Whodunnit

1. Summary ..................................................................................... (2,3)
2. Current Status and Roadmap ........................................................ (3)
3. Selection and Development Progression ...................................... (4)
4. Issues Encountered ....................................................................... (4)
5. Conclusion and Links ..................................................................... (5)

Author

Cam Downs – 1cysw0rdk0

1. Summary

Whodunnit is a forensic tool initially created with the intention of allowing the user to quickly sort windows event logs based solely on the user that was responsible for the generation of that log. Over time, this has evolved to allow for filtering on far more criteria than simply a username. The overreaching goal of this project was to reduce the time at keyboard for a forensic investigator, by allowing for an automatable sorting process, and that is still the end goal. However, currently the command line interface is not operational, and the GUI is fully functional.

Whodunnit allows for a user to read windows event logs either from a previously exported dump, or from the local machine. At the time, it reads all the logs that it has access to, meaning that in order to acquire the Security Logs, it must be run with proper permissions. Following reading in logs, the user has the option to load a filter from a previously exported filter, or create a custom filter.

Currently, there are 6 properties that can be used to filter the logs loaded, including usernames, both a starting and ending date and time, event codes, event types, and event sources. The list of usernames entered is a negative search list, meaning all usernames included in that list will not be included in the final list of logs matching this filter. The time window filter selects only logs that fall between both of the times specified. If the start time is omitted, all logs before the end time are selected, and if the end time is omitted, all logs after the start time are selected. The list of event codes is a positive search list, with a default value of ‘\*’. If ‘\*’ is anywhere in the list of specified codes, all event codes are accepted. The ‘event types’ criteria allows the user to select event types to include, from a list. Available options are Error, Warning, Information, Success Audit, and Failure Audit. Finally, the ‘event sources’ criteria allows the user to select which sources to include in the final set of logs. Available sources are Application logs, Hardware Event logs, Internet Explorer logs, Key Management logs, OAlerts logs, Security logs (requires administrative rights), System logs, Windows Azure logs, and Windows PowerShell logs.

The active filter is applied immediately after backing out of the editing menu, or immediately after loading a filter from a file. This may take a significant amount of time, especially if system logs are included, or if a significant number of logs are being sorted through. The filtering algorithm is O(n \* (j + k + l)), where ‘n’ is the number of logs to filter through, note that the sources are stored independently, and only sources in the current filter are looked at. ‘j’, k’, and ‘l’ are the lengths of usernames, event codes and event types lists.

1. Summary (cont.)

The display function is intended to give the user a quick glimpse at how many logs will be exported compared to the total number read in. As such, the display function does not list out all of the logs read in. Instead, a table is printed with the numbers of logs in each log source, both before and after filtering them.

Finally, the export function allows the user to write logs to a file. The user is prompted to select whether to export all logs, or only logs matching the current filter. Then the user is prompted for a file path. An XML file is written to that path, containing either all or only the filtered logs, dependent on the user’s selection.

1. Current Status and Roadmap

In its current state, Whodunnit is ready for use on local machines. Everything described as above functions in all tested scenarios, which admittedly are fairly limited, as I only have access to one windows machine at the moment.

However, there is still work to be done on this tool. In the coming weeks, I will be implementing a command line interface for Whodunnit, which will require a second mode of operation, and should not be too much of a challenge.

As for automation of tasks, that is not possible with the current state of Whodunnit. In its current state, the logs read in, the filtered logs, and the currently applied filter are stored as script global variables, which prevents calling subroutines from the command line, or dot sourcing the script and calling individual functions from outside the scope of the script. Without a command line interface, or the ability to import functions, automation is not possible. Refactoring the project to allow for dot sourcing is likely a far larger undertaking than creating a command line interface, which is planned to begin in late January of 2019.

Following the above mentioned changes, I plan to add support for ranges of event codes, support for exporting logs to different formats, support for reading logs from a remote machine, investigate an issue holding back the use of QuietusPlus’s interactive PowerShell menu script (linked at the bottom of the README.md for this project). The long term project is investigating the possibility of recovering logs from a forensic image of a windows machine.

1. Selection and Development Progression

After consulting with a friend from the Connecticut State Police’s Computer Forensic Department, it became apparent that they had no tool which allowed them to sort windows event logs by the user that created them, and as a result spent far too long digging through arbitrary and irrelevant logs to find the ones they needed. Thus, Whodunnit was born, with a singular goal: reduce the number of irrelevant logs when investigating a windows machine.

My original approach was to only allow the user to filter based on usernames, since criminal investigations tend to focus on an individual’s wrong doing. However, it became apparent that even focusing on a single user leaves investigators with an enormous amount of logs to sift through. It was then that Whodunnit evolved to support selecting events in a time window, then searching for specific events, or event types. Finally, in an effort to reduce runtime, I added an option to completely exclude massive portions of the logs which may be irrelevant, without having to iterate over those logs. This was done by storing logs from each source independently of each other, and simply not running the filter routine on log sources which were not to be included in the output.

1. Issues Encountered

Windows PowerShell is a very powerful language, once you get the hang of its quirks. While working on this project, I encountered several of these quirks which forced me to change my expectations of the project. One of these issues was with the Export-CSV function in PowerShell, which exports an object input to a csv file. The objects I created to store the information for this project have several properties, and some of these properties happen to be arrays. Unfortunately, the Export-CSV function calls the .ToString() method on all properties, and writes the return of that to the file. Calling .ToString() on an array in PowerShell simply returns the string “System.Object[]”. This means whatever data was in that array is lost when it is exported to a csv. I chose the simplest workaround to this, and exported everything as an XML file instead.

1. Conclusion and Links

In its current state, Whodunnit does fulfill its initial goal of saving investigators’ time at a keyboard. However, in the coming weeks and months, the time saved could be drastically increased with the introduction of a command line interface, and the refactoring of the code to allow for modularity, and use in outside scripts. As for the GUI, in the future it may receive an overhaul if I can get the Write-Menu functions to work.

Links

Whodunnit Repository: <https://github.com/1cysw0rdk0/DFIR_cmd7983_CSEC464/tree/master/whodunnit>

QuietusPlus’s Write-Menu (not currently used):

<https://github.com/QuietusPlus/Write-Menu>