R.A.M.

Camila Fonseca
DETI
University of Aveiro
Aveiro, Portugal
cffonseca@ua.pt

Eduardo Monteiro
DETI
University of Aveiro
Aveiro, Portugal
eduardo.monteiro@ua.pt

Rodrigo Lima
DETI
University of Aveiro
Aveiro, Portugal
rodrigoflima@ua.pt

***Abstract*—The advancement of technology has led to a considerable growth in the number of cybercrime related cases, and this has raised a considerable challenge to be tackled. Extracting RAM is not a trivial task due to its volatile nature and mechanisms the Operative System has in place to prevent RAM manipulation-based attacks. Considering the constraints mentioned above, it was decided to use LiME [16], a Linux Kernel module which allows to map out privileged system data.**

The main objective of this project is to develop a tool that can easily retrieve RAM from a system that is running in a remote location, so that it can be later analysed by someone who's performing a forensics investigation on the target system. This tool aims to be a complement to other forensics tools, such as the ones used to create disk storage forensic images.

***Index Terms*—RAM, cybercrime, forensics, remote**

I. Introduction

The collection and analysis of RAM is a lively area of research in cybersecurity. The continuous advancement of technology also brings with it the growth of cybercrime, which can vary from downloading pirated movies and software to being targeted with fileless malware, where the malware exists only in RAM.

RAM analysis also offers a great vision over other malicious vectors, as it contains fragments of encrypted files, list of all running processes, network connections, and running modules. These contents are either difficult or impossible to extract from ROM (non-volatile memory). For these reasons, this project focused on developing a tool to allow volatile memory dumping on remote systems.

The R.A.M tool was developed with its primary focus on aiding forensics investigations as this tool can provide investigators with digital evidence from a system which they don't have physical access to, such as it being in a remote location. This is possible due to how the R.A.M tool was designed; its design allows for multiple clients (systems that are subject to a forensics analysis) to connect to a server, which will communicate with these clients and may then request their volatile memory contents.

II. State of the Art

A. Random Access Memory

Random Access Memory, henceforth referred to as 'RAM', is a type of (usually) volatile computer memory, which allows extremely fast write and read operations, with random access - reading/writing from memory takes always the same amount of time, no matter where the target location is - being possible. [1]

RAM is not meant to be used as any kind of long-term storage. Data is loaded into RAM as needed, operated over, and then freed and written over so other data can be loaded. For instance, whenever a script is executed, it is loaded into memory in its entirety, (compiled) code, variables, and required libraries. Once execution is over, that memory is freed, and may be overwritten at any time.

During normal computer operation, every bit of data accessed will at some point be in RAM [2] - the Operating System itself, all kinds of files, and most importantly, sensitive data such as passwords and private keys. As such, RAM becomes an essential source of data to be explored in a forensic environment [6], since everything that happens on a computer will leave traces on RAM, at least at a given point in time. Even more importantly, there exists data that will exist only on memory, such as data accessed through a browser. This can be weaponized in the form of Memory-Resident Viruses. [5]

However, and even though RAM does have an internal architecture, it is not structured as a computer's file system and considerably harder to analyze, requiring specialized forensic tools for the purpose.

B. Memory Dumping

To complicate matters further, due to its volatile nature, a copy of the data present in RAM cannot be obtained as easily as simply cloning a hard drive - and even that may not be as straightforward to do in a manner that enables the copy to be admissible evidence for an investigation. [7]

The process of taking a copy of the working memory of a program is called **Memory Dumping**. Depending on the needs and what's considered to be acceptable risk, the methods for this can vastly differ. It is important to keep in mind that dumping memory may leave a system unstable, and it is not a trivial process to access a modern's Operating System RAM - To avoid many RAM manipulation-based attacks (Stack Smashing, various arbitrary code execution techniques, among others [9]) it is highly protected [8], and the system will have multiple protection mechanisms, one of them, simply crashing. (In linux, a Kernel Panic.) Plus, since a running system's memory is constantly being altered, it may be hard to obtain a correct RAM snapshot. Due to these limitations, there are multiple types of current RAM aquisition tools.

Keeping in mind that these are not often useful due to needing to have hardware installed *a priori*, hardware-based solutions are a possibility by exploiting Direct Memory Access [10] [11], making it possible to transfer data at high speeds through specific hardware connections.

One such possibility is FireWire [12], also known as the IEEE 1394 Interface. It allows to retrieve the first 4GiB of the address space without needing any administrator privileges on the target system, and can be used even with a locked system. The 4GiB limitation makes this a valuable tool for taking RAM dumps from 32-bit systems, but for newer equipment a different approach is needed to obtain a complete RAM copy. [3]

Other methods for accessing memory directly, are via ExpressCard [13], PCI/PCI Express [14] and CardBus [15], which not only may have Direct Memory access themselves but also allow a FireWire expansion card to be installed.

Software solutions for RAM dumping require root access to the machine, as well as a live machine. These have a large downside to them, that improper usage may invalidate the forensic evidence they're trying to obtain, by using RAM and overwriting other, possibly important, data. Therefore, these need to be as lightweight as possible, to avoid as much as possible RAM being overwritten, and all RAM dumped should be stored off-site, remotely, or in a separate physical medium (USB pendrive, for instance).

Tools used for dumping RAM are, for instance, LiME [16] - Linux Memory Extractor, FTK Imager [17], dd [18] and memdump [19]. LiME, for instance, is a Linux Kernel module, running in kernel mode, which allows it to map out privileged system data.

C. Remote Memory Dumping

Some of the previously mentioned tools explicitly allow for transferring the acquired RAM over network, and workarounds like piping to a TCP socket enable it on CLI tools, if such feature isn't implemented natively. This is even highly encouraged to avoid overwriting the RAM being dumped.

To remotely trigger the dumping procedure, and then RAM retrieval, there are available commercial software that allow a remote server/node to request RAM extraction from an agent. Some of these are Belkasoft [4],F-Response [20] and Detego Global [21].

III. Implementation

To aid in the retrieval of Remote Access Memory from remote hosts, the tool R.A.M was designed. The tool aims to provide an intuitive way to retrieve memory from hosts where access is restricted or where the only access to the host is through the network.

To achieve the task of being able to retrieve random access memory remotely through the network, some considerations should be taken.

RAM, also known as Remote Access Memory, is volatile due to its origin and role in a computational system. Due to the retrieval and storage of this memory, some roadblocks arise. There are also some considerations to be made about the malleability of the tool.

A. Retrieval

In order for RAM to be requested from an Agent, that Agent must first be registered with the Server. For this, the server opens a listener that will receive a sent "Hello" message. This message contains the Agent's IP, and the port it will use to send the RAM on. By receiving this message, the Agent is added to the Server's agent list, from which it allows us to pick from when selecting to retrieve RAM.

After picking which Agent we wish to fetch the RAM from, the server sends a message to that agent and stays listening on a certain port. On the Agent's side, LiME is called and it does the RAM retrieving, sending through TCP to the Server's Listener, to avoid writing anything to memory on the agent and tampering with the RAM being retrieved.

B. Storage

The received RAM is written to disk on the Server, for later analysis. It is never written to disk or memory on the Agent's side, as previously mentioned.

C. Malleability

The development of this tool also aims for it to be malleable to its users. This malleability will enable users to adapt the tool to their environment. It will also enable them to tweak it so that other goals can be achieved with the tool besides retrieving RAM.

This malleability is achieved by carefully choosing the language in which the tool is designed to be built. Furthermore, we chose an easily extensible approach that would allow users to extend the functionalities of the tool, thus making it more adaptable to users.

For the development of the tool, Go was chosen as the programming language. Developed by Google, this language simplifies network communications. Additionally, it supports cross-compilation of the code for a variety of architectures and has built-in concurrency.

IV. Architecture

To meet the strict requirements needed to achieve the goals of the tool, an appropriate architecture should be used.

The tool aims to support multiple clients. The aim is to be able to request RAM from multiple clients through a single command and control server. To meet this requirement the server has a built-in command to listen for incoming connections to be added. As a result, the configuration will be simpler, the only requirement being that the server must be in the add-client mode and that the client must be run. In the future, if a client operator wants to retrieve RAM from a specific connected client, he just needs to query the clients' list. He can then request the RAM with the client ID.

The handling of adding new clients and RAM requests is done through the GO standard libraries. This enables better maintainability and scalability if other commands were to be

added to the functionalities of the tool. However, the handling of data transmission, handling of the transfer of the requested RAM, is done through the Netcat tool. This choice was made due to the requirements for the transfer and retrieval of RAM. If it were to be handled within the client's host besides relying on a socket the data would corrupt the very data being transferred.

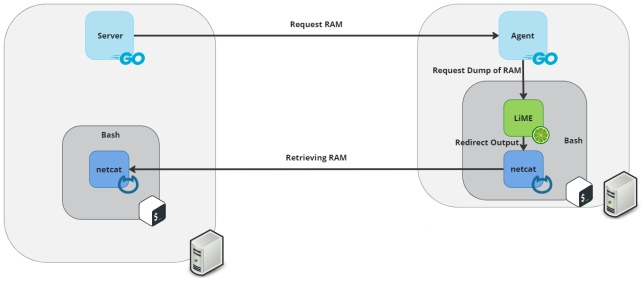

Fig. 1. Project Architecture

With the architecture adopted, users can easily develop custom modules or plugins. Furthermore, the client is decoupled, allowing it to be written in different languages or programs, as shown in figure 1.

V. Findings

During the development of R.A.M, we noticed that memdump [19], the tool that was initially chosen to retrieve volatile memory from target systems, was not as stable as expected as it often crashed the target system midway the RAM retrieval process. Taking this into account, a decision was made to choose another tool to dump the client's system RAM content. This new tool is called LiME [16], which is a Linux Kernel module which allows to map out privileged system data, and it was observed that LiME [16] dumps the RAM contents in full, and in a more stable way for the operating system it is running on, not causing any crashes.

However, given a system with large amounts of RAM, the dumped memory most likely won't be atomic. Due to the time LiME takes to create and send a copy of the RAM's contents, it is too slow when compared to the speed with which the contents change in memory, so this could lead to small sections of memory to change between the start and end of the dumping process. This is a serious issue, as it makes it nearly impossible to obtain a proper snapshot of the memory state in any modern system.

VI. Discussion

The R.A.M. project if used in forensic investigations could have several positive impacts. One potential benefit of this tool is that it could allow investigators to obtain valuable digital evidence from a computer without physically accessing the device. This could be particularly useful in cases where the device is located in a remote location, or if it is not possible or practical to physically access the device.

Another potential benefit of a tool that extracts RAM remotely is that it could provide investigators with the ability to obtain evidence from a live system, rather than from a forensic image of the hard drive. This could be useful in cases where the investigator needs to obtain evidence quickly, such as in situations where a suspect may be trying to cover their tracks or destroy evidence. In these cases, extracting RAM remotely could provide investigators with the ability to obtain valuable evidence before it is lost or destroyed.

Additionally, extracting RAM remotely could potentially provide investigators with access to volatile data, such as data that is stored in memory and is not written to the hard drive. This could include data that is generated by running programs, data that is temporarily stored in memory while it is being processed, and data that is associated with network connections and other active processes. This type of data can be difficult or impossible to obtain through traditional forensic methods, but extracting RAM remotely could provide investigators with a new way to access this valuable evidence.

Overall, a tool that enables users to extract RAM remotely could have several positive impacts on forensic investigations. It could provide investigators with the ability to obtain valuable digital evidence from a live system, access volatile data that is not typically available through traditional forensic methods, and obtain evidence from remote or inaccessible devices.

VII. Conclusion

The R.A.M. tool is an innovative and useful tool that can be used by forensic analysts to extract RAM remotely. This capability is a significant improvement over traditional forensic methods, which typically require physical access to the device in order to obtain digital evidence.

One of the key benefits of the R.A.M. tool is that it allows forensic analysts to obtain evidence from a live system. This means that the investigator can obtain evidence quickly, without having to wait for the device to be physically accessed and imaged. This is particularly useful in cases where the suspect may be attempting to cover their tracks or destroy evidence, as the investigator can obtain the evidence before it is lost or destroyed.

Overall, the R.A.M. tool is a valuable addition to the forensic analyst's toolkit. It provides a new and efficient way to obtain digital evidence from live systems, and allows investigators to access volatile data that is not typically available through traditional forensic methods.

References

[1] P. Haugen, I. Myers,B. Sadler, and J. Whidden, "A Basic Overview of Commonly Encountered types of Random Access Memory (RAM),"pp.2–3.

[2] K. Hausknecht, D. Foit and J. Burić, "RAM data significance in digital forensics," 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2015, pp. 1372-1375, doi: 10.1109/MIPRO.2015.7160488, section III

[3] Witherden, Freddie D.. "Memory Forensics over the IEEE 1394 Interface." 2010

[4] A. R. Javed, W. Ahmed, M. Alazab, Z. Jalil, K. Kifayat and T. R. Gadekallu, "A Comprehensive Survey on Computer Forensics: State-of-the-Art, Tools, Techniques, Challenges, and Future Directions," in IEEE Access, vol. 10, pp. 11065-11089, 2022, doi: 10.1109/ACCESS.2022.3142508.

[5] B. N. Sanjay, D. C. Rakshith, R. B. Akash and D. V. V. Hegde, "An Approach to Detect Fileless Malware and Defend its Evasive mechanisms," 2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS), 2018, pp. 234-239, doi: 10.1109/CSITSS.2018.8768769., section I

[6] K. Hausknecht, D. Foit and J. Burić, "RAM data significance in digital forensics," 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2015, pp. 1372-1375, doi: 10.1109/MIPRO.2015.7160488. , section IV

[7] Y. -T. Chang, M. -J. Chung, C. -F. Lee, C. -T. Huang and S. -J. Wang, "Memory Forensics for Key Evidence Investigations in Case Illustrations," 2013 Eighth Asia Joint Conference on Information Security, 2013, pp. 96-101, doi: 10.1109/ASIAJCIS.2013.22., Section III

[8] Y. Kim, J. Lee and H. Kim, "Hardware-based Always-On Heap Memory Safety," 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 2020, pp. 1153-1166, doi: 10.1109/MICRO50266.2020.00095., Section I

[9] Z. Wang, X. Ding, C. Pang, J. Guo, J. Zhu and B. Mao, "To Detect Stack Buffer Overflow with Polymorphic Canaries," 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2018, pp. 243-254, doi: 10.1109/DSN.2018.00035.

[10] W. Ahmed and B. Aslam, "A comparison of windows physical memory acquisition tools," MILCOM 2015 - 2015 IEEE Military Communications Conference, 2015, pp. 1292-1297, doi: 10.1109/MILCOM.2015.7357623., Section II

[11] Cai, Liming, Jing Sha and Wei Qian. "Study on Forensic Analysis of Physical Memory." (2013)., Section II.A1

[12] "1394 Trade Association", https://web.archive.org/web/20170209003417/http://1394ta.org/, (accessed Nov. 26, 2022).

[13] "ExpressCard — USB-IF", https://www.usb.org/expresscard, (accessed Nov. 26, 2022).

[14] "PCI-SIG", https://pcisig.com/,(accessed Nov. 26, 2022).

[15] "Understanding PC Card, PCMCIA, Cardbus, 16-bit, 32-bi - hpcfactor.com", https://www.hpcfactor.com/support/cesd/200194/ understanding_pc_card_pcmcia_cardbus_16-bit_32-bit/, (accessed Nov. 26, 2022).

[16] 504ensicsLabs, "LiME Linux Memory Extractor", https://github.com/504ensicsLabs/LiME,(accessed Nov. 27, 2022).

[17] Exterro, "FTK Imager - Exterro",https://www.exterro.com/ftk-imager,(accessed Nov. 27, 2022).

[18] "dd(1) - Linux manual page", https://man7.org/linux/man-pages/man1/dd.1.html,(accessed Nov. 27, 2022).

[19] doudou, "Memdump", https://github.com/doudou/memdump, (accessed Nov. 27, 2022).

[20] "F-Response - Extend Your Arsenal", https://www.f-response.com/ (accessed Nov. 28, 2022)

[21] "Fast Data Extraction Software & Solution - Detego Global", https://detegoglobal.com/remote-acquisition/, (accessed Nov. 27, 2022)