SAFARI: a Scalable Air-gapped Framework for Automated Ransomware Investigation

Tommaso Compagnucci[1], Franco Callegati[1], Saverio Giallorenzo[1,2], Andrea Melis[1], Simone Melloni[3], and Alessandro Vannini[1]

[1] Alma Mater Studiorum - Università di Bologna, Bologna, Italy
[2]Olas Team, INRIA, Sophia Antipolis, France
[3]ARPAE Emilia-Romagna, Italy

Abstract. Ransomware poses a significant threat to individuals and organisations, compelling tools to investigate its behaviour and the effectiveness of mitigations. To answer this need, we present SAFARI, an open-source framework designed for safe and efficient ransomware analysis. SAFARI's design emphasises scalability, air-gapped security, and automation, democratising access to safe ransomware investigation tools and fostering collaborative efforts. SAFARI leverages virtualisation, Infrastructure-as-Code, and OS-agnostic task automation to create isolated environments for controlled ransomware execution and analysis. The framework enables researchers to profile ransomware behaviour and evaluate mitigation strategies through automated, reproducible experiments. We demonstrate SAFARI's capabilities by building a proof-of-concept implementation and using it to run two case studies. The first analyses five renowned ransomware strains (including WannaCry and LockBit) to identify their encryption patterns and file targeting strategies. The second evaluates Ranflood, a contrast tool which we use against three dangerous strains. Our results provide insights into ransomware behaviour and the effectiveness of countermeasures, showcasing SAFARI's potential to advance ransomware research and defence development.

Keywords: Analysis Automation, Ransomware, Infrastructure as Code

1 Introduction

Ransomware is malware that seizes users' and organizations' data, demanding payment to restore access to the rightful owners [16]. Ransomware [28,25] is one of the most pressing cybersecurity threats facing organisations and individuals worldwide. In its most general definition, ransomware extorts victims through their data. Economically, ransomware impose sizable costs to society [10,23]. Indeed, ransomware forms a sophisticated, multi-billion dollar industry that continues to adapt and overcome defensive measures [22].

Despite increased awareness and investment in cybersecurity, ransomware attacks continue to proliferate, causing significant financial losses, operational disruptions, and potential harm to human life and safety [5].

The study of the behaviour of ransomware in different contexts and conditions provides fundamental knowledge for its contrast. The scientific and technical literature abounds with solutions to counteract ransomware [1,21,2,25,20,8]. Studying the behaviour of these solutions against real-world malware is important to both evaluate their effectiveness and efficiency and help their evolution.

We respond to such needs with a platform for the *safe* and *efficient* investigation of ransomware and its mitigations. Safety and efficiency of investigation are the main challenges we address in the design of our proposal. To perform an evaluation of ransomware attacks and of contrast strategies, one needs numerous tests in which real ransomware attempts to encrypt users' data. We provide a framework to run these tests *safely*, ensuring that the ransomware infection is fully contained, and *efficiently*, automating the management of the experiment parametrisation, to generate test batteries in a consistent and reproducible manner, and to collect and analyse indicators useful to understand the behaviour and estimate the effectiveness of ransomware and defensive configurations.

SAFARI We call our proposal Scalable, Air-gapped Framework for Automated Ransomware Investigation (SAFARI), available as an open-source project at `https://github.com/Flooding-against-Ransomware/SAFARI`.

One of the foundational principles behind SAFARI is the *democratisation of security research*. Indeed, while SAFARI's architecture is general and can encompass the usage of Cloud virtual machines (VM), its design—and the prototype we build to showcase it—targets on-premises deployments where investigators can use local resources to assemble the system. In this way, small research groups, as well as student collectives who can access home-lab-sized hardware can configure their own deployment and run experiments.

Given the dangerous and highly infective nature of ransomware, SAFARI run experiments in *air-gapped* environments. Thanks to the virtualisation of the testing environment, SAFARI isolates malware processes within VMs from those of the hosting environment and the other VMs (to avoid interference).

SAFARI integrates complementary technologies for the definition of experiments and the collection and analysis of attack data. Our proof-of-concept, presented in Section 2, illustrates the architecture of SAFARI both in terms of concepts and of software (off-the-shelf and purpose-made by us) that SAFARI integrates to analyse the behaviour of ransomware and their contrast solutions.

While SAFARI's architecture can accommodate the study of malware in general (combined with countermeasures), we focus our proposal on *ransomware* of the crypto kind, since it targets the data of the user, which we use as the metric to measure the effectiveness of both malware and contrast tools. In Section 3, we showcase SAFARI through two case studies. The first investigates the behaviour and efficiency of real-world, infamous ransomware such as WannaCry, LockBit, and Phobos. The second analyses the effectiveness and efficiency of a recent tool, Ranflood [3,4], which contrasts ransomware attacks through data flooding — an innovative technique that mixes dynamic honeypots and moving-target defence to contrast ransomware attacks.

Notably, while we provide the experimental data in Section 3 as evidence of the potential of SAFARI, the analysis itself constitutes a contribution that adds new knowledge about the analysed malware's behaviour. We position our contribution within the literature in Section 4 and discuss final remarks and future development in Section 5.

2 Implementing a SAFARI Prototype

In this section, we present our prototypical implementation of SAFARI. We outline its architecture and detail the concepts, technologies, and tools employed for and their respective roles in the implementation.

The fundamental concepts behind SAFARI's efficiency are: *a*) *Infrastructure as Code* (IaC), which is a paradigm where an orchestrator manages the provisioning of infrastructure components, like computing, network, and storage devices, using code rather than manual configuration, *b*) *OS-agnostic Task Automation* (OTA), which offers a uniform interface that allows users to execute processes independently of the underlying operating system, and *c*) *Investigation Tools*, which are the software that provide visibility/metrics on malware's behaviour.

Our prototype reifies IaC with Terraform[1] and OTA with Ansible[2]. Since our application context is crypto-ransomware, we concretise IT with ad-hoc tools that generates a report of the files and checksums of a target VM and compare the reports from before and after running tests to collect the experiments' data.

Before delving into our prototype's architecture, we briefly describe which hypervisor technology we choose. Indeed, while SAFARI's design abstracts away from a specific virtualisation technology, one needs to fix it when considering a specific deployment. We choose Proxmox[3], which is a widely-used and renowned open-source type-2 hypervisor (hosted in Linux/Debian) that supports enterprise-level virtualisation and includes an integrated web-based interface that complements the API-based one for the management of virtual machines, software-defined storage, and networking.

2.1 Software Architecture

We illustrate the architecture and experiments behaviour of our prototype in Figure 1. The main components of the architecture are the prototype software components (labelled "SAFARI" in the figure, for brevity), test VMs, analysis VMs, and remote persistence. We use borders, in the figure, to represent ephemerality with a dashed line and persistence with a solid one. The VMs that run the experiments (test VM) and the related analysis (analysis VM) correspond to a test run and are ephemeral and discarded within the related test session. The prototype's software and persistence components respectively orchestrate the experiments and store their results (encompassing all the test sessions).

[1] `https://www.terraform.io/`
[2] `https://www.redhat.com/en/ansible-collaborative`
[3] `https://www.proxmox.com/`

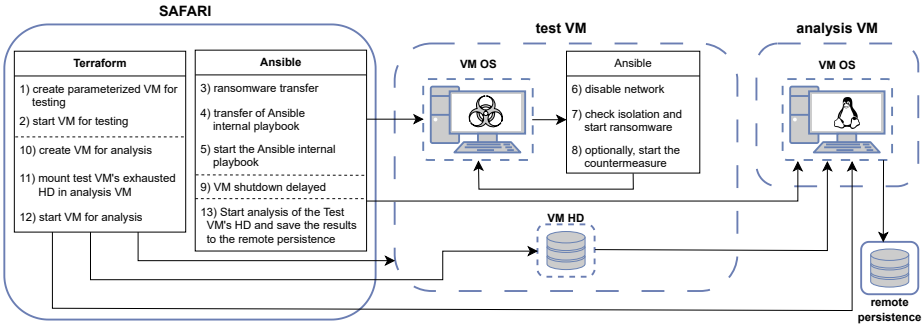

Fig. 1. SAFARI's prototype architecture and functionalities.

In Figure 1, from the left, we find the core components of the SAFARI prototype, which orchestrate the operations required to carry out the tests. Terraform and Ansible scripts mainly conduct these operations. The numbered operations shown in the schema correspond to a test session, which one can run multiple times and in parallel to gather statistical experimental data.

We describe the workflow of the prototype following the sequence in Figure 1.

At the start of an experiment, the system interacts with the hypervisor to create a VM (1) and to start it (2). Then, control passes to OTA, which transfers the malware to the VM (3), along with a list of tasks that the VM must execute (4). This step is peculiar and particularly important for the *air-gapping* principle mentioned in Section 1. We require the VMs to run an internal OTA engine. In this way, we avoid the network-based orchestration of the ransomware and rather inject a part of the OTA tasks within the VM. We adopt a multi-tiered approach to enforce isolation, which leads us to divide the logic of OTA orchestration into two parts, one external and one internal. The internal one disables the network connection (6) and makes sure that the VM is isolated before starting the ransomware (7). The in-VM OTA component can also start possible countermeasures present in the VM (8), according to the experiment's design.

All experiments have a set timeout, which triggers (9) the shutdown of the test VM. Then, control passes back to IaC for the creation of the analysis VM (10), which mounts the disk of the test VM for analysis and the remote persistence disk for storing the analysis results (11, 12). OTA orchestrates this step (13), which uses the IT tools to analyse the files and generate the results.

Since SAFARI simplifies managing VMs running different operating systems, our recommendation is, e.g., to use Linux VMs for the analysis of Windows-specific ransomware, to avoid possible accidental activations of the ransomware that would threaten the reliability of the analysis.

2.2 Filechecker and Profiler as IT

We close this section with details of the other software contribution presented in this paper for the implementation of IT tools in SAFARI to study ransomware.

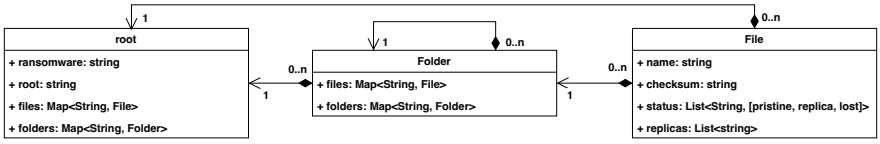

Fig. 2. Schema of the Hierarchical Profile.

To implement the IT part of our SAFARI prototype, we develop a set of open-source tools, available at `https://github.com/Flooding-against-Ransomware/profiling`. We use these tools to compare the state of VMs before and after a crypto-ransomware attack. Specifically, we use a Filechecker tool to create VM file *reports*. The software generates a JSON-formatted list of the *path*s of all the files (descending within folders) contained within a given *root* location, associated with their *checksum* signature (MD5). We expect investigators to launch the Filechecker manually (one can automate this task with tools like Vagrant[4].) when they assemble a test VM template, so they generate a report of the pristine test VM for later use in the analysis VM. Once obtained the report of the test VM, we run a second tool, called Profiler. The Profiler, given two report files, a reference one and a post-attack one, generates a JSON *profile* file of the difference between the two input files. Specifically, the profile indicates which files are *pristine*, which have been *lost* due to encryption, and possible *replicas* that one can use to restore the original files of the user (i.e., files with a different location but the same content of files of the reference report). Interestingly, while having the pristine-lost ratio would already give us a reading of the effectiveness/efficiency of ransomware and contrast tools, measuring the replicas sheds further light on the modality of execution of ransomware, since many of these create copies of the files to encrypt before deleting them [14]. In addition to a file-to-file comparison, the Profiler generates a *hierarchical* view of the profile, following file locations, i.e., starting from the root, we find a list *files* contained therein, with their related status (pristine/lost/replica) and a list of *folders*, each containing a list of files and subfolders. For reference, we report in Figure 2 the structure of hierarchical profiles—which also contain contextual information of the experiment/analysis, like the name of the ransomware and the root location (according to the reports). The Profiler can process a hierarchical profile to generate a *summary* profile that indicates measures such as the total numbers of pristine/lost/replica files found in hierarchical order—so that the root reports the overall numbers, then broken down by the folders it contains—and also aggregated by extension (useful to investigate e.g., which file formats a given ransomware mainly targets). Finally, the Profiler can *aggregate* multiple summary profiles of the same kind of experiment to provide statistical data about it. The result of is a JSON profile with the same structure of the summary one but with additional elements that report statistical measures on the files such as

[4] `https://github.com/hashicorp/vagrant`

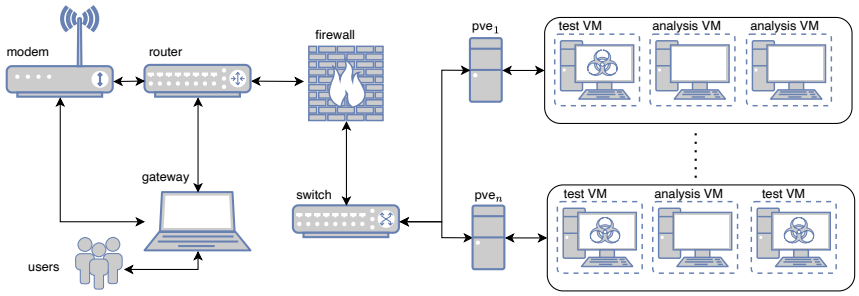

Fig. 3. SAFARI's Prototype Deployment Infrastructure.

averages and standard deviations. By integrating the Filechecker and Profiler in our SAFARI prototype, we obtain the automatic generation of statistics about the execution of ransomware and their countermeasures.

3 Case Studies

To illustrate the usage of SAFARI, we use our prototype to conduct two case studies. The first regards the analysis of the behaviour of a set of known ransomware, to profile their activity. The second showcases the integration within tests of a ransomware countermeasure, Ranflood, to benchmark its effectiveness against attacks. Before describing the case studies and their results, we report on the deployment used to run an instance of our SAFARI prototype.

3.1 SAFARI's Prototype Deployment Infrastructure

The case studies helps us illustrate an on-premises deployment of our prototype. We build the infrastructure shown in Figure 3 as a companion contribution to SAFARI's definition, designed to let users remotely run safe experiments.

Following Figure 3, users can authenticate and execute the prototype's functionalities through a gateway connected to the Internet. From the gateway, users can interact with the nodes, named $\text{pve}_1, \ldots, \text{pve}_n$ (where pve stands for Proxmox virtualisation environment), that make up the cluster and host the test and analysis VMs. All these nodes and VMs are behind a firewall that regulates access to the Internet through the router and via the modem.

We detail the main features of the components found in Figure 3. The *gateway* runs Debian v.12 (although unlikely, we use a Linux machine to further stave off possible infections from Windows ransomware) and represents the only network access point that experimenters use to connect to the nodes. Users connect to the gateway via the ZeroTier VPN and use sshuttle[5] to access directly the nodes, i.e., to run commands as if executed on machines in their local network.

[5] Resp. `https://www.zerotier.com/` and `https://github.com/sshuttle/sshuttle`.

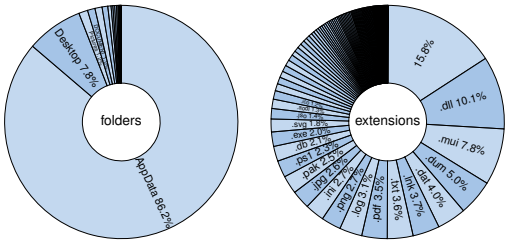

Fig. 4. Visualisation of the user's file profiles.

The *router*, mounting OpenWrt [24], can give Internet connection to the nodes in the cluster, but by default its *firewall* prevents VMs from accessing the Internet to avoid the possible propagation of ransomware.

The hardware of the pve cluster represents typical office/desktop personal computers, and it encompasses eight machines. The nodes run ProxMox 7.0-8 and include two VM templates, Linux Ubuntu 24.04 for the analysis VM and a Windows 10 (x64) 1809 template for the test VM. While each node can run multiple VMs, to have good performance, we run at most two VMs per node.

3.2 Case Study: Ransomware Profiling

To conduct the testing of our prototype, we start by defining a testing protocol, comprising various types of ransomware. We base the selection of these ransomware samples on an analysis of the history of ransomware and its evolution over time [26]. We source the samples used for the tests from repositories previously used and recognised in other academic contexts [18], such as VirusTotal.

In the first case study, we demonstrate our SAFARI prototype's capability to analyse the encryption patterns of five well-known Windows ransomware: Ryuk, Vipasana, WannaCry, LockBit, and Phobos. To simulate a real-world scenario, we configured the test VM with a typical user profile, modelled after Hansley [9], including the primary directories of the Windows 10 file system. The profile features 2 GB of user data, aligned with file type recommendations from Scaife et al. [30] and Kaspersky [13], along with file names commonly targeted by ransomware [15]. Additionally, it incorporates user-interactive programs such as a web browser and an office suite [29]. The profile includes 13 "user" folders, such as "Documents", "Desktop", "Music", and "Pictures" with the breakdown reported in Figure 4, using the prototype's Filechecker and Profiler tools.

For each ransomware strain, we run 32 experiments with a timeout of 10 minutes for each experiment. We obtain at least 16 valid experiments for each ransomware — 22 for LockBit, 32 for Ryuk and Vipasana, and 16 for Phobos and Wannacy, where we discard the ones where the ransomware did not start.

We visualise the results in Figure 5, reporting in the rows the data of a given ransomware and divide these (from left to right) respectively between the percentage summary of pristine/lost/replica files and the percentage breakdown

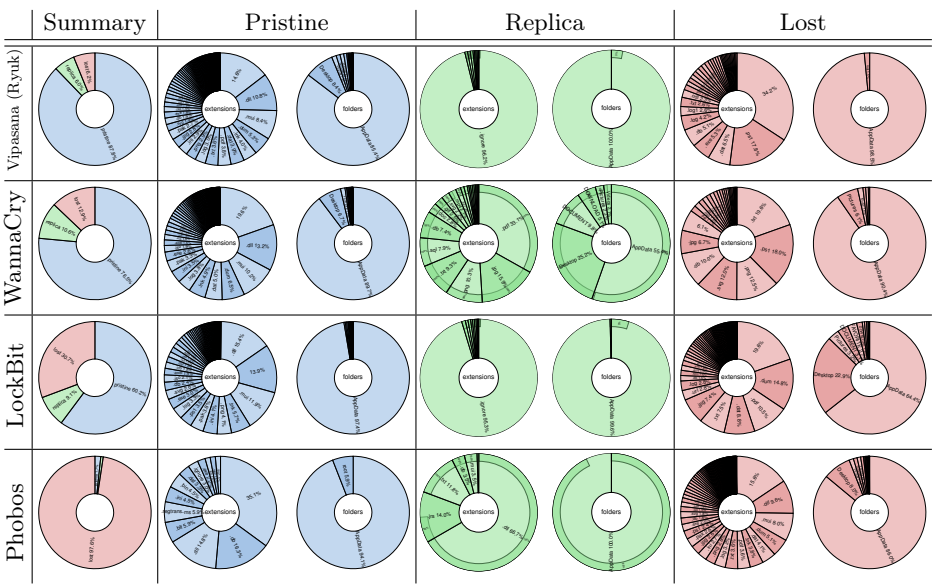

Fig. 5. Attack profiles of Vipasana/Ryuk, WannaCry, LockBit, and Phobos.

of pristine, replica, and lost files by extensions and folders. As Ryuk and Vipasana exhibit similar behavior, only Vipasana's results are shown for brevity. Replica file plots include smaller outer segments indicating the percentage of distinct replicas within a partition, e.g., the 3% in Ryuk/Vipasana folder replicas denotes that only 3% are unique, with the remainder having multiple copies.

Before analysing the results, we observe that all ransomware generate replicas, leading to two key insights. First, the samples employ a copy-based strategy, involving a copy phase followed by encryption — the replicas found are temporary files left unencrypted due to the VM shutdown. Second, the presence of replicas is a performance indicator, suggesting that the 10-minute delay was insufficient for the ransomware to encrypt all target files.

We comment on the results in Figure 5 from top to bottom.

Ryuk and Hiragana exhibit similar behaviour, primarily targeting the "AppData" folder (containing application files restorable through reinstallation). They create ".ignore" files in "AppData", often making multiple copies (3% are distinct). This strategy likely aims to prevent recovery by simple renaming.

WannaCry also focuses on "AppData" but spreads its attack across other directories like "Pictures", "Music", "Desktop", and "Documents". It creates unique copies before encryption and targets common file types such as ".png", ".jpg", ".svg", ".pdf", ".txt", and ".doc" while skipping files without extensions.

LockBit outperforms WannaCry, encrypting 30% of files compared to WannaCry's 12%. It targets user directories like "Desktop" (23%), "Pictures", and "Documents" while still focusing on "AppData". Like Ryuk/Vipasana, LockBit generates ".ignore" copies but also appears to use a three-stage strategy involving

direct copying, renaming to ".ignore", and encryption. Phobos seems the most dangerous one, reaching a staggering 98% of lost files. Given the almost-complete coverage of the attack area, looking at the lost and pristine charts gives little insight on its behaviour, although we might infer that its encryption routine might skip specific locations, found within the user profile and "AppData" folders that mainly contain executable and configuration files, e.g., preserved by the ransomware to allow minimal system functioning to show the ransom message.

Summarising the results, we notice that the severity of the different types of ransomware is not directly related to the mere percentage of lost files (which, of course, is an important measure, per se). For example, Phobos has the highest percentage of lost files while LockBit seems less dangerous since it encrypt only ca. 30% of the user's files. However, the breakdown of the encrypted files tells a more nuanced story. Phobos is much more "blunt" in its behaviour, since it indiscriminately encrypts most of the user files, irrespective of the folder and the extension. On the contrary, LockBit, is much more efficient, since it focuses its effort only on the files that mainly matter to the user, i.e., the files found under "Desktop", whose sequester compel the user to pay the ransom.

3.3 Ransomware Mitigation Profiling

The second case study provides an example of how one can use SAFARI to evaluate mitigation tools and techniques in a realistic yet safe scenario.

Concretely, we profile a recent ransomware tool, called Ranflood [4], exponent of a family of solutions, called Data Flooding against Ransomware [3]. Essentially, Ranflood contrasts ransomware attacks by confounding the files of the user with a "flood" of decoy ones, stymieing the attack both file- and resource-wise (by contending IO access with the ransomware).

Ranflood provides two families of flooding strategies: *random* and *copy-based*. The *random* family, implemented by the *Random* strategy, generates files of varying sizes and formats (mimicking those targeted by ransomware) with random content, requiring only a disk location to flood.

The *copy-based* family, implemented by the *On-the-fly* and *Shadow* strategies, require a target location and a preliminary setup under normal conditions: *On-the-fly* collects checksums of pristine files to avoid copying corrupted ones, while *Shadow* creates backups of user files to use as the source for flood copies.

Since Vipasana/Ryuk do not attack the files of the user (cf. Figure 5), we concentrate on WannaCry, LockBit, and Phobos. We report, in Figure 6, the results of the experiments after running 12 attack-and-contrast instances on the template VM used in the first case study. For each selected ransomware, we run the different Ranflood strategies with a 30-second delay since starting the ransomware, launching 13 flooding instances in parallel on distinct folders of the user, including "Documents", "Desktop", "Music", and "Pictures", purposefully avoiding flooding the "AppData" folder which contains application files rather than the user's ones and stopping the VM after 10 minutes.

For brevity, we show, in Figure 6, only the best-performing strategy, i.e., the one that minimises the percentage of lost files, which is Shadow. The interested

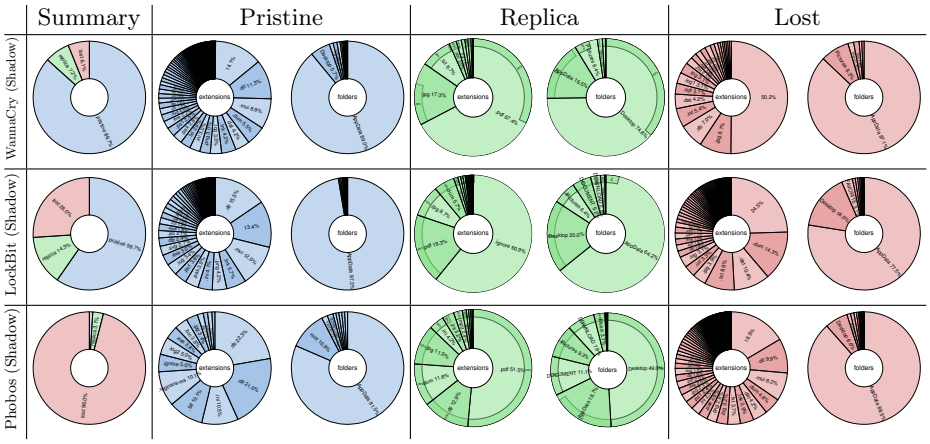

Fig. 6. Ranflood's contrast profiles against WannaCry, LockBit, and Phobos.

reader can find all data and plots at `https://zenodo.org/records/13891513`. The data in the plots is the average of 12 runs, amounting to an average standard deviation of 20.66% (across pristine, lost, and replica files).

Considering the specific ransomware, we find that Ranflood reduces the number of lost files in all cases, albeit with different impacts—WannaCry -52%, LockBit -15%, and Phobos -1.6%. As expected, since the Shadow strategy is copy-based, we find an increased number of replicas in most cases—LockBit +46.1% and Phobos +442.8%—except for WannaCry, where we observe a decrease of 32%. While the latter result deserves further investigation, we conjecture it might derive from the high number of replicas used by WannaCry during attacks (the largest among the tested ransomware, cf. Figure 5), which Ranflood hindered via IO access contention. The breakdown of the WannaCry replicas in Figure 6 sheds some light on the phenomenon. While, in Figure 5, we find most (55.4%) replicas under the "AppData" folder and fewer (25.2%) in "Desktop", the scenario with Ranflood reverses the ratios, with most replicas under "Desktop" (74.8%) and fewer under "AppData", which we attribute to the flooding performed by Ranflood that hinders the attacks of the ransomware.

The replica plots of LockBit and Phobos greatly differ from the respective ones in Figure 5, hinting at the "fight" put up by Ranflood against the ransomware on folders like "Desktop" and "Pictures".

4 Related work

Since SAFARI simulates real-world scenarios of hosts in an emulated air-gapped network, we can see it as an implementation of a Digital Twin (DT); a transformative technology that predicts system failures and identifies anomalies.

SAFARI fits into such a technological paradigm, particularly Cyber Ranges. A Cyber Range [27], when viewed as a DT application, is a virtualised environ-

ment designed to replicate real-world IT systems, networks, and infrastructures for cybersecurity testing and research [17], without impacting live operations.

In the literature, practical technological implementations of this type are scarce, or are developed around a vertical use-case scenario.

De Benedictis et al. [7] proposed an IIoT anomaly detection architecture based on DT and autonomic computing paradigms, utilizing the MAPE-K feedback loop to monitor, analyze, plan, and execute reconfiguration or mitigation strategies based on deviations from prescriptive behavior stored as shared knowledge. While effective for anomaly detection, it is limited to this scope and cannot test malicious software like ransomware due to the absence of air-gapping.

Another related work is the one presented by Masi et al. [19], who derive a cybersecurity DT as part of the security-by-design practice for Industrial Automation and Control Systems used in Critical Infrastructures. Although conceptually similar to SAFARI, this work offers only an architectural overview without detailing the enabling technologies required for implementation. In contrast, our work focuses on technological specifics, providing detailed guidance on implementing SAFARI and its practical applications in experiments.

SCASS [6], although completely Open Source, proposes a DT oriented to the implementation of a specific Industrial Component System Use Case, which is composed by a mix of virtualized components and "Hardware-in-the-loop" physical components. This hybrid testbed is then utilized to test cyber attacks specific to the ICS context. Similar to SCASS, EPICTWIN [12] proposes an Open Source virtualization of a highly specific ICS Use Case and allows for live attack simulations on SCADA systems. Differently from the last two cited works, SAFARI doesn't aim to reproduce a specific Use Case, but allows the user to configure any kind of virtualized network, will it be IT or OT/ICS oriented.

In terms of structure and implementation, one work close to SAFARI is PANDORA [11], which is a safe testing environment that allows users to conduct experiments on automated cyber-attack tools. The differences between the two proposals lie in their focus. PANDORA is designed mainly for testing automated cybersecurity tools, such as scanners or IDS systems. In contrast, SAFARI focuses on creating test scenarios that are as close as possible to real-world ones, prioritising open tools and highly modular networking virtualisation technologies, ensuring greater fidelity and enhancing flexibility in adapting to various testing needs. We report the key points of the comparison in Table 1.

5 Conclusion

Studying malware behaviour and testing detection and mitigation tools is challenging, requiring strict safety protocols and reliable statistical data. SAFARI addresses these needs by offering a democratised framework for ransomware investigation, designed for scalability, air-gapped security, and automation, catering to users including small research groups and educational institutions. At its core, SAFARI is an open-source framework that automates ransomware (and countermeasure) testing in realistic environments. Built with modern technologies, it

Compared solutions	Configurable Virtualized Infrastructure	Sandboxing Capabilities	General Purpose	Hardware Agnostic	Air-gapped by design	Open Source Code
De Benedictis et al. [7]	●	○	◐	◐	○	○
Masi et al.[19]	●	○	◐	●	○	○
PANDORA[11]	●	●	●	◐	○	○
SCASS[6]	●	●	○	○	○	●
EPIC [12]	●	●	○	◐	○	●
SAFARI	●	●	●	●	●	●

Table 1. Tabular comparison of SAFARI (last row) with related work (one work per row) under the main characteristics of the considered proposals.

integrates Infrastructure-as-Code and OS-agnostic Task Automation for reproducible and consistent experiment setups across diverse hardware. We present a prototype that demonstrates SAFARI's feasibility and effectiveness.

As an additional contribution, we conduct two case studies on profiling the behaviour of five renowned ransomware strains (Ryuk, Vipasana, WannaCry, LockBit, and Phobos) and evaluating the effectiveness of a ransomware mitigation tool (Ranflood) against three of these strains. Thanks to SAFARI we described the distinct attack strategies of the strains. Briefly, Ryuk and Vipasana primarily target the "AppData" folder, leaving behind many ".ignore" files as part of their encryption approach, while WannaCry shows a broader attack pattern, affecting multiple user directories and targeting common file extensions. LockBit exhibits the most targeted strategy, focusing on user-important files such as those on the "Desktop", "Documents", and "Pictures" folders. Phobos seems the most aggressive ransomware, encrypting nearly all files indiscriminately. Studying the effectiveness of Ranflood, we found that the copy-based strategies are the most effective in preserving user-critical data.

Future work aims to expand SAFARI's capabilities to support a wider range of malware types, such as exfiltration ransomware [20], by incorporating tools to track malicious network activities. Another direction involves enabling hybrid on-premises-cloud deployments, allowing users to scale on-premises capacity with cloud resources. For the prototype, we plan to integrate and automate additional analysis tools, such as those leveraging dynamic ransomware behaviour analysis and automated malware behaviour pattern recognition, to aid in interpreting results. To make SAFARI accessible to non-technical users, we are exploring user-

friendly interfaces to help define high-level testing plans, which the framework would automatically translate into corresponding IoC and OTA components.

Acknowledgments

We thank Matteo Cicognani for supporting the collaboration between ARPAE and Università di Bologna. This study was carried out within the "CYBER RANGE FOR INDUSTRIAL SECURITY - CRI4.0 NELL'AMBITO DEL BANDO PER PROGETTI DI RICERCA INDUSTRIALE STRATEGICA" (PR FESR 2021–2027 AZIONE 1.1.2 CUP: E37G22000490007).

References

1. Al-rimy, B.A.S., Maarof, M.A., Shaid, S.Z.M.: Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions. Computers & Security **74**, 144–166 (2018). https://doi.org/10.1016/j.cose.2018.01.001
2. Beaman, C., Barkworth, A., Akande, T.D., Hakak, S., Khan, M.K.: Ransomware: Recent advances, analysis, challenges and future research directions. Computers & Security **111**, 102490 (2021). https://doi.org/10.1016/j.cose.2021.102490
3. Berardi, D., Giallorenzo, S., Melis, A., Melloni, S., Onori, L., Prandini, M.: Data flooding against ransomware: Concepts and implementations. Computers & Security p. 103295 (2023). https://doi.org/10.1016/j.cose.2023.103295
4. Berardi, D., Giallorenzo, S., Melis, A., Melloni, S., Prandini, M.: Ranflood: A mitigation tool based on the principles of data flooding against ransomware. SoftwareX **25**, 101605 (2024). https://doi.org/10.1016/j.softx.2023.101605
5. Connolly, A.Y., Borrion, H.: Reducing ransomware crime: Analysis of victims' payment decisions. Comput. Secur. **119**, 102760 (2022). https://doi.org/10.1016/J.COSE.2022.102760
6. d'Ambrosio, N., Capodagli, G., Perrone, G., Romano, S.P.: Scass: Breaking into scada systems security. Comp. Sec. **151**, 104315 (2025). https://doi.org/10.1016/j.cose.2025.104315
7. De Benedictis, A., Flammini, F., Mazzocca, N., Somma, A., Vitale, F.: Digital twins for anomaly detection in the industrial internet of things: Conceptual architecture and proof-of-concept. IEEE TII **19**(12), 11553–11563 (2023). https://doi.org/10.1109/TII.2023.3246983
8. Dey, A., Totel, E., Costé, B.: Daemon: dynamic auto-encoders for contextualised anomaly detection applied to security monitoring. In: IFIP SEC. pp. 53–69. Springer (2022)
9. Halsey, M.: Windows 10 File Structure in Depth, pp. 449–457. Apress, Berkeley, CA (2016). https://doi.org/10.1007/978-1-4842-0925-7_27
10. Hernandez-Castro, J., Cartwright, A., Cartwright, E.: An economic analysis of ransomware and its welfare consequences. RSOS **7**(3), 190023 (2020). https://doi.org/10.1098/rsos.190023
11. Jiang, H., et al.: Pandora: A cyber range environment for the safe testing and deployment of autonomous cyber attack tools. In: SSCC. pp. 1–20. Springer (2021)
12. Kandasamy, N.K., Venugopalan, S., Wong, T.K., Leu, N.J.: An electric power digital twin for cyber security testing, research and education. Computers and Electrical Engineering **101**, 108061 (2022). https://doi.org/10.1016/j.compeleceng.2022.108061

13. Kaspersky: Ransomware attacks and types (2021)
14. Kharraz, A., Arshad, S., Mulliner, C., Robertson, W.K., Kirda, E.: UNVEIL: A large-scale, automated approach to detecting ransomware. In: Holz, T., Savage, S. (eds.) USENIX Security Symposium. pp. 757–772. USENIX Association (2016)
15. Kroll: Data exfiltration ransomware attacks (2021)
16. Liska, A., Gallo, T.: Ransomware: Defending Against Digital Extortion. O'Reilly Media, Inc., 1st edn. (2016)
17. Mahmoud, R.V., Anagnostopoulos, M., Pedersen, J.M.: Detecting cyber attacks through measurements: Learnings from a cyber range. IEEE IMM **25**(6), 31–36 (2022). https://doi.org/10.1109/MIM.2022.9847127
18. Maigida, A., Abdulhamid, S., Olalere, M., Alhassan, K., Chiroma, H., Dada, E.: Systematic literature review and metadata analysis of ransomware attacks and detection mechanisms. Journal of Reliable Intelligent Environments **5** (07 2019). https://doi.org/10.1007/s40860-019-00080-3
19. Masi, M., Sellitto, G.P., Aranha, H., Pavleska, T.: Securing critical infrastructures with a cybersecurity digital twin. Software and Systems Modeling **22**(2), 689–707 (2023)
20. McIntosh, T., Susnjak, T., Liu, T., Xu, D., Watters, P., Liu, D., Hao, Y., Ng, A., Halgamuge, M.: Ransomware reloaded: Re-examining its trend, research and mitigation in the era of data exfiltration. ACM Comput. Surv. (Aug 2024). https://doi.org/10.1145/3691340
21. McIntosh, T., et al.: Ransomware mitigation in the modern era: A comprehensive review, research challenges, and future directions. ACM Comput. Surv. **54**(9) (Oct 2021). https://doi.org/10.1145/3479393
22. Meland, P.H., Bayoumy, Y.F.F., Sindre, G.: The ransomware-as-a-service economy within the darknet. Comput. Secur. **92**, 101762 (2020). https://doi.org/10.1016/J.COSE.2020.101762
23. Meurs, T., Junger, M., Tews, E., Abhishta, A.: Ransomware: How attacker's effort, victim characteristics and context influence ransom requested, payment and financial loss. In: APWG eCrime. pp. 1–13. IEEE (2022). https://doi.org/10.1109/ECRIME57793.2022.10142138
24. OpenWrt Team: Openwrt. https://openwrt.org/, [Online; accessed Sept. 2024]
25. Oz, H., Aris, A., Levi, A., Uluagac, A.S.: A survey on ransomware: Evolution, taxonomy, and defense solutions. ACM Comput. Surv. **54**(11s) (Sep 2022). https://doi.org/10.1145/3514229
26. Oz, H., Aris, A., Levi, A., Uluagac, A.S.: A survey on ransomware: Evolution, taxonomy, and defense solutions. ACM Comput. Surv. **54**(11s) (Sep 2022). https://doi.org/10.1145/3514229
27. Pokhrel, A., Katta, V., Colomo-Palacios, R.: Digital twin for cybersecurity incident prediction: A multivocal literature review. pp. 671–678. ICSEW'20, ACM, New York, NY, USA (2020). https://doi.org/10.1145/3387940.3392199
28. Richardson, R., North, M.: Ransomware: Evolution, mitigation and prevention. International Management Review **13**(1), 10–21,101 (2017), https://www.proquest.com/scholarly-journals/ransomware-evolution-mitigation-prevention/docview/1881414570/se-2
29. Rossow, C., Dietrich, C.J., Grier, C., Kreibich, C., Paxson, V., Pohlmann, N., Bos, H., Van Steen, M.: Prudent practices for designing malware experiments: Status quo and outlook. In: IEEE S&P. pp. 65–79. IEEE (2012)
30. Scaife, N., Carter, H., Traynor, P., Butler, K.R.B.: Cryptolock (and drop it): Stopping ransomware attacks on user data. In: IEEE ICDCS. pp. 303–312 (2016). https://doi.org/10.1109/ICDCS.2016.46