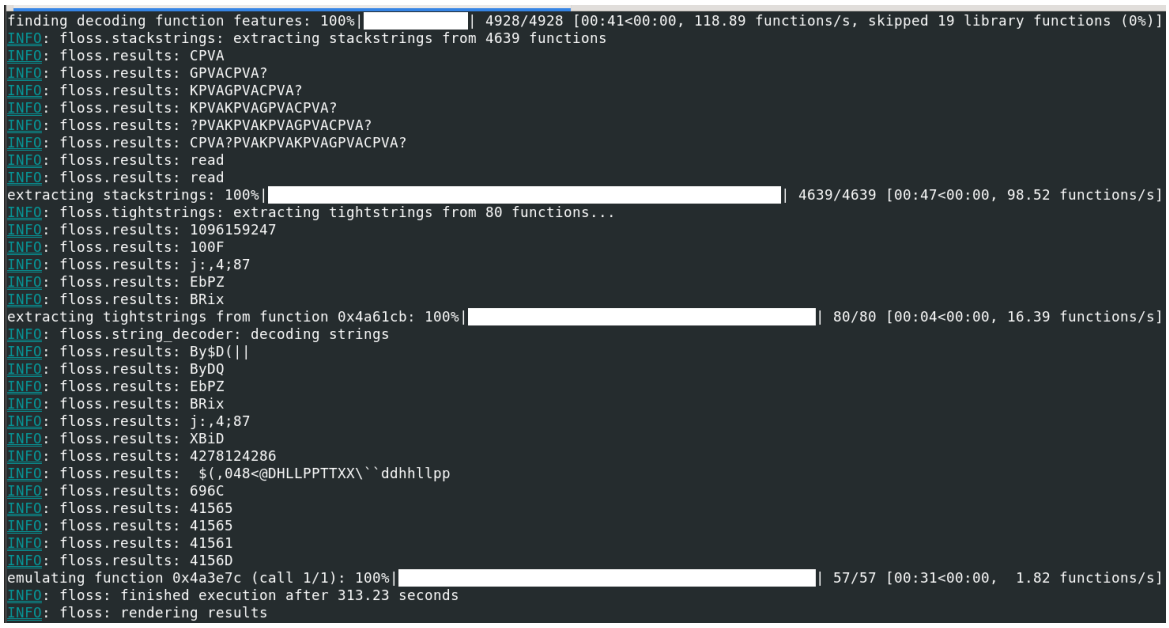


Figure 6.5: Local FLOSS output showing recovered installer and setup-loader strings. Used to confirm the Inno Setup wrapper signature and the HWiNFO Monitor product theme.

FLOSS recovered installer-related strings, including setup-data labels, command-line help text, password-prompt text, and the HWiNFO Monitor product theme. No validated C2 endpoint, encryption key, or final-payload artifact was recovered from the static string set.

The absence of a plaintext C2 configuration is consistent with packed or encrypted payload storage and with the C2-gated activation model documented in public reporting [3]. The correct conclusion is narrow: no C2 endpoint is statically recoverable from the wrapper. This does not refute family attribution; it constrains what static evidence alone can demonstrate.

6.7 YARA, peframe Summary, and Excluded Tooling

MalwareBazaar shows public YARA matches for this sample (Chapter 5). peframe's local summary aggregates the packing, crypto-constant, anti-debug, registry, token, and file-operation flags discussed above. `capa` produced an empty output file for this sample; no capa-derived capability claims appear anywhere in this report.

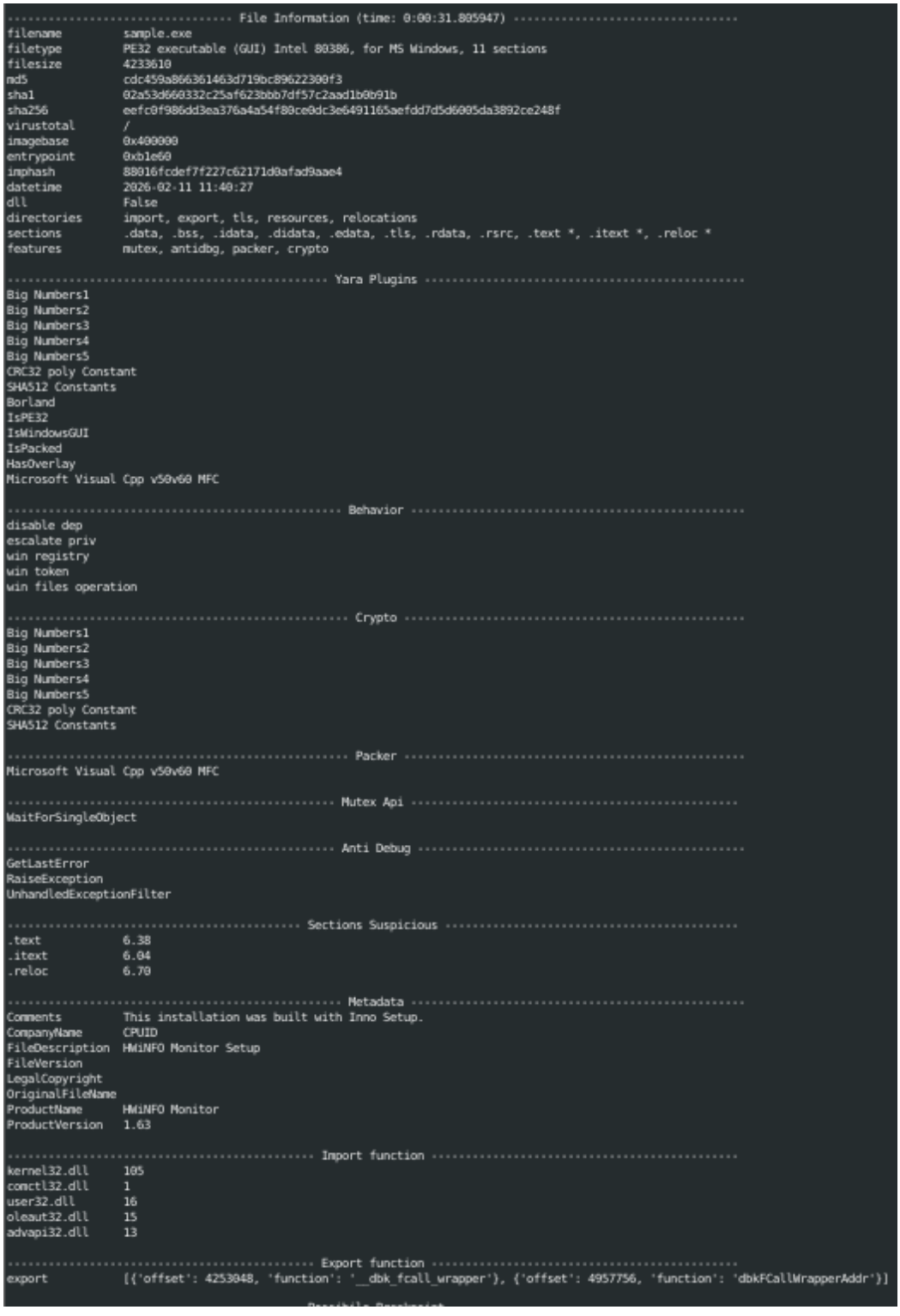


Figure 6.6: Local peframe summary: file information, feature flags, metadata, and suspicious sections. Consolidates the static triage findings used throughout this chapter.

6.8 Static Analysis Conclusion

Static analysis supports three sample-level findings: the file is the selected PE32 EXE; it presents as an Inno Setup-style installer with setup-loader exports and overlay; and it exhibits packing/loader-consistent indicators (`IsPacked`, elevated section entropy, memory-management and process-creation imports). Static analysis does not, on its own, prove credential theft, persistence, dropped files, or C2 contact.

7 Dynamic Analysis

7.1 Approach and Sources

Local interactive execution in the FLARE VM was prevented by the host's anti-malware engine, as documented in Chapter 4. Dynamic evidence in this chapter therefore comes from three public sandbox sources:

- an **ANY.RUN interactive session** (the primary substitute for local interactive analysis; provides the decoy-UI, process-tree, PowerShell, MSBuild-alert, installed-files, and installer-error screenshots);
- an automated **Hatching Triage** execution (process tree and signature mapping);
- an automated **CAPE** execution (file activity, dropped binaries, behavior summary, network result);
- a **FileScan** emulation summary (additional indicator corroboration).

Local FLARE VM screenshots are limited to lab preparation and confirm only that monitoring was configured before the AV block; they do not contain executed-malware telemetry.

7.2 Local FLARE VM Preparation (Pre-Block)

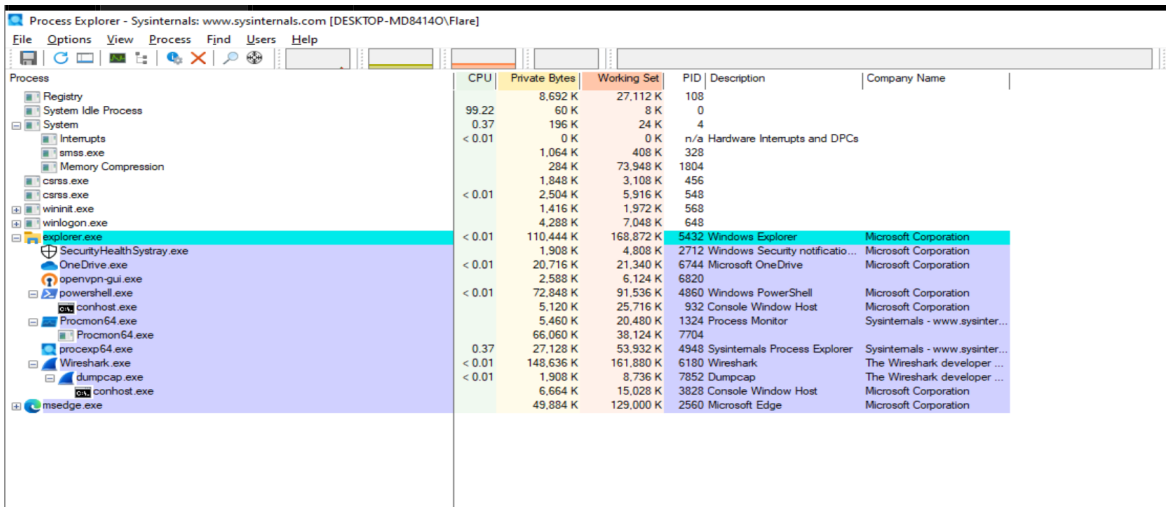


Figure 7.1: Local FLARE VM Process Explorer immediately before the launch attempt, showing Procmon, Process Explorer, Wireshark, and PowerShell already running. Establishes that monitoring was active; contains no malware-related processes.

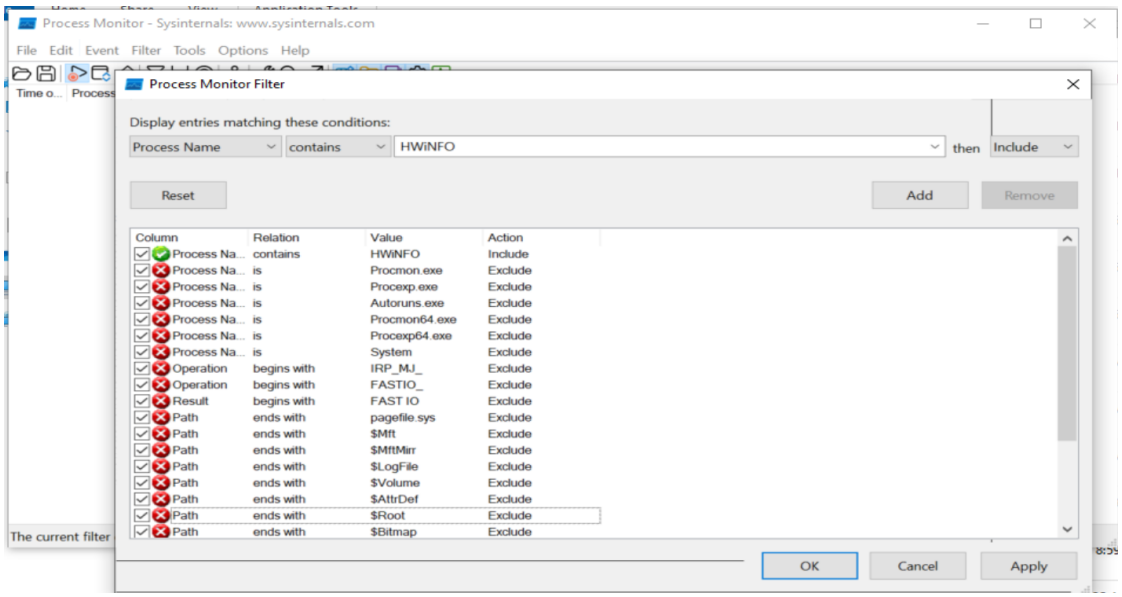


Figure 7.2: Local FLARE VM Procmon process-name filter configured to scope capture to `HWiNFO`-related processes. Confirms capture intent; no matching events were recorded because the AV engine prevented process creation.

The two preparation screenshots above are the only local-VM evidence in this chapter. They demonstrate that the analysis environment was instrumented correctly before launch; they do not show any malware behavior.

7.3 ANY.RUN Interactive Sandbox

The ANY.RUN session was used after the local AV block to obtain interactive dynamic evidence. The sandbox is a Windows 10 64-bit environment with a browser-mediated desktop view; all subsequent screenshots in this section are interface captures of that session, not of the local FLARE VM.

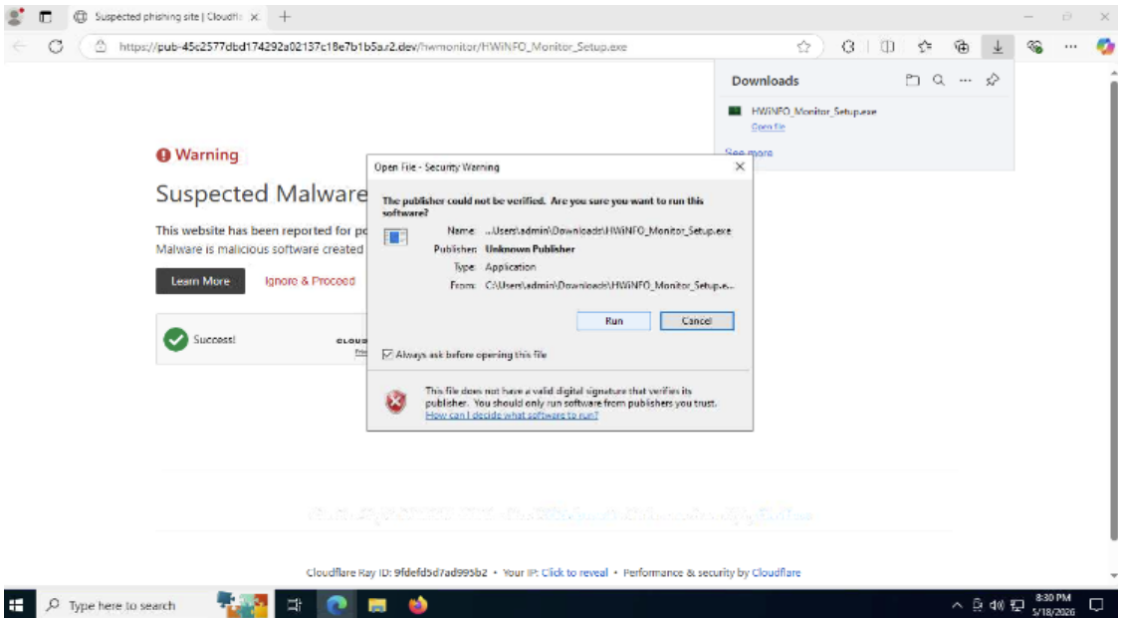


Figure 7.3: ANY.RUN session: Windows SmartScreen-style launch warning for `HWiNFO_Monitor_Setup.exe` from the user download path. Confirms the EXE was launched interactively as a user inside the sandbox.

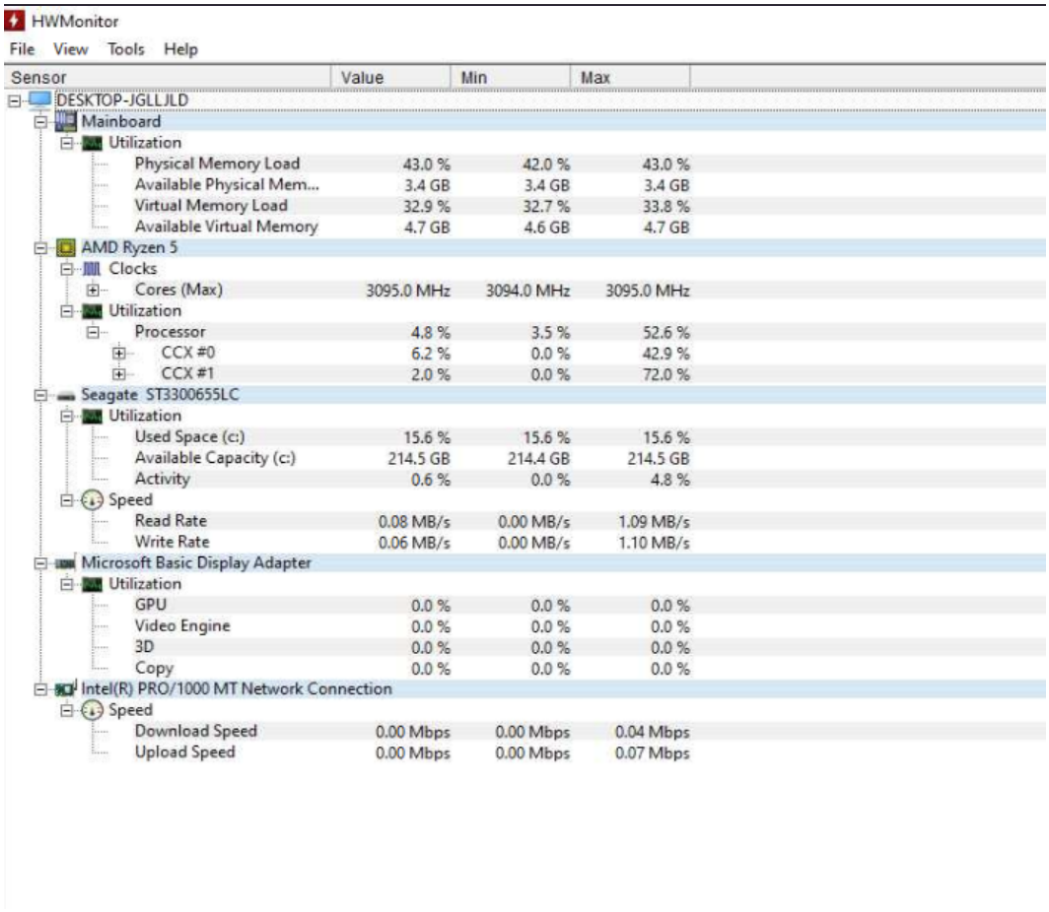


Figure 7.4: ANY.RUN session: CPUID HWMonitor-style hardware-monitor interface displayed after the installer ran. Demonstrates the decoy/legitimate-looking UI presented to the user; does not by itself prove the final STX RAT payload activated.

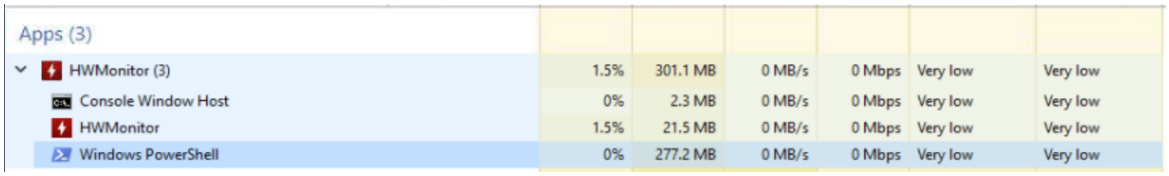


Figure 7.5: ANY.RUN session: Task Manager showing `HWMonitor` grouped with Console Window Host (`conhost.exe`) and Windows PowerShell. Establishes a parent-child relationship between the installed HWMonitor binary and a PowerShell process; full command line is not visible in this view.

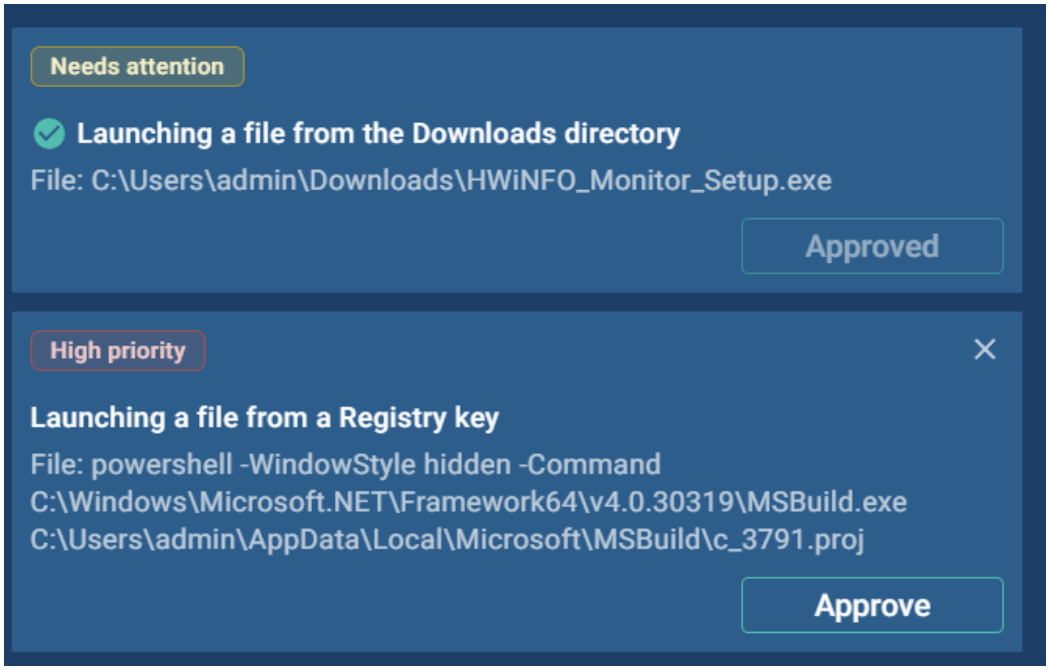


Figure 7.6: ANY.RUN session: behavioral alert reporting hidden-window PowerShell launched from a registry-resident command, invoking `MSBuild.exe` against a project file under `AppData\T1\textbackslashLocal\T1\textbackslashMicrosoft\T1\textbackslashMSBuild`. Indicates registry-based execution staging and MSBuild abuse; the project file contents and full registry value are not recovered.

The ANY.RUN evidence above is the strongest dynamic signal in the report. It shows that, when allowed to execute, the sample presents a legitimate-looking hardware-monitor decoy while spawning a PowerShell process whose command line is launched from a registry key and invokes `MSBuild.exe` with an analyst-relevant project path. MSBuild is a known living-off-the-land binary frequently abused to execute inline-task or project-embedded code. The behavior is consistent with the PowerShell-staged execution pattern documented in public reporting [3].

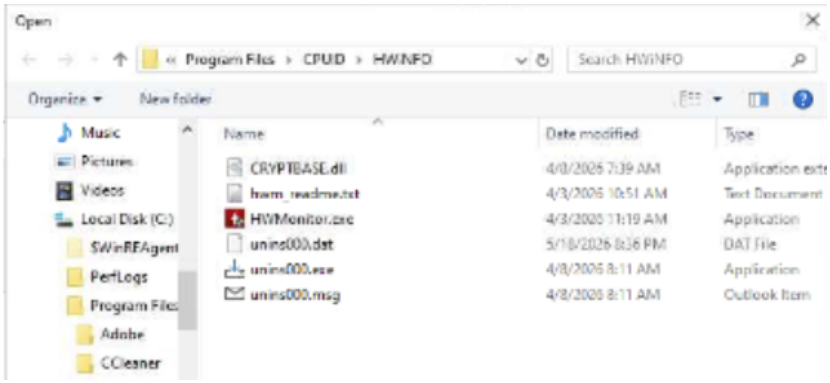


Figure 7.7: ANY.RUN session: files placed under `C:\T1\textbackslashProgramFiles\T1\textbackslashCPUID\T1\textbackslashHWINFO`, including `HWMonitor.exe`, `CRYPTBASE.dll`, and uninstall artifacts. Demonstrates installer-style file placement; file hashes were not captured from the sandbox session, so these files cannot yet be independently classified.

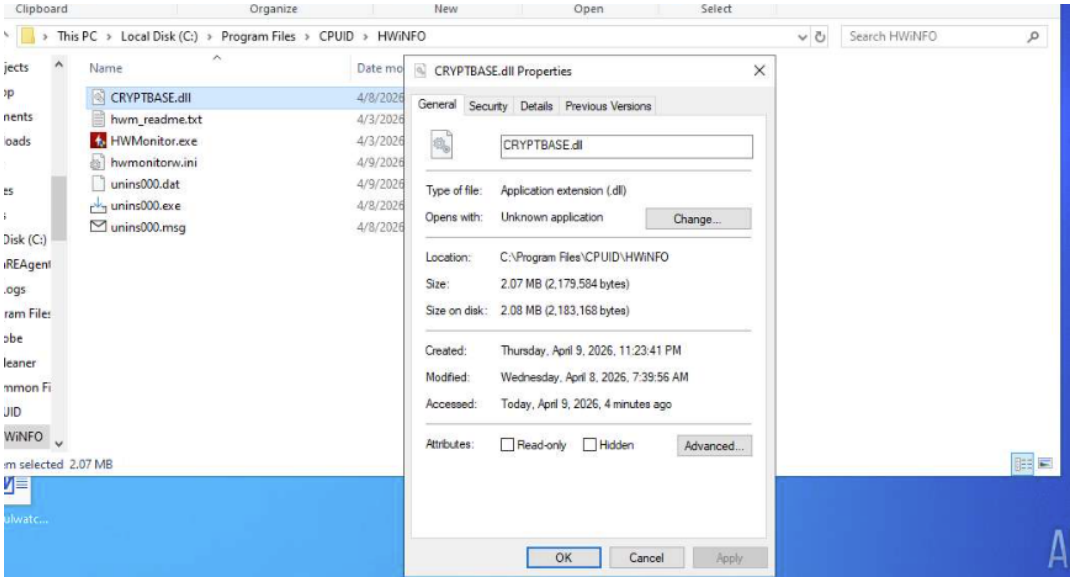


Figure 7.8: ANY.RUN session: Windows file-properties view for the installed `CRYPTBASE.dll` under the CPUID/HWINFO directory. This records the suspicious side-loaded DLL candidate location and file size, but it does not prove maliciousness without a hash or content review.

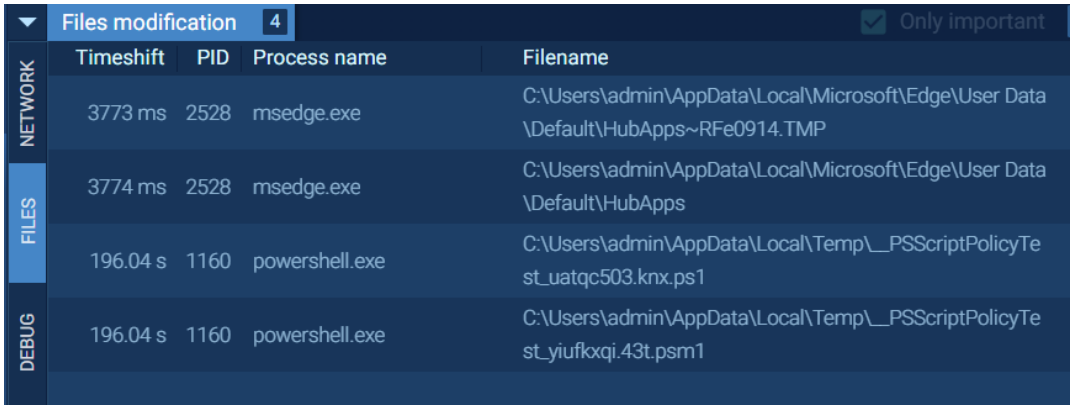


Figure 7.9: ANY.RUN session: file-modification view showing browser temporary files and PowerShell script-policy test artifacts during the run. This supports runtime file-activity context, while the installed CPUID/HWINFO directory remains the stronger sample-specific artifact.

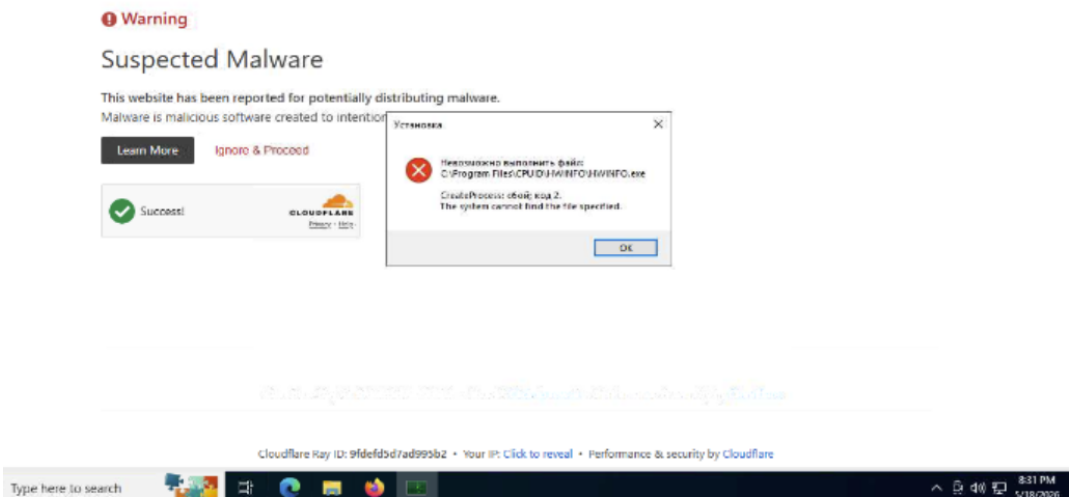


Figure 7.10: ANY.RUN session: installer error referencing a missing `C:\T1\textbackslashProgramFiles\T1\textbackslashCPUID\T1\textbackslashHWINFO\T1\textbackslashHWINFO.exe`. Suggests the visible installer path did not complete cleanly while the suspicious PowerShell/MSBuild activity still occurred.

The installer-error screenshot is operationally interesting: the user-visible component of the chain fails (missing `HWINFO.exe`) while the analyst-relevant component (hidden PowerShell launching MSBuild from a registry-resident command) is the one that produced behavioral alerts. The legitimate-looking installer activity is therefore best read as a decoy layer over the actual execution path, not as the actual purpose of the binary.

7.4 Hatching Triage Automated Sandbox

Hatching Triage executed the sample automatically and assigned a score of `7/10` with tags including `discovery` and `installer`. The target was identified as `HWiNFO_Monitor_Setup.exe` with SHA256 eefc0f986dd3ea376a4a54f80ce0dc3e6491165aefdd7d5d6005da3892ce248f.

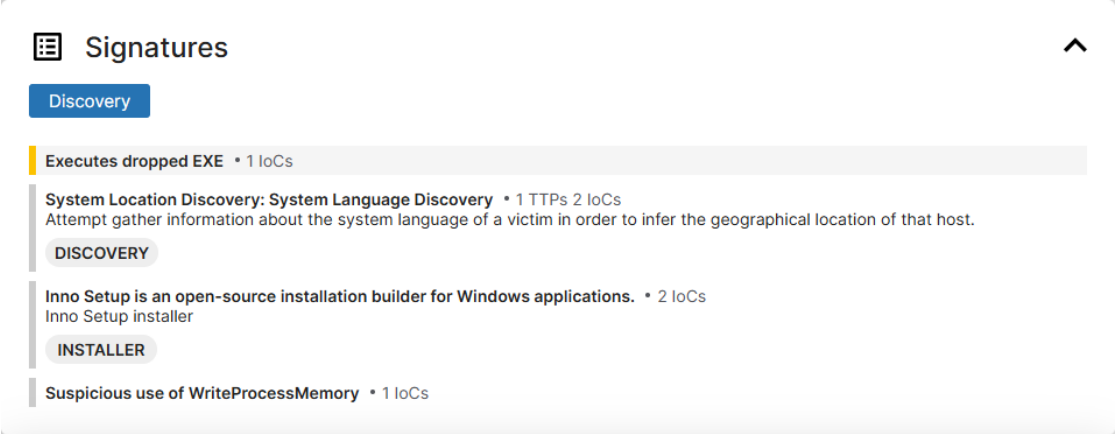


Figure 7.11: Triage signature panel: dropped-EXE execution, system-language discovery, Inno Setup installer recognition, and `WriteProcessMemory` use. These signatures support installer-wrapper plus inter-process memory writing.

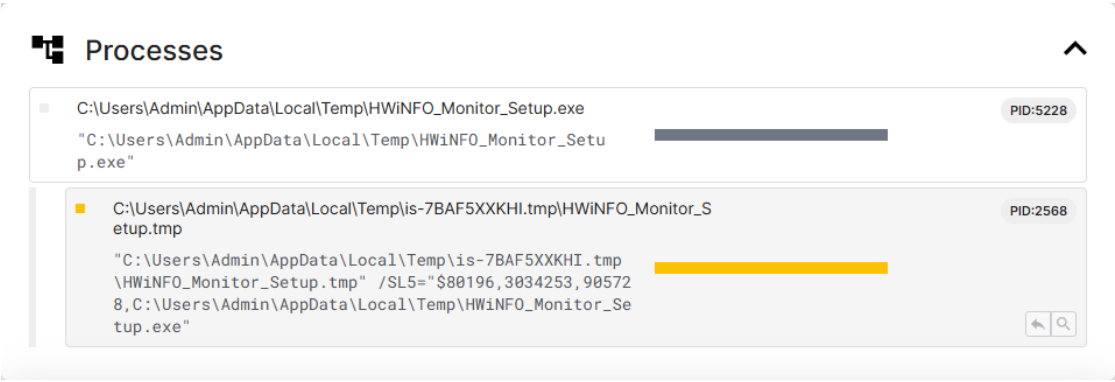


Figure 7.12: Triage process tree: the installer-themed EXE launches a temporary `HWiNFO_Monitor_Setup.tmp` child process from an Inno Setup extraction directory. Corroborates the wrapper-and-dropped-child execution pattern.

Triage's reported behaviors align with the static profile (Inno Setup wrapper, dropped-EXE stage) and with the ANY.RUN observation that a child binary continues execution after the wrapper. The `WriteProcessMemory` signature is suggestive of inter-process memory manipulation but is not, by itself, proof of process injection.

7.5 CAPE Automated Sandbox

CAPE executed the sample using the `exe` package and recorded the analyzed file as `HWiNFO_Monitor_Setup.exe`. The saved report independently verifies the PE32 GUI classification, file size of 4,233,610 bytes, and the SHA256, SHA1, and MD5 values used throughout this report.

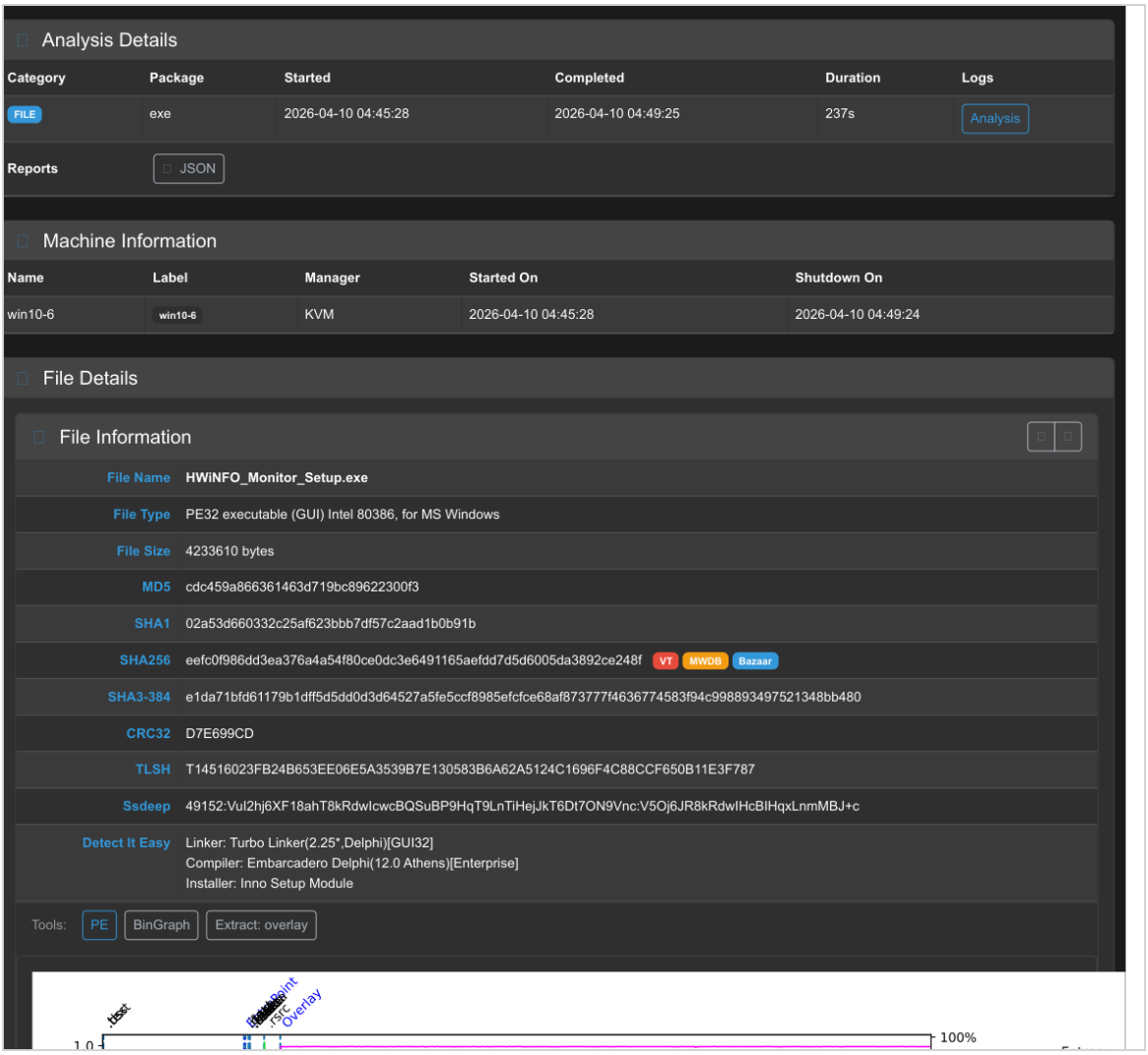


Figure 7.13: CAPE saved overview: EXE-package execution, machine context, hashes, and entropy/overlay visualization. Independently confirms the sample identity and the packed/overlay structure also seen in Chapter 6.

CAPE's analysis log records execution from `%LocalAppData%\Temp\HWiNFO_Monitor_Setup.exe`, followed by creation and execution of `%LocalAppData%\Temp\is-U85DTLA4LM.tmp\HWiNFO_Monitor_Setup.tmp` and a further Inno Setup helper at `%LocalAppData%\Temp\is-345P5RZPYA.tmp\_isetup\_setup64.tmp`. This matches the wrapper-then-child execution pattern reported by Triage.

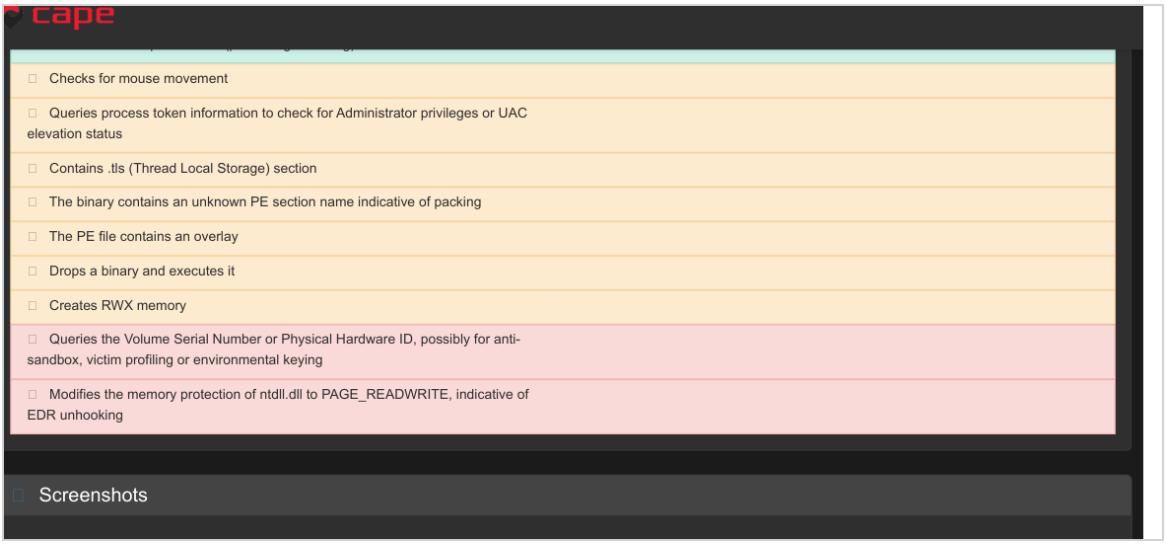


Figure 7.14: CAPE behavior summary: mouse-movement check, token/UAC status check, TLS section presence, overlay, dropped-binary execution, RWX memory allocation, hardware-identifier queries, and memory-protection changes. Indicates environment-aware execution plus memory behavior of interest.

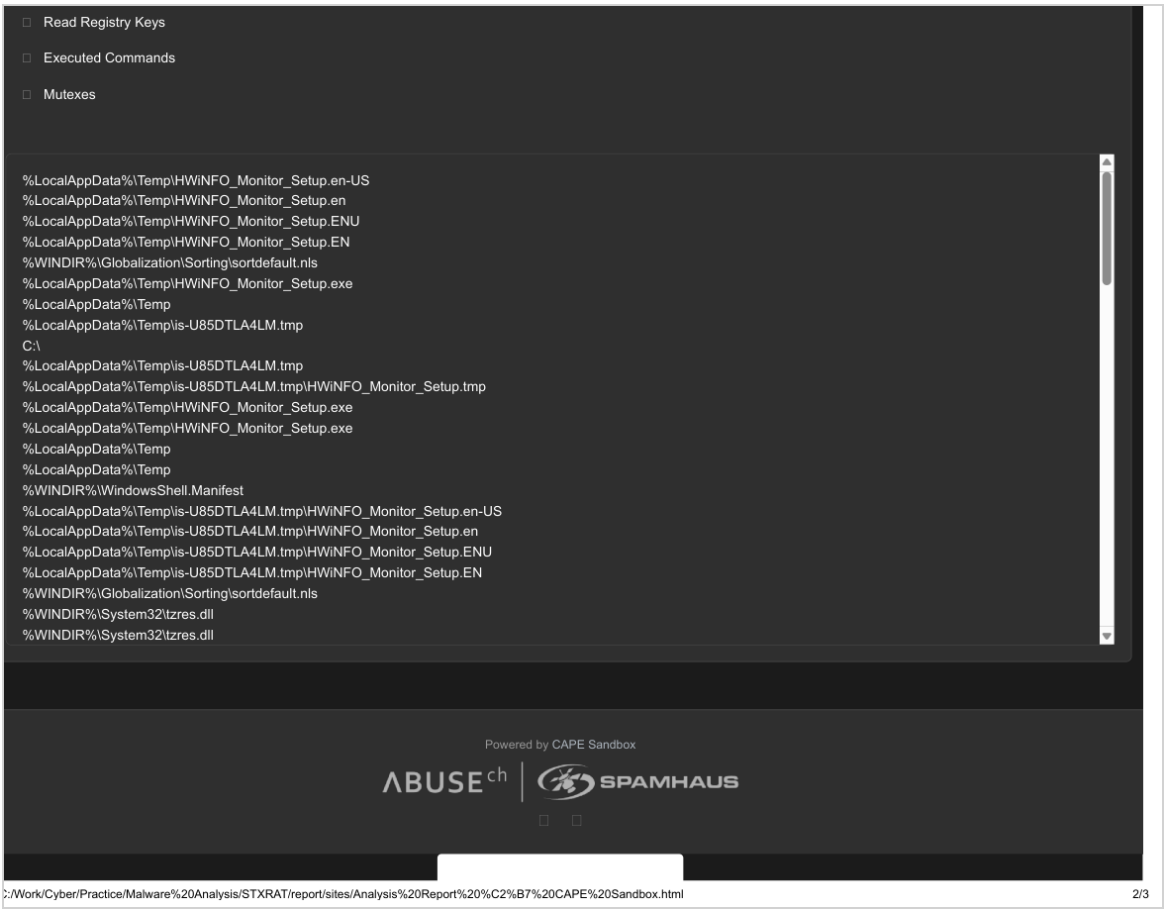


Figure 7.15: CAPE file-activity summary: accessed paths and temporary installer paths associated with `HWiNFO_Monitor_Setup.exe`. Cross-references the Triage drop paths and the ANY.RUN-installed CPUID/HWINFO directory.

CAPE's behavior summary contributes two evidence points the other sources do not provide as cleanly. First, the explicit RWX memory allocation and subsequent memory-protection changes are characteristic of an in-memory unpacking step, consistent with the loader pattern in public reporting [3]. Second, CAPE's network section records *no* DNS, TCP, UDP, or HTTP(S) requests for this execution; under this run, there is no live C2 contact to claim.

7.6 FileScan

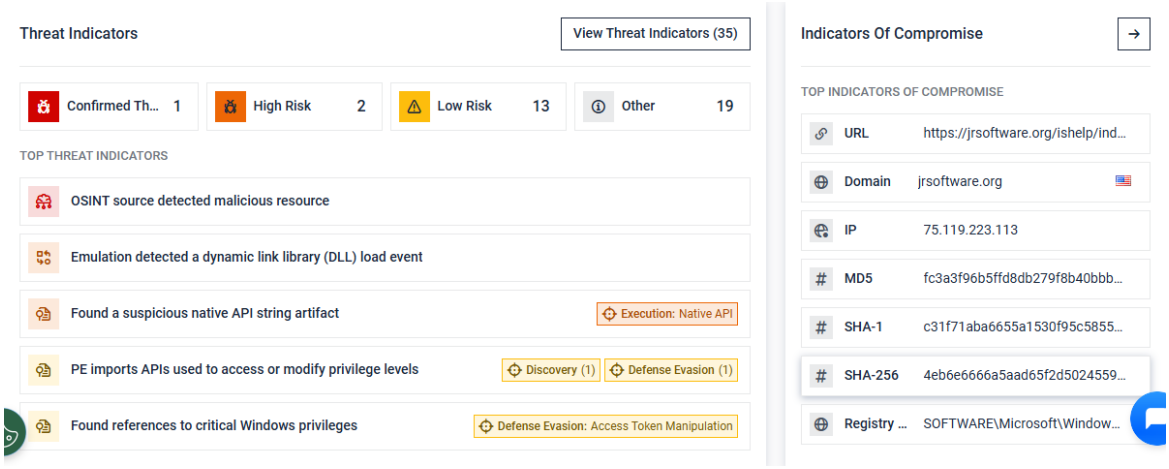


Figure 7.16: FileScan emulation-indicator summary for the selected sample. Used here only for indicator-level corroboration; behavioral claims are drawn from ANY.RUN, Triage, and CAPE.

FileScan's indicator list corroborates packing, installer-wrapper, and environment-discovery signals already seen in the other sources. It does not add behavior beyond what Triage and CAPE already record.

7.7 Cross-Source Behavior Correlation

Table 7.1: Dynamic behavior correlation across sources

Behavior	Local FLARE VM	ANY.RUN	Triage	CAPE	Interpretation	Conf.
User-launched EXE	Blocked at launch by AV	Launch warning shown	Sample target shown	EXE package execution	Selected artifact was launched as an EXE	High
Decoy installer / UI	Not observed	HWMonitor UI shown	Installer tag	Installer behavior	Supports legitimate-looking hardware-monitor theme	High
Temp extraction / dropped EXE	Not observed	Installed CPUID/H-WINFO files shown	Reported	Reported	Installer-wrapper plus dropped child is well-supported	High
PowerShell activity	Not observed	Task Manager + behavioral alert	Not explicit	Not explicit	Supports script/loader activity; command line and script body not recovered	Medium-High

MSBuild invocation	Not observed	Behavioral alert reports MSBuild project path	Not shown	Not shown	Suspicious LOLBin abuse; project file contents not recovered	Medium
System language / location discovery	Not observed	Not directly shown	Reported	Language-related registry reads shown	Environment profiling indicator	Medium
Memory API behavior	Not observed	Not directly shown	`WriteProcessMemory` reported	RWX + memory-protection changes	Consistent with in-memory unpacking; not proof of injection	Medium
Network activity	Not observed	Interactive traffic visible but not validated as C2	Not detailed	No DNS/TCP/UDP/HTTP(S) recorded	No confirmed live C2 contact in this evidence set	Low
Persistence	Not observed	Registry-resident PowerShell alert only	Not shown	Not shown	Suspicious but not confirmed durable persistence	Low-Medium

7.8 Limitations

The dynamic evidence is sufficient to characterize the sample as a decoy installer plus PowerShell/MSBuild loader candidate associated with STXRAT, but several specific items remain unrecovered: the PowerShell command line and any pipe-fed script body; the exact registry key and value used to launch the hidden PowerShell; the contents of the referenced `c_3791.proj` MSBuild project file; hashes for the installed files under `C:\T1\textbackslashProgramFiles\T1\textbackslashCPUID\T1\textbackslashHWINFO`; and any unpacked STX RAT payload bytes. CAPE's null network result also means live C2 cannot be claimed from this evidence set, even though family-level research describes a specific C2 protocol [3].

Network indicators from public sandboxes must not be treated as C2 without corroboration. Certificate checks, update traffic, CDN traffic, and generic Microsoft endpoints visible in any sandbox session are not STX RAT C2 indicators unless tied to decoded configuration, protocol fingerprinting, repeated beaconing, or trusted family-level reporting.

8 Correlation and Defensive Analysis

8.1 Probable Execution Chain

Combining static and sandbox evidence, the supported execution chain is:

1. the user (or an automated sandbox) launches the installer-themed EXE `HWiNFO_Monitor_Setup.exe`;
2. the Inno Setup wrapper extracts content into a temporary `is-*.tmp` directory (Triage, CAPE);
3. a temporary child binary `HWiNFO_Monitor_Setup.tmp` executes from that directory (Triage, CAPE);
4. the visible installer flow places legitimate-looking files under `C:\T1\textbackslashProgramFiles\T1\textbackslashCPUID\T1\textbackslashHWINFO` as a decoy (ANY.RUN);
5. a hidden-window PowerShell process is launched from a registry-resident command and invokes `MSBuild.exe` against an analyst-relevant project file in `AppData\T1\textbackslashLocal\T1\textbackslashMicrosoft\T1\textbackslashMSBuild` (ANY.RUN behavioral alert);
6. environment profiling and memory operations consistent with an in-memory loader occur (CAPE: RWX, memory protection changes; Triage: `WriteProcessMemory`; language/locale reads);
7. the visible installer path completes with an error (missing `HWINFO.exe`) while the hidden execution path continues (ANY.RUN);
8. live C2 contact and follow-on operator tasking are not confirmed in this evidence set.

Public family-level research [3] adds an alternative initial-access path (VBScript → JScript → PowerShell in-memory loader) that is not the path observed here; for this sample, the entry point is the installer-themed EXE itself.

8.2 MITRE ATT&CK Mapping

Table 8.1: MITRE ATT&CK mapping with evidence basis

Tactic	ID	Technique name	Evidence basis	Evidence type	Conf.
Execution	T1204.002	User Execution: Malicious File	Installer-themed EXE; ANY.RUN launch warning; Triage and CAPE EXE execution [9]	Public sandbox	High
Defense Evasion	T1027	Obfuscated Files or Information	Overlay, high-entropy executable sections, packer indicators (Ch. 6); CAPE overlay/entropy view [5]	Local static + public sandbox	High
Execution	T1059.001	PowerShell	ANY.RUN Task Manager and behavioral alert showing hidden-window PowerShell launched from a registry-resident command [6]	Public sandbox	Medium-High
Defense Evasion	T1218.014	Signed Binary Proxy Execution: MSBuild	ANY.RUN behavioral alert: PowerShell invokes `MSBuild.exe` against a user-AppData project file [3]	Public sandbox	Medium
Defense Evasion	T1497	Sandbox Evasion / Anti-Analysis	Local anti-debug indicators (Ch. 6); CAPE mouse-movement check and hardware-ID queries [10]	Local static + public sandbox	Medium
Discovery	T1614.001	System Language Discovery	Triage signature; CAPE language-related registry reads [8]	Public sandbox	Medium
Defense Evasion	T1055	Process Injection (suggestive)	Triage `WriteProcessMemory`; CAPE RWX allocation and memory-protection changes; not directly demonstrated [7]	Public sandbox	Medium
Persistence	T1547.001	Registry Run Keys / Startup Folder (suggestive)	ANY.RUN alert references PowerShell launched from a registry-resident command; exact key/value not recovered	Public sandbox	Low-Medium
Command and Control	T1095	Non-Application Layer Protocol	Family-level reporting of proprietary TCP; not observed in this evidence set [3]	Family research	Low
Command and Control	T1090.003	Multi-hop Proxy (Tor)	Family-level reporting of onion fallback; not observed in this evidence set [3]	Family research	Low
Credential Access	T1555	Credentials from Password Stores	Family-level capability; no local or sandbox credential-store access evidence in this set [3]	Family research	Low

8.3 Indicators of Compromise

Table 8.2: Sample-level IOCs

Type	Defanged value	Basis
SHA256	eefc0f986dd3ea376a4a54f80ce0dc3e6491165aefdd7d5d6005da3892ce248f	Local + public

SHA1	02a53d660332c25af623bbb7df57c2aad1b0b91b	Local + public
MD5	cdc459a866361463d719bc89622300f3	Local + public
Filename theme	`HWiNFO_Monitor_Setup.exe`	Local + public
File type	PE32 executable, Intel 80386, GUI	Local + public
File size	4,233,610 bytes	Local + public
imphash	`88016fcdef7f227c62171d0afad9aae4`	Local

Table 8.3: ANY.RUN interactive-session artifacts

Type	**Defanged value**	**Basis**
Launch path	`C:\T1\textbackslashUsers\T1\textbackslashadmin\T1\textbackslashDownloads\T1\textbackslashHWiNFO_Monitor_Setup.exe`	ANY.RUN session
Installed folder	`C:\T1\textbackslashProgramFiles\T1\textbackslashCPUID\T1\textbackslashHWINFO`	ANY.RUN session
Installed EXE	`HWMonitor.exe`	ANY.RUN session
Installed DLL	`CRYPTBASE.dll`	ANY.RUN session
Hidden PowerShell command (alert)	`powershell-WindowStylehidden-CommandC:\Windows\Microsoft.NET\Framework64\v4.0.30319\MSBuild.exeC:\Users\admin\AppData\Local\Microsoft\MSBuild\c_3791.proj`	ANY.RUN session
Referenced MSBuild project	`C:\T1\textbackslashUsers\T1\textbackslashadmin\T1\textbackslashAppData\T1\textbackslashLocal\T1\textbackslashMicrosoft\T1\textbackslashMSBuild\T1\textbackslashc_3791.proj`	ANY.RUN session

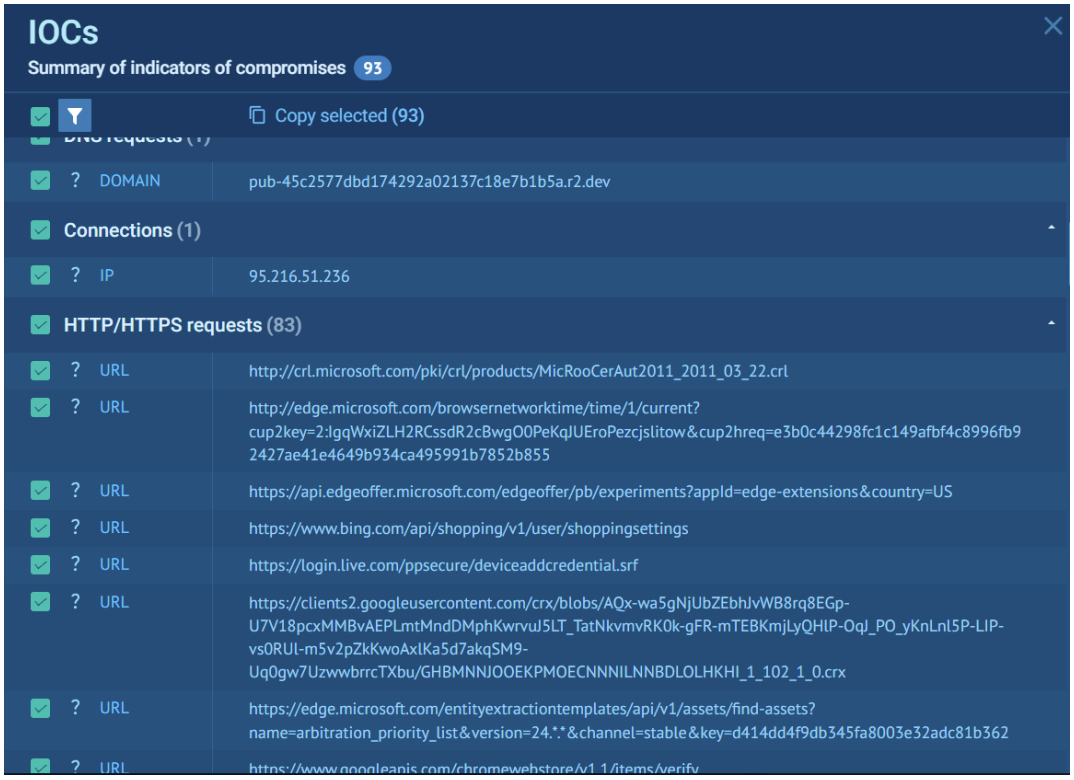
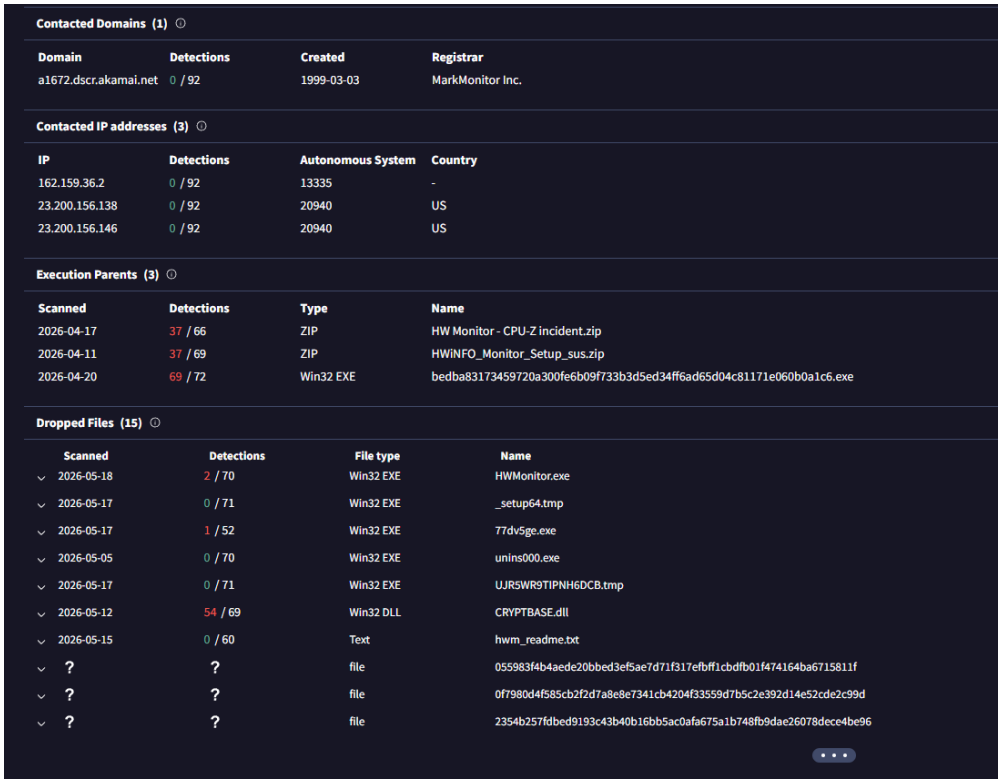


Figure 8.1: ANY.RUN IOC summary showing the sandbox's extracted domains, connections, and HTTP/HTTPS requests. These are treated as sandbox IOCs and triage leads, not confirmed STX RAT C2 for this sample.

CRYPTBASE.dll enrichment

The ANY.RUN file view shows `CRYPTBASE.dll` under the installed CPUID/HWINFO folder. VirusTotal relation and detection screenshots provide public enrichment for similarly named or related DLL artifacts. This is useful for investigation, but the report does not claim that the installed `CRYPTBASE.dll` is malicious unless its exact hash is recovered and matched.



Contacted Domains (1)

Domain	Detections	Created	Registrar
a1672.dscr.akamai.net	0 / 92	1999-03-03	MarkMonitor Inc.

Contacted IP addresses (3)

IP	Detections	Autonomous System	Country
162.159.36.2	0 / 92	13335	-
23.200.156.138	0 / 92	20940	US
23.200.156.146	0 / 92	20940	US

Execution Parents (3)

Scanned	Detections	Type	Name
2026-04-17	37 / 66	ZIP	HW Monitor - CPU-Z incident.zip
2026-04-11	37 / 69	ZIP	HWiNFO_Monitor_Setup_sus.zip
2026-04-20	69 / 72	Win32 EXE	bedba83173459720a300fe6b09f733b3d5ed34ff6ad65d04c81171e060b0a1c6.exe

Dropped Files (15)

Scanned	Detections	File type	Name
2026-05-18	2 / 70	Win32 EXE	HWMonitor.exe
2026-05-17	0 / 71	Win32 EXE	_setup64.tmp
2026-05-17	1 / 52	Win32 EXE	77dv5ge.exe
2026-05-05	0 / 70	Win32 EXE	unins000.exe
2026-05-17	0 / 71	Win32 EXE	UJR5WR9TIPNH6DCB.tmp
2026-05-12	54 / 69	Win32 DLL	CRYPTBASE.dll
2026-05-15	0 / 60	Text	hwm_readme.txt
?	?	file	055983f4b4aede20bbed3ef5ae7d71f317efbff1cbdfb01f474164ba6715811f
?	?	file	0f7980d4f585cb2f2d7a8e8e7341cb4204f33559d7b5c2e392d14e52cde2c99d
?	?	file	2354b257fdbed9193c43b40b16bb5ac0afa675a1b748fb9dae26078dece4be96

Figure 8.2: VirusTotal relations view showing public relationships around a `CRYPTBASE.dll`-named artifact. Used as enrichment for the suspicious installed DLL lead; exact local or sandbox file-hash matching remains future work.

Figure 8.3: VirusTotal detection view for a `CRYPTBASE.dll`-named artifact. This supports follow-up priority for DLL side-loading analysis but is not treated as proof for the ANY.RUN-installed DLL without hash confirmation.

Table 8.4: Automated sandbox dropped/unpacked artifacts

Type	Defanged value	Basis
Temporary EXE	`%LocalAppData%\Temp\is-U85DTLA4LM.tmp\HWiNFO_Monitor_Setup.tmp`	CAPE
Temporary setup helper	`%LocalAppData%\Temp\is-345P5RZPYA.tmp\_isetup\_setup64.tmp`	CAPE
Temporary EXE	`C:\Users\Admin\AppData\Local\Temp\is-7BAF5XXKHI.tmp\HWiNFO_Monitor_Setup.tmp`	Triage

Unpacked SHA256 candidate	cd0cd2b40bdea39eae382825ee30645b83cbd4164cf7e94b419bc720ea137c19	Public platform

Table 8.5: Sandbox network indicators

Type	Defanged value	Source
Delivery URL	`hxxps://pub-45c2577dbd174292a02137c18e7b1b5a[.]r2[.]dev/hwmonitor/HWiNFO_Monitor_Setup.exe`	ANY.RUN
Network result	No DNS, TCP, UDP, or HTTP(S) requests recorded	CAPE

Table 8.6: Family-level infrastructure and delivery candidates (not observed in this sample's evidence set)

Type	Defanged value	Basis
Initial-stage download IP	`147.45.178[.]61`	Family research [3]
Elevated script command	`"C:\Windows\System32\wscript.exe""C:\Users\<Username>\AppData\Local\Temp\business-structure.xlsx.js"/elevated`	Family research [3]
Fileless PowerShell command	`powershell.exe-Command"[Console]::In.ReadToEnd()` Invoke-Expression"|	Family research [3]
C2 IP	`95.216.51[.]236`	Family research [3]
C2 port	`31415/tcp`	Family research [3]
Onion C2	`yu7sbzk2tgm4vv56qgvsq44wnwgct6sven4akbb2n3onp46f42fcstid[.]onion`	Family research [3]

8.4 Detection Opportunities

Defenders should hunt for installer-themed executables launched from user-writable paths, particularly files mimicking hardware-monitoring tools. Priority telemetry: creation of `is-*.tmp` directories under Temp; execution of temporary `*.tmp` binaries from those directories; `/SL5=` command-line tokens characteristic of Inno Setup helper launches; RWX

memory allocation and memory-protection changes near installer execution; locale and environment discovery; short-lived child processes spawned from installer extraction directories.

The ANY.RUN-observed behavior adds higher-value detection leads specific to this sample: hidden-window PowerShell launched from a registry-resident command; PowerShell invoking `MSBuild.exe`; MSBuild project files in `AppData\T1\textbackslashLocal\T1\textbackslashMicrosoft\T1\textbackslashMSBuild`; and creation of files under `ProgramFiles\T1\textbackslashCPUID\T1\textbackslashHWINFO` when no legitimate CPUID software is approved in the environment. Family-level detections should cover script staging through `wscript.exe` and PowerShell that reads from standard input and invokes the received content [3].

Network detection should separate three categories: delivery URLs (sample download), generic platform traffic (certificates, updates, CDN, Microsoft endpoints), and confirmed malware C2. None of the public-sandbox traffic captured here qualifies as confirmed STX RAT C2 for this sample.

8.5 Incident Response Recommendations

If this sample is found on a host: isolate the endpoint at the network layer, preserve volatile memory before powering down, collect process and network telemetry, hash any files in Temp extraction directories (`is-*.tmp`), enumerate and preserve registry Run-key style autostart locations and the `AppData\T1\textbackslashLocal\T1\textbackslashMicrosoft\T1\textbackslashMSBuild` project directory, and preserve browser and credential-store evidence only if endpoint telemetry shows access to those stores. Credential rotation and session invalidation should follow only if credential access or successful C2 tasking is confirmed by local evidence.

8.6 Hardening Recommendations

Constrain execution of unsigned or untrusted binaries from user-writable paths using AppLocker or WDAC. Apply ASR rules that block child-process creation by Office and script hosts and block credential-store theft. Enable Sysmon (or equivalent EDR telemetry) for process creation with command line, file creation under Temp, registry-key writes in autostart locations, memory-related events where supported, and outbound network connections. Restrict or monitor PowerShell script-block logging and MSBuild execution from non-developer accounts; both are recurring living-off-the-land vectors.

9 Conclusion

9.1 Key Findings

The selected sample is a PE32 Windows GUI executable associated with STXRAT in public malware-intelligence sources. Local static evidence confirms an installer-themed artifact with Inno Setup strings, setup-loader exports, packing indicators, suspicious entropy, an overlay, and imports consistent with setup staging and loader-like behavior. Public sandbox evidence from ANY.RUN, Hatching Triage, CAPE, and FileScan shows installer-wrapper execution, temporary dropped-child execution, a hardware-monitor decoy UI, and a hidden-window PowerShell process launched from a registry-resident command that invokes `MSBuild.exe` against a user-AppData project file. CAPE's behavior summary additionally records RWX memory allocation and memory-protection changes consistent with an in-memory loader stage.

9.2 Limitations

Local interactive execution did not produce telemetry because the FLARE VM's anti-malware engine blocked the launch; ANY.RUN was used as a substitute, with the consequence that the strongest interactive evidence is public sandbox evidence rather than local FLARE VM evidence. The PowerShell command body, the exact registry key and value, the MSBuild project file contents, and hashes of the installed files were not captured. `capa` produced no usable output and is excluded from analysis. Family-level capabilities (credential theft, hidden remote desktop, full C2 cryptographic stack) are not demonstrated by this sample's evidence set; they remain documented capabilities of the family in public reporting.

9.3 Final Assessment

`HWiNFO_Monitor_Setup.exe` should be handled as a malicious STXRAT-associated installer-and-loader artifact. The combined evidence supports blocking the selected hash, hunting against the wrapper-plus-dropped-child plus hidden-PowerShell-and-MSBuild pattern, and treating any host showing those artifacts as compromised pending forensic review. Validated runtime characterization of the final unpacked payload, persistence durability, and live C2 behavior remains future work and requires either AV-bypass in a controlled lab or recovery of the unpacked payload for separate analysis.

A Static Evidence Checklist

Table A.1: Static evidence checklist

Evidence	Status	Folder	File
Hash and file type	Present	`figures/static/`	`hash_filetype_verification_panel.png`
Original hash screenshot	Present	`figures/static/`	`hash_filetype_terminal.png`
PE metadata	Present	`figures/static/`	`metadata.png`
rabin2 metadata	Present	`figures/static/`	`rabin2_metadata.png`
Sections and entropy	Present	`figures/static/`	`pe_sections_entropy.png`
Imports and exports	Present	`figures/static/`	`imports_exports.png`
peframe output	Present	`output/static/`	`peframe_output.txt`
FLOSS output	Present	`output/static/`	`floss_output.txt`
strings output	Present	`output/static/`	`strings_output.txt`
rabin imports	Present	`output/static/`	`rabin_imports.txt`
rabin exports	Present	`output/static/`	`rabin_exports.txt`
capa output	Present but empty (excluded)	`output/static/`	`capa_output.txt`
Ghidra screenshots	Not collected	—	—

B Dynamic Evidence Checklist

B.1 Local FLARE VM Evidence Status

Table B.1: Local FLARE VM evidence status

Evidence	Status	Required proof or current basis
Clean VM snapshot	Taken (not exported)	Baseline before sample transfer
Network isolation	Configured	Host-only networking
Hash before execution	Verified	Matches selected SHA256
Process Explorer baseline	Collected	`dynamic_procexplorer_baseline.png`
Procmon capture setup	Collected	`dynamic_procmon_hwinfo_filter.png`; no events captured (AV block)
Sample execution	Blocked by AV	Local AV engine detected and terminated launch
Procmon event export	Not produced	Execution did not proceed
Wireshark capture	Not produced	Execution did not proceed
FakeNet-NG / INetSim	Not engaged	Execution did not proceed
Regshot before/after	Not produced	Execution did not proceed
Autoruns review	Not produced	Execution did not proceed
Sysmon / Event Viewer	Not produced	Execution did not proceed
Dropped/installed file hashes	Not produced	No local files installed

B.2 Public Sandbox Evidence Status

Table B.2: Public sandbox evidence status

Evidence	Status	Required proof or current basis
ANY.RUN interactive session	Collected	Launch warning, decoy UI, Task Manager view, behavioral alert, installed files, installer error
Hatching Triage automated run	Collected	Signatures, process tree, score `7/10`
CAPE automated run	Collected	Overview, behavior summary, file activity, null network result
FileScan emulation	Collected	Indicator summary screenshot
ANY.RUN raw network log	Not exported	Only summary view captured
ANY.RUN command-line capture	Not exported	PowerShell command body and MSBuild project contents not recovered
Triage raw report	Not exported	Only screenshots captured
CAPE raw report	Partial	HTML report retained; PCAP/PML not exported

B.3 Screenshots Used

Table B.3: Local and sandbox screenshots used

Screenshot path	Evidence type	Finding supported
`figures/dynamic/dynamic_procexplorer_baseline.png`	Local FLARE VM	Analysis tools running before launch
`figures/dynamic/dynamic_procmon_hwinfo_filter.png`	Local FLARE VM	Procmon filter configured before launch
`figures/dynamic/dynamic_sample_launch_warning.png`	ANY.RUN session	User-launched EXE path in the sandbox
`figures/dynamic/dynamic_hwmonitor_decoy_ui.png`	ANY.RUN session	Visible HWMonitor-style decoy UI
`figures/dynamic/dynamic_taskmanager_powershell_child.png`	ANY.RUN session	HWMonitor process group includes `conhost.exe` and PowerShell
`figures/dynamic/dynamic_registry_key_powershell_alert.png`	ANY.RUN session	Behavioral alert: hidden PowerShell launched from registry key, invokes MSBuild

figures/dynamic/dynamic_installed_files_cpuid_hwinfo.png	ANY.RUN session	Files placed under CPUID/HWINFO folder
figures/proposed/dynamic/anyrun_cryptbase_dll_properties.png	ANY.RUN session	File properties for suspicious installed `CRYPTBASE.dll`
figures/proposed/dynamic/anyrun_file_modifications.png	ANY.RUN session	File-modification context during sandbox execution
figures/dynamic/dynamic_installer_error_after_execution.png	ANY.RUN session	Installer error for missing `HWINFO.exe`
figures/proposed/dynamic/anyrun_ioc_summary.png	ANY.RUN session	Extracted sandbox IOC summary
figures/proposed/automated/virustotal_properties_file_history.png	VirusTotal	File metadata and submission-history context
figures/proposed/automated/virustotal_behavior_summary.png	VirusTotal	Public behavioral summary for triage context
figures/proposed/automated/virustotal_cryptbase_relations.png	VirusTotal	Public relationship context for `CRYPTBASE.dll`-named artifact
figures/proposed/automated/virustotal_cryptbase_detection.png	VirusTotal	Public detection view for `CRYPTBASE.dll`-named artifact
figures/proposed/static/die_pe32_overview.png	Detect It Easy	PE32 and installer/packer overview
figures/proposed/static/pescan_section_summary.png	pe-scan	Packed-file and PE section summary
figures/proposed/static/die_imports.png	Detect It Easy	Supplementary import view
figures/proposed/static/die_exports.png	Detect It Easy	Supplementary export view
figures/proposed/automated/virustotal_network_communications.png	VirusTotal	Supplementary network-communication triage view
figures/proposed/automated/virustotal_mitre_tactics.png	VirusTotal	Supplementary ATT&CK mapping view

figures/automated/triage_signatures.png	Hatching Triage	Dropped-EXE, language discovery, Inno Setup, WriteProcessMemory
figures/automated/triage_process_tree.png	Hatching Triage	Temporary child process from extraction directory
figures/automated/cape_overview.png	CAPE	EXE package execution, hashes, file type, entropy/overlay
figures/automated/cape_behavior_summary.png	CAPE	Dropped-binary execution, RWX memory, anti-analysis checks
figures/automated/cape_file_activity.png	CAPE	Temp extraction and installer-related file activity
figures/automated/filescan_indicators.png	FileScan	Emulation indicator corroboration

C Supplementary Screenshot Evidence

This appendix stores supporting screenshots that are useful for auditability but too detailed or repetitive for the main narrative. They are grouped here so the main report remains readable while still preserving the analyst's original evidence trail.

C.1 Supplementary Static Tooling

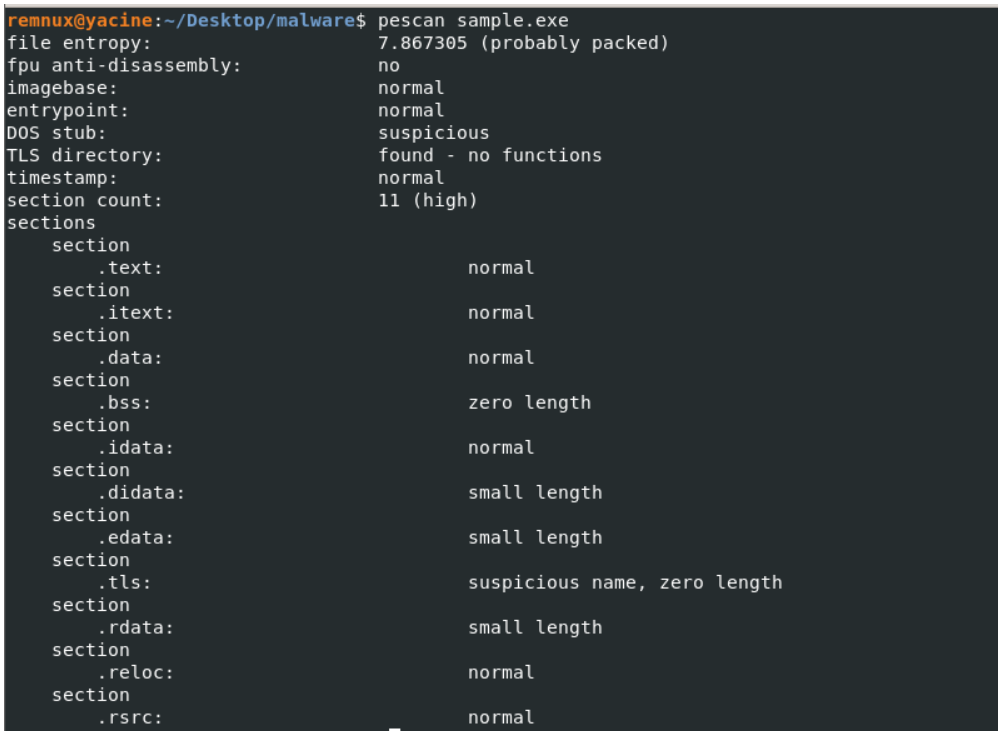


Figure C.1: pe-scan terminal output for `sample.exe`. It reports the file's packed status, PE characteristics, and section-level observations; this is supplementary static evidence for the packing discussion in Chapter 6.

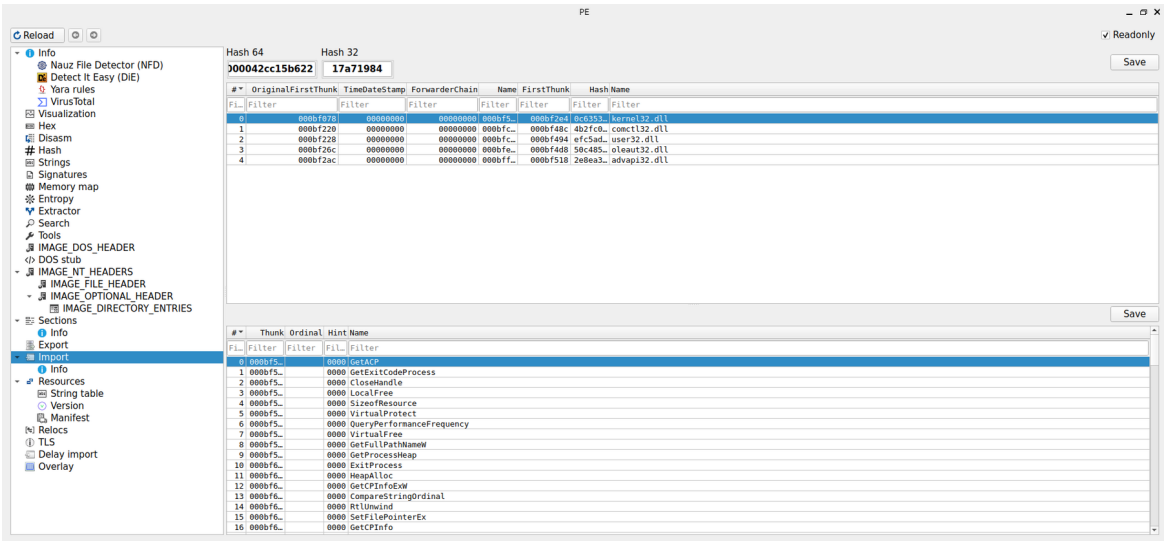


Figure C.2: Detect It Easy import view for the selected sample. This supplements the rabin2 import screenshot by showing imported DLLs and APIs in a GUI view.

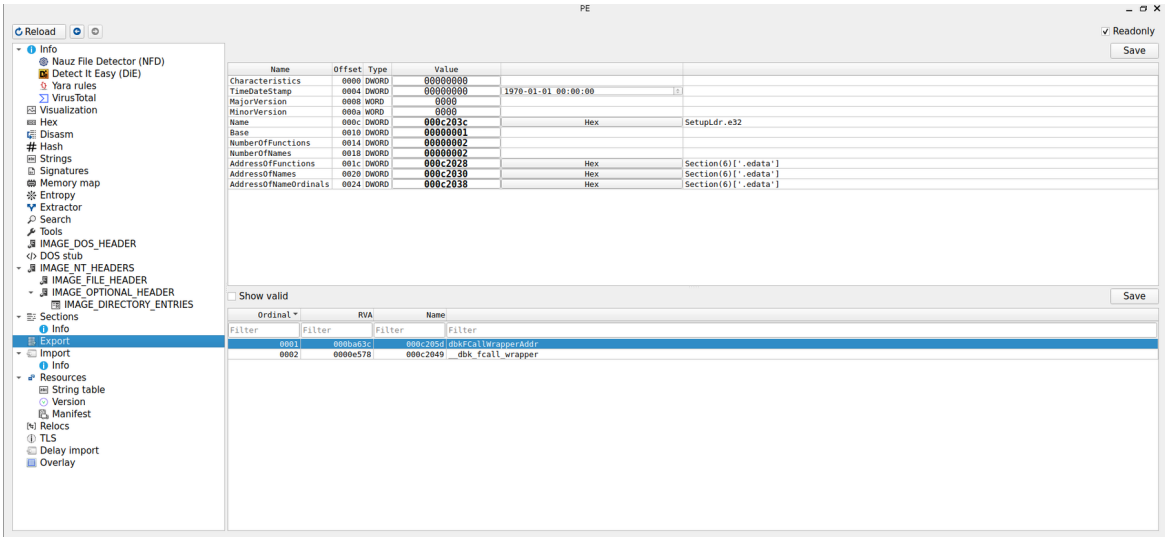


Figure C.3: Detect It Easy export view for the selected sample. This supplements the export analysis and supports the setup-loader interpretation.

C.2 Supplementary VirusTotal Views

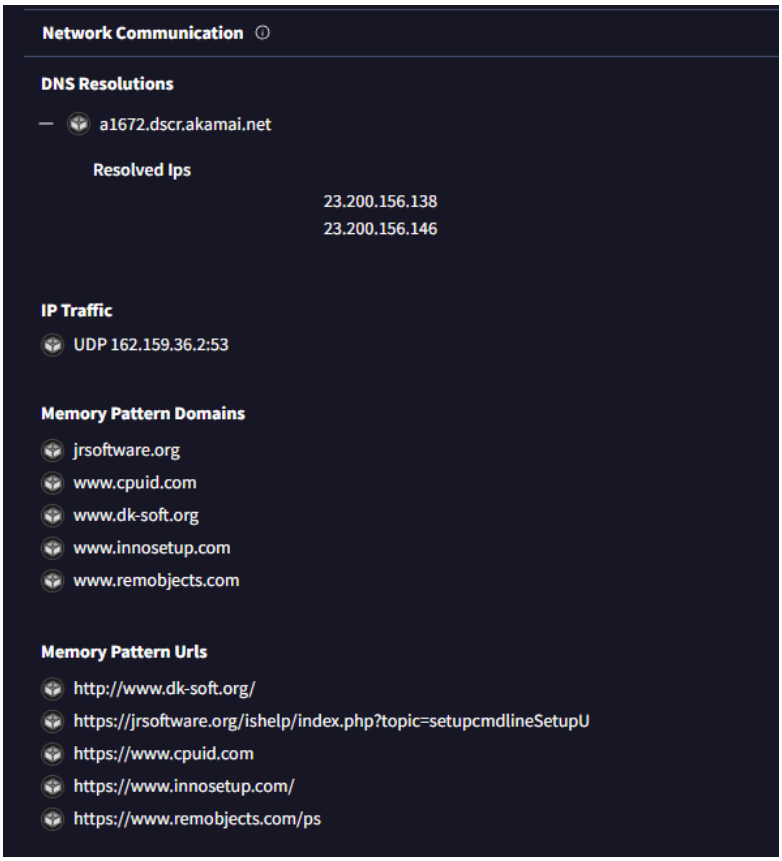


Figure C.4: VirusTotal network-communications view. Included as public triage context only; the report does not treat this as confirmed STX RAT C2 because sandbox and platform traffic can include benign browser, certificate, update, and CDN activity.

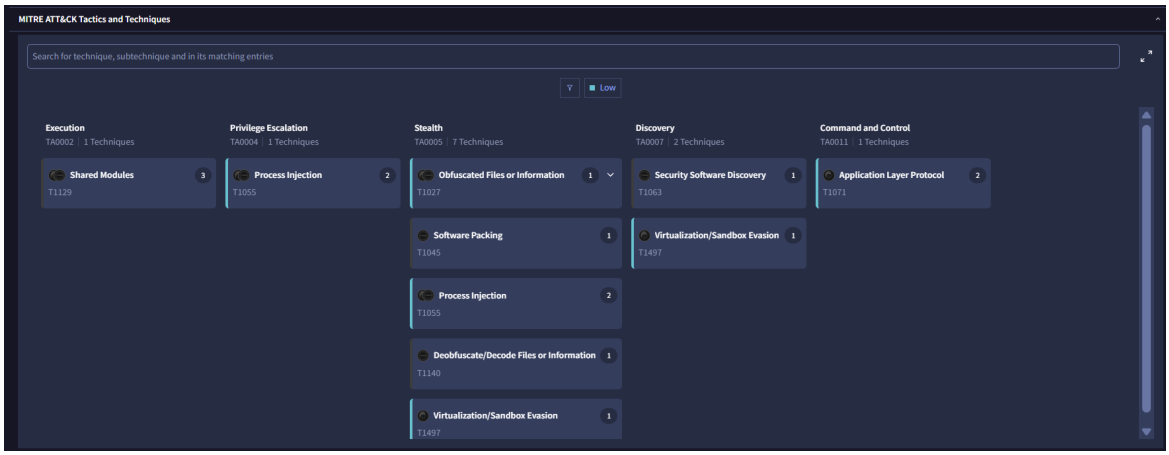


Figure C.5: VirusTotal MITRE tactics and techniques view. Included as supplementary platform mapping; the report's ATT&CK table remains the authoritative mapping because it separates direct evidence from public-sandbox and family-level claims.

D Commands Used and Source Acknowledgments

D.1 Static Workflow Commands

The following commands describe the static workflow represented by the collected artifacts. They are listed for reproducibility and were executed only inside the isolated analysis environment.

```
sha256sum sample.exe
sha1sum sample.exe
md5sum sample.exe
file sample.exe
peframe sample.exe > output/static/peframe_output.txt
rabin2 -i sample.exe > output/static/rabin_imports.txt
rabin2 -E sample.exe > output/static/rabin_exports.txt
strings -a -n 6 sample.exe > output/static/strings_output.txt
floss sample.exe > output/static/floss_output.txt
capa sample.exe > output/static/capa_output.txt
```

The empty `capa_output.txt` confirms that capa did not produce capability matches for this sample under the local configuration; capa is therefore not cited as evidence anywhere in this report.

D.2 Public Reporting Image Acquisition

The family-level research figures shown in Chapter 2 are screenshots taken from the public report cited as [3] (eSentire Threat Response Unit, "A New RAT in 2026 with Infostealer Capabilities"). They were downloaded as static images using `Invoke-WebRequest` and stored locally under `figures/theory/`. No malware was executed during image collection; only static image URLs from the publisher's CDN were fetched.

```
Invoke-WebRequest -Uri "https://esentire-dot-com-assets.s3.amazonaws.com/
   assetsV3/Blog/Blog-Images/A-new-RAT-in-2026-with-Infostealer-Capabilities-1.
   png"  -OutFile "figures/theory/esentire_initial_wscript.png"
Invoke-WebRequest -Uri "https://esentire-dot-com-assets.s3.amazonaws.com/
   assetsV3/Blog/Blog-Images/A-new-RAT-in-2026-with-Infostealer-Capabilities-3.
   png"  -OutFile "figures/theory/esentire_powershell_loader.png"
Invoke-WebRequest -Uri "https://esentire-dot-com-assets.s3.amazonaws.com/
   assetsV3/Blog/Blog-Images/A-new-RAT-in-2026-with-Infostealer-Capabilities-4.
   png"  -OutFile "figures/theory/esentire_unpacking_routine.png"
Invoke-WebRequest -Uri "https://esentire-dot-com-assets.s3.amazonaws.com/
   assetsV3/Blog/Blog-Images/A-new-RAT-in-2026-with-Infostealer-Capabilities
   -12.png" -OutFile "figures/theory/esentire_antivm_check.png"
Invoke-WebRequest -Uri "https://esentire-dot-com-assets.s3.amazonaws.com/
   assetsV3/Blog/Blog-Images/A-new-RAT-in-2026-with-Infostealer-Capabilities
   -17.png" -OutFile "figures/theory/esentire_c2_key_exchange.png"
Invoke-WebRequest -Uri "https://esentire-dot-com-assets.s3.amazonaws.com/
   assetsV3/Blog/Blog-Images/A-new-RAT-in-2026-with-Infostealer-Capabilities
   -26.png" -OutFile "figures/theory/esentire_fileless_powershell.png"
```

The family-level command lines reproduced in Chapter 2 (WScript-elevated launch; PowerShell standard-input pipe) are taken from the same public report and are included for defensive understanding. They are not local command-line evidence for the selected `HWiNFO_Monitor_Setup.exe` sample.

Bibliography

[1] abuse.ch MalwareBazaar. Selected STXRAT Sample Report. https://bazaar.abuse.ch/sample/eefc0f986dd3ea376a4a54f80ce0dc3e6491165aefdd7d5d6005da3892ce248f/, 2026. Accessed 2026-05-18.

[2] abuse.ch MalwareBazaar. STXRAT Signature Page. https://bazaar.abuse.ch/browse/signature/STXRAT/, 2026. Accessed 2026-05-18.

[3] eSentire Threat Response Unit. STX RAT: A new RAT in 2026 with Infostealer Capabilities. https://www.esentire.com/blog/stx-rat-a-new-rat-in-2026-with-infostealer-capabilities, April 2026. Accessed 2026-05-19.

[4] Hybrid Analysis. Selected STXRAT File Report. https://hybrid-analysis.com/sample/eefc0f986dd3ea376a4a54f80ce0dc3e6491165aefdd7d5d6005da3892ce248f, 2026. Accessed 2026-05-18.

[5] MITRE ATT&CK. Obfuscated Files or Information, T1027. https://attack.mitre.org/techniques/T1027/, 2026. Accessed 2026-05-18.

[6] MITRE ATT&CK. PowerShell, T1059.001. https://attack.mitre.org/techniques/T1059/001/, 2026. Accessed 2026-05-19.

[7] MITRE ATT&CK. Process Injection, T1055. https://attack.mitre.org/techniques/T1055/, 2026. Accessed 2026-05-19.

[8] MITRE ATT&CK. System Language Discovery, T1614.001. https://attack.mitre.org/techniques/T1614/001/, 2026. Accessed 2026-05-19.

[9] MITRE ATT&CK. User Execution: Malicious File, T1204.002. https://attack.mitre.org/techniques/T1204/002/, 2026. Accessed 2026-05-18.

[10] MITRE ATT&CK. Virtualization/Sandbox Evasion, T1497. https://attack.mitre.org/techniques/T1497/, 2026. Accessed 2026-05-18.

[11] VirusTotal. Selected STXRAT File Report. https://www.virustotal.com/gui/file/eefc0f986dd3ea376a4a54f80ce0dc3e6491165aefdd7d5d6005da3892ce248f, 2026. Accessed 2026-05-18.