

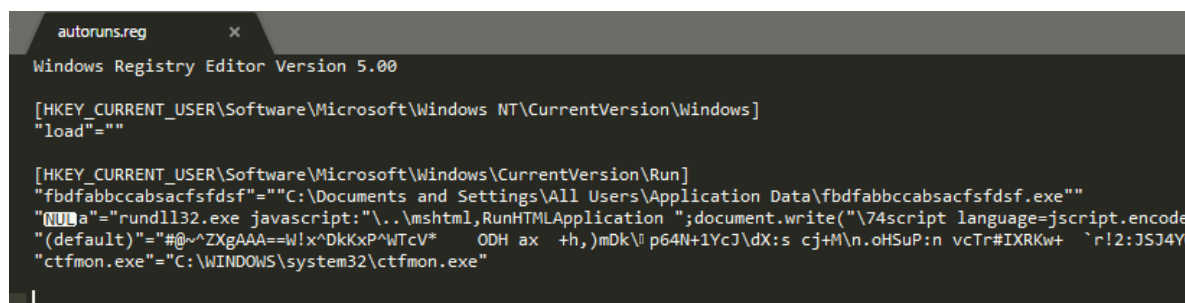
RegParser - 0.5.5

Description:

RegParser (rp) is a python wrapper script for python-registry framework (@williballenthin [FireEye]). This command-line utility is designed to slightly extend and facilitate framework's capabilities. In general it's used to parse any offline windows registry hives during malware hunting or forensic investigations.

```
C:\Users\Python3\python.exe C:/repos/regparser/rp.py -s c:/bin/hives/NTUSER-Trojan.Poweliks.DAT -p autoruns -E autoruns.reg
2014-09-04 13:12:49.937500 UTC,,Software\Microsoft\Windows NT\CurrentVersion\Windows,load,
2014-09-04 13:12:25.703125 UTC,,Software\Microsoft\Windows\CurrentVersion\Run,fdbfabbccabsacsfdsf,"C:\Documents and Settings\
2014-09-04 13:12:25.703125 UTC,,Software\Microsoft\Windows\CurrentVersion\Run, a,rundll32.exe javascript:"..\mshtml,RunHTMLAp
2014-09-04 13:12:25.703125 UTC,,Software\Microsoft\Windows\CurrentVersion\Run,(default),#@~^ZXgAAA=W!x^DkKxP^WTcV* ODH ax +h
2014-09-04 13:12:25.703125 UTC,,Software\Microsoft\Windows\CurrentVersion\Run,ctfmon.exe,C:\WINDOWS\system32\ctfmon.exe

Process finished with exit code 0
```



The screenshot shows the Windows Registry Editor (Version 5.00) with the file 'autoruns.reg' open. The left pane shows the tree structure expanded to 'HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Windows'. The right pane displays the following registry values:

- Value: "load" = ""
- Value: "fdbfabbccabsacsfdsf" = "C:\Documents and Settings\All Users\Application Data\fdbfabbccabsacsfdsf.exe"
- Value: "a" = "rundll32.exe javascript:"..\mshtml,RunHTMLApplication ";document.write("\74script language=jscript.encode"
- Value: "(default)" = "#@~^ZXgAAA=W!x^DkKxP^WTcV* ODH ax +h,)mDk\i p64N+1YcJ\dX:s c j+m\n.oHSuP:n vcTr#IXRKw+ `r!2:JSJ4Y"
- Value: "ctfmon.exe" = "C:\WINDOWS\system32\ctfmon.exe"

It comes with following major features:

- Search for a registry key, value name or value data patterns described by a comma separated: strings, regex strings or utf8 hex binary strings
- Search for value data by its size, specified by operators like range, equality or inequality
- Search for registry modified keys at given date and time, specified by regex string pattern or range, or inequality operators
- Query the registry keys or values (including partial wildcard support)
- Enumerate and display hidden keys and values
- Hash registry value content
- Detect hive type
- Export results to .REG format (Simplifies malware analysis/infection reproduction based on file-less registry load points)
- Export results to SQLite (Used by regparser for plugin's baseline)
- Export results to CSV or stout
- Customize output data (21 different format fields)
- Easy plugin implementation and support with built in plugins like "autoruns", "services"
- Plugins baseline support

Requirements:

- Python 3.6.1 Framework
- **python-registry** module (1.2.0 at least):

In proxy enabled environment use: **pip --proxy http://PROXY_IP:PORT install %package name%**

- Operating system: **Windows, Linux, MacOS**

Installation:

```
pip install --upgrade python-dateutil
pip install --upgrade enum34
pip install --upgrade unicodescv
pip install --upgrade https://github.com/williballenthin/python-registry/archive/master.zip
```

Via Proxy:

```
pip --proxy http://30.207.31.50:8080 install --upgrade python-dateutil
pip --proxy http://30.207.31.50:8080 install --upgrade enum34
pip --proxy http://30.207.31.50:8080 install --upgrade unicodescv
pip --proxy http://30.207.31.50:8080 install --upgrade https://github.com/williballenthin/python-registry/archive/master.zip
```

Final step:

```
wget/save RegistryParser
Unzip RegistryParser and run: python rp.py -h (To review all available commands)
```

Upgrade:

Upgrade via **git pull** (Set appropriate settings if necessary)

Enhancement requests:

- Continue plugin development (Tasks with COM tasks support, Prefetch, etc.)
- Improve wildcard support in query_value_wd (Dynamic count of *)
- Improve error handling and logging (Implement python logging, custom errors in settings.py etc.)
- Add multi-threading support

Search logic and supported pattern type:

Registry property	Code	Search arguments	String	Regex	Binary	Special
Key	key.path()	-ks, -kr, -kb	✓	✓	✓	
Key LastWriteTime	key.timestamp()	-tm, -td		✓		✓
Value name	value.value()	-vs, -vr, -vb	✓	✓	✓	
Value Data	value.content()	-ds, -dr, -db	✓	✓	✓	
Data size	len(value.raw_data())	-dl				✓

Note: Special column would indicated the use of special search arguments like **_date_time()** and **_data_size()** which are described in Command line reference

Format fields:

Following list represents all available format fields with example values; all these fields can be used in any order to manipulate the output data (option **-f**):

special	Data depends on the plugin (Services plugin would log CurrentSet00X here)
key_timestamp:	2017-01-20 20:48:05.429764 UTC
key_subkeys	0
key_values	1
key_path	Software\Microsoft\Windows\CurrentVersion\Run
key_path_unicode	b'...\x00o\x00n\x00\\\x00R\x00u\x00n\x00'
hive_type	HiveType.NTUSER
hive_path	hives\NTUSER.DAT
hive_name	NTUSER.DAT
hive_root	\$\$\$PROTO.HIV (Note: The "{CMI-CreateHive{unique_identifier}" would appear as root key on all Windows Vista+ hosts, while the "\$\$PROTO.HIV" would be set as root on Windows XP hosts)
user_sid	S-1-5-21-790525478-688789844-854245398-1003
value_type	1
value_type_str	RegSZ
value_name	TestVirus
value_content	calc.exe
value_content_hash	32392C813DD09699130B08295844A13D
value_size	...
value_name_unicode	b'T\x00e\x00s\x00t\x00V\x00i\x00r\x00u\x00s\x00'
value_content_unicode	b'c\x00a\x00l\x00c\x00.\x00e\x00x\x00e\x00\x00\x00'
key_value	Software\Microsoft\Windows\CurrentVersion\Run\TestVirus
plugin_name	autoruns

Command line reference:

Parameter	Description
-h, --help	Opens the help page
-s, --source %INPUT_FOLDER%	File/Folder pointing to offline registry hive(s) [raw or .zip format supported]
-r, --recursive	Enables recursive search for -s, --source folder
-c, --case-sensitive	Enables case sensitive search (Default: case-insensitive)
-v, --verbose	Sends verbose debug data to stdout (Used for debugging)
Query arguments:	
-qk, --query-key "REG_KEY1", "REG_KEY2*", "..."	Query given comma separated registry keys, example: -qk "Software\Microsoft\Windows\CurrentVersion\Run" (all registry values from queried key(s) would be printed). If the key ends with "*" all its subkeys would be queried

-qv, --query-value "REG_KEY\VALUE_NAME", "..."	Query given comma separated registry values, example: -qv "Software\Microsoft\Windows\CurrentVersion\Run\ctfmon.exe" (only the queried value would be printed) [See examples for limited wildcard support]
-qs, --query-subkeys	Display root sub keys of given registry hive (Default: False)
Search arguments:	
-tm, --time-modified "REGEX_PATTERN "	Comma separated regex date patterns: -tm "2014-09-04 09:50","2013 2014" would search for all keys modified in 2014-09-04 at 09:50 and on 2013 or 2014
-td, --time-modified-datetime "DATE_TIME_PATTERN"	Comma separated Date time patterns: [datetime]..[datetime] or [Operator][datetime], where operator is one of: ">", "<". Example: -td "2014-09-04 13:12:19..2014-09-04 13:12:25.703125",">2014-09-04"
-ks, --key-string "STRING"	Comma separated string values: -ks "\\Run" would search for all keys having given string anywhere in their key path string
-kr, --key-regex "REGEX_PATTERN"	Comma separated regex strings: -kr "\\Run\$" would search for all keys which ends with "\\Run" string
-kb, --key-bin "ESCAPED_BYTES"	A binary value expressed in UTF8 escaped hex string, example: -kb "\\x5C\\x52\\x75\\x6E" would search for all keys having given sequence of bytes anywhere their key path bytes
-vs, --value-string "VALUE_NAME"	A string value: -vs "ctfmon" would search for all values having given string anywhere in their value name
-vr, --value-regex "REGEX_PATTERN"	A regex string: -vr "ctfmon\\.exe\$" would search for value name which ends with "ctfmon.exe" string
-vb, --value-bin "ESCAPED_BYTES"	A binary value expressed in UTF8 escaped hex string, example: -vb "\\x63\\x74\\x66\\x6D\\x6F\\x6E" would search for all values having given sequence of bytes anywhere their name bytes
-ds, --data-string "STRING"	A string value: -ds "CTF" would search for all values having given string anywhere in their data value
-dr, --data-regex "REGEX_PATTERN"	A regex string: -dr "^CTF" would search for all values which starts with "CTF" string
-db, --data-bin "ESCAPED_BYTES"	A binary value expressed in UTF8 escaped hex string, example: -db "\\x43\\x54\\x46" would search for all values having given sequence of bytes anywhere their data bytes
-dl, --data-size "DATA_SIZE_PATTERN"	Comma separated Data size patterns: [int]..[int] or [Operator][int], where operator is one of: ">", "<" or "=". Example: -dl "3000..3225","=3226"
Output arguments:	
-b, --baseline-output	Output to SQLite3 format (Used for baseline)
-o, --output-file "FILE_PATH"	Filepath to output file (Default: stdout)
-e, --export-registry-folder "FOLDER_PATH"	Folder path for exported .reg files
-E, --export-registry-file "FILE_PATH"	File path for exported single .reg file
-a, --output-format-all	Output all format fields [Time consuming task]
-f, --output-format "FORMAT_STRING"	Format of output data. (Default="key_timestamp,special,key_path,value_name,value_content")
-d, --output-delimiter "DELIMITER_STRING"	Used to separate the output fields (Default: ",")
Plugin arguments:	
-p, --plugins "PLUGIN_NAME", "PLUGIN_NAME"	Specify plugins to run: autoruns , dromedan , "..."

Remarks:

- Parameters can be used in any order
- Parameters containing space shall be put in double quotes (like: -s "folder/src folder")

Theory:

As mentioned in the tool's description, it's able to scan registry hives, but what the hive is exactly? A hive is a logical group of keys, sub keys and values in the registry that has a set of supporting files containing backups of its data. Each time a new user logs on to a computer, a new hive is created for that user with a separate file for the user profile (This is called the user profile hive). In general all user profile hives are mounted under the **HKEY_USERS** root in associated user SID sub keys (This mapping is dynamically created by Windows Registry Editor and built out of several ntuser.dat hive files). Please have a look on following standard mapping schema:

Standard hive mapping:

Hive mapping	Hive file
HKEY_CURRENT_CONFIG	System
HKEY_CURRENT_USER	Ntuser.dat
HKEY_LOCAL_MACHINE\SAM	Sam
HKEY_LOCAL_MACHINE\Security	Security
HKEY_LOCAL_MACHINE\Software	Software
HKEY_LOCAL_MACHINE\System	System
HKEY_USERS\.\DEFAULT	Default

According to my research, Windows Registry Editor keeps track of all mounted hives in the following key: **HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\hivelist** [Taken from Windows 7 SP1 x86]

(Default)	REG_SZ	(value not set)
\REGISTRY\MACHINE\BCD00000000	REG_SZ	\Device\HarddiskVolume1\Boot\BCD
\REGISTRY\MACHINE\HARDWARE	REG_SZ	
\REGISTRY\MACHINE\SAM	REG_SZ	\Device\HarddiskVolume2\Windows\System32\config\SAM
\REGISTRY\MACHINE\SECURITY	REG_SZ	\Device\HarddiskVolume2\Windows\System32\config\SECURITY
\REGISTRY\MACHINE\SOFTWARE	REG_SZ	\Device\HarddiskVolume2\Windows\System32\config\SOFTWARE
\REGISTRY\MACHINE\SYSTEM	REG_SZ	\Device\HarddiskVolume2\Windows\System32\config\SYSTEM
\REGISTRY\USER\DEFAULT	REG_SZ	\Device\HarddiskVolume2\Windows\System32\config\DEFAULT
\REGISTRY\USER\S-1-5-19	REG_SZ	\Device\HarddiskVolume2\Windows\ServiceProfiles\LocalService\NTUSER.DAT
\REGISTRY\USER\S-1-5-20	REG_SZ	\Device\HarddiskVolume2\Windows\ServiceProfiles\NetworkService\NTUSER.DAT
\Registry\User\S-1-5-21-3683099907-3949995901-678473381-1000	REG_SZ	\Device\HarddiskVolume2\Users\lab\NTUSER.DAT
\Registry\User\S-1-5-21-3683099907-3949995901-678473381-1000_Classes	REG_SZ	\Device\HarddiskVolume2\Users\lab\AppData\Local\Microsoft\Windows\UsrClass.dat

Additionally based on the observation, some registry hives have their own mappings via registry symbolic links like CurrentControlSet:

The registry key **HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet** is a symbolic registry link, which points to a copy of current control set, which finally holds the current configuration of Windows Services loaded by Windows loader.

The OS maintains several copies of control sets and stores them in the keys with following naming standard: **HKEY_LOCAL_MACHINE\SYSTEM\ControlSetXXX** (Where XXX is a set number like 001, 002 or 003). The key **HKEY_LOCAL_MACHINE\SYSTEM\Select** and its values determine the role of each control set respectively:

Current - If set to 1, the CurrentControlSet would point to ControlSet001

Default – Unclear and untested. On the machine without a failure, seems to point to Current

Failed - Holds the set number for the last unsuccessful boot (By default 0, indicates no failure)

LastKnownGood - Holds the set number, for the last successful boot like ControlSet002

Ref:

<https://msdn.microsoft.com/en-us/library/bb742541.aspx>

Windows Internals Book

- **Conclusion:** IF you want to search for services in CurrentControlSet key, you shall load the « system » hive and enumerate appropriate ControlSetXXX directly.

Some other examples could be:

HKEY_LOCAL_MACHINE\Software -> (hive: **software**) directly exposes the sub keys of Software*

HKEY_CURRENT_USER\Software -> (hive: **ntuser.dat**) has several sub keys, including Software

HKEY_CURRENT_USER\Software\Classes -> (hive: **classes**) directly exposes sub keys of Classes key

Remark: Consider using option “-qs” if you are unsure how to properly build the search pattern. This option would only print all sub keys of given hive(s) with associated hive_path (without the root key)

Examples

Following query would execute autoruns-like scan and enumerate services on all hives from the hives/ folder recursively. (Result would be printed into standard output)

```
rp.py -r -s hives -p autoruns,services
```

Following query would display immediate sub keys of every registry hive in “hives/ folder”. (This option ignores the output format specified by -f option)

```
rp.py -s hives -qs
```

Output :

```
hives/NTUSER.DAT,AppEvents
hives/NTUSER.DAT,Console
hives/NTUSER.DAT,Control Panel
hives/NTUSER.DAT,Environment
hives/NTUSER.DAT,Identities
hives/NTUSER.DAT,Keyboard Layout
hives/NTUSER.DAT,Printers
hives/NTUSER.DAT,Software
hives/NTUSER.DAT,UNICODE Program Groups
hives/NTUSER.DAT,Windows 3.1 Migration Status
```

Following query looks for value named “StubPath” in every sub key of “...\Installed Components” key. (It demonstrates limited wildcard support)

```
rp.py -s hives/sys_SOFTWARE.tmp -qv "Microsoft\Active Setup\Installed Components\*\StubPath"
```

Output (Shortened):

```
sys_SOFTWARE.tmp,,CMI-CreateHive{199DAFC2-6F16-4946-BF90-5A3FC3A60902}\Microsoft\Active
Setup\Installed Components\{89B4C1CD-B018-4511-B0A1-
5476DBF70820},StubPath,C:\Windows\system32\Rundll32.exe C:\Windows\system32\mscories.dll,Install
```

Following query looks for all keys which end on “\Run” or “\RunOnce” (It demonstrates the regex support)

```
rp.py -s hives/sys_SOFTWARE.tmp -kr "\\Run$", "\\RunOnce$"
```

Output (Shortened):

```
sys_SOFTWARE.tmp,,CMI-CreateHive{199DAFC2-6F16-4946-BF90-
5A3FC3A60902}\Classes\SDRun.AutoPlayHandler\shell\run,None,None
sys_SOFTWARE.tmp,,CMI-CreateHive{199DAFC2-6F16-4946-BF90-5A3FC3A60902}\Microsoft\Group
Policy\Client\RunOnce,{ECDBB9F9-F918-4B9C-BCA5-BC7518869245},
sys_SOFTWARE.tmp,,CMI-CreateHive{199DAFC2-6F16-4946-BF90-
5A3FC3A60902}\Microsoft\Windows\CurrentVersion\Run,IgfxTray,C:\Windows\system32\igfxtray.exe
sys_SOFTWARE.tmp,,CMI-CreateHive{199DAFC2-6F16-4946-BF90-
5A3FC3A60902}\Wow6432Node\Microsoft\Windows\CurrentVersion\Run,IMSS,"C:\Program Files
(x86)\Intel\Intel(R) Management Engine Components\IMSS\PlconStartup.exe"
```

Following query demonstrates how to search for hidden value names containing a NULL byte (\x00). Thanks to **-v** option you may observe what’s the real binary search pattern. Additionally via **-f** option you may adjust the output format, this time we are especially interested in “**value_name_unicode**” so we can see the exact position of the NULL byte:

```
rp.py -v -s hives/NTUSER-Trojan.Poweliks.dat -vb "\x00" -f "key_path,value_name_unicode,value_content"
```

Output (Shortened):

```
PARSER Started: 2017-08-13 13:21:21
WARNING: Binary arguments (-kb, -vb, -db) do not support case insensitive mode. Enabling cases-sensitive
mode...
Loading hives from: hives/NTUSER-Trojan.Poweliks.dat
Hives:
- hives/NTUSER-Trojan.Poweliks.dat

Parameters:
- Export to: None
- Case sensitive: True

Search entries:
- { VALUE | BIN | b'\x00\x00' }

Load hive: hives/NTUSER-Trojan.Poweliks.dat
$$$PROTO.HIV\Software\Microsoft\Windows\CurrentVersion\Run,b'\x00\x00a\x00',rundll32.exe
javascript:"..\mshtml,RunHTMLApplication ";document.write("\74script language=jscript.encode>...
```

Useful registry locations:

-qv "ControlSet001\services\Tcpip\Parameters\Interfaces*\DHCPIPAddress"

- Get the IP addresses of interfaces configured by Dynamic Host Configuration Protocol (DHCP):

-qv "Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tasks*\Path"

-qk "Software\Microsoft\Windows\CurrentVersion\Internet Settings"

-qk "ControlSet001\Control\Session Manager\Memory Management\PrefetchParameters"