Forensic Technologies to Automate the Acquisition of Digital Evidences

David García, Llanos Tobarra, Antonio Robles-Gómez, Rafael Pastor-Vargas
Departamento de Sistemas de Comunicación y Control
Escuela Técnica Superior de Ingeniería Informática
Universidad Nacional de Educación a Distancia (UNED)
Madrid, Spain
dgarcia2011@alumno.uned.es; {llanos,arobles,rpastor}@scc.uned.es

***Abstract*—The main goal of this work is to propose the automatic acquisition of evidences in a remote way. This automated capacity becomes interesting for companies with extensive networks and/or several locations, as it allows them to delegate and centralize the acquisition task at a single point in their structure, while saving time and travel costs. This research has been carried out through the initial implementation of a virtual laboratory made up of a network and different scenarios, by including an experimentation process. The virtual network includes both the machine from which automatic acquisitions are performed and the devices from retrieving the evidence. The group of devices will be made up of various experiments. The aim is to analyze the viability of the acquisition in different scenarios, since distributed networks are not homogeneous in the real world.**

***Index Terms*—CSCI-RTCW, Forensic technologies, automation, digital evidence acquisition, security policies**

I. Introduction

The generalization of the use of technology has led at the same time to an increase in the number of crimes committed through it. Therefore, digital evidence has gained fundamental importance in fields as diverse as civil litigation, criminal cases, administrative matters or military intelligence [1] [2]. The discipline that is responsible for dealing with this digital evidence is forensic analysis. Kruse and Heiser indicated in 2002 that forensics analysis implies the preservation, identification, extraction, documentation and interpretation of digital data that, like any other discipline, must follow clear and well-defined methodologies, maintaining enough flexibility when facing unusual situations [3]. On the other hand, Sammons defines forensics analysis as a discipline of forensic science, the latter being the application of science to resolve a legal problem [2], echoing the proposal of Ken Zatyko, who in 2007 defined forensic analysis as *"The application of computer science and investigative procedures for a legal purpose involving the analysis of digital evidence after proper search authority, chain of custody, validation with mathematics, use of validated tools, repeatability, reporting, and possible expert presentation" (Zatyko, 2007)."* [4].

Authors would like to acknowledge the support of the I4Labs UNED research group, the CiberGID UNED innovation group with the CiberScratch 2.0 project, the SUMA-CITeL research project for the 2022-2023 period, as well as the E-Madrid-CM Network of Excellence (S2018/TCS-4307). The authors also acknowledge the support of SNOLA, officially recognized Thematic Network of Excellence (RED2018-102725-T) by the Spanish Ministry of Science, Innovation and Universities.

In addition to this, the acquisition of evidence must guarantee that an acquired evidence is not compromised. To achieve this, it is necessary to adhere to the current legal regulation. This regulation in Europe is found in two standards. *UNE 71506:2013* [5] establishes a methodology for preservation, acquisition, documentation, analysis and presentation of digital evidences; and *UNE-EN ISO/IEC 27037:2016* [6] is the standard which contains the guidelines that must lead the acquisition phase and, therefore, the model to follow. This last standard establishes that digital evidence is governed by principles of relevance, reliability and sufficiency, in addition to emphasizing the need for a properly documented chain of custody.

According to this, it is proposed in this work to use the framework of *Ansible*, a configuration management tool, for the execution of utilities that allow the acquisition of evidence for forensic analysis in an automated way. This evidence acquisition will be carried out using recognized tools for its storage and guarantee of integrity. Everything following the standards set by current regulations.

In short, our proposed solution will also be able to:

1) Collect evidence reducing the impact on the original.
2) Guarantee that the acquired evidence has not been altered.
3) For all acquired evidence, identify at least where, when and by whom they were acquired.
4) Preserve the collected evidence in a suitable storage place.

Our proposal seeks, on the one hand, to minimize the human factor in the acquisition process that may cause the loss of probating value of the evidence, while reducing the costs of forensic analysis, as well as using open source tools saving costs in licenses. Finally, our proposal seeks to maintain flexibility and scalability in the acquisition system.

This paper is organized as follows. First, a review of existing proposals is detailed in Section II. Additionally, Section III presents details of our proposal for the automation of the evidence acquisition process. Section IV discusses the first results obtained in this work. A discussion is provided in Section V. Finally, Section VI presents our conclusions and further works.

II. State of the Art

A set of both commercial and open-source tools can be found in the market that, although they cannot be considered a complete solution, could be adapted to cover said needs. For instance, ISEEK XtremeForensics [7] [8], which is a tool developed for electronic discovery, forensics, malware detection, and regulatory compliance.

Additionally, a commercial tool is Cortex XDR Forensics [9]. It is integrated in the *Cortex XDR* suite, which is oriented to incident response and that includes capacities for collection and data analysis, search and correction of threats, as well as event logging and a state-of-the-art antivirus.

Another commercial tool is IBM QRadar Incident Forensics [10], offered by IBM, which constitutes a complete suite with monitoring, detection, research and response capabilities.

The Panda Adaptative Defense 360 [11] service, offered by the owners of Panda Security, WatchGuard, is a complete EDR (Endpoint Detection and Response based on light agents and with deployment in the cloud.

As an open-source tool, Velociraptor [12] stands out. Developed as a DFIR (Digital Forensic and Incident Response) tool, Velociraptor is an open-source suite that enables endpoints monitoring, forensic analysis and incident response.

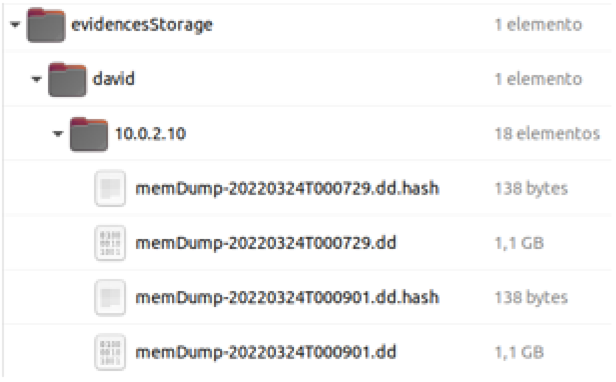
All these tools combine various functionalities in the same software. This has advantages such as centralization of information and the elimination of the need to switch from one application to another in each of the forensic analysis steps.

Our solution complies a modular proposal to allow flexibility when choosing between acquisition tools, thus avoiding the introduction of dependency on a specific vendor and, on the other hand, the need for incident response teams to abandon the use of their favorite tools. Additionally, we have detected the existing weaknesses of the current tools within the context in which we are moving, there are still needs that are not covered.

III. Our proposal

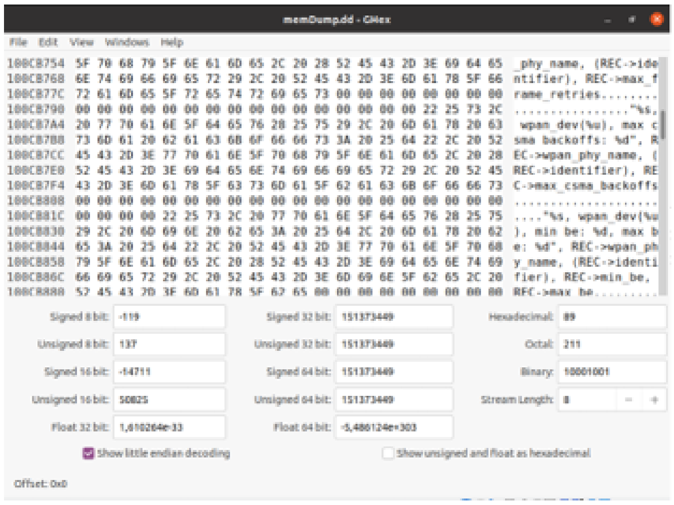
The determining criteria for choosing *Ansible* as the tool to perform configuration management and remote administration are: 1) its ability to centrally manage a large number of endpoints; 2) a minimum impact of configuration and training. An inventory of the network on which the system has to act will be carried out.

The system must check that the endpoint is ready so that the acquisition can be carried out and, if not, it must ensure that it is left in the appropriate state. For each type of evidence, this preparation will be different, choosing in each case the most convenient tool for the acquisition of the evidence. For the experiments detailed in this work, *LiME (Linux Memory Extractor)* is used to capture memory and *The SleuthKit* to obtain file system information, both of them with *Linux*.



```
▾ evidencesStorage                              1 elemento
  ▾ david                                       1 elemento
    ▾ 10.0.2.10                                 18 elementos
        memDump-20220324T000729.dd.hash         138 bytes
        memDump-20220324T000729.dd              1,1 GB
        memDump-20220324T000901.dd.hash         138 bytes
        memDump-20220324T000901.dd              1,1 GB
```

Fig. 1. Folder structure for stored evidences (1 node; 1 controller).



```
100CB754 5F 70 68 79 5F 6E 61 6D 65 2C 20 28 52 45 43 2D 3E 69 64 65 _phy_name, (REC->ide
100CB768 6E 74 69 66 69 65 72 29 2C 20 52 45 43 2D 3E 6D 61 78 5F 66 ntifier), REC->max_f
100CB77C 72 61 6D 65 5F 72 65 74 72 69 65 73 00 00 00 00 00 00 00 00 rame_retries........
100CB790 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 22 25 73 2C ................"%s,
100CB7A4 20 77 70 61 6E 5F 64 65 76 28 25 75 29 2C 20 6D 61 78 20 63  wpan_dev(%u), max c
100CB7B8 73 6D 61 20 62 61 63 6B 6F 66 66 73 3A 20 25 64 22 2C 20 52 sma backoffs: %d", R
100CB7CC 45 43 2D 3E 77 70 61 6E 5F 70 68 79 5F 6E 61 6D 65 2C 20 28 EC->wpan_phy_name, (
100CB7E0 52 45 43 2D 3E 69 64 65 6E 74 69 66 69 65 72 29 2C 20 52 45 REC->identifier), RE
100CB7F4 43 2D 3E 6D 61 78 5F 63 73 6D 61 5F 62 61 63 6B 6F 66 66 73 C->max_csma_backoffs
100CB808 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....................
100CB81C 00 00 00 00 22 25 73 2C 20 77 70 61 6E 5F 64 65 76 28 25 75 ...."%s, wpan_dev(%u
100CB830 29 2C 20 6D 69 6E 20 62 65 3A 20 25 64 2C 20 6D 61 78 20 62 ), min be: %d, max b
100CB844 65 3A 20 25 64 22 2C 20 52 45 43 2D 3E 77 70 61 6E 5F 70 68 e: %d", REC->wpan_ph
100CB858 79 5F 6E 61 6D 65 2C 20 28 52 45 43 2D 3E 69 64 65 6E 74 69 y_name, (REC->identi
100CB86C 66 69 65 72 29 2C 20 52 45 43 2D 3E 6D 69 6E 5F 62 65 2C 20 fier), REC->min_be, 
100CB880 52 45 43 2D 3E 6D 61 78 5F 62 65 00 00 00 00 00 00 00 00 00 REC->max_be.........
```

Signed 8 bit:	-119	Signed 32 bit:	151373449	Hexadecimal:	89
Unsigned 8 bit:	137	Unsigned 32 bit:	151373449	Octal:	211
Signed 16 bit:	-14711	Signed 64 bit:	151373449	Binary:	10001001
Unsigned 16 bit:	50825	Unsigned 64 bit:	151373449	Stream Length:	8
Float 32 bit:	1,610264e-33	Float 64 bit:	-5,486124e+303		

[✓] Show little endian decoding
[ ] Show unsigned and float as hexadecimal

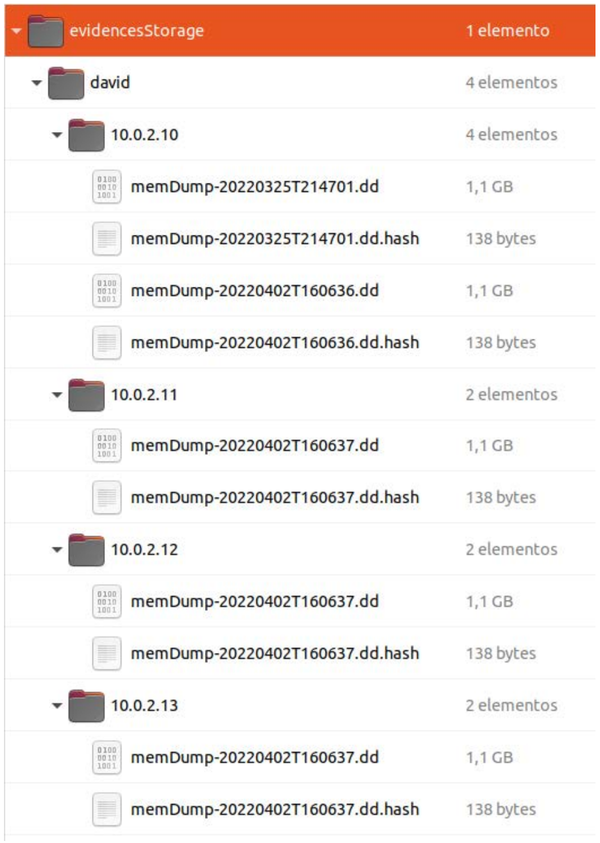
Offset: 0x0

Fig. 2. Acquired memory dump viewed in a hex editor.

After that, the acquisition of evidence and the labeling will be launched, identification and storage actions will be carried out in such a way that the custody chain of the said evidence can be maintained and its probating value is not lost. This approach seeks to maximize modularity and flexibility. Thus, any change in the requirement regarding the acquisition tools will not imply a global change in the proposal.

As for the concrete experiment taken in this work, an experiment for remote memory acquisition from a *Linux* node is first designed. This experiment is performed on a virtual network consisting of two machines, a node and a controller. This core case allows us to confirm the suitable use of *LiME* for memory acquisition by automating its execution using *Ansible*. This experiment also confirms that the SSH configuration on both the controller and the node is correct Similarly, it is confirmed that the format of the inventory is as expected.

As can be seen in Fig. 1, the ability to properly label and store the acquired evidence is verified, thus maintaining its probating value. The automation is achieved by making use of several Ansible modules: *apt*, *git*, *shell* and *fetch*.



evidencesStorage	1 elemento
david	4 elementos
10.0.2.10	4 elementos
memDump-20220325T214701.dd	1,1 GB
memDump-20220325T214701.dd.hash	138 bytes
memDump-20220402T160636.dd	1,1 GB
memDump-20220402T160636.dd.hash	138 bytes
10.0.2.11	2 elementos
memDump-20220402T160637.dd	1,1 GB
memDump-20220402T160637.dd.hash	138 bytes
10.0.2.12	2 elementos
memDump-20220402T160637.dd	1,1 GB
memDump-20220402T160637.dd.hash	138 bytes
10.0.2.13	2 elementos
memDump-20220402T160637.dd	1,1 GB
memDump-20220402T160637.dd.hash	138 bytes

Fig. 3. Folder structure for stored evidences for several nodes.

As observed in Fig. 2, the file with the memory dump is available, which, as can be analyzed with a forensic analysis application or opened with a hex editor for review its content. If the experiment is run for a second time some tasks are skipped. The omitted tasks are those related to the LiME installation. Once the node is ready and LiME is installed on that node, there is no need to re-install it again.

In addition to this, this experiment allow us to adapt the result to regulations, in such a way that a memory capture is obtained, labeled with the type of evidence, date and hour of acquisition, identification of the user who performs the acquisition and the machine from which it is performed, in addition to a hash value that guarantees the integrity of the evidence.

On the other hand, our next experiment deals with the remote acquisition from five *Linux* nodes. For this experiment, the set of nodes is increased, being made up of a controller machine and a group of five nodes, all of them with the Ubuntu operating system. It seeks to confirm the ability of the proposal to automate memory acquisition remotely in various nodes (see Fig. 3). Again, the ability to properly label and store the acquired evidence is also verified, thus maintaining its probating value.

Finally, the remote acquisition of file system information in *Linux* has been carried out. In this experiment, the size of the network is expanded again, with a controller machine and seven nodes, all of them with the Ubuntu operating system. The aim is to confirm the ability of the proposal to acquire other types of evidence, for which the *The Sleuthkit* suite will be used.

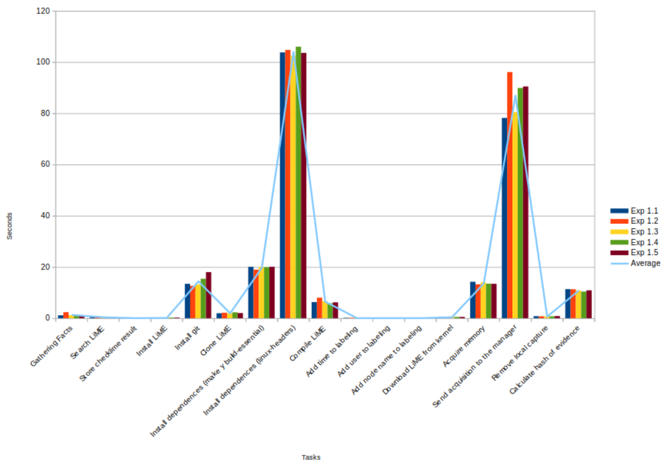


Fig. 4. Execution time (in seconds) for 5 nodes without a previous preparation.

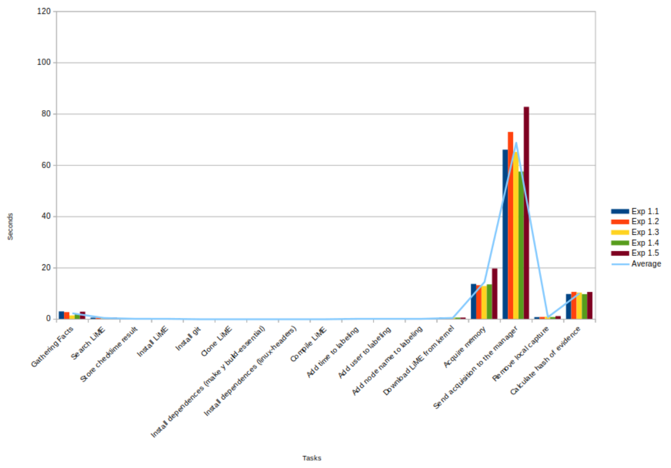


Fig. 5. Execution time (in seconds) for 5 nodes with a previous preparation.

IV. Results

All the experiments have been carried out on a network of virtual machines whose virtualization has been carried out using *VirtualBox 6.1*. The machine that hosts this virtual network is a Kubuntu 20.04 running on Intel© Core™ i7-8750H 2.20GHz processor with 12 cores, 32GB of DDR4 at 2666MHz RAM and an SSD disk.

First, the remote memory acquisition from a *Linux* node has been performed. The most remarkable finding of this experiment is the confirmation of the possibility of acquiring the evidence, in this case a complete memory dump, automatically and remotely through the use of *Ansible*.

TABLE I
EXECUTION TIME (IN SECONDS) FOR 5 NODES WITHOUT A PREVIOUS PREPARATION

Task	*Exp 01*	*Exp 02*	*Exp 03*	*Exp 04*	*Exp 05*	*Mean*
Gathering facts	1.13	2.35	1.2	1.31	1.15	**1.43**
Search LiME	0.48	0.46	0.58	0.51	0.52	**0.51**
Store checklime result	0.15	0.15	0.19	0.14	0.17	**0.16**
Install LiME	0.23	0.24	0.25	0.24	0.24	**0.24**
Install git	13.42	12.62	13.16	15.38	17.99	**14.51**
Clone LiME	1.92	2.13	1.91	2.29	1.99	**2.05**
Install dependences (make y build-essential)	20.05	18.93	20.04	19.91	20.06	**19.80**
Install dependences (linux-headers)	103.74	104.71	102.45	105.97	103.55	**104.08**
Compile LiME	6.29	8	6.61	5.8	6.19	**6.58**
Add time to labeling	0.15	0.16	0.16	0.17	0.17	**0.16**
Add user to labeling	0.16	0.16	0.16	0.17	0.16	**0.16**
Add node name to labeling	0.16	0.16	0.15	0.15	0.15	**0.15**
Download LiME from kernel	0.6	0.52	0.48	0.5	0.5	**0.52**
Acquire memory	14.21	13.21	14.02	13.47	13.42	**13.67**
Send acquisition to the manager	78.17	96.08	80.5	89.87	90.45	**87.01**
Remove local capture	0.77	0.74	0.73	0.73	0.82	**0.76**
Calculate hash of evidence	11.34	11.3	10.82	10.38	10.87	**10.94**

TABLE II
EXECUTION TIME (IN SECONDS) FOR 5 NODES WITH A PREVIOUS PREPARATION

Task	*Exp 01*	*Exp 02*	*Exp 03*	*Exp 04*	*Exp 05*	*Mean*
Gathering facts	2.9	2.63	1.33	1.81	2.79	**2.29**
Search LiME	0.46	0.45	0.54	0.47	0.49	**0.48**
Store checklime result	0.16	0.17	0.16	0.15	0.17	**0.16**
Install LiME	0.15	0.16	0.16	0.15	0.15	**0.15**
Install git	0	0	0	0	0	**0**
Clone LiME	0	0	0	0	0	**0**
Install dependencies (make y build-essential)	0	0	0	0	0	**0**
Install dependencies (linux-headers)	0	0	0	0	0	**0**
Compile LiME	0	0	0	0	0	**0**
Add time to labeling	0.14	0.16	0.14	0.16	0.15	**0.15**
Add user to labeling	0.15	0.16	0.16	0.15	0.15	**0.15**
Add node name to labeling	0.14	0.14	0.15	0.15	0.14	**0.14**
Download LiME from kernel	0.5	0.51	0.52	0.5	0.5	**0.51**
Acquire memory	13.61	13.12	12.8	13.43	19.63	**14.52**
Send acquisition to the manager	65.96	72.89	65.1	57.45	82.67	**68.81**
Remove local capture	0.68	0.74	0.85	0.7	1.06	**0.81**
Calculate hash of evidence	9.68	10.53	10.24	9.63	10.49	**10.11**

The difference in execution time between machines prepared for the acquisition and machines no prepared is shown for the experiment carried out for five nodes and a controller. As observed in Fig. 4 and Table I, the average acquisition time is 4 minutes and 22 seconds for the case of unprepared machines with previous installation.

This average time is considerably reduced in the case of prepared machines to one minute and 22 seconds, as shown in Table II and Fig. 5. This means a reduction of approximately 66%. For this reason, it is very relevant for companies to be aware about the topic of forensic analysis.

The concrete tasks to be performed for this experiment are:

1) Gathering facts.
2) Search LiME.
3) Store checklime result.
4) Install LiME.
5) Install git.
6) Clone LiME.
7) Install dependencies (make y build-essential).
8) Install dependencies (linux-headers).
9) Compile LiME.
10) Add time to labeling
11) Add user to labeling.
12) Add node name to labeling
13) Download LiME from kernel.
14) Acquire memory.
15) Send acquisition to the manager.
16) Remove local capture.
17) Calculate hash of evidence.

After that, the remote acquisition of file system information in *Linux* was performed. The contribution of this experiment is the confirmation of the capacity of the proposal to acquire various types of evidence. In this case, a text file is retrieved for each of the target nodes that includes the list of partitions for each node.

In this case, a set of the running tasks are different to the previous experiment, which are:

1) Gathering facts.

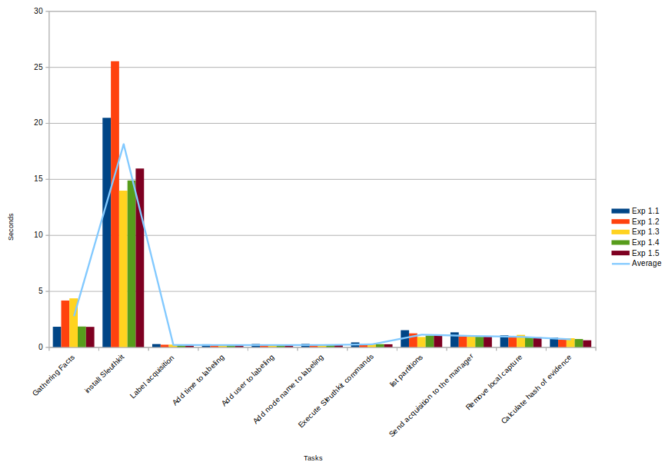


Fig. 6. Execution time (in seconds) for 7 nodes without a previous preparation.

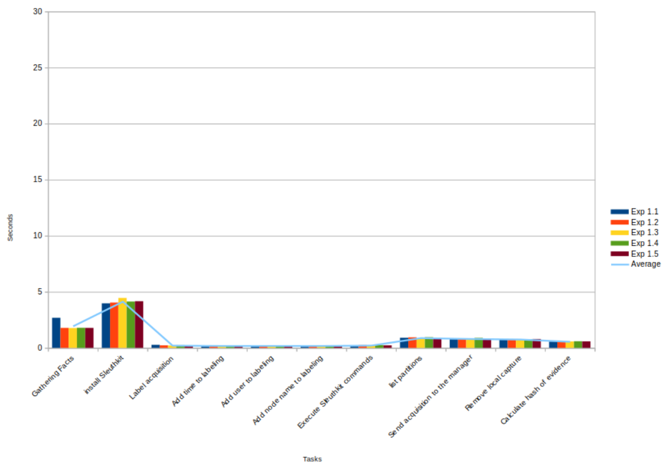


Fig. 7. Execution time (in seconds) for 7 nodes with a previous preparation.

2) Install sleuthkit.
3) Generate acquisition label.
4) Add time to labeling.
5) Add user to tagging.
6) Add node name to labeling.
7) Run sleuthkit commands.
8) List partitions.
9) Send evidence to controller.
10) Eliminate local evidence.
11) Calculate hash of evidence.

The average times for this type of acquisition in a 7-node network are 25 seconds and 10 seconds, depending on whether the machines are ready for the acquisition or not. These fact are shown in Figs. 6 and 7. We can observe a time reduction of a 60%. Tables III and IV show the detailed obtained values, respectively, without and with a previous preparation.

V. Discussion

In view of the experiments and tests carried out, it can be stated that the following statements have been confirmed:

- *The proposal is capable of performing memory acquisition remotely.* This objective has been covered thanks to the use of *shell* module, which gives *Ansible* virtually unlimited flexibility by allowing the use of any command that Shell can execute. *Ansible* allows the preparation of the node so that it has the necessary tools to carry out the acquisition, executes the acquisition itself and is responsible for sending the evidence to the controller machine.
- *The probating value of remote evidence is preserved.* Again, *Ansible* allows us to label and classify the evidence in such a way that both the evidence and the person who has acquired it can be uniquely identified. In addition, it allows the calculation of the *hash* value of the evidence at the time of its storage in order to guarantee its integrity. With this, the objective can be considered as covered.
- *The proposal is capable of acquiring other types of evidence remotely.* As a proof of concept, the acquisition of information from the file system of *Linux* nodes has been achieved. For this, the *Sleuthkit* suite has been used together with the capacity of the *Shell* module of *Ansible*.
- *The viability of the designed solution is established.* No major impediments have been found that could indicate the infeasibility of the proposed solution. Taking into account the tests carried out, it can be stated that by using *Ansible* is possible to automatically acquire evidence for forensic analysis.

VI. Conclusions and Future Works

Our proposal covers the needs raised for this work, by proposing a solution that automatically allows evidence acquisition for forensic analysis from a controller machine.This way, it eliminates, or at least reduces, the need to physically move an incident response team to all the locations where it is located a machine of the distributed network for the acquisition of evidence.

Ansible provides power, based on the wide catalog of available modules, which cover most of the most common needs, and flexibility, thanks to the *command* and *shell* modules. They allow us to go where the rest of the modules have not arrive yet, being able to execute commands of ad-hoc orders against the node machine. *Ansible's* learning curve is very smooth and fast.

As future work, we plan to evaluate deeply the defined experiments, as well as incorporate additional experiments with others operating systems, such as Windows. In this case, with Windows nodes, it will be interesting to check the connection capacity via WinRM and, additionally, to study if the use of this connection protocol is viable for the proposed use or if, on the contrary, other alternatives must be sought. It will also be interesting to see how the acquisition tools available for Windows perform.

TABLE III
EXECUTION TIME (IN SECONDS) FOR 7 NODES WITHOUT A PREVIOUS PREPARATION

Task	*Exp 01*	*Exp 02*	*Exp 03*	*Exp 04*	*Exp 05*	*Mean*
Gathering facts	1.81	4.15	4.34	1.83	1.80	**2.79**
Install sleuthkit	20.46	25.51	13.96	14.87	15.93	**18.15**
Generate acquisition label	0.26	0.20	0.21	0.20	0.20	**0.21**
Add time to labeling	0.24	0.19	0.20	0.19	0.19	**0.20**
Add user to tagging	0.27	0.18	0.19	0.20	0.18	**0.20**
Add node name to labeling	0.28	0.17	0.19	0.22	0.17	**0.21**
Run sleuthkit commands	0.41	0.25	0.23	0.25	0.24	**0.28**
List partitions	1.50	1.22	0.89	1.02	1.05	**1.14**
Send evidence to controller	1.31	0.96	1.00	0.91	0.91	**1.02**
Eliminate local evidence	1.04	0.85	1.08	0.86	0.87	**0.94**
Calculate hash of evidence	0.83	0.66	0.78	0.71	0.60	**0.72**

TABLE IV
EXECUTION TIME (IN SECONDS) FOR 7 NODES WITH A PREVIOUS PREPARATION

Task	*Exp 01*	*Exp 02*	*Exp 03*	*Exp 04*	*Exp 05*	*Mean*
Gathering facts	2.68	1.78	1.77	1.79	1.78	**1.96**
Install sleuthkit	3.97	4.04	4.45	4.13	4.16	**4.15**
Generate acquisition label	0.27	0.22	0.21	0.24	0.21	**0.23**
Add time to labeling	0.20	0.20	0.19	0.20	0.19	**0.20**
Add user to tagging	0.19	0.20	0.19	0.22	0.19	**0.20**
Add node name to labeling	0.18	0.20	0.20	0.21	0.19	**0.20**
Run sleuthkit commands	0.23	0.25	0.22	0.25	0.22	**0.23**
List partitions	0.90	0.94	0.84	0.95	0.88	**0.90**
Send evidence to controller	0.77	0.84	0.80	0.89	0.80	**0.82**
Eliminate local evidence	0.76	0.77	0.77	0.78	0.79	**0.77**
Calculate hash of evidence	0.56	0.56	0.57	0.59	0.58	**0.57**

REFERENCES

[1] M. W. Graves, *Digital Archaeology: The Art and Science of Digital Forensics*. Addison-Wesley, 2014.

[2] J. Sammons, *The Basics of Digital Forensics, 2nd edn*. Syngress, 2014.

[3] J. G. Heiser and W. G. Kruse, *Computer Forensics: Incident Response Essentials*. Addison-Wesley, 2002.

[4] K. Zatyko, "Commentary: Defining digital forensics," *Forensic Magazine*, pp. February/March(2007), 1–5, 2007.

[5] "UNE-EN ISO/IEC 71506:2013 - Information Technologies (IT). Methodology for the digital evidences forensic analysis," 2013.

[6] "UNE-EN ISO/IEC 27037:2016 - Information technology - Security techniques - Guidelines for identification, collection, acquisition and preservation of digital evidence," 2016.

[7] R. Adams *et al.*, "Iseek, a tool for high speed, concurrent, distributed forensic data acquisition," *The Proceedings of 15th Australian Digital Forensics Conference 5-6 December 2017*, pp. 19–25, December 2017.

[8] "Iseek the intelligent agent," 2021, https://www.xtremeforensics.com/iseekdiscovery-1, visited in October 2022.

[9] "Cortex xdr forensics," 2021, https://www.paloaltonetworks.com/resources/datasheets/cortex-xdr-forensics, visited in October 2022.

[10] "Ibm security qradar xdr," 2021, https://www.ibm.com/es-es/qradar, visited in October 2022.

[11] "Panda advanced endpoint security," 2021, https://www.watchguard.com/wgrd-resource-center/docs/adaptive-defense-360, visited in October 2022.

[12] "Velociraptor documentation," 2021, https://docs.velociraptor.app/docs/, visited in October 2022.