VolHunter SOP v 3.0



VolHunter SOP

[Purpose 3](#_Toc26361259)

[Notes 3](#_Toc26361260)

[Capture Requirements – Utilizing the DIP 3](#_Toc26361261)

[Capture Requirements – MPNet Systems 3](#_Toc26361262)

[Target System Requirements 4](#_Toc26361263)

[Preparing the Windows System 4](#_Toc26361264)

[VolHunter Powershell Process 4](#_Toc26361265)

[VolHunter Powershell Module Commands 5](#_Toc26361266)

[Preparing the Linux System 6](#_Toc26361267)

[Example Usage – Loading Data to Elastic 6](#_Toc26361268)

[Reading Data in Kibana 6](#_Toc26361269)

[Investigating Suspicious Findings 6](#_Toc26361270)

[Cleaning Up 6](#_Toc26361271)

[Currently Supported Plugins 6](#_Toc26361272)

# Purpose

The ability to monitor a network of systems and identify anomalous behaviors, processes, or actions, e.g. baselining, provides a powerful manner through which an operator may detect misconfigurations or adversary activities. As anti-virus programs and other signature based detection methods have become more prevalent, hackers have moved from custom tools and backdoors to less easily detectable methods. These methods may include “living off the land”, using only tools that already exist on a network when they gain access, in order to appear to be a legitimate user, or maintaining their presence purely in volatile memory, where many anti-virus vendors do not or cannot search.

Standard approaches to analyzing memory require the capturing of RAM from a suspect system, often between two and 16 gigabytes of data, and moving that capture file to a trusted system for initial triage. As memory analysis is more consistently necessary to detect intrusions, the ability to analyze a large network of systems presents a problem of scale, both from the amount of data being transferred around the network, as well as storage and processor availability for analysis. This document provides guidance on the usage of a set of scripts called VolHunter that attempts to solve this problem utilizing tools already present on the CVA/H weapon system.

# Notes

**Sections marked bold are best practices/warnings/recommendations** and should be reviewed and understood before utilizing this toolset.

*Italicized Words / Are examples of code / or Powershell commands*

*Powershell Commands are started with PS:*

*Linux Commands with sudo privileges are started with #*

*Linux Commands without elevated privileges start with $*

VolHunter is a set of scripts, python and powershell, Kibana dashboards, and is under active testing and development. Please pay particular attention to the version number located in the comments of scripts. The latest stable version is available at www.github.com/monpolo/volhunter

# Capture Requirements – Utilizing the DIP

To effectively utilize all functions of VolHunter, operators should utilize either a Windows based virtual machine (VM) with python installed, or a Windows VM and a Linux VM. These VMs should be located in the DIPs DMZ or otherwise allowed to communicate using WMIC and WinRM to the mission partner network (MPNet). These systems should not be allowed to communicate with anything internal to the DIP other than an Elastic ingestion point, typically over TCP port 9200.

# Capture Requirements – MPNet Systems

MPNet systems must be configured to accept remote WinRM connections. These systems must also have the system you are launching from in its TrustedHosts list. For ease of use and the least amount of changes to MPNet systems, it is recommended that the launching system be a Windows computer part of the MPNet domain.

If for any reason you are unable to utilize a MPNet computer to run VolHunter from, work with the mission partner to either have your system added to their computers TrustedHosts list, or a wildcard is temporarily placed in.

To add a system to the TrustedHosts list on your current machine, in Powershell run:

*Winrm set winrm/config/client ‘@{TrustedHosts=”machine,machine”}’*

# Target System Requirements

Systems you plan to collect & triage from at a minimum must run Powershell version 3 or later. This is typically Windows 7 SP1 and later, but it is possible that Windows 8, 8.1, and certain Windows Server version are not running the necessary version of Powershell or newer.

# Preparing the Windows System

To successfully implement VolHunter, your launching system requires a specific placement of folders and files, as shown below. While the actual parent folder of VolHunter is irrelevant, the subfolders (yellow) and files (blue) are strict requirements.

VolHunter

bin

GatheredLogs

JobLogs

VHLogs

VHSwarm.ps1

Targetlist.txt

VolHunter.psm1

DumpIt-64.exe

DumpIt-86.exe

Volatility.exe

# VolHunter Powershell Process

The below commands are the most common string of commands an operator will run in order to deploy, analyze, and collect results. Situations may arise that call for deviation and operators should be familiar with each cmdlet inside of VolHunter before deviating.

**Ensure you have built out your desired target list file prior to executing these commands**

PS: *Import-Module .\VolHunter.psm1*

PS: *Set-VHEnvironment –TargetList [.\targetlist.txt] –MaxThreads [10] –credName [domain\admin]*

**Just typing Set-VHEnvironment will populate the variables with default values**

**You will be prompted to enter the administrator account’s password. Be sure you type this accurately**

PS: *Start-VHInvestigation*

**The system will now confirm connectivity to each system in [TargetList] and report back which do not appear to be online. After testing connections to each target, VolHunter will begin pushing tools to each system in order, up to [MaxThreads] simultaneously. After deploying tools, a command will be pushed to each remote system to initiate remote execution. At the end of this, VolHunter will produce text to the terminal detailing how many systems were executed against.**

**It is recommended that the operator wait 30-60 minutes before proceeding to allow remote systems to complete execution at this point. In order to validate if remote systems are complete or not, run:**

**CHECK ALL SYSTEMS**

PS: *Get-VHStatusAll*

**CHECK 1 SYSTEM STATUS**

PS: *Get-VHStatus –Target [computername]*

**Once all target systems have completed (there may be 1-3% that fail to complete in an acceptable time frame) execute the following to retrieve all text outputs. Include the optional switch parameter “skipOfflines” to perform an additional connectivity check to targets and skip systems that have lost connection in the time since beginning execution**

PS: *Get-VHOutput [-skipOfflines]*

Now all output files from Volatility will be stored in .\VolHunter\GatheredLogs console reports of remote Powershell connections in .\VolHunter\JobLogs and logs from the remote Powershell script on targets in .\VolHunter\VHLogs

Output files are named in the following manner [plugin]-[hostname].txt

# VolHunter Powershell Module Commands

**Display the values held in environment variables**

Display-VHEnvironment

**Pull an individual systems’ memory dump file**

Get-VHMemDump –Target [hostname]

**Retrieve VolHunter output files from all targets**

Get-VHOutput

**Check individual target’s status during execution**

Get-VHStatus –Target [hostname]

**Check all target statuses during execution**

Get-VHStatusAll

**Remove tools, output files, and memory dump from targets**

Remove-VHRemote

**In case individual systems fail to run or a plugin fails on an individual system, the below command will get status from each system, and rerun the steps that failed**

Start-VHExecutionCleanup

**Displays the VHLog from a remote machine to the powershell console**

Display-VHLog

**To monitor the status of target systems, retrieving status updates of only systems that haven’t completed every 30 seconds**

Watch-VHStatus

# Preparing the Linux System

After your Windows system has collected the Volatility output from target terrain, and you have transferred the contents of the .\GatheredLogs folder to your Linux system, in whatever approved fashion you have available, take the following steps.

Ensure you have Python3 installed.

Ensure you have installed the ElasticSearch python module. This can be accomplished with:

*# pip install ElasticSearch*

Or by using the requirements text file

*# pip install -r requirements.txt*

Or by manually downloading the package and placing it in your Python path.

Ensure you have the latest VolHunter Python scripts and gathered output data organized in the following manner, take note that the root folder name of “VHConversion” here can be modified to fit the users needs.

All volatility plugin output files should be placed in the \VHConversion\VHdata\gatheredLogs\ folder path. The converted, output, and processed folders are intermediary folders for the different stages of work for formatting, tagging, and uploading output data into Elastic.

processed

output

gatheredLogs

converted

volindexer.py

tagger.py

jsonformat.py

encoder.py

convert.py

VHConversion

VHdata

VHprocessor.py

# 

# VolHunter Python

To begin processing results, run the following command from an elevated prompt:

To see the help screen and possible commands:

*# python3 VHprocessor.py -h*

Basic usage:

*# python3 VHprocessor.py all -i* [IP of elastic server]

Action options:

Convert – Convert, format, locally enrich, and ingest data to Elastic

Investigate – Investigate for standard lineage

CAR – Run MITRE CAR rules

All – Run convert, investigate, and car

Delete – Delete data from VolHunter Index – CANT UNDO

Mark – Mark \_id’s provided in .txt file as investigated. Must specify -f with file path.

Optional Arguments:

IP (--ip or -i) – IP of Elastic Stack server. Default = 20.25.20.142

Port (--port or -p) – Port of Elastic Stack server. Default = 9200

Directory (--dir or -d) – Root path for VHData. Default = [Current working directory]

File (--file or -f) – TXT file containing \_id’s to be marked as investigated separated by a new line.

## Cleaning Up

To delete data from Elastic server:

*# python3 VHprocessor.py delete -I* [IP of elastic server]