Collection of state information in live digital forensics

Fábio Freitas[1] and António Pinto[2]

[1] GCC, CIICESI, ESTG, Politécnico do Porto, Portugal
8080176@estgf.ipp.pt
[2] GCC, CIICESI, ESTG, Politécnico do Porto
and CRACS & INESC TEC, Porto, Portugal
apinto@inesctec.pt

Abstract. In a digital forensic investigations, the investigator usually wants to get as much state information as possible. Examples of such scenarios are households with wireless networks connecting multiple devices where a security incident occurs. USB devices present themselves as interesting vehicles for the automated collection of state information, as it can store the applications that collect the information, can store the results and can also facilitate the information collection by enabling its automatic operation. This paper proposes a USB solution to facilitate the collection of state information with integrity guarantees and multi-platform operation. Moreover, the proposed solutions is the only one that performs an extensive and homogeneous artifact collection, independently of the underlying operating system.

Keywords: Digital forensics; Live; State information.

1 Introduction

In Incident Response (IR) [1] situations, the investigator tries to collect as much information as possible. The goal of an IR investigation can be to confirm the existence of the incident, to provide rapid detection and containment, to identify facts and collect information, to minimize business interruption or network operation, to recover from the incident, to manage the public perception of the incident, to collect evidence that enables legal or civil action against the perpetrators, to inform the top management, or, finally, to improve the organization's reaction to future security incidents [4].
State information can be included within the collectible information and its analysis emerges as an important aspect of digital forensics, especially when dealing with IR situations that involve multiple networked equipment.This type of procedure is usually named as live forensics. The objective of this type of forensic analysis is to collect volatile data before shutting down the system to be analyzed. In live forensics, one collects information such as a copy of Random Access Memory (RAM) memory or the list of running processes. State information is volatile and will be lost once the equipment is turnoff. State information may contain artifacts

of interest to the analyst, such as [12]: the list of running processes; the history of commands executed on the console; recently used passwords (some in readable text); instant messages; IP addresses; who is logged on to the system; which network ports and applications are listening for IP connections; system information; list and history of connected devices, among others.
The collection of state information in live forensics should not be seen as a substitute of traditional forensic analysis. Instead, it should be seen as complementary. Live forensics will collect information that is not available otherwise. Traditional forensic analysis involves the bit-by-bit copying of data stored in media and its subsequent analysis. Such bit-by-bit copy and analysis is typically performed on a reduced set of equipment since it is a lengthy process. On the other hand, the collection of state information, which can be automated, becomes a source of complementary information that can be very useful. It enables the identification of the connections established between various devices and, hence, check for viruses, spyware, malware or other malicious programs that use the network to communicate with a control server, for example.
Universal Serial Bus (USB) storage devices are interesting vehicles for building an automated mechanism that collects state information by storing both the applications required for collecting the information and the collection result. These are also easily transportable, can have large storage capabilities and facilitate the automatic collection after insertion into the PC.
Live forensic procedures are not risk-free. On the one hand, in order to collect state information, we are required to use the system under analysis that may be compromised and, possibly, may compromise the collected information. On the other hand, the collection of state information is an intrusive process, in the sense that the collection of state information changes the state of the system being analyzed. Traditional digital forensic analysis focuses on avoiding changes to the analyzed system.
This article is organized in sections. Section 2 describes and compares the existing applications that perform artifact collection in the context of digital forensic investigations or of incident response situations. Section 3 details the proposed solution. Section 4 presents the tests carried out in order to validate de proposed solution and, Section 5, concludes this paper.

2 Related work

Multiple applications that collect artifacts in the context of live digital forensics and incident response are available online and can be obtained freely. Examples being: the ir-triage-toolkit (ITT) [13], the IRKIT [11], the TR3Secure (TR3S) [3], the TR3Secure's fork by Neely (NTR3S) [9], the Live Response Collection (LRC) [5], and the Live Response Scripts (LRS) [6].
The ITT consists of a set of small applications and scripts. The author includes applications of other entities in his set, but states that no license agreements or copyright restrictions are broken. These scripts aim to

automate the collection of artifacts to facilitate their subsequent analysis, as well as, the screening of events in incident response scenarios. It can be used in Windows and Linux Operating Systems (OS). It also provides copies of RAM and includes scripts to create tool kits for Linux and Windows.

The IRKIT was created by Bill Dean for his own use. He put together a set of free applications that collect volatile data in a script. All applications should fit on a USB storage medium and be ready for use in IR situations. Can only be used in systems with the Windows OS.

The TR3S is yet another set of scripts for collecting digital artifacts on connected systems. It was developed by Corey Harrell and is made available in his blog *Journey Into Incident Response*. Harrell needed a set of applications to respond to systems during attack simulations and one of the applications had to quickly collect volatile data. The commands required for the proper functioning of TR3S are not made available directly due to copyright limitations. Can only be used in systems with the Windows OS. The NTR3S is a fork of the TR3S project by Neely. The motivation behind this fork was related to the need to adapt TR3S to the way that Neely's response and screening team worked in malware detection situations.

The LRC was developed by Brimor Labs and consists of a collection of commands and scripts for the collection of artifacts in digital forensic investigation scenarios. This has the particularity of being able to be used in multiple OS, such as Windows, Linux and macOS. The application was tested on a wide variety of OS versions.

The LRS takes the form of three scripts for the collection of artifacts in digital forensic investigation scenarios, one script for each reference OS. Scripts for Linux and macOS do not depend on external applications.

Table 1 compares the tools identified. Three main factors were considered in this comparison: the multi-platform support, the amount of information collected, and if the collected information was homogeneous among the various supported OS. The table shows the percentage of artifacts collected by each tool, organized by information groups and by client OS (Windows (W), Linux (L) and macOS (M)). Each percentage was obtained by calculating the number of artifacts each tool collected over the total number of artifacts collected by all tools, per information group.

The network information group considers artifacts such as: the network interfaces configuration, IP and MAC addresses, routing table, ARP table, DNS cache, network connections, and associated processes. The status information group considers artifacts such as: open files, running processes (process identification number, running time), and a complete list of files in the file system. The system information group considers artifacts such as: the system date and time, including the time zone, OS version, list of services and programs configured to automatically run at startup, system configuration, and the list of installed software. The user information group considers artifacts such as: the list of tasks scheduled to run automatically, the list of local user and group accounts, and the history of system accesses. Device information group considers artifacts such as: the loaded drivers and modules, memory capacity, hard disk information, and the list of mounted file systems. Integrity information

Table 1. Collected artifacts by information group (in %)

	ITT			IRKIT			TR3S			NTR3S			LRC			LRS		
Information	W	L	M	W	L	M	W	L	M	W	L	M	W	L	M	W	L	M
Network	60	20	-	80	-	-	100	-	-	100	-	-	60	80	100	80	60	60
State	67	67	-	67	-	-	67	-	-	67	-	-	100	67	67	67	67	67
System	-	50	-	75	-	-	50	-	-	50	-	-	100	50	50	75	50	50
User	33	67	-	100	-	-	100	-	-	100	-	-	67	33	33	33	67	67
Devices	100	50	-	100	-	-	50	-	-	50	-	-	100	50	100	-	100	100
Integrity	100	100	-	-	-	-	-	-	-	-	-	-	100	100	100	-	-	-
Logs	-	-	-	-	-	-	100	-	-	100	-	-	50	100	100	-	-	-
Autorun	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
RAM copy	100	100	-	100	-	-	100	-	-	100	-	-	100	-	100	-	-	-

group considers artifacts such as: result of multiple cryptographic hash functions (MD5 [10], SHA1 [7], SHA512 [8]) over all collected artifacts and the tool itself. Finally, the log information group considers artifacts such as: the Operating System (OS) and application logs [4].
In conclusion, none of the identified tools performs an extensive and homogeneous artifact collection that is independent of the underlying OS.

3 Digital Forensic and Incident Response USB

In a digital forensic investigation, or in a response to a computer security incident, the investigator must comply with standardized procedures that include the use o specific applications and the validation of the integrity of the collected artifacts. The proposed solution, named Digital Forensic and Incident Response USB (DFIRU), aims to provide the investigator with an application that enables him to automatically collect artifacts in a standardized way and with integrity validation. Moreover, the artifact collection must be extensive, homogeneous, independent of the underlying operational system, portable and automatically executed when possible.

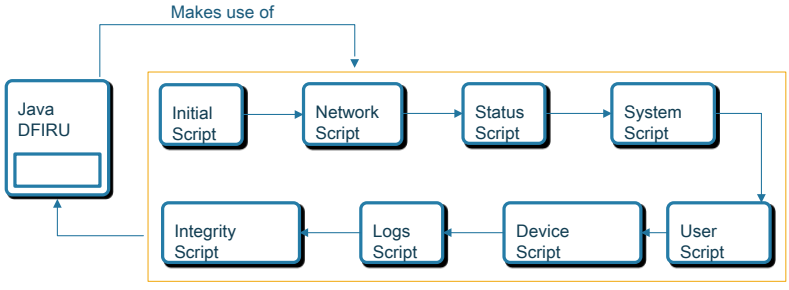

Fig. 1. DFIRU architecture

The proposed solution, depicted in Figure 1, consists of a main application, developed in Java, that runs OS specific collection scripts. The scripts are arranged in 7 information groups. In turn, these scripts will execute commands that will perform the artifact collection on the target system. The DFIRU application is stored in a USB drive. The storage device can be formatted in multiple formats, in the prototype the Extended File Allocation Table (exFAT) [2] format was selected due to its native read and write support in the target operating systems (Windows, Linux and macOS). Moreover, the exFAT supports files larger than 4GB, which are expected if memory dumps are performed. The automatic execution of the proposed solution is implemented using the USB *autorun* functionality. This feature has native supported on both Windows and Linux. In macOS, and newer versions of the Windows OS, the user is required to manually launch the proposed solution because the *autorun* functionality is being deprecated due to security considerations.

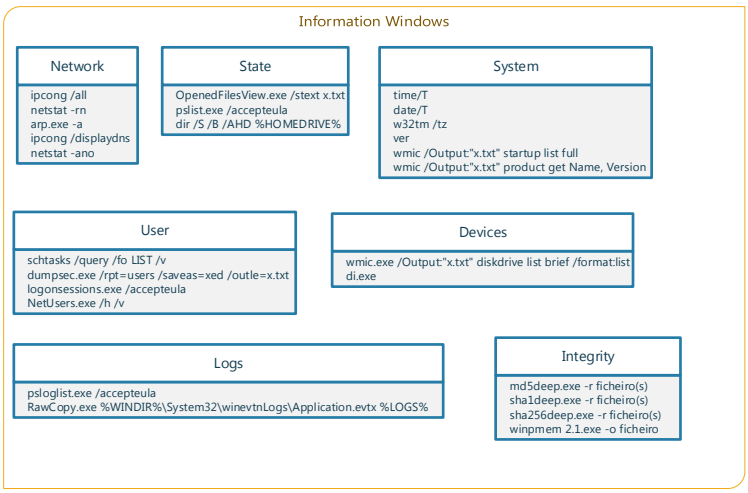

Fig. 2. Sample commands per information group (Windows)

Figure 2 presents a sample of the commands that are used by the scripts in the multiple information groups for the Windows OS. In order to collect artifacts of all information groups, 21 scripts were created, which were distributed by groups as follows: 2 for network, 2 for state, 2 for system, 2 for user, 2 for devices, 2 for logs, 8 for integrity, and one additional script that is used to prepare the output files and folders needed to save the collected artifacts. The need for 2 scripts per group is related to the level of privileges of the user that initiates the artifact collection (with or without administrator privileges). The integrity group required additional scripts because the application (see Figure 3) has extra features for this group. These were then duplicated for users with or without administrator privileges.

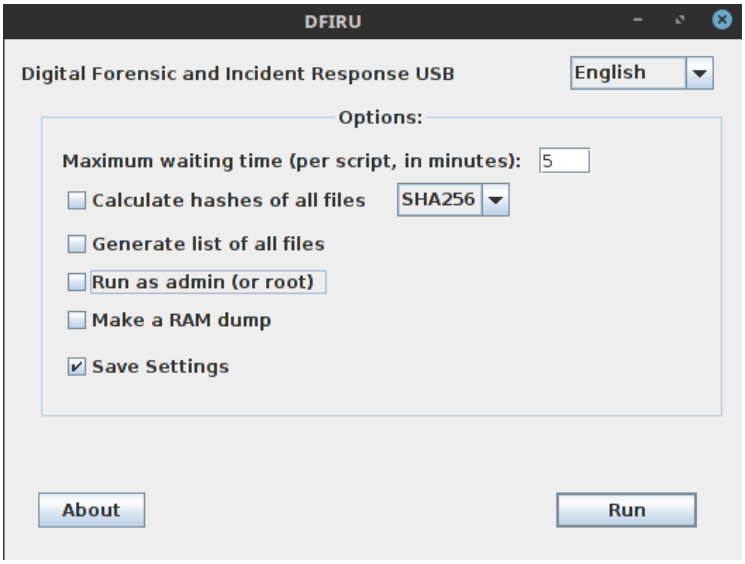

Fig. 3. Main window of the DFIRU java application

4 Validation

The proposed solution was installed on a USB storage device using a device setup script that was develop for that purpose. The installation script includes the formatting of the USB storage device, the copying of the java application and associated scripts to the correct folders within the USB storage device, and the copying or downloading of the utilities required by the various scripts. Some utilities, due to distribution restrictions imposed by their developers, are downloaded at this time. The installation script has variants for the various supported operating systems (Linux, Windows and macOS). This script also facilitated the process of validating the proposed solution.

In particular, the proposed solution was validated in terms of its overall functionality, portability and self-execution. A USB device was setup successfully with the proposed solution in all the reference operating systems. The USB device, after setup, was then disconnected and reconnected to test the *autorun* functionality. The *autorun* was successful in older versions of Windows and some Linux distributions, such as Fedora, but failed in recent versions of macOS and Windows 10. Moreover, the artifact collection with all available options selected was tested multiple times in all reference operation systems. The artifacts were correctly collected, resulting in a USB device containing the proposed solution and the collected artifacts. The proposed solution successfully registered its own activity and calculated cryptographic hash values of the collected artifacts and generated files.

5 Conclusion

The proposed solution consists of an application developed in Java that collects state information that is both extensive and homogeneous in all supported operating systems. Both the solution and the information collected are stored on a portable storage medium. Whenever allowed by the operating system, the proposed solution makes use of the self-execution mechanisms to automate the process of collecting artifacts. The proposed solution registers its own activity and calculates cryptographic hash values of the collected information and of the generated files. To the best of our knowledge, the proposed solutions is the only one that performs an extensive and homogeneous artifact collection, independently of the underlying operating system.

References

1. Al-Zarouni, M., Al-Hajri, H.: A proof-of-concept project for utilizing U3 technology in Incident Response. In: Australian Digital Forensics Conference. p. 15 (2007)
2. Hamm, J.: Extended FAT file system. Tech. rep., Paradigms Solutions (2009)
3. Harrell, C.: TR3Secure. http://journeyintoir.blogspot.pt/2011/12/jiir-updates.html (2011)
4. Luttgens, J.T., Pepe, M., Mandia, K.: Incident response & computer forensics. McGraw-Hill Education Group (2014)
5. Moran, B.: Live response collection. https://www.brimorlabs.com/tools/ (2015)
6. Musick, E.: Live response scripts. http://erikmusick.com/live-response-scripts/ (2011)
7. National Institute of Standards and Technology: Secure Hash Standard (SHS). FIPS 180-1, NIST (April 1995)
8. National Institute of Standards and Technology: Secure Hash Standard (SHS). FIPS 180-4, NIST (August 2015), http://dx.doi.org/10.6028/NIST.FIPS.180-4
9. Neely, K.: Tr3secure volatile data collection kit. https://github.com/ktneely/Tr3Secure (2014)
10. Rivest, R.: The MD5 Message-Digest Algorithm. RFC 1321, RFC Editor (April 1992), http://www.rfc-editor.org/rfc/rfc1321.txt
11. Sanabria, A., Compton, A., Dean, B.: Free volatile data collection kit. https://www.swordshield.com/2010/09/free-volatile-data-collection-kit/ (2010)
12. Shipley, T.G., CFE, C., Reeve, H.R.: Collecting evidence from a running computer. SEARCH, The National Consortium for Justice and Internationals Standards p. 6 (2006)
13. Shipp, R.: ir-triage-toolkit. https://github.com/rshipp/ir-triage-toolkit (2013)