Registry Decoder

By Russ McRee – ISSA member, Puget Sound (Seattle), USA Chapter

Join the Discussion
Connect

Prerequisites

Binaries require no external dependencies; working from a source checkout requires Python 2.6.x or 2.7.x and additional third-party apps and libraries.[1]

Merry Christmas: *"Christmas is not a time nor a season, but a state of mind. To cherish peace and goodwill, to be plenteous in mercy, is to have the real spirit of Christmas."* - Calvin Coolidge

Readers of the SANS Computer Forensics Blog[2] or Harlan Carvey's Windows Incident Response[3] blog have likely caught wind of Registry Decoder.[4] Harlan even went so far as to say "sounds like development is really ripping along (no pun intended). If you do any analysis of Windows systems and you haven't looked at this tool as a resource, what's wrong with you?" When Registry Decoder was first released in September 2011, I spotted it via Team Cymru's Dragon News Bytes mailing list and filed it away for future use. Then, in most fortuitous fashion, Andrew Case, one of the Volatility developers I'd reached out to for September's Volatility column, contacted me regarding Registry Decoder in early November. Andrew co-develops Registry Decoder with Lodovico Marziale as part of Digital Forensic Solutions and kindly provided me with content for the remaining entirety of this introduction.[5]

Registry Decoder is open source (GPL), written completely in Python, and is downloadable via Google Code projects.[6] It was initially funded by the National Institute of Justice and now is funded by Digital Forensics Solutions.

Registry Decoder was devised to automate the acquisition, analysis, and reporting of registry contents. To accomplish this, there are actually two projects. The first is Registry Decoder Live, which allows for the safe acquisition of registry files from a live machine by forcing a system restore point, thus putting the currently active registry files into a read-only state in backup. It then reads these files from backup either in System Restore Points for XP or from the Volume Shadow Service on Windows Vista and Windows 7. As Registry Decoder Live acquires files, it creates a database that can then be imported into the second tool, Registry Decoder.

Registry Decoder can analyze registry files from a number of sources and then provide a number of GUI-driven analysis capabilities. The current version of the tool (1.1 as this is written) can import individual registry files, raw (dd) disk images, raw (dd) split images, Encase (E01) images, and databases from the live tool. Once evidence is imported and pre-processed, the investigator then has a number of analysis tools available, and new evidence can be added to a case at any time.

Registry Decoder's analysis capabilities include:

- Browsing Hives (similar to Access Data's Registry Viewer)
- Hive Searching (more on this below)
- Plugin System (similar to regripper)
- Hive Differencing
- Timelining based on last write time
- Path-Based Analysis
- Automated reporting of all of the above

Registry Decoder automates all of this functionality for any number of registry hives, and the reporting can handle exporting results from multiple hives and analysis types into one report.

Andrew's favorite Registry Decoder use case is USBSTOR analysis. Almost every case involving investigating a specific employee requires determining which (if any) USB drives were in use. To do this with Registry Decoder, all an investigator has to do is create a case with the disk images or hives acquired, run the *USBSTOR* plugin, and then export the results. After pre-processing is done, it takes mere minutes to have a report created with the device name, serial number, etc. of any devices connected. Also, since Registry Decoder pulls historical files from live machines and disk images (System Restore and Volume Shadow Service), this analysis can be run across hives going back months or years.

1 http://code.google.com/p/registrydecoder/downloads/detail?name=RegistryDecoder-Offline-Analysis-Instructions-v1.1.pdf.

2 http://computer-forensics.sans.org/.

3 http://windowsir.blogspot.com/.

4 http://www.digitalforensicssolutions.com/registrydecoder/.

5 Content provided by Andrew Case, Registry Decoder developer

6 http://code.google.com/p/registrydecoder/ and http://code.google.com/p/regdecoderlive/.

Similarly, while investigating data exfiltration between multiple employees of a company, Andrew needed to know if they shared USB drives. To make the determination he took the SYSTEM files from each machine, loaded them into Registry Decoder and then used the plugin differencing ability on the *USBSTOR* plugin. It immediately revealed what drives were shared between computers, including their serial number. Another common use of the differencing feature is with the Services plugin as this quickly identifies malware if you difference your known good disk image vs. a disk image of a machine suspected to be infected.

Registry Decoder's search feature is one of its strongest features. It allows you to search across any number of hives and filter by keys/values/names, last write time range, wildcard searching, and bulk searching with keyword files.

For a recent case, Andrew had to determine if a person was accessing files he shouldn't have been looking at. They had a desktop and a laptop, both running XP and both with many System Restore Points. In less than 30 minutes with Registry Decoder, Andrew needed only load the disk images from the two machines into Registry Decoder, make a text file with all the search terms, and then search all the terms across all the hives in the case (including historical ones). This returned results that he then exported into one report and was finished. Another useful search is noted when viewing the search results tab, right click on any result, and immediately jump into the *Browse* view positioned at that key.

Another good use case includes path-based analysis, which allows you to determine if a registry path exists in any number of files. For whichever files it is present in, one can then export the path and optionally its key/value pairs. This is extremely useful in two situations:

1. Determining if certain software is installed (P2P, cracked software, etc.), as you can simply search any of the paths that the program creates and then export its key/values inclusive of when and where the software was installed.
2. During malware analysis as most malware writes to the registry. Searching across numerous suspect systems for the malware's path allows investigators to immediately determine the extent of infection.

Registry Decoder's roadmap includes more analysis plugins and added support for memory analysis (integrate with Volatility's existing in-memory registry functionality).

The developers also want to add support for analyzing previously deleted keys and name/value pairs within hives. The library utilized for enumerating hives, *reglookup*, already supports this functionality, so it is just a matter of integration.

Running the Registry Decoder online acquisition component

I ran regdecoderlive32 on a 32bit Windows XP SP3 virtual machine infected with Lurid and regdecoderlive64 on a Windows 7 SP1 64bit machine.

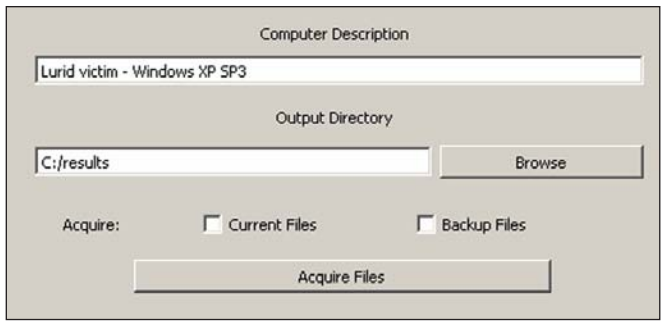


Figure 1 – Registry Decoder Live

One note for regdecoderlive32 on Windows XP systems with drives formatted with NTFS. Even when running regdecoderlive32 with administrator privileges, the hidden System Volume Information directory is protected with unique ACLs. To circumvent this issue, issue `cacls "C:\System Volume Information" /E /G <username>:F` from a command prompt at the root of C: (this assumes the OS is installed on C:).

As seen in Figure 1, running regdecoderlive is as simple as executing and defining a few parameters including description, output directory (must be empty) and check boxes for acquisition of current and backup files.

Once acquisition is complete, the results directory will be populated with `registryfiles/acquire_files.db` and related files. This results directory can (should) be written to portable storage mounted on the target system or a network share, which can then be consumed by Registry Decoder for offline analysis.

Running the Registry Decoder offline analysis component

Registry Decoder can consume individual registry files, raw (dd) disk images, and Encase (E01) images, including split images. Building a case is as easy as adding a case name and number, investigator, comments, and case directory. Adding evidence to a case after initial processing is created is quite simple; you'll be prompted to add new evidence after choosing *Start Case* and opening an existing case.

I only tested Registry Decoder with the acquisition database acquired from a Lurid-infected Windows XP VM via Registry Decoder Live.

Initial processing can take some time depending on the number of restore points or volume shadows. Once initial processing is complete however, Registry Decoder is nimble and effective.

I mimicked some of Andrew's use cases in this analysis of a Lurid[7] victim. From runtime analysis of the Lurid sample I had (md5: 84d24967cb5cbacf4052a3001692dd54) I knew a few key attributes to test Registry Decoder with. Services and registry keys created include `WmdmPmSp`. As the search

7 http://us.trendmicro.com/imperia/md/content/us/trendwatch/researchandanalysis/12802_trend_micro_lurid_whitepaper.pdf

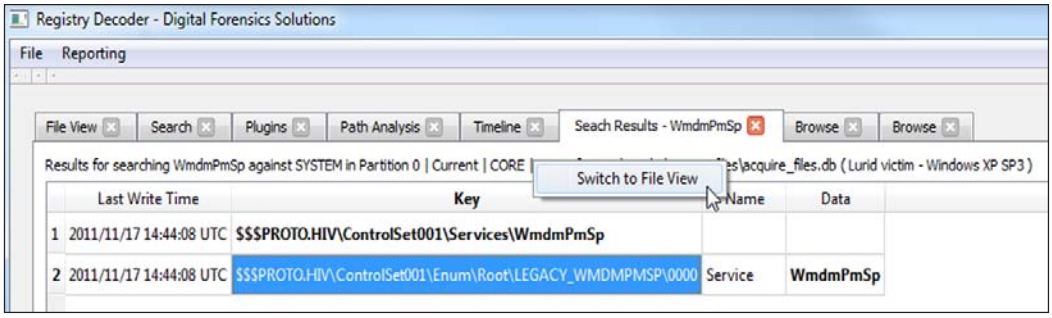


Figure 2 – Registry Decoder search results

functionality is a strong suit, I selected *CORE* from the current snapshot acquired and searched `WmdmPmSp`. Right-click search results and select *Switch to File View* then navigate to the *Browser* tab for key values, etc. as seen in Figure 2.

I made use of the timeline functionality and was amply rewarded. Imagine a scenario where you have a ballpark time window for a malware compromise or unauthorized access. You can filter the timeline window accordingly and produce output that is compliant to the SleuthKit's mactime format. It's not human readable currently (next release) so read it in with Autopsy or TSK. Timeline gathering and results are combined in Figure 3. It clearly identified exactly when Lurid wrote to `HKLM\SYSTEM\CONTROLSET001\SERVICES\WmdmPmSp`.

I also tested USBSTOR (unrelated to Lurid) on both acquisitions (Windows 7 and Windows XP) and the results were accurate and immediate in both cases as seen Figure 4.

Explore the *Plugins* options included with Registry Decoder; the possibilities are endless. *SYSTEM* will provide you a nice summary overview as you begin; *IE Typed URLs* is great for inappropriate browser use; *Services with Perform Diff* enabled is excellent for malware hunting; *System Runs* will give you instant gratification regarding what's configured to run on startup; *ACMRU* queries the registry keys that have been typed into the Windows Search dialog box; and on and on and on. ☺ Brilliant!

In conclusion

I'm extremely excited about this tool and imagining its use at scale to be of incredible use for enterprise incident responders and forensic examiners. I've been chatting with Andrew at length while writing this and he continuously mentions pending features including some visualization options and the aforementioned Volatility interaction. I can't wait; check out Registry Decoder out for yourself ASAP.

Merry Christmas!

Ping me via email if you have questions (russ at holisticinfosec dot org).

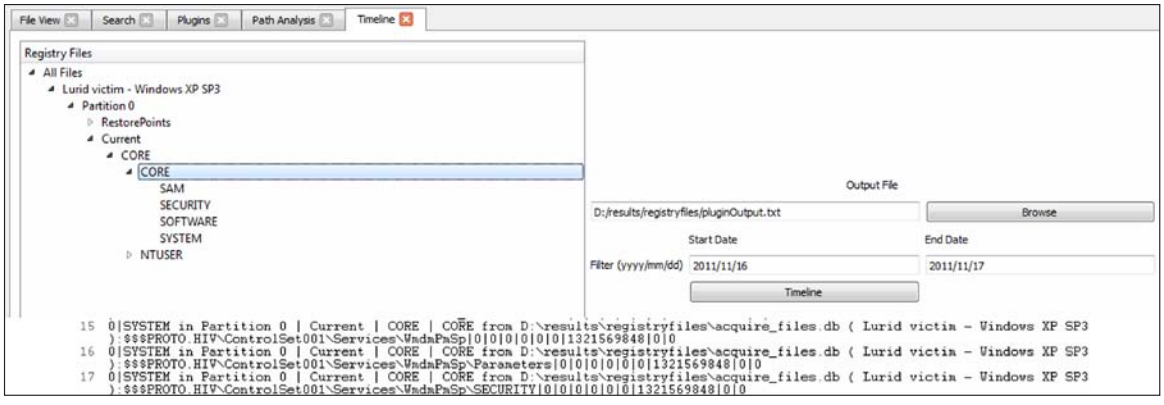
Cheers…until next month.

Acknowledgements

—Andrew Case, Registry Decoder developer and project lead

About the Author

Russ McRee, GCIH, GCFA, GPEN, CISSP, is team leader and senior security analyst for Microsoft's Online Services Security Incident Management team. As an advocate of a holistic approach to information security, Russ' website is holisticinfosec.org. Contact him at russ at holisticinfosec dot org.



(Left)
Figure 3 – Registry Decoder timeline results

(Below)
Figure 4 – Registry Decoder USBSTOR results