CHAPTER 4

Remote Triage Tools

Chapter Overview

01

WMIC

Windows Management Instrumentation for remote forensic queries

02

PowerShell

Cmdlets, Remoting, and CIM for fleet-wide triage

03

IR Frameworks

Velociraptor, GRR, OSQuery, and KAPE for enterprise-scale response

WMIC

Forensically sound remote queries

WMIC Fundamentals

wmic: command-line interface to Windows Management Instrumentation

Read-only by default — queries do not alter system state

Syntax: wmic [context] [where clause] get [properties]

Can target remote systems: wmic /node:192.168.1.10 ...

Output formats: /format:csv, /format:list, /format:htable

Key aliases: process, service, useraccount, netuse, startup, logicaldisk

Always document commands run and output collected (chain of custody)

WMIC Example Commands

Processes & Network

wmic process get Name,ProcessId,ParentProcessId,ExecutablePath

wmic process where ProcessId=1234 get CommandLine

wmic /node:HOST process list brief

netstat -naob (via cmd, not wmic)

Services & Accounts

wmic service get Name,StartName,PathName,State

wmic useraccount list full

wmic startup get Caption,Command,Location

wmic /node:HOST service where State='Running' list brief

WQL — WMI Query Language

WQL: SQL-like query language used by WMIC and CIM cmdlets

Syntax: SELECT * FROM Win32_Process WHERE Name = 'cmd.exe'

Key classes: Win32_Process, Win32_Service, Win32_UserAccount, Win32_NetworkConnection

Win32_StartupCommand: all autostart entries visible to WMI

Win32_ShadowCopy: VSS snapshots present on the system

Filtering: WHERE Name LIKE '%svch%' — find process name substrings

Remote: wmic /node:HOST /user:DOMAIN\admin /password:PWD process get Name,ProcessId

Output to CSV: wmic process get Name,ProcessId /format:csv > c:\ir\procs.csv

WMIC Remote Forensics Workflow

Step 1 — Processes: wmic /node:HOST process get Name,ProcessId,ExecutablePath,CommandLine /format:csv

Step 2 — Services: wmic /node:HOST service where State='Running' get Name,PathName,StartName /format:csv

Step 3 — Accounts: wmic /node:HOST useraccount where LocalAccount=TRUE get Name,Disabled,SID

Step 4 — Startup: wmic /node:HOST startup get Caption,Command,User

Step 5 — Network shares: wmic /node:HOST share get Name,Path,Type

Step 6 — Logon sessions: wmic /node:HOST path Win32_LogonSession get LogonType,AuthenticationPackage

Pipe to /format:csv for import into spreadsheets or SIEM

Document all commands with timestamps — forensic chain of custody

PowerShell

The modern remote triage platform

Key PowerShell Cmdlets for IR

Get-Process: running processes with path, PID, handles

Get-Service: Windows services and their status

Get-NetTCPConnection: active TCP connections (replaces netstat)

Get-LocalUser / Get-LocalGroupMember: accounts and group membership

Get-ScheduledTask: all scheduled tasks with actions

Get-ItemProperty HKLM:\...\Run: autostart registry keys

Get-FileHash: compute SHA256 of suspicious files for verification

PowerShell Remoting & CIM

PowerShell Remoting

Enable-PSRemoting (requires admin on target)

Enter-PSSession -ComputerName HOST: interactive remote shell

Invoke-Command -ComputerName HOST -ScriptBlock {...}: single commands

New-PSSession: persistent session for multi-command workflows

Requires WinRM service running on target

CIM / WMI via PowerShell

Get-CimInstance Win32_Process: modern replacement for wmic process

Get-CimInstance Win32_NetworkAdapterConfiguration

Get-CimInstance -ComputerName HOST: remote queries

Invoke-CimMethod: call WMI methods (use with caution)

CIM preferred over WMI cmdlets — uses WS-Man, not DCOM

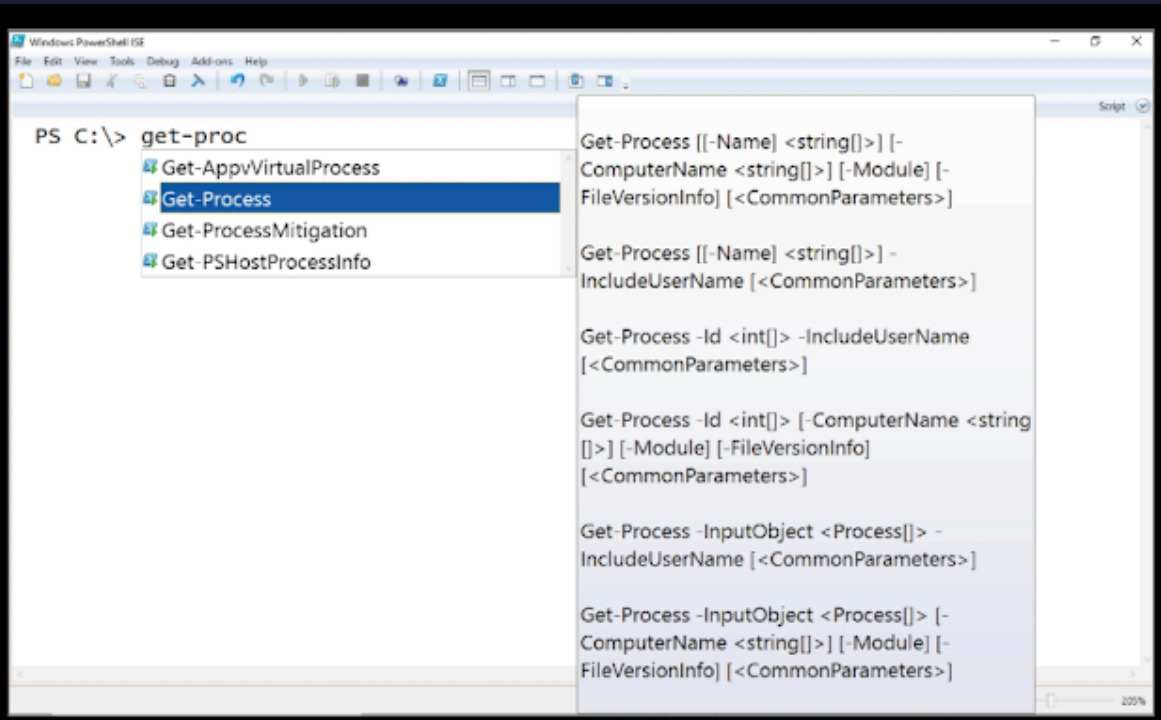
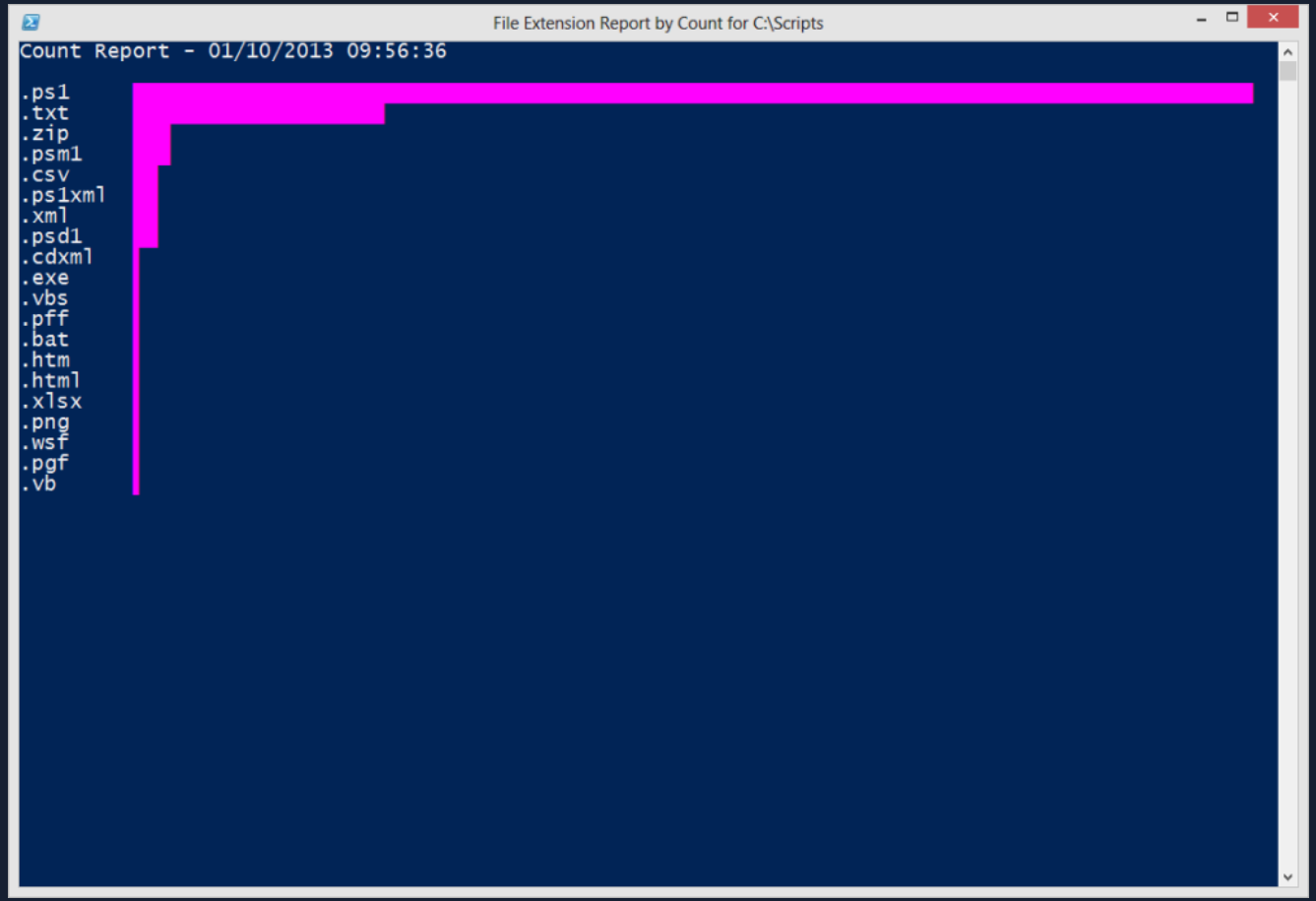


Figure 4.5: PowerShell ISE IntelliSense feature at work as the user types



PowerShell console — the IR responder's primary tool for live system triage

PowerShell Security — Execution Policy & Logging

Execution policy: controls which scripts run — NOT a security boundary (easily bypassed)

Bypass: powershell.exe -ExecutionPolicy Bypass -File script.ps1

Set centrally via GPO — local policy can be changed by attackers

Constrained Language Mode (CLM): restricts powerful .NET calls — a real mitigation

Enable Script Block Logging (4104) via GPO: captures all executed script content

Module Logging (4103): records which modules and commands were loaded

Transcription: full session transcript saved to a central share

AMSI: feeds PowerShell content to AV engine — even obfuscated scripts get inspected

PowerShell One-Liners for IR Triage

Running processes with paths: Get-Process | Select Name,Id,Path | Export-Csv processes.csv

Network connections: Get-NetTCPConnection | Select LocalPort,RemoteAddress,RemotePort,State,OwningProcess

Services: Get-Service | Where-Object {$_.Status -eq 'Running'} | Select Name,DisplayName,StartType

Scheduled tasks: Get-ScheduledTask | Where-Object {$_.State -ne 'Disabled'} | Select TaskName,TaskPath

Autostart registry: Get-ItemProperty 'HKLM:\Software\Microsoft\Windows\CurrentVersion\Run'

Recent files in TEMP: Get-ChildItem $env:TEMP -Recurse | Sort-Object LastWriteTime -Descending | Select -First 20

Local admins: Get-LocalGroupMember -Group Administrators | Select Name,PrincipalSource

File hash: Get-FileHash suspicious.exe -Algorithm SHA256

IR Frameworks

Agent-based fleet triage at scale

Incident Response Frameworks Overview

Purpose-built tools for enterprise-wide IR triage and collection

Velociraptor: open-source, VQL query language, artifact library, real-time hunts

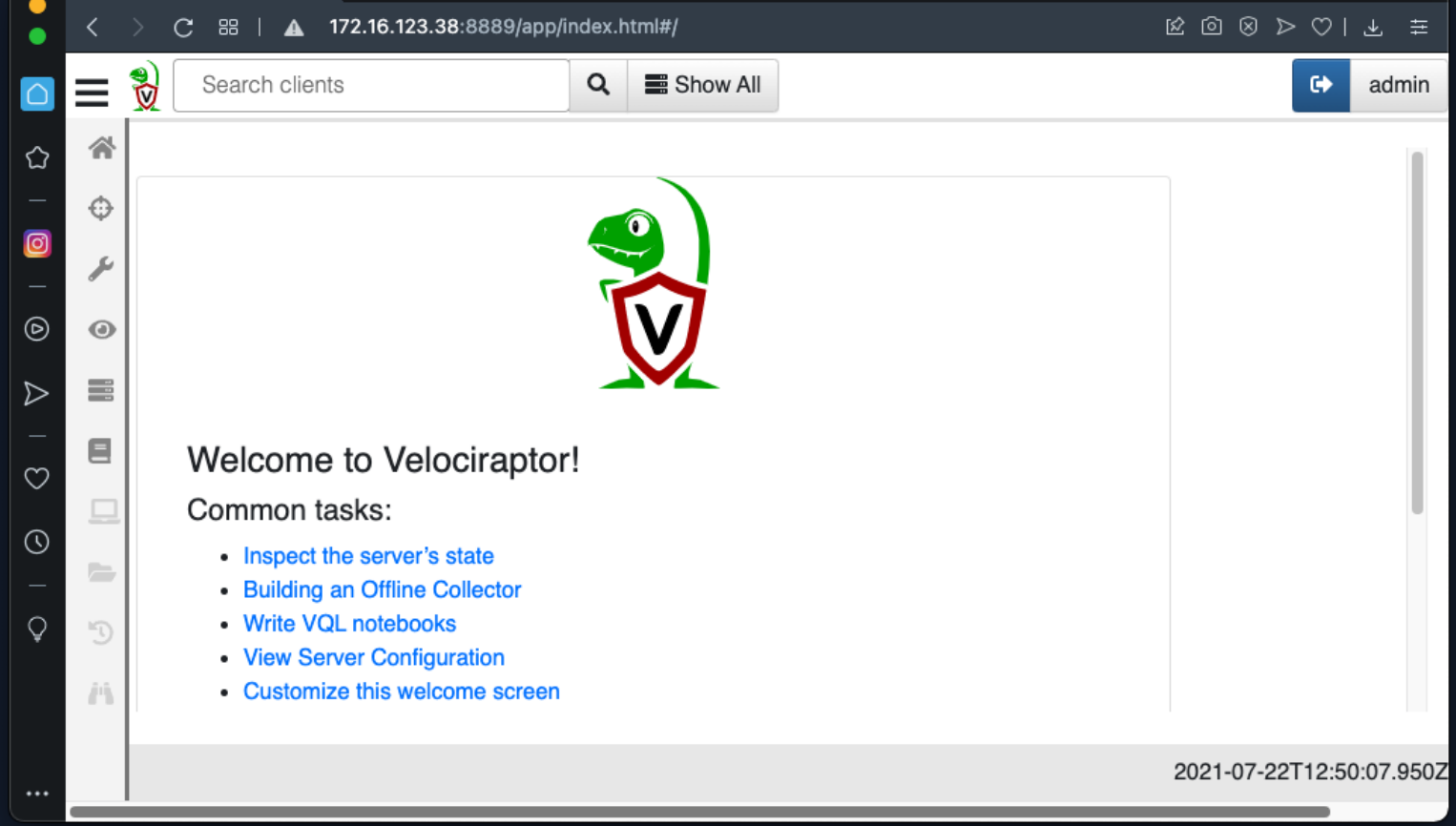
GRR Rapid Response (Google): Python agents, hunt flows, timeline analysis

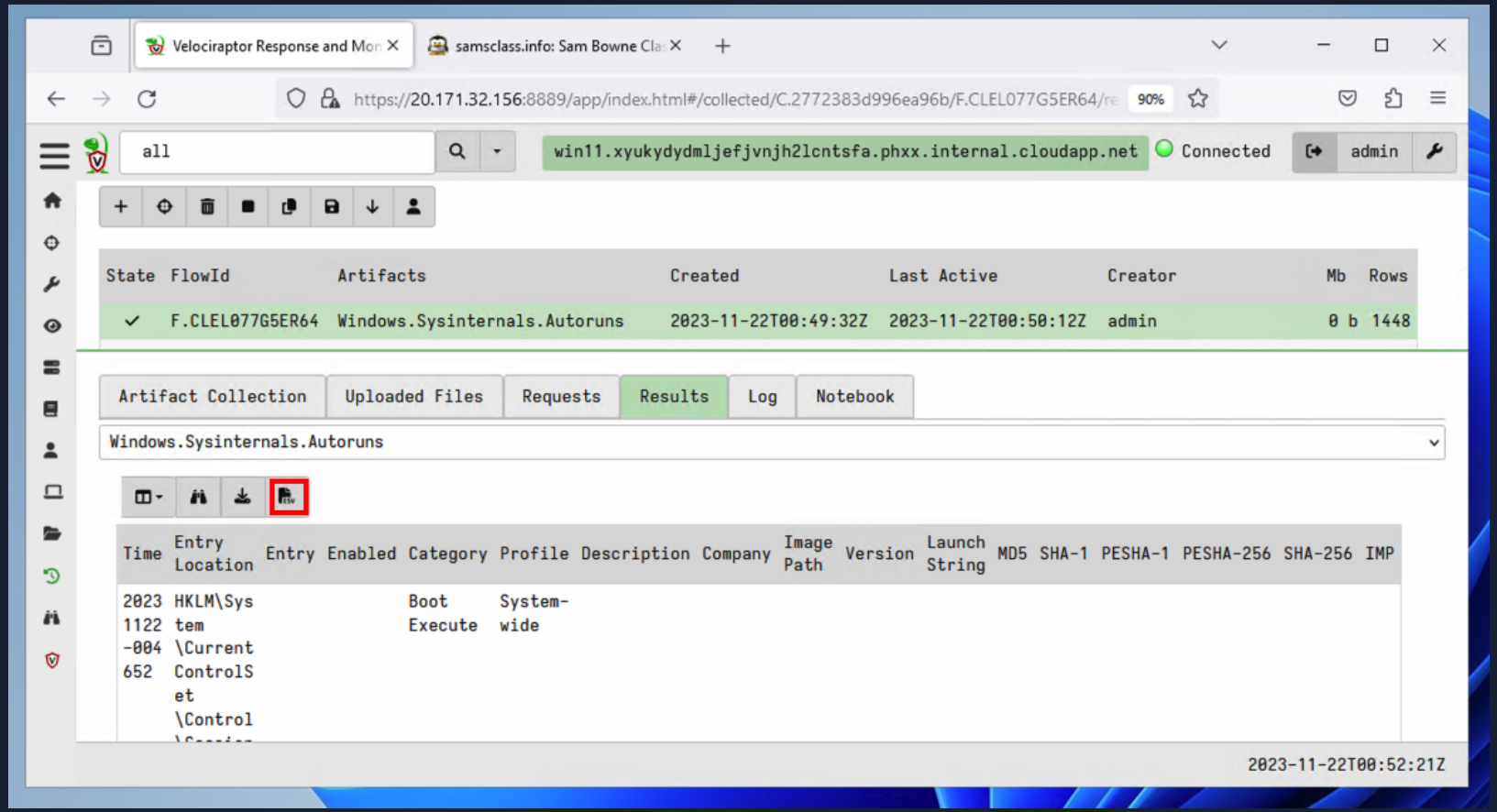
OSQuery: SQL interface to OS state — cross-platform (Windows/Mac/Linux)

KAPE (Kroll Artifact Parser and Extractor): targeted artifact collection + processing

Advantages: scale to thousands of endpoints, central logging, repeatable hunts

IR frameworks > manual WMIC/PS when scope is enterprise-wide


172.16.123.38:8889/app/index.html#/
Search clients
Show All
admin
Welcome to Velociraptor!
Common tasks:
Inspect the server's state
Building an Offline Collector
Write VQL notebooks
View Server Configuration
Customize this welcome screen
2021-07-22T12:50:07.950Z


Velociraptor Response and Mon
samsclass.info: Sam Bowne Cla
https://20.171.32.156:8889/app/index.html#/collected/C.2772383d996ea96b/F.CLEL077G5ER64/re
90%
all
win11.xyukydydmljefjvnjh2lcntsfa.phxx.internal.cloudapp.net
Connected
admin
State FlowId Artifacts Created Last Active Creator Mb Rows
F.CLEL077G5ER64 Windows.Sysinternals.Autoruns 2023-11-22T00:49:32Z 2023-11-22T00:50:12Z admin 0 b 1448
Artifact Collection
Uploaded Files
Requests
Results
Log
Notebook
Windows.Sysinternals.Autoruns
Time Entry Location Entry Enabled Category Profile Description Company Image Path Version Launch String MD5 SHA-1 PESHA-1 PESHA-256 SHA-256 IMP
2023-1122-004652 HKLM\System\CurrentControlSet\Control Boot Execute System-wide
2023-11-22T00:52:21Z

Velociraptor — VQL and Artifact Hunting

VQL (Velociraptor Query Language): SQL-like language for endpoint queries and hunts

Artifacts: pre-built query packs covering processes, services, users, files, and more

Example: SELECT * FROM pslist() WHERE CommandLine =~ 'powershell' — find PS processes

Fleet-wide hunts: push query to all agents simultaneously; results stream back centrally

Offline collection: generate standalone collector EXE for systems without connectivity

Built-in artifacts: Windows.System.Pslist, Windows.Network.Netstat, Windows.EventLogs.*

Hunt dashboards: real-time results across entire fleet in a browser UI

Open source, Apache license — deploy on-premises with no per-endpoint cost

GRR Rapid Response & OSQuery

GRR (Google Rapid Response): Python agents, hunt flows, and file timeline analysis

GRR flows: automated multi-step collection jobs (collect file → hash → compress → retrieve)

File finder flow: recursive search across filesystem with regex and hash matching

OSQuery: SQL interface to OS state — same query syntax on Windows, Mac, and Linux

osqueryi: interactive SQL shell for live system queries

SELECT * FROM processes WHERE name = 'cmd.exe': cross-platform process query

Fleet (by Fleet DM): centralized OSQuery management and hunt dashboard

Best use case: mixed-OS environments where one query must work across all platforms

KAPE — Targeted Artifact Collection

KAPE (Kroll Artifact Parser and Extractor): targeted collection and processing tool

Targets: define WHAT to collect — Prefetch, Event Logs, Registry Hives, Browser History, etc.

Modules: define HOW to process — parse collected artifacts automatically

Two phases: Collect (copy artifacts to output folder) → Process (run parsers against output)

Module examples: PECmd (Prefetch), RegRipper (Registry), EvtxECmd (Event Logs)

Typical command: kape.exe --tsource C:\ --tdest D:\evidence --target !BasicCollection --module !EZParser

Speed: KAPE collects and parses a full Windows artifact set in under 10 minutes

Build repeatable, documented collection packages per incident type

Conclusion

WMIC and PowerShell provide powerful, read-only access to remote system state

PowerShell Remoting and CIM enable fleet-wide triage without agents

Document all commands: forensic soundness means knowing what you ran and when

IR frameworks (Velociraptor, GRR) scale investigation to thousands of systems

KAPE reduces triage time from hours to minutes with targeted artifact collection

Choose the right tool for scope: single host vs. enterprise-wide hunt

Knowledge Check