Team Members: Tyler Dionne, Kendall Kelly, Garrett Gmeiner
CYB 4890
Dr. Sudhakaran
November 25, 2024

Autorun Analyzer Plugin: Windows Memory Forensics Tool

Overview

The Autorun Analyzer plugin for Volatility3 is a memory forensics tool designed to identify persistence mechanisms in Windows memory dumps by analyzing registry keys commonly used by attackers to maintain access to a system.

In Windows systems, entries under the “Run” registry key start automatically when the system boots up or a user logs in [1][2][3]. This registry key is popular among attackers who are looking to achieve persistence on a system. The term “achieve persistence” means maintaining access to a compromised system. By analyzing autorun locations in memory dumps, forensic analysts can identify possible malicious programs configured to execute automatically upon startup.

Methodology

1. Analysis of Common Autorun Locations

The plugin specifically focuses on the analysis of registry keys in autorun locations that Windows executes automatically during system startup. The following autorun keys are analyzed [6][7][8].

```
autorun_keys = [
    "Microsoft\\Windows\\CurrentVersion\\Run",
    "Microsoft\\Windows\\CurrentVersion\\RunOnce",
    "Microsoft\\Windows\\CurrentVersion\\RunServices",
    "Microsoft\\Windows\\CurrentVersion\\Policies\\Explorer\\Run",
    "Wow6432Node\\Microsoft\\Windows\\CurrentVersion\\Run",
    "Wow6432Node\\Microsoft\\Windows\\CurrentVersion\\RunOnce",
    "Wow6432Node\\Microsoft\\Windows\\CurrentVersion\\Policies\\Explorer\\Run",
]
```

The list shown above defines the common persistence locations in the Windows Registry, includes both 32-bit (Wow6432Node) and 64-bit paths (Not prefixed with Wow6432Node). This list also covers both immediate execution (Run) and one-time execution (RunOnce) keys. These registry keys are known to be commonly used by malware for persistence on windows machines.

The plugin shall scan the entries under these autorun registry keys and identify possible threats.

2. Detection of Suspicious Autorun Entries

The analyze() function is used to perform two tasks: 1. Collect registry hives from the memory dump 2. Scan the specific autorun registry keys (specified above) within those hives to identify entries.

Two lists are created: 1. “reg_keys” to store the registry hives discovered in the memory dump 2. “results” to store the entries under the keys in our list

The plugin uses the HiveList class from Volatility3 to enumerate all the available registry hives in the memory dump. The hives are collected using the “list_hives” function which retrieves the metadata.

For each hive returned from “list_hives” the plugin iterates through the list of predefined autorun keys “autorun_keys” which are known autorun locations in the Windows Registry commonly used by attackers to achieve persistence.

The “get_key” method tries to access the autorun key in the current hive and if it exists it obtains all the associated values (entries). For each entry the function obtains the:

- Key Path: The registry key’s full path
- Value Name: The name of the registry entry
- Data String: The associated data (ex. Path to an .exe or .bat)

  Note: If decoding the data string fails then <Unable to decode> is used

The metadata for each entry is appended to a “results” list which is used in the generator to print the final output.

The current method of identifying malicious autorun entries is by comparing the data string against a predefined list of keywords “malicious_keywords” commonly associated with Windows malware. This approach is efficient because of the fact that using regex to find uncommon or arbitrary names may result in false positives while this list can be tailored to common malware on Windows systems and specific keywords that regex may not be able to identify.

Testing

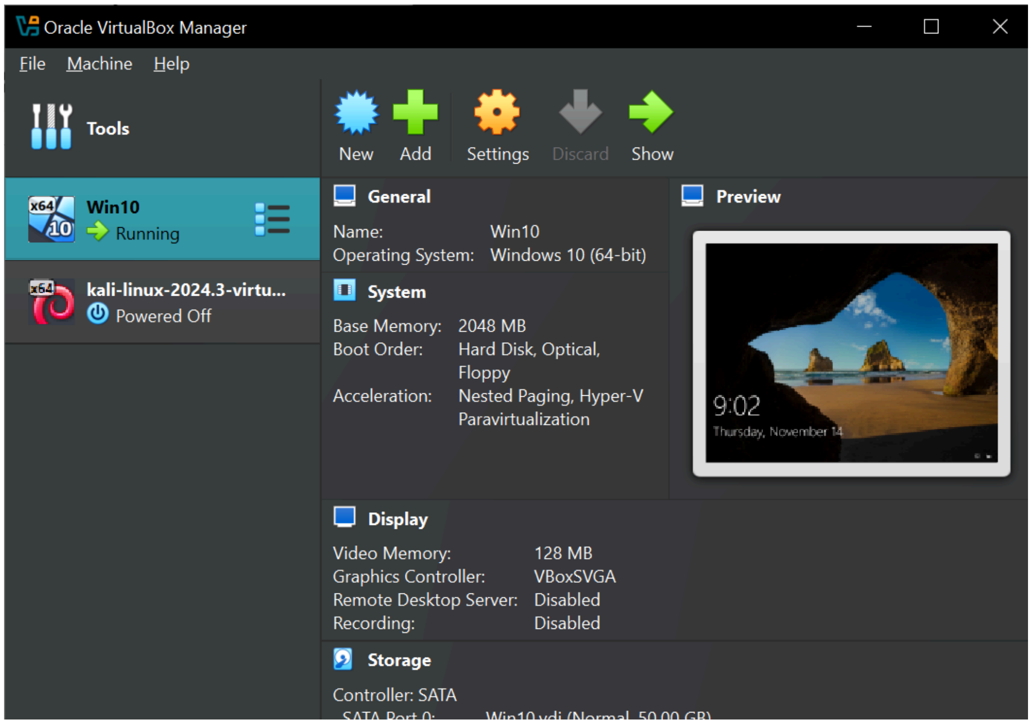
In order to test the functionality of our plugin we must prepare a Windows system and capture two memory dumps.

The first dump will simulate a clean capture of a Windows system prior to any tampering with autorun locations. This will help to ensure our plugin can identify non malicious autorun entries correctly and not produce false positives.

The second dump will simulate the capture of a compromised system in which attackers used autorun registry keys to achieve persistence on the machine. This will show us if our plugin is able to identify both known and suspicious autorun entries.

1. Windows 10 Virtual Machine

To prepare our test cases we must spawn a new Windows 10 virtual machine in VirtualBox.



Win10
Running

kali-linux-2024.3-virtu...
Powered Off

General

Name: Win10
Operating System: Windows 10 (64-bit)

System

Base Memory: 2048 MB
Boot Order: Hard Disk, Optical, Floppy
Acceleration: Nested Paging, Hyper-V Paravirtualization

Display

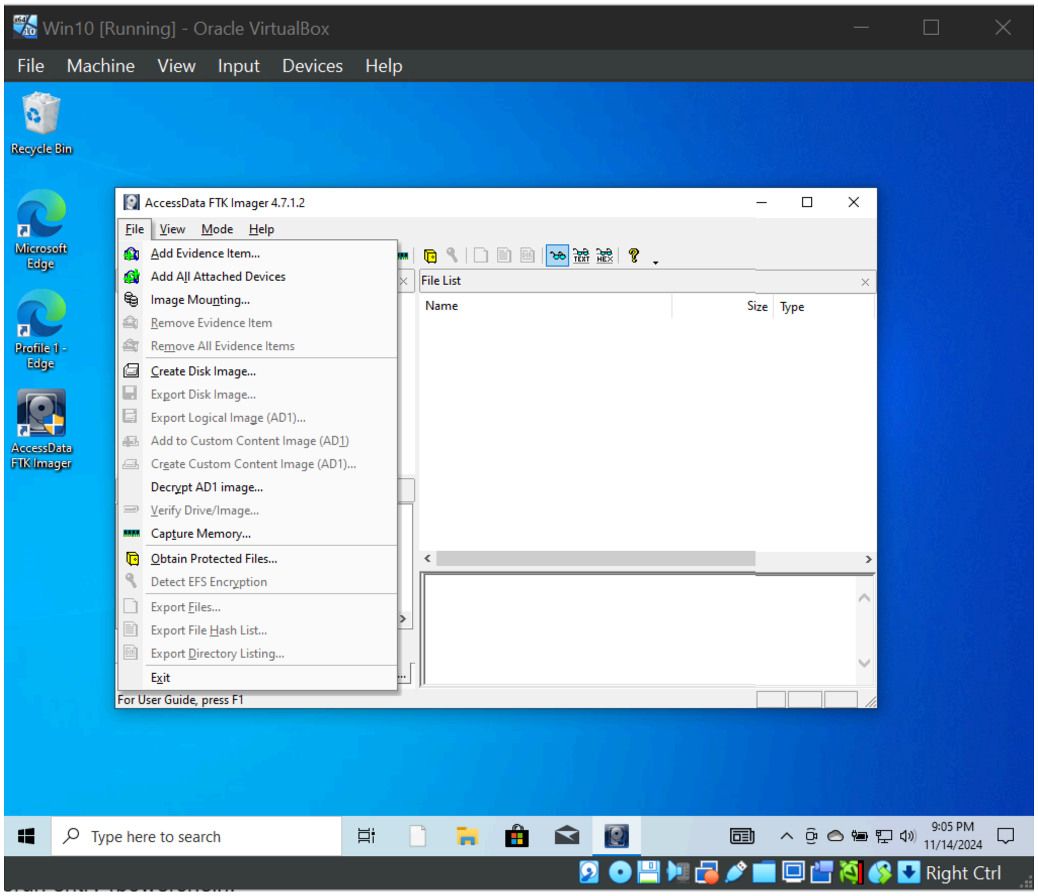
Video Memory: 128 MB
Graphics Controller: VBoxSVGA
Remote Desktop Server: Disabled
Recording: Disabled

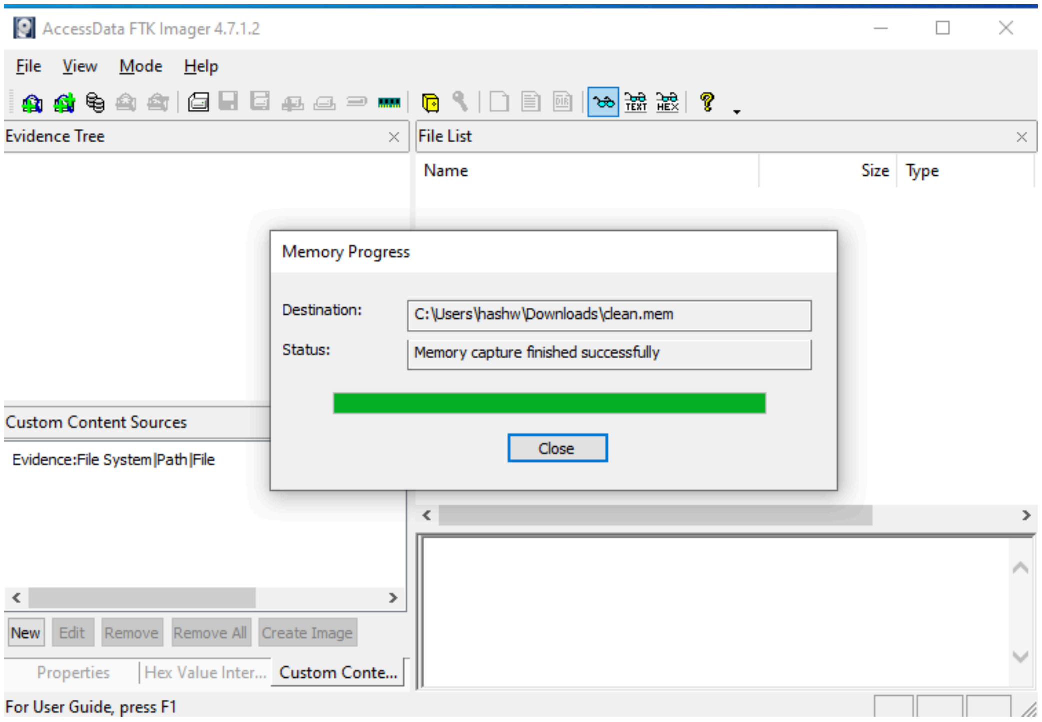
Storage

Controller: SATA
SATA Port 0: Win10.vdi (Normal, 50.00 GB)

2. Clean Memory Dump Acquisition

Use FTK Imager to capture memory of the clean system.




AccessData FTK Imager 4.7.1.2
File View Mode Help
Evidence Tree
File List
Name Size Type
Memory Progress
Custom Content Sources
Evidence:File System|Path|File
Close
New Edit Remove Remove All Create Image
Properties Hex Value Inter... Custom Conte...
For User Guide, press F1

Destination: C:\Users\hashw\Downloads\clean.mem

Status: Memory capture finished successfully

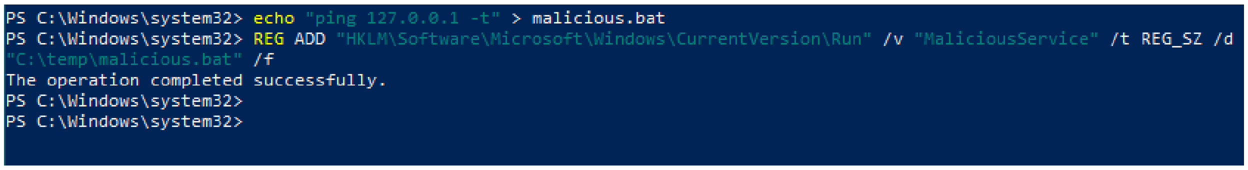
3. Compromised Memory Dump Acquisition

First we must create some “malicious” autorun entries.

As stated in the overview, the goal is to add entries under the “Run” Windows registry key which causes specified programs to automatically execute each time the system starts.

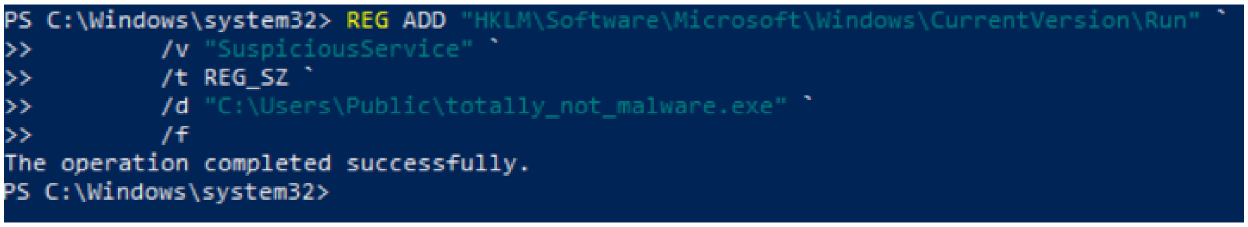
$ echo "ping 127.0.0.1 -t" > malicious.bat

$ REG ADD "HKLM\Software\Microsoft\Windows\CurrentVersion\Run" /v "MaliciousService" /t REG_SZ /d "C:\temp\malicious.bat" /f



```
PS C:\Windows\system32> echo "ping 127.0.0.1 -t" > malicious.bat
PS C:\Windows\system32> REG ADD "HKLM\Software\Microsoft\Windows\CurrentVersion\Run" /v "MaliciousService" /t REG_SZ /d
"C:\temp\malicious.bat" /f
The operation completed successfully.
PS C:\Windows\system32>
PS C:\Windows\system32>
```

$ REG ADD "HKLM\Software\Microsoft\Windows\CurrentVersion\Run" /v "SuspiciousService" /t REG_SZ /d "C:\Users\Public\totally_not_malware.exe" /f



```
PS C:\Windows\system32> REG ADD "HKLM\Software\Microsoft\Windows\CurrentVersion\Run" `
>>         /v "SuspiciousService" `
>>         /t REG_SZ `
>>         /d "C:\Users\Public\totally_not_malware.exe" `
>>         /f
The operation completed successfully.
PS C:\Windows\system32>
```

To understand how these commands work we can break each section down:

1. REG ADD

- This is a command line tool used to create/modify Windows Registry entries

2. HKLM\Software\Microsoft\Windows\CurrentVersion\Run

- This is the path to the registry key the user wants to create/modify entries in.
- In this case the goal is to add entries under the “Run” key.
- HKLM (HKEY_LOCAL_MACHINE) is a top level registry hive and holds settings that apply to the whole computer, not just one user [9] [10]. There exist other registry hives in Windows systems such as HKCU (HKEY_CURRENT_USER) and HKCR (HKEY_CLASSES_ROOT).

3. /v “MaliciousService” and /v “SuspicuousService”

- The /v flag sets the name of the new value being added to the key.

4. /t REG_SZ

- The /t flag sets the type of registry value.
- In this case REG_SZ means the value is a plain text string.

5. /d “C:\temp\malicious.bat” and /d “C:\Users\Public\totally_not_malware.exe”

- The /d flag sets the data for the value being added such as a file or executable.
- In this case we set the value to an executable and a batch file.

6. /f

- This flag forces the command to overwrite an existing value without confirmation.

Use FTK Imager to capture memory of the compromised system.

[Refer to previous section for images]

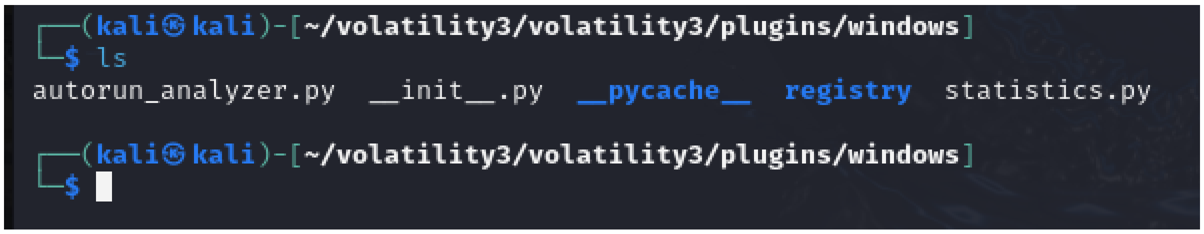
Plugin Installation

To install the plugin simply create a file named autorun-analyzer.py. Paste the source code.

$ nano autorun_analyzer.py

Then move the file to the Volatility3 plugins directory:

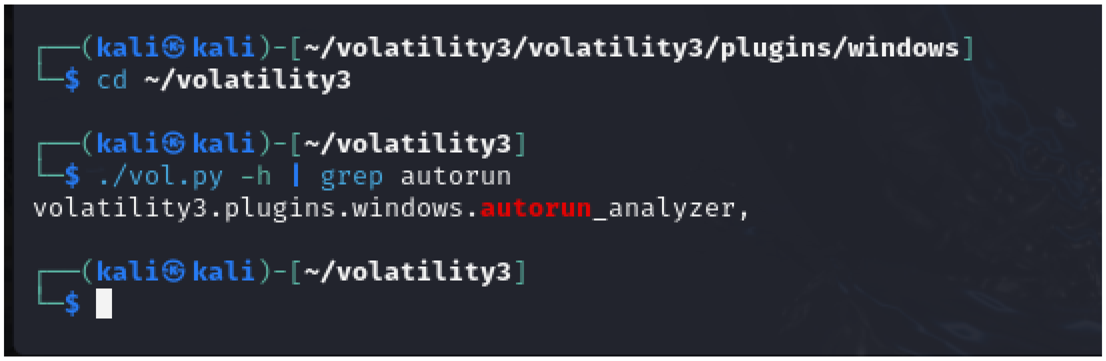
$ cp ~/autorun-analyzer.py ~/volatility3/volatility3/plugins/windows



```
┌──(kali㉿kali)-[~/volatility3/volatility3/plugins/windows]
└─$ ls
autorun_analyzer.py  __init__.py  __pycache__  registry  statistics.py

┌──(kali㉿kali)-[~/volatility3/volatility3/plugins/windows]
└─$ 
```

Verify plugin installation:

$ cd ~/volatility3
$ ./vol.py -h | grep autorun



```
┌──(kali㉿kali)-[~/volatility3/volatility3/plugins/windows]
└─$ cd ~/volatility3

┌──(kali㉿kali)-[~/volatility3]
└─$ ./vol.py -h | grep autorun
volatility3.plugins.windows.autorun_analyzer,

┌──(kali㉿kali)-[~/volatility3]
└─$ 
```

Example Usage:

$ cd volatility3

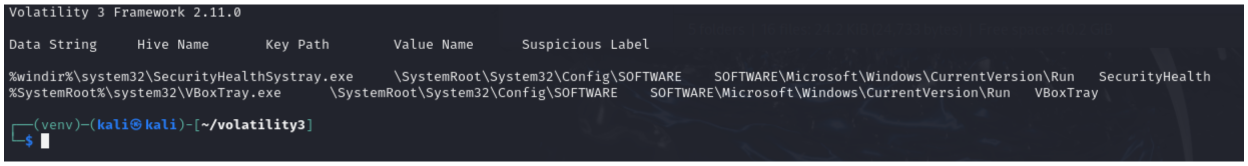
$ python3 vol.py -f ~/dump.raw windows.autorun-analyzer

Analysis

1. Clean System Analysis

Once the clean dump has been moved to ~ run the following command to analyze the dump:

$ python3 vol.py -f ~/clean.mem windows.autorun-analyzer

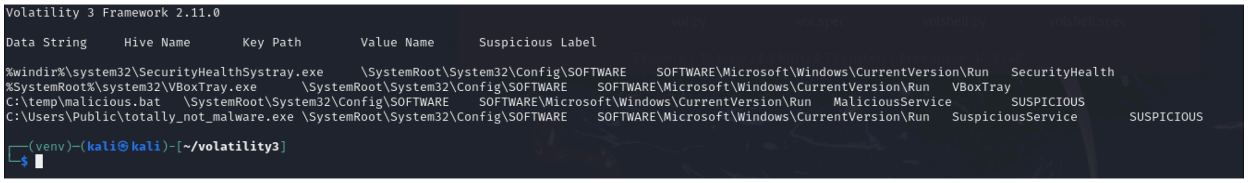


```
Volatility 3 Framework 2.11.0

Data String     Hive Name       Key Path        Value Name      Suspicious Label

%windir%\system32\SecurityHealthSystray.exe     \SystemRoot\System32\Config\SOFTWARE    SOFTWARE\Microsoft\Windows\CurrentVersion\Run   SecurityHealth
%SystemRoot%\system32\VBoxTray.exe      \SystemRoot\System32\Config\SOFTWARE    SOFTWARE\Microsoft\Windows\CurrentVersion\Run   VBoxTray

┌──(venv)─(kali㉿kali)-[~/volatility3]
└─$ 
```

2. Compromised System Analysis

Once the compromised dump has been moved to ~ run the following command to analyze the dump:

$ python3 vol.py -f ~/dirty.raw windows.autorun-analyzer



```
Volatility 3 Framework 2.11.0

Data String     Hive Name       Key Path        Value Name      Suspicious Label

%windir%\system32\SecurityHealthSystray.exe     \SystemRoot\System32\Config\SOFTWARE    SOFTWARE\Microsoft\Windows\CurrentVersion\Run   SecurityHealth
%SystemRoot%\system32\VBoxTray.exe      \SystemRoot\System32\Config\SOFTWARE    SOFTWARE\Microsoft\Windows\CurrentVersion\Run   VBoxTray
C:\temp\malicious.bat   \SystemRoot\System32\Config\SOFTWARE    SOFTWARE\Microsoft\Windows\CurrentVersion\Run   MaliciousService        SUSPICIOUS
C:\Users\Public\totally_not_malware.exe \SystemRoot\System32\Config\SOFTWARE    SOFTWARE\Microsoft\Windows\CurrentVersion\Run   SuspiciousService       SUSPICIOUS

┌──(venv)─(kali㉿kali)-[~/volatility3]
└─$ 
```

3. Key Observations

The plugin was able to list entries under the “Run” registry key and was able to differentiate between standard autorun entries such as %winddir%\system32\SecurityHealthSystray.exe and %SystemRoot%\system32\VboxTray.exe and malicious autorun entities such as C:\temp\malicious.bat and C:\Users\Public\totally_not_malware.exe demonstrating the plugin’s ability to identify potential threats during forensic analysis.

4. Detection Effectiveness

Despite our plugin correctly identifying 100% of our sample malicious autorun entries the scope of autorun keys was limited to “HKLM\Software\Microsoft\Windows\CurrentVersion\Run” because this was the only location entries were created before the dump was captured. Our plugin also has the functionality to discover malicious entries in 6 additional autorun keys (refer to Methodology section). If the commands used to create the registry key entries were modified to incorporate these locations the results are expected to be the same. To create a more comprehensive plugin to detect all autorun locations in Windows systems additional functionality would have to be implemented for the locations outside of the scope of this project.

Debugging

1. Source Code Errors

Run the following command to help see errors with the source code for the plugin.

```
$ python3 vol.py -vv -f ~/clean.mem windows.autorun_analyzer 2>&1 | tee debug.log
```

2. Missing Python Libraries

Run the following sequence of commands:

```
$ python3 -m venv venv

source venv/bin/activate

pip install pycryptodome

pip install yara-python pefile
```

Once complete run the following command:

```
$ source venv/bin/activate
```

Conclusion

In conclusion, autorun registry keys are used by attackers to achieve persistence on Windows systems by allowing their payload to execute every time the system reboots. The autorun analyzer plugin for Volatility3 demonstrates its ability to identify both standard autorun entries and suspicious autorun entries through the simulated scenarios.

References

[1] https://learn.microsoft.com/uk-ua/sysinternals/downloads/autoruns
[2] https://www.ghacks.net/2016/06/04/windows-automatic-startup-locations/
[3] https://en.wikipedia.org/wiki/AutoRun
[4] https://www.ccslearningacademy.com/what-is-persistence-in-cybersecurity/
[5] https://www.netscout.com/what-is-mitre-attack/persistence
[6] https://www.bleepingcomputer.com/tutorials/windows-program-automatic-startup-locations/
[7] https://learn.microsoft.com/en-us/windows/win32/setupapi/run-and-runonce-registry-keys
[8] https://www.ghacks.net/2016/06/04/windows-automatic-startup-locations/
[9] https://en.wikipedia.org/wiki/Windows_Registry
[10] https://www.lifewire.com/hkey-local-machine-2625902