Digital Forensics: Accessing the Windows Registry with Python

Windows Registry = Digital Forensics Gold Mine

Hi! In this tutorial, I'll show you how to access the Windows Registry with Python.

For this tutorial, we will not need to install any dependencies because Python has a module that we can utilize to achieve our goal, which will be introduced later. At the end of the article, an example Python script is provided that extracts potentially important forensic information from the Windows Registry. Let's get to it!

What is the Windows Registry?

The Windows Registry is a tree-structured database that contains critical information for all users on a

Windows operating system. There are several folders that make up the root of the database known as *hives*. Each node of the database is known as a *key* and each key contains a list of *subkeys* and

corresponding *values*. Some information a digital forensic analyst can obtain from the Windows Registry include:

- Previously browsed URLs
- Auto-runs (applications that run on system start-up)
- Details about previously or currently connected USB drives
- Recently used applications/files
- Installed Applications
- List of all services installed on a system
- Network Configuration (Connected Access point, IP address assigned by DHCP server)
- Previously mounted drives that are mounted or were removed

and a whole ton of other things that may prove critically important for digital forensic investigations.

Although, the Windows Registry is ideal for the digital forensic analyst, adversaries also love to use it to gain persistent access to a compromised machine. For example, malware writers use the autoruns Registry key located at `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run`to run malicious programs on system start up.

A more specific example of malware that manipulates the Windows Registry would be the SUNBURST malware recently used in the SolarWinds supply chain attack which turned off blocklisted services to prevent system immunity by changing the value of this Registry key,
`HKLM\SYSTEM\CurrentControlSet\services\<service_name>\Start`

to a value of 4 (where `<service_name>` is the name of the blockedlisted service). This allowed the malware to run no matter if there were blocklisted services that could have thwarted the execution of the malware.

For more on the SUNBURST malware, check out my recent article.

Reading Values from the Windows Registry

To read values from a key within the Windows Registry, we will use a built-in Python module called winreg. Let's look at a simple example that retrieves information from the Registry key `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion` that contains information regarding the currently installed instance of the Windows OS:

Let's explain what the script above is doing (**Note**: to run the script above successfully, you need to run the script with administrative privileges).

First, we import all the necessary functions and constants defined within the `winreg` module that we will use to enumerate the values of a given registry key (**Lines 1–9**).

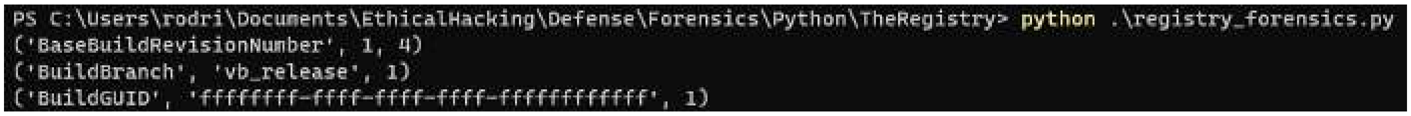
Second, we connect to one of the root keys (hive) known as **HKEY_LOCAL_MACHINE** using the `ConnectRegistry` function (line 11), which stores most of the configuration details for the software installed on a given Windows OS. It also stores much of the information for the Windows OS itself (more info on this hive here).

Third, we open the subkey that we want to access located within the hive using the `OpenKey` function (**line 12**), which takes in the connected hive returned from the `ConnectRegistry` function, the subkey we wish to open, an integer for the keyword argument "reserved" which has to be 0 according to the `winreg` documentation, and lastly, the permissions we are trying to open this registry key with,

KEY_ALL_ACCESS. Within the `winreg` module, their is a constant defined as KEY_ALL_ACCESS, and it does exactly what it sounds like it does and grants us "all access" to this subkey.

Finally, we use the `QueryInfoKey` function (**line 13**) and pass in the opened registry subkey to retrieve the number of values found within this key. To get the total number of values, we extract the number located in the second index ([1]) of the returned tuple. This will allow us to enumerate all the values using the `EnumValue` function (**lines 14–16**) which takes in the opened subkey and the index of the value we are trying to get.

Here is a portion of the output from running the script above:



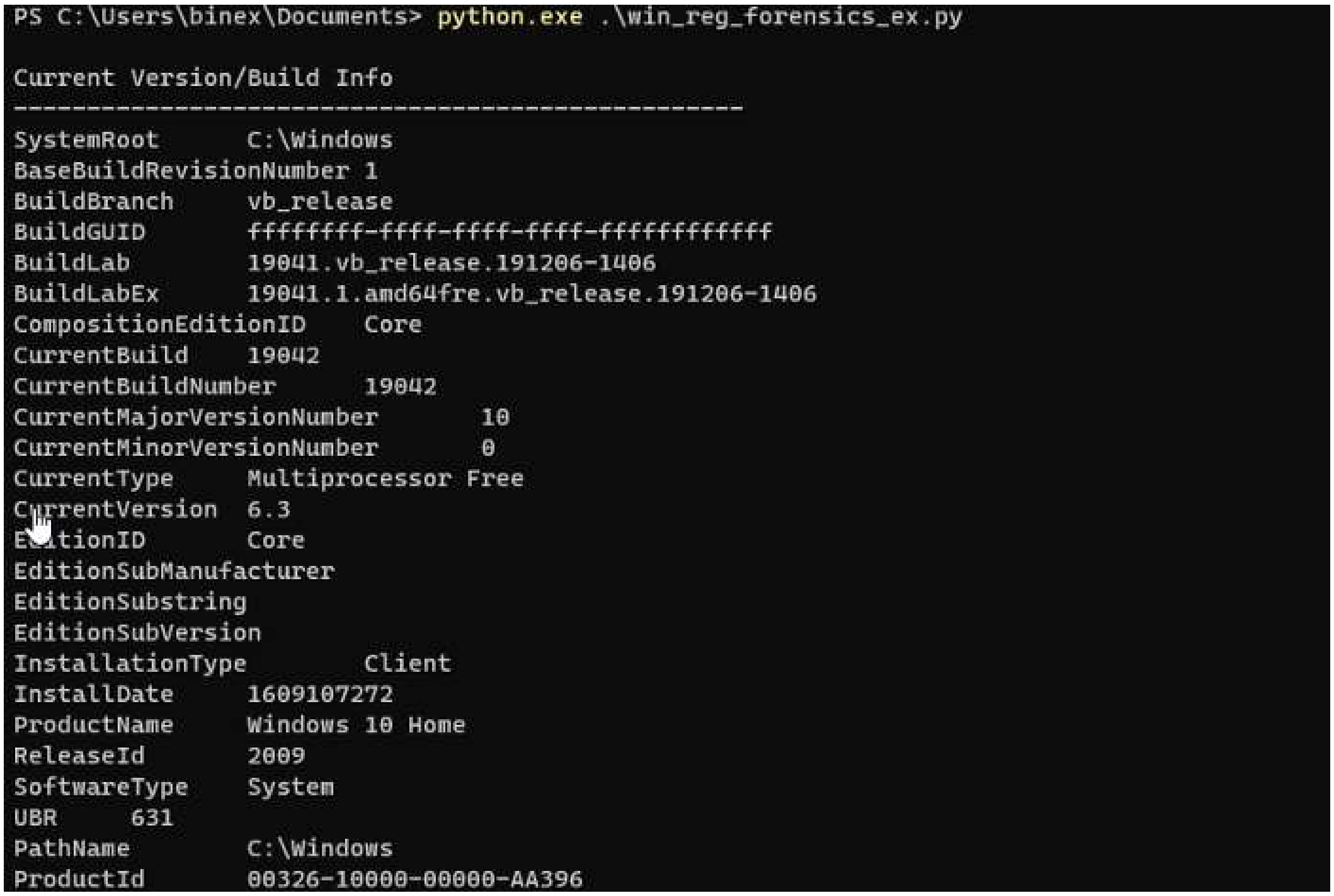
```
PS C:\Users\rodri\Documents\EthicalHacking\Defense\Forensics\Python\TheRegistry> python .\registry_forensics.py
('BaseBuildRevisionNumber', 1, 4)
('BuildBranch', 'vb_release', 1)
('BuildGUID', 'ffffffff-ffff-ffff-ffff-ffffffffffff', 1)
```

That's it! You can use now most of the logic defined within the script above to access any other subkey within the Windows Registry.

Creating a Basic Windows Registry Forensics Tool

Here is the final script I promised, showing you how to extract information from several subkeys within the Windows Registry that can provide essential evidence for a digital forensics investigation:

The output generated by running the script above is a little long so below is just a snippet from the output:



```
PS C:\Users\binex\Documents> python.exe .\win_reg_forensics_ex.py

Current Version/Build Info
--------------------------------------------------
SystemRoot      C:\Windows
BaseBuildRevisionNumber 1
BuildBranch     vb_release
BuildGUID       ffffffff-ffff-ffff-ffff-ffffffffffff
BuildLab        19041.vb_release.191206-1406
BuildLabEx      19041.1.amd64fre.vb_release.191206-1406
CompositionEditionID    Core
CurrentBuild    19042
CurrentBuildNumber      19042
CurrentMajorVersionNumber       10
CurrentMinorVersionNumber       0
CurrentType     Multiprocessor Free
CurrentVersion  6.3
EditionID       Core
EditionSubManufacturer
EditionSubstring
EditionSubVersion
InstallationType        Client
InstallDate     1609107272
ProductName     Windows 10 Home
ReleaseId       2009
SoftwareType    System
UBR     631
PathName        C:\Windows
ProductId       00326-10000-00000-AA396
```

EOF

In this article, you have learned how to use Python to access the Windows Registry in order to help automate the acquisition of Registry data for a forensic analyst. If you wish to read more articles like this, please follow me on Medium for more cybersecurity-related articles and on GitHub (link below) for more cybersecurity-related projects! Thanks for reading!

References