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

Tommaso Compagnucci[a], Saverio Giallorenzo[a,b], Andrea Melis[a], Simone Melloni[c], Marco Prandini[a], Alessandro Vannini[d]

[a]*Alma Mater Studiorum – Università di Bologna, Bologna, Italy*
[b]*INRIA, Sophia Antipolis, France*
[c]*ARPAE Emilia-Romagna, Italy*
[d]*Bureau Veritas Cybersecurity, Italy*

Abstract

Ransomware poses a significant threat to individuals and organisations, creating a need for tools to investigate its behaviour and the effectiveness of mitigations. To address 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 conduct two case studies: the first analyses seven ransomware strains – including WannaCry and LockBit – to identify their encryption patterns and file-targeting strategies; the second evaluates Ranflood, a countermeasure tool, against five 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 organisations' data, demanding payment to restore access to the rightful owners [1]; it is one of the most pressing cybersecurity threats [2, 3] facing organisations and individuals worldwide. In its most general definition, ransomware extorts victims through their data. Economically, ransomware imposes sizeable costs on society [4, 5]: indeed, it forms a sophisticated, multi-billion dollar industry that continues to adapt and overcome defensive measures [6].

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 [7].

Recent reports confirm that the threat remains acute. Chainalysis estimated that ransomware payments exceeded $1 billion in 2023, marking a record year for extorted cryptocurrency payments, and its 2024 mid-year update reported $459.8 million in payments already recorded in the first half of 2024 [8, 9]. Complementarily, the FBI's IC3 received more than 3,600 ransomware complaints in 2025 alone, while explicitly warning that direct monetary losses understate the overall impact, overlooking business interruption, remediation costs, and other indirect damage [10]. These figures reinforce that ransomware is not only persistent, but also operationally disruptive at scale.

The study of ransomware behaviour in different contexts and conditions provides fundamental knowledge for countering it. The scientific and technical literature abounds with solutions to counteract ransomware [11, 12, 13, 3, 14, 15]. Studying the behaviour of these solutions against real-world malware is important both for evaluating their effectiveness and efficiency and for helping their evolution.

We respond to such needs with a platform for the examination of ransomware and the evaluation of mitigation actions. *Safety* and *efficiency* of investigation

are the main challenges we address in the design of our proposal. To evaluate ransomware attacks and 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.

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. The tool's architecture is based on the use of virtual machines (VMs), whether hosted on cloud platforms or on premises. We favour the latter choice to enable investigators to use local resources to assemble the system; accordingly, we designed our prototype for that deployment modality. In this way, small research groups, as well as student collectives with access to home-lab-sized hardware, can configure their own deployment and run experiments.

Given the dangerous and highly infectious nature of ransomware, our tool runs 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).

The proposed framework integrates complementary technologies for the definition of experiments and the collection and analysis of attack data. Our proof-of-concept is based on the architecture described in Section 3, where SAFARI is explained both in terms of concepts and software choices needed for an effective simulation of the behaviour of ransomware and its contrast solutions.

While our tool'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 user data, which we use as the metric to measure the effectiveness of both malware and contrast tools. In Section 4, we showcase our framework 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 [16, 17], 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 4 as evidence of the potential of SAFARI, the analysis itself constitutes a contribution that adds new knowledge about the analysed malware's behaviour.

This article is a revised and expanded version of earlier conference work published at the 40th IFIP International Conference ICT Systems Security and Privacy Protection [18]. Besides a revision of the whole original presentation, we introduce (a) two new meta-analyses, (b) a new presentation of the logic of attack-profile visualisation, and (c) new case study scenarios through the analysis of new ransomware strains. The first meta-analysis, presented in Section 3.1, regards SAFARI's prototypical implementation and reviews the available alternatives for the implementation of the three main concepts behind the tool's architecture: Infrastructure as Code, OS-agnostic Task Automation, and Investigation Tools, justifying the adoption of specific off-the-shelf components as well as the need to integrate them with tailor-made software. The second meta-analysis is about the ransomware mitigation solutions considered in our case studies (Section 4.2) – expanded with new experiments on additional ransomware strains. The other noteworthy addition, presented in Section 3.4, is a detailed explanation of the visualisation logic underlying both the original *doughnut-chart profiles* and the newly introduced *file-system-tree profiles* of the attacks, which provide a structural visualisation of attack patterns that were not explicit in the aggregated view presented in the conference version [18]. This new two-level representation of attack profiles allows operators to assess the impact of a studied scenario at a glance, while also enabling a deeper investigation of the attacker's behaviour.

We position our contribution within the literature in Section 2 and discuss final remarks and future development in Section 5.

2. 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 [24], when

Compared solutions	Configurable Virtualised Infrastructure	Sandboxing Capabilities	General Purpose	Hardware Agnostic	Air-gapped by design	Open Source
De Benedictis et al. [19]	●	○	◐	◐	○	○
Masi et al. [20]	●	○	◐	●	○	○
PANDORA [21]	●	●	●	◐	○	○
SCASS [22]	●	●	○	○	○	●
EPIC [23]	●	●	○	◐	○	●
SAFARI	●	●	●	●	●	●

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

viewed as a DT application, is a virtualised environment designed to replicate real-world IT systems, networks, and infrastructures for cybersecurity testing and research [25], without impacting live operations.

In the literature, practical technological implementations of this type are scarce, and those that exist are often developed around vertical use-case scenarios.

De Benedictis et al. [19] proposed an IIoT anomaly detection architecture based on DT and autonomic computing paradigms, using the MAPE-K feedback loop to monitor, analyse, plan, and execute reconfiguration or mitigation strategies based on deviations from prescriptive behaviour 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 contribution is the one presented by Masi et al. [20], 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 [22], although completely open source, proposes a DT tailored to a specific Industrial Control System use case, composed of a mix of virtualised components and "hardware-in-the-loop" physical components. This hybrid test bed is then used to test cyberattacks specific to the ICS context. Similar to SCASS, EPICTWIN [23] proposes an open-source virtualisation of a highly specific ICS use case and allows live attack simulations on SCADA systems. Unlike these two works, SAFARI does not aim to reproduce a single use case; rather, it allows the user to configure arbitrary virtualised networks, whether IT- or OT/ICS-oriented.

In terms of structure and implementation, one work close to SAFARI is PANDORA [21], 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.

Beyond framework-oriented proposals, the literature also includes a broad set of ransomware identification and detection methods across platforms. On Windows, UNVEIL [26] is an early large-scale dynamic-analysis approach based on file-system interactions, while more recent works explore real-time system-call monitoring [27], explainable dynamic learning-based detection [28], and static structural feature analysis for identifying previously unseen ransomware families [29]. On Android, researchers have proposed both feature-engineering and learning-based solutions, including ARdetector [30] and traffic-analysis-based supervised detection [31]. Taken together, these studies show that the research landscape spans Windows and Android, static and dynamic analyses, and detection pipelines based on both handcrafted features and machine learning. They are complementary to SAFARI: they mainly target the classification or early detection of ransomware samples, whereas our contribution is an experimental infrastructure that enables the safe, auto-

mated, and reproducible execution of ransomware campaigns and countermeasure evaluations across configurable environments.

As a final remark, we mention the work by Callegati et al. [32] on the SAFARI framework applied beyond the IT domain, applying it to Operational Technology (OT) environments, and demonstrating its versatility as a general-purpose foundation for Security-Investigation-as-Code. The authors instantiate SAFARI as a digital twin and cyber range, exploiting its Infrastructure-as-Code layer – realised with Terraform and Proxmox SDN – to construct faithful, air-gapped replicas of industrial network architectures. Rather than the file-centric Filechecker/Profiler pair employed in the present work, this OT-oriented instantiation adopts MITRE Caldera [33] as the Investigation Tools (IT) component, leveraging its automated adversary emulation capabilities to generate and execute attack scenarios and systematically probe each replica for exploitable vulnerabilities. The authors evaluate three industrial network topologies of increasing segmentation, finding that SAFARI automates complex, multistep OT attack scenarios end-to-end, supports regression testing of architectural refinements, and delivers quantifiable evidence of the security improvements that network segmentation provides. Taken together, the two works establish SAFARI as a modular, domain-agnostic investigation platform whose layered architecture – IaC, OTA, and pluggable IT – accommodates both IT ransomware analysis and OT vulnerability assessment without modifying the framework's core.

3. 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, along with 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 tools that provide visibility and 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 generate 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.

3.1. Meta-analysis

The design of SAFARI deliberately abstracts each of its three foundational layers – Infrastructure as Code (IaC), OS-agnostic Task Automation (OTA), and Investigation Tools (IT) – from any specific technological realisation, so that concrete tool choices can be made independently for each deployment context. Thus, before detailing our prototypical implementation, we contextualise our choices within the landscape of available alternatives. For each layer, we characterise the relevant design space, survey the principal candidate technologies, and argue why the tools selected for our prototype – respectively Terraform, Ansible, and the Filechecker-Profiler pair – represent an appropriate realisation, given SAFARI's requirements.

IaC Alternatives. From a practical standpoint, several mature IaC solutions exist as alternatives. Terraform [34] and its community-driven, open-source fork OpenTofu [35] are cloud-agnostic, declarative frameworks based on HashiCorp Configuration Language, offering multi-cloud portability, a mature provider ecosystem, reusable modules, and a predictable plan/apply execution model. Pulumi [36] differs by enabling infrastructure definition in general-purpose programming languages such as TypeScript, Python, and Go, allowing richer abstractions and tighter integration between infrastructure and application code. Cloud-native solutions – AWS CloudFormation and AWS CDK [37, 38],

[1] https://www.terraform.io/

[2] https://www.redhat.com/en/ansible-collaborative

[3] https://www.proxmox.com/

and Azure ARM templates and Bicep [39, 40] – provide deep integration with their respective cloud ecosystems, but are inherently vendor-specific and unsuitable in multi-cloud or on-premises architectures. Another alternative regards Kubernetes-native control plane solutions, such as Crossplane [41], which support unified infrastructure and application lifecycle management via Kubernetes APIs, at the cost of requiring a dedicated, operational Kubernetes cluster as a prerequisite.

Among these alternatives, we deem Terraform a natural choice for SAFARI for two complementary reasons. First, and most critically, Terraform is the only framework in the survey with a mature, community-maintained provider for on-premises, open-source hypervisors, like Proxmox, which we foresee constituting one of the main scenarios for SAFARI's deployment. Cloud-native alternatives have no support for on-premises hypervisors by design, Pulumi's Proxmox support is nascent and unstable, and Crossplane would demand a full Kubernetes control plane solely to manage a handful of experimental VMs – an overhead incompatible with SAFARI's democratisation principle. Second, Terraform's declarative, state-reconciling model provides strong idempotency guarantees, where repeated applications of the same configuration converge to an identical infrastructure state. Should licensing transparency be a concern, OpenTofu would preserve all of the above properties without any modification to the configuration codebase.

OTA Alternatives. Within the landscape of OS-agnostic task automation, several mature frameworks offer alternative realisations, each instantiating different trade-offs in deployment model, execution semantics, and operational overhead [42]. SaltStack [43] adopts an event-driven, agent-based architecture in which a central coordinator communicates with distributed agents over a publish-subscribe messaging bus, enabling high-throughput, concurrent remote execution and reactive orchestration. Puppet [44] provides a declarative, pull-based model in which agents periodically fetch and enforce configurations expressed in the Puppet DSL from a central controller. Puppet provides a rich module ecosystem that makes it well-suited for continuous enforcement across large, heterogeneous fleets, yet it similarly demands dedicated master-agent infrastructure and presents a steep operational on-boarding cost. Chef [45] diverges from the declarative paradigm by offering an imperative, Ruby-based specification model in which recipes and cookbooks describe the sequential steps the system must execute, affording fine-grained procedural control and tight integration with test-driven infrastructure workflows. Fabric [46] is an SSH-based task-execution library based on direct, scripted remote-command execution. Thanks to its approach, Fabric enjoys a minimal footprint, but it provides no idempotency guarantees, no declarative state model, and no structured mechanism for parameterised, reproducible experiment orchestration.

Against the backdrop of the alternatives surveyed above, a final alternative, Ansible [47], emerges as the most appropriate OTA solution for SAFARI, primarily by virtue of its agentless architecture and its native support for both push-based and local execution modes. The decisive factor derives from SAFARI's multi-tiered OTA model: while the external orchestration layer coordinates the transfer of tasks and malware binaries to the test VMs via SSH, the internal OTA layer – which disables network connectivity, verifies air-gap isolation, and triggers the ransomware execution entirely from within the VM – must operate without any outbound network connectivity. Technically, Ansible operates as a transient process – invoked, executed, and terminated without leaving any running background component – while agent-based tools install persistent daemons that remain active throughout the lifetime of the managed node. Ansible avoids this contamination entirely. Once the external OTA layer injects the playbook and control passes to the internal OTA tier, Ansible executes its tasks sequentially – disabling the network, verifying isolation, and triggering the ransomware – and then exits cleanly, leaving the VM in a quiescent, daemon-free state before and during ransomware execution. This transient execution model is a structural property of Ansible's architecture and cannot be replicated by agent-based alternatives.

Fabric, while agentless and SSH-based, lacks the idempotency guarantees and declarative state model required by SAFARI's reproducibility objectives: without a structured notion of desired system state, Fabric cannot ensure that repeated invocations of an experiment produce consistent configurations across different VMs, thereby undermining the statistical validity of experiments. On the contrary, Ansible's strong idempotency semantics ensure that the experimental environment is consistently reproduced across test sessions.

IT Alternatives. Several complementary categories of investigation tools exist for ransomware experimentation. Adversary emulation platforms such as MITRE Caldera [33, 32] provide automated, scriptable generation of attack scenarios, enabling systematic exploration of adversary tactics, techniques, and procedures. At the opposite end of the spectrum, volatile-memory foren-

sics approaches proposed by Davies et al. [48] and Bajpai et al. [49] focus on acquiring RAM snapshots during live ransomware execution to extract cryptographic keys directly from memory, enabling file decryption without ransom payment. At the filesystem level, McIntosh et al. [50] and Kharraz et al. [51] instrument the OS kernel to monitor I/O Request Packets and Master File Table modifications in real time, deriving behavioural signatures and anomaly-based detection rules from burst write patterns and directory targeting. Finally, cross-layer recovery solutions such as FFRecovery [52] integrate file system forensics analysis to restore overwritten data, at the cost of requiring storage-level modifications and tight system integration.

We deem that none of the tools surveyed above is fully aligned with the investigative objective of SAFARI. Adversary emulation platforms such as Caldera [33] are positioned upstream of the measurement problem: they generate attacks but provide no mechanism to quantify their file system impact. Volatile-memory forensics approaches [48, 49] target a different research question – key extraction and file decryption – and are by design time-sensitive and non-reproducible, since the cryptographic material they seek is ephemeral and present only during active encryption. Kernel-level monitoring solutions [50, 51] are oriented toward real-time detection rather than post-mortem quantification: their instrumentation runs concurrently with the ransomware, generating file system and memory events that contaminate exactly the signal one would wish to measure in a controlled benchmarking experiment. FFRecovery [52], while post-incident in orientation, is recovery-focused and requires invasive storage-level integration, making it unsuitable as a lightweight, portable measurement layer deployable across heterogeneous VM templates. The shared limitation is the absence of a static, non-interfering, and reproducible methodology for the quantitative assessment of ransomware impact and countermeasure effectiveness across repeated experimental runs.

To close this gap, in this paper, we propose two new IT tools: the Filechecker and the Profiler. This pair of purpose-built, open-source tools operates exclusively in two quiescent windows – once on the pristine VM template before the attack, and once on the offline disk image mounted in the analysis VM after it – so that no investigative process is ever co-resident with the ransomware during execution. Together, they produce per-file and per-directory breakdowns of pristine, lost, and replica file ratios that serve as the primary quantitative output of SAFARI's experiments.

3.2. Software Architecture

We illustrate the architecture and experimental 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. In the figure, we use borders 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, i.e. 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).

In Figure 1, from the left, we find the core components of the SAFARI prototype, which orchestrates 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 an operator can run multiple times and in parallel to gather statistical experimental data.

We describe the prototype's workflow following the sequence shown 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,

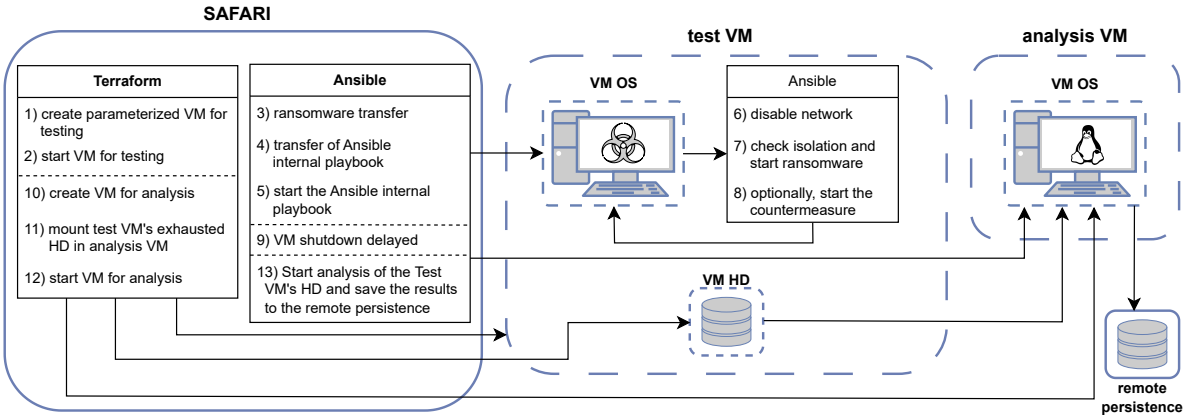

Figure 1: SAFARI's prototype architecture and functionalities.

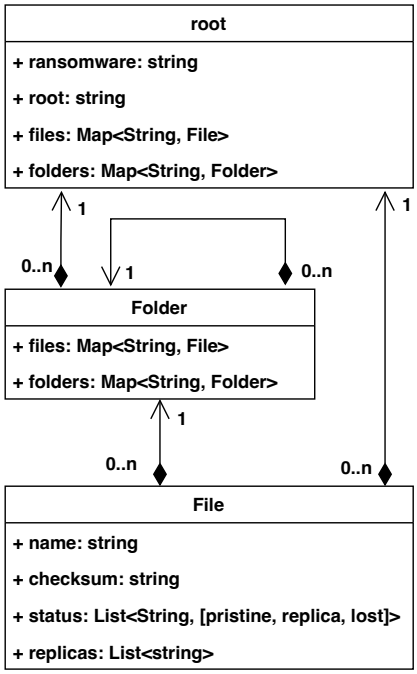

Figure 2: Schema of the Hierarchical Profile.

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.

3.3. 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.

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 we obtain 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 as files in the reference report)[5]. 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

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

[5] The "reading" one can attribute to the presence of these files is context-dependent. If the profile corresponds to an uncontrasted attack, the replicas are probably created by a piece of ransomware that temporarily copied the user's files before encrypting the originals. If the profile corresponds to the usage of contrasting tools, the replicas could be a byproduct of their behaviour, e.g., as it happens with the copy-based contrast modalities of Ranflood [16, 17].

the modality of execution of ransomware, since many of these create copies of the files to encrypt before deleting them [26]. 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 is a JSON profile with the same structure as the summary one but with additional elements that report statistical measures on the files such as 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.

3.4. IT Tools: Visualisation Methodologies

As a final item, we briefly discuss the implementation of visualisation routines useful to represent the data gathered by the IT tools discussed in Section 3.3. Indeed, while the essential point of our approach "stops" at the means for gathering data on attacks, we deem it important to discuss effective ways for operators to quickly grasp the evidence carried by the data – case in point, we use these visualisations to support the analysis of the data from our experiments (cf. Section 4.2).

More specifically, in this context, we deem it paramount to define visualisation methodologies that can compactly summarise one or more aspects of the data – i.e., they are expressive – but do not give way to ambiguity – i.e., they help the operator in forming correct intuitions about the data.

In this work, we adopt a multi-level visualisation strategy that operates on the aggregated, statistically averaged profiles generated by the IT tools. At the first level, we represent each profile as a set of proportional doughnut charts. At the second level, we complement these per-profile charts with a file-system-tree visualisation that renders the spatial distribution of the attack surface across the directory hierarchy, providing a more comprehensive view of the attack behaviour. The third level analyses the per-extension outcome distribution of a ransomware attack using an additive colour space that conveys threat severity at a glance and allows side-by-side comparison of multiple ransomware families.

Doughnut-Chart Profiles. Doughnut-Chart profiles employ a uniform layout in which each doughnut/ring breaks down the profile of a specific aspect of an experimental result.

We consider four aspects: one represents the overall profile of the attack, called *summary*, and three break down the distribution of *pristine*, *lost*, and *replica* files. The break-down of the last three aspects concerns two categories: the percentage of either the *folders* that contain files that belong to that aspect (e.g., the percentage of folders whose files have been lost during an attack) – in this work, we limit the scope of folders to the top-level ones, for compactness – or the *file types (extensions)* that belong to that aspect (e.g., the type-aggregate percentage of files lost during an attack).

The doughnut-chart layout supports simultaneous reading at two resolutions, where an operator can first assess the severity of the attack from the summary aspect and then drill into the extension and folder panels to characterise which classes of files and areas of the file system have been affected.

File-System-Tree Profiles. The doughnut-chart profiles excel at providing an aggregate, overall view of the file counts across the entire file-system, but they obscure the *structural* pattern of the attack. Viewed under a negative lens, doughnut-chart profiles do not reveal which paths sustain the main losses or experience the most "contention" by a given contrast tool. To address this limitation, we introduce a complementary file-system-tree visualisation. More specifically, whereas doughnut charts present a *marginal* summary of the attack profile, tree views present a *joint* spatial distribution of the profile across the file-system's hierarchy.

To build this visualisation, we model the file-system as a rooted tree $T = (V, E, r)$, where $V = V_F \cup V_D$ partitions nodes into file leaves V_F and folder nodes V_D (since we have files and folders, we disambiguate by using D for "directory"), E encodes parent-child containment relationships, and $r \in V_D$ is the unique root vertex corresponding to the user's top-level data directory. Although the underlying structure is strictly a tree – acyclic and singly-rooted – we render and lay out T as

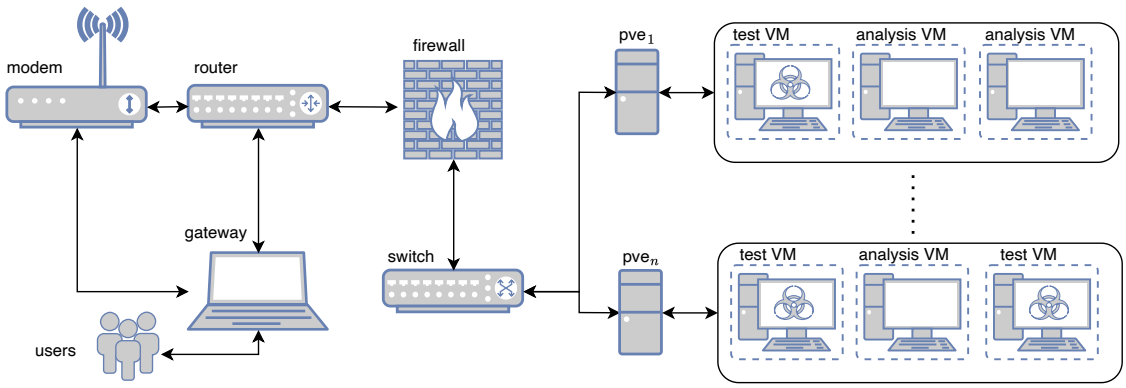

Figure 3: SAFARI's Prototype Deployment Infrastructure.

a general undirected graph using a force-directed algorithm, as detailed below.

We assume that the visualisation ingests an encoding of the averaged outcomes of multiple ransomware attack profiles. Each node $v \in V$ carries a normalised triple $(\pi_v, \lambda_v, \rho_v) \in [0,1]^3$ satisfying $\pi_v + \lambda_v + \rho_v = 1$, where π_v, λ_v, and ρ_v denote the fraction of attack instances in which the entity remained pristine, was lost, or was recoverable through a backup replica, respectively – note that the interpretation slightly changes whether v is a file or a folder; if $v \in V_F$ the triple represents the aggregate of the file status over the attack instances, while for $v \in V_D$ the values represent the averaged distribution of the aggregate statuses of the folder's files and subfolders over the attack instances.[6]

We complete the definition by introducing the operator num(v), which denotes the number of files found directly within a folder $v \in V_D$ and its subfolders – this measure propagates quantities upward to ancestor nodes, so each folder node reflects the cumulative count of its entire descendant file population.

Due to the potentially large cardinality of real-world folder trees, we avoid rendering the complete vertex set. Instead, we apply a two-tier size-based pruning strategy:

- rendering set: we retain the top 20% of folder nodes ranked by num(v) in decreasing order with lexicographic ordering on directory path as tie-breaking policy, removing all remaining nodes (both file nodes and small directories) from the graph prior to layout computation.
- labelling set: we label the top 5% of folder nodes by num(v).

This strategy ensures that the visualisation focuses cognitive load on the highest-impact regions of the file-system, consistent with Pareto-inspired analysis of attack surfaces in prior work [53, 54].

We encode each folder v along three independent visual dimensions: chromaticity, size, and force-direction.

We use chromaticity to represent the attack profile, mapping the aggregate of the normalised distribution of files (pristine, lost, and replicas) in a folder to a node colour resulting from the additive RGB vector $(\lambda_d, \pi_d, \rho_d)$. In this way, the visualisation yields a purely red node for total loss (λ_d), a purely green node for intact folders (π_d), and a purely blue node for a folder always recoverable from replicas (ρ_d). Intermediate damage profiles produce mixed hues that intuitively communicate the consistency of each attack profile at a glance.

We render each node's size proportional to the folder's relative file population num(v)/num(r), where num(r) is the aggregate file count of the root, which corresponds to the count across all folders.

The dual encoding of chromaticity and size allows an operator to assess the spatial extent of the damage and its severity, without requiring separate panels.

To structure the relationships among the entities in the graph, we position nodes using the Fruchterman-Reingold spring-embedder algorithm [55] on a weighted version of the pruned graph, assigning edge weights as per Equation (1). In the layout, edges with higher weight translate into tighter springs and closer nodes, while lower weights correspond to looser springs and nodes positioned farther apart. Formally, assuming ord(v), $v \in V_D$, returns the natural number ordinal positioning of v in the decreasing-order folder se-

[6] Of course, one could use the same visualisation to represent a single instance, where, for $v \in V_F$, one element of the triple is 1 and the others are 0s, while folders continue to normalise these values.

quence induced by num(v) – used above to cut off folders below the top 20% – the weight $w(u, v)$ of the edge between vertices $\{u, v\} \subseteq V_D$ corresponds to

$$w(u, v) = \omega * \log(\text{ord}(u) + \text{ord}(v)) \quad (1)$$

In Equation (1), $\omega \in \mathbb{R}^+$ acts as a global scaling factor, uniformly shifting the magnitude of all edge weights without altering their relative ordering – in our implementation, $\omega = 0.01$. In the measure, the adoption of a logarithmic formulation for $w(u, v)$ implements a principle of *diminishing marginal relevance*. Indeed, the argument of the logarithm, $\text{ord}(u) + \text{ord}(v)$, encodes the combined ordinal rank of both endpoint folders: pairs involving high-ranking folders (i.e., those with many files, hence low ord values) yield a smaller argument and thus lower weights and looser springs, whereas pairs involving lower-ranked directories produce larger arguments, higher weights, and tighter springs. The logarithmic compression prevents the raw sum of ranks from inducing disproportionately large weight disparities between edges, guaranteeing that $w(u, v)$ grows *sublinearly* with the rank sum, bounding the dynamic range of the weight distribution and conferring a "natural" and well-distributed disposition of the vertices. While we apply the above logic to all folders, we treat the root and user folders differently. We assign the minimal weight ω to edges where one of the two vertices is the root, so we can properly display the main subfolders (the root node might be quite large and overlap with/hide its subfolders). Moreover, we enforce the inclusion of user folders (such as Desktop, Documents, Pictures, Videos, etc.) in the visualisation and show their labels – so that they appear in visualisations although the filtering by num(d) might have excluded them.

The algorithm lays out the nodes by first setting them at random positions, and on each iteration it computes attractive forces along edges and repulsive forces between all pairs of vertices, sums the forces on each vertex, and moves it slightly in the resulting direction. Since our implementation – which relies on the NetworkX library [56] – provides the definition of a fixed seed, given the same set of nodes (and relative total sizes, which should remain the same in our settings, given that the base profile is the same for all attack profiles) all visualised profiles preserve the same graph layout, substantially facilitating the task of operators in comparing attack profiles of different pieces of ransomware and contrast tools.

Format Spectrum Profiles. To further investigate the selective behaviour of modern ransomware – which frequently discriminates between file types, skipping system-critical extensions to preserve operability while maximising ransom leverage – we introduce the *format spectrum* profiles, a colour-coded visualisation that maps the per-extension outcome distribution of a ransomware attack onto an additive RGB colour space. The format spectrum follows three guiding principles: (i) *information density*, so that each visual element encodes three simultaneous quantities without requiring multiple subplots; (ii) *semantic immediacy*, given by the colour of a bar that conveys threat severity at a glance (exploiting the same pre-attentive perceptual association between red and danger found in tree profiles); and (iii) *comparability*, through a fixed format category taxonomy and a deterministic sort order to support side-by-side comparison of multiple ransomware types.

The colouring of each bar of the spectrum follows a logic similar to the one discussed for the tree profiles. Formally, let $\mathcal{E}$ denote the set of file extensions observed across an aggregated experiment comprising N independent runs of the same ransomware. For each extension $e \in \mathcal{E}$, let $n_l(e)$, $n_p(e)$, and $n_r(e)$ denote the total number of files classified as *lost*, *pristine*, and *replica*, respectively. Then, we define the normalised status mass $S(e)$ as $S(e) = n_l(e) + n_p(e) + n_r(e)$ and the three outcome proportions as

$$r_l(e) = \frac{n_l(e)}{S(e)}, \qquad r_p(e) = \frac{n_p(e)}{S(e)}, \qquad r_r(e) = \frac{n_r(e)}{S(e)}$$

with $r_l(e) + r_p(e) + r_r(e) = 1$ by construction.
Each extension is then assigned the RGB triple

$$c(e) = (r_l(e), r_p(e), r_r(e))$$

where the red channel encodes destructive impact, the green channel encodes resilience, and the blue channel encodes replica injection activity. Visually, the format spectrum profile renders the extensions following a fixed sorting order and each extension as a vertical bar. A pure-red bar indicates a fully lost format, a pure-green bar indicates a fully spared format, and a pure-blue bar indicates total replica injection. Intermediate hues represent mixed outcomes and capture partial targeting patterns throughout experiments.

The taxonomy partitions extensions into semantically coherent categories such as Document, Image, Database, Executable/Library, etc., and fixes the relative position of these groups and of the contained formats.

This organisation provides two complementary benefits. First, it preserves semantic structure, making it

possible to quickly assess which formats a piece of ransomware concentrates on. Second, it yields a stable visual scaffold for multi-sample comparison.

Broadly, the format spectrum makes explicit that ransomware impact is semantic – e.g., two attacks with similar global loss rates may differ substantially in the classes of files they disrupt, and those differences matter for contrasting attacks.

4. 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.

4.1. SAFARI's Prototype Deployment Infrastructure

The case studies help 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 placed 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[7] to access directly the nodes, i.e., to run commands as if executed on machines in their local network.

The *router*, running OpenWrt [57], can provide Internet connectivity to the nodes in the cluster, but by default its *firewall* prevents VMs from accessing the Internet to avoid the possible propagation of ransomware.

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

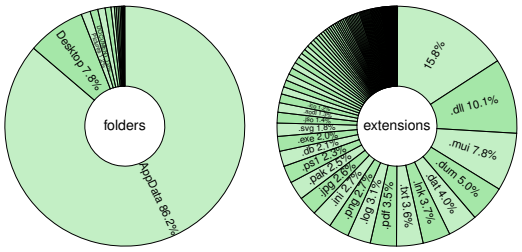

Figure 4: Visualisation of the profiles of user files.

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.

4.2. 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 [58]. We source the samples used for the tests from repositories previously used and recognised in other academic contexts [59], such as VirusTotal. The selection includes seven renowned Windows ransomware: Ryuk, Vipasana, WannaCry, LockBit, Akira, Chaos, and Phobos – we report in Table 2 the SHA signatures of the ransomware samples.

Experiments. In the first case study, we demonstrate our SAFARI prototype's capability to analyse the encryption patterns of seven well-known Windows ransomware: Ryuk, Vipasana, WannaCry, LockBit, Akira, Chaos, and Phobos. To simulate a real-world scenario, we configured the test VM with a typical user profile, modelled after Hansley [60], 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. [61] and Kaspersky [62], along with file names commonly targeted by ransomware [63]. Additionally, it incorporates user-interactive programs such as a web browser and an office suite [64]. The profile includes 13 "user" folders,

Ransomware	SHA-256 Signature
Vipasana	1733b199a7063443c167e3caeae7dda2315f590341ea2152a9b132e1ad8e94a8
Ryuk	23f8aa94ffb3c08a62735fe7fee5799880a8f322ce1d55ec49a13a3f85312db2
WannaCry	ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa
LockBit	af0821ba5e0889e21f6246c7f33d89caf320a5452e620d28d39948fde3ea20ac
Akira	06c2a137c31aae5d02b4d7df61ffd31f1af9a9e59978f15b3f7265cc751bff1f
Chaos	d174cf908b7bcee10cc0458955554e5ba81beffd8544ae95a427fff643a41c86
Phobos	a91491f45b851a07f91ba5a200967921bf796d38677786de51a4a8fe5ddeafd2

Table 2: SHA-256 signatures of the ransomware samples included in the case studies.

such as “Documents”, “Desktop”, “Music”, and “Pictures” with the breakdown reported as a doughnut chart in Figure 4, using the prototype’s Filechecker and Profiler tools.

For each ransomware strain, we run at least 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, Vipasana, Akira, and Chaos, and 16 for Phobos and WannaCry, discarding runs in which the ransomware did not start.

Doughnut-Chart Profiles. We start the discussion of the results with the doughnut-chart profiles of the attack, reported in Figure 5. In the figure, we display each ransomware’s data in a row and divide it (from left to right) between the percentage summary of pristine-lost-replica files and the percentage breakdown of pristine, replica, and lost files by extensions and folders. As Ryuk and Vipasana exhibit similar behaviour, 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 Vipasana 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.

We classify Akira and Chaos at the higher end of the aggressiveness spectrum, since they encrypt 49.8% and 57% of files, respectively, while leaving 5.9% and 4.0% as replica files. Akira’s attack is mainly focused on the “AppData” folder, similarly to Ryuk/Vipasana. Looking at the replica files, we notice that, like LockBit and Ryuk/Vipasana, Akira also creates many “.ignore” replica files, probably due to a copy-and-encrypt attack strategy. Chaos’ attack seems more oriented towards user files, signalled by the files lost in “Desktop”, “Pictures”, and “Documents”. Chaos also seems to implement a copy-and-encrypt attack strategy, however, it does not move or rename the replicas it creates, as visible from the breakdown of replica files found in folders like “Desktop”, “Pictures”, “Documents”, “Videos”, etc.

Phobos appears to be the most dangerous, 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 into 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

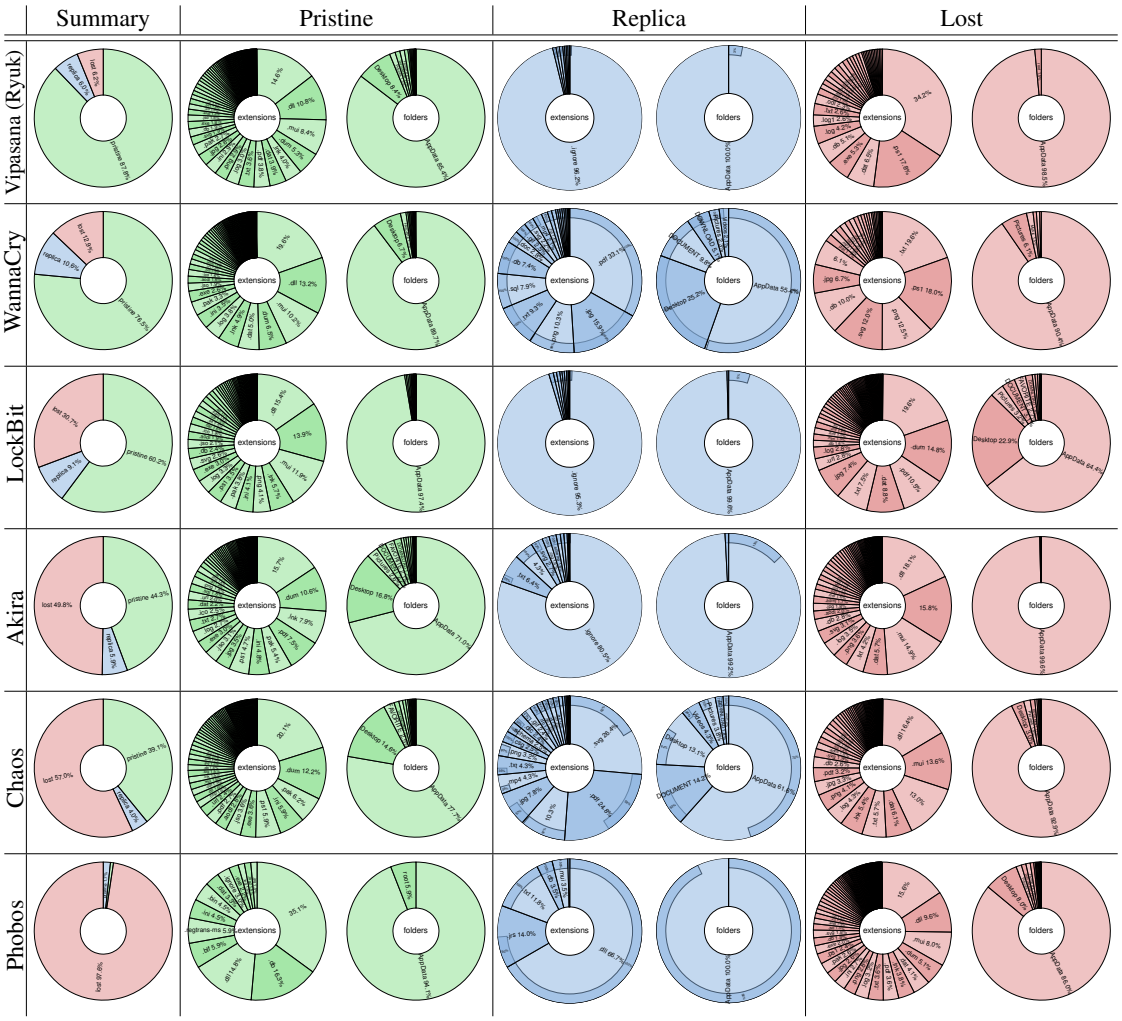

Figure 5: Doughnut-chart visualisation of the attack profiles of Vipasana/Ryuk, WannaCry, LockBit, Akira, Chaos, and Phobos.

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 is, of course, an important measure in itself). For example, Phobos has the highest percentage of lost files while LockBit seems less dangerous since it encrypts 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. By contrast, LockBit is much more efficient, since it focuses its effort only on the files that matter most to the user, i.e., the files found under "Desktop", whose sequestration compels the user to pay the ransom.

File-System-Tree Profiles. We proceed with our analysis by presenting the file-system-tree profiles of the attacks in Figure 6.

Overall, as expected, the tree visualisation provides an at-a-glance impression of the spatial extent of the damage and its severity, given by the size of the nodes (which represent the total number of files reachable from a given folder, directly or through subfolders) and their colouring. Starting from Vipasana/Ryuk, we notice – as already visualised through the doughnut charts – that the attack is substantially limited. Looking at the red-shaded nodes, the tree visualisation helps us localise the main sites of attack under the "AppData" folder (LocalLow, Local, etc.). Moving to WannaCry, we find a more interesting attack profile: many files are lost under the "AppData" folder, but some files are also lost under the user's profile, such as "Desktop" (and, from the

doughnut chart, “Pictures”, etc.). As noted through the doughnut-chart analysis, we find many replicas, which the tree-shaped visualisation helps locate precisely in the blue-shaded folders. From the distribution, we notice that there is not a specific location for replicas; it rather seems that the general behaviour of the ransomware is to first create a copy of each file it wants to encrypt, use the latter to encrypt the file, and later delete it. The tree visualisation confirms the narrow focus of LockBit, which mainly attacks user files (“Desktop”, etc.), leaving files under “AppData” almost untouched and recoverable (although it would be interesting to further investigate the reason behind the encryption of files under the AppData/Local folder). Moving to Akira and Chaos, we notice the markedly redder shade of most nodes with respect to the previous tree visualisations, signalling a generally increased percentage of lost files. While Akira concentrates its attack under the “Local” folder and subfolders, Chaos’ attack seems more spread out, reaching both the user’s data (“Desktop”, “Documents”, “Pictures”, etc.) and most of the folders under “AppData”. Closing our analysis with Phobos, we mainly notice the dark-red nodes under the user profile (Desktop, etc.) that correspond to severe losses.

Format Spectrum Profiles. Figure 7 presents the format spectrum of the six ransomware types.

Vipasana/Ryuk exhibit a predominantly green spectrum in the comparison, with narrow intermittent red spikes, suggesting that the ransomware targets a tightly scoped set of extensions. A non-trivial blue component emerges in the spectrum, indicating moderate replica activity, probably used as temporary copies (.ignore files). Operationally, we can characterise these ransomware as restrained encryptors that slowly damage selected formats while preserving most of the host, thereby maintaining system usability and maximising the probability that the victim can still receive, evaluate, and pay the ransom demand.

WannaCry also remains largely green-dominant, but it differs from Vipasana/Ryuk in two important respects. First, the blue contribution is visibly stronger and more widespread, making it the most replica-heavy spectrum in the comparison. Second, the spectrum shows more pronounced disruption of formats like documents and source code, while executable and library regions remain mostly green. Overall, we can place WannaCry close to Vipasana/Ryuk in aggressiveness, but distinct in operational texture – both are selective, yet WannaCry seems noisier and more reliant on temporary replicas.

LockBit marks a clear transition from selective to broad-spectrum behaviour. Its spectrum is visually balanced between red and green, and it displays sustained red segments across images, audio/video, database, and system-oriented categories. However, it does not collapse into full red saturation. Several segments remain mixed or green-dominant, especially around code and executable/library formats, which suggests a calibrated strategy rather than blind destruction. In this sense, LockBit resembles a more aggressive version of WannaCry, which still discriminates, but with a much wider targeting scope and an attack surface beyond the narrow class of user files. Notably, LockBit also uses .ignore files, like Vipasana/Ryuk, for temporary copies.

Akira is close to LockBit in global intensity, but the category pattern is not identical. Where LockBit appears more balanced in the middle of the spectrum, Akira shows stronger red concentration in image-like, database-like, code-like, and log-oriented regions, while some document and archive segments remain mixed. The most interesting parallel is with LockBit, as both seem to belong to the same broad family of semi-selective, high-impact ransomware, although Akira appears more willing to disrupt operational and application-related artefacts.

Chaos appears at an intermediate position between LockBit/Akira and the more conservative top spectra – its spectrum is more uneven than that of LockBit – and regions like database and code/script segments are markedly dark red, indicating the presence of pristine and/or replica files.

Phobos is visually distinct from all other families in the figure and emerges as an indiscriminate encryptor. Its spectrum is overwhelmingly red and only thin traces of green or blue remain. Unlike the more selective families above it, Phobos shows red dominance across nearly every format group, including images, audio/video, archive, database, and system-related regions.

4.3. 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.

Meta-analysis. SAFARI places no intrinsic restriction on the choice of contrast tool: any solution that can be installed within a “physical” machine (and, thus, a VM) is a candidate, including kernel-level frameworks, detection-based monitors, or deception systems, provided the experimenter configures the VM template accordingly.

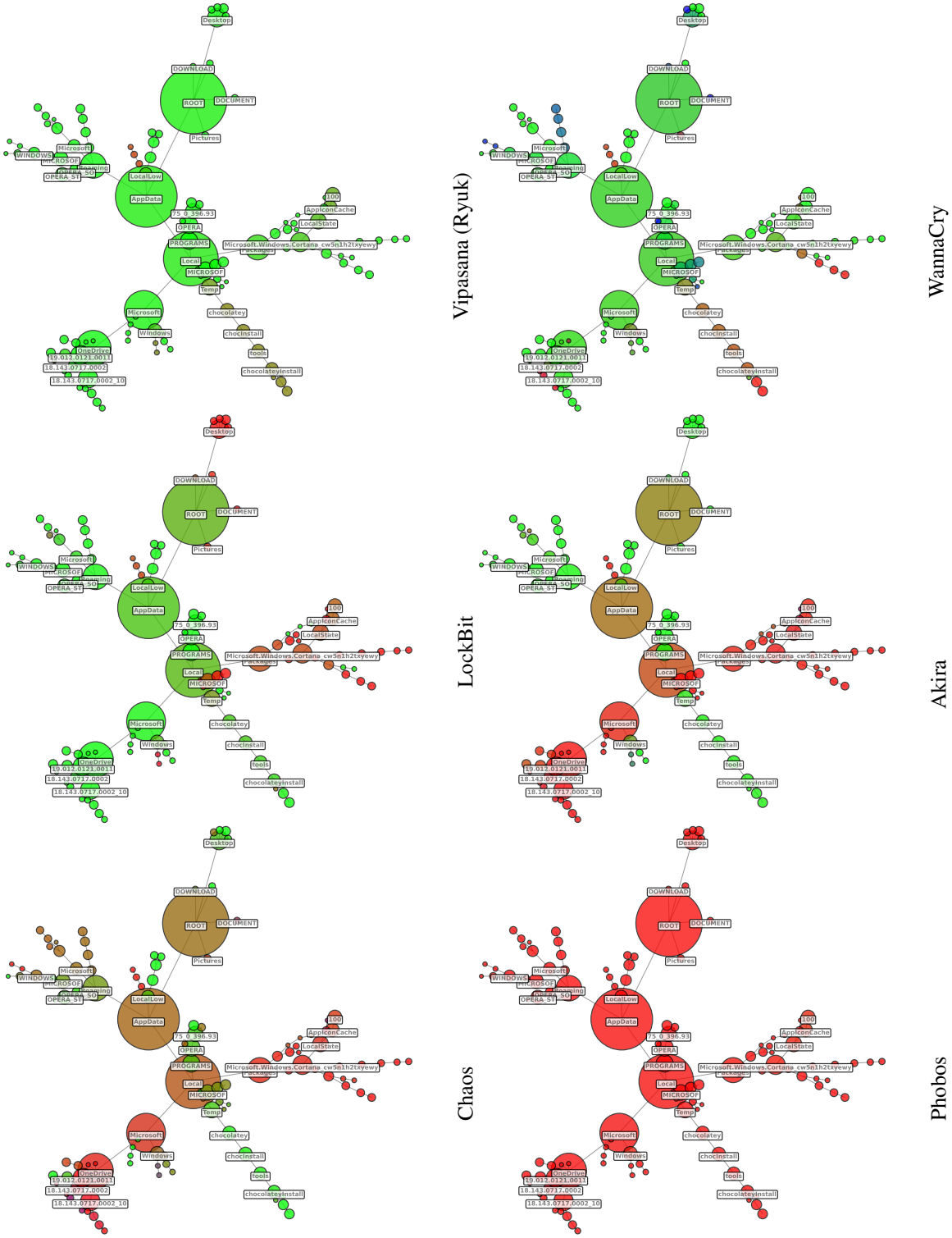

Figure 6: File-system-tree-chart visualisation of the attack profiles of Vipasana/Ryuk, WannaCry, LockBit, Akira, Chaos, and Phobos.

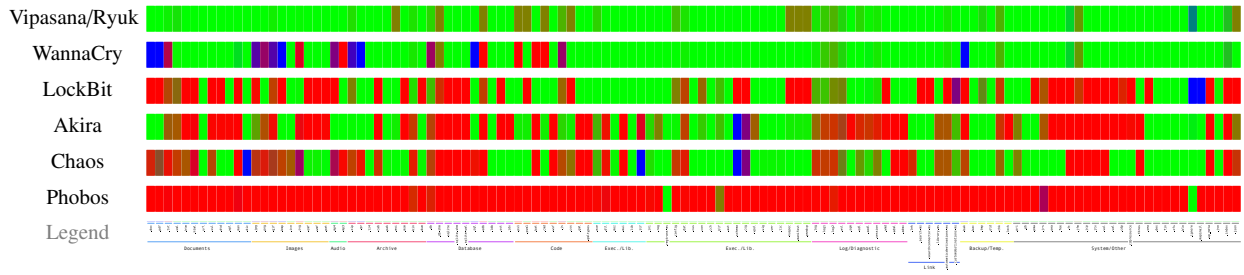

Figure 7: Spectrum visualisation of the attack profiles of Vipasana/Ryuk, WannaCry, LockBit, Akira, Chaos, and Phobos.

To choose one of these tools for our second case study, we considered several recent proposals for ransomware mitigation.

Ranflood [16, 17] implements a relatively recent mitigation paradigm – data flooding at the file system level – which mixes moving-target defence and dynamic honeypots to increase I/O contention and slow the ransomware's encryption process. Unlike detection-based approaches such as Ranker [65] and CryptoDrop [61], which act by identifying and terminating the malicious process, or kernel-level mediation frameworks such as ShieldFS [66], Redemption [67], GuardFS [68], and RansomWall [69], which protect data through write interception and rollback, Ranflood does not attempt to detect or block the ransomware and operates concurrently with it, contesting the file system. This paradigm is substantially less studied than detection or kernel-mediation approaches, and the question of its effectiveness against different real-world ransomware strains and encryption strategies is genuinely open. SAFARI's benchmarking methodology – with its quantitative, per-file and per-directory metrics of pristine, lost, and replica ratios – is precisely the kind of rigorous experimental infrastructure needed to answer it. Furthermore, Ranflood's flooding behaviour produces replica files as a structural byproduct of its copy-based modes [16], which are directly observable in the Profiler's output, yielding a particularly information-rich experimental signal that reflects not only whether the contrast tool is effective but also how it interacts with the ransomware's encryption strategy at the directory level.

Of course, SAFARI's pipeline can accommodate the other kinds of contrast tools we mentioned. Detection-based approaches and kernel-level solutions would require experimenters to prepare VM templates with the appropriate drivers, kernel modules, or event-tracing infrastructure pre-installed – a one-time configuration step that is well within SAFARI's IaC-managed provisioning model.

Experiments. Given the above considerations, in this case study, we profile Ranflood [17]. 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, requires 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 user files (cf. Figure 5), we concentrate on WannaCry, LockBit, and Phobos.

Doughnut-chart Profiles. As done in Section 4.2, we start our report with the doughnut-chart profile visualisation of 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 after starting the ransomware, launching 13 flooding instances in parallel on distinct user folders, including "Documents", "Desktop", "Music", and "Pictures", purposefully avoiding flooding the "AppData" folder which contains application files rather than user files and stopping the VM after 10 minutes.

For brevity, we show, in Figure 8, only the best-performing strategy, i.e., the one that minimises the percentage of lost files, which is Shadow. The interested reader can find all data and plots at `https://zenodo.org/records/18911874`. The plots report averages over at least 12 runs, amounting to an average standard

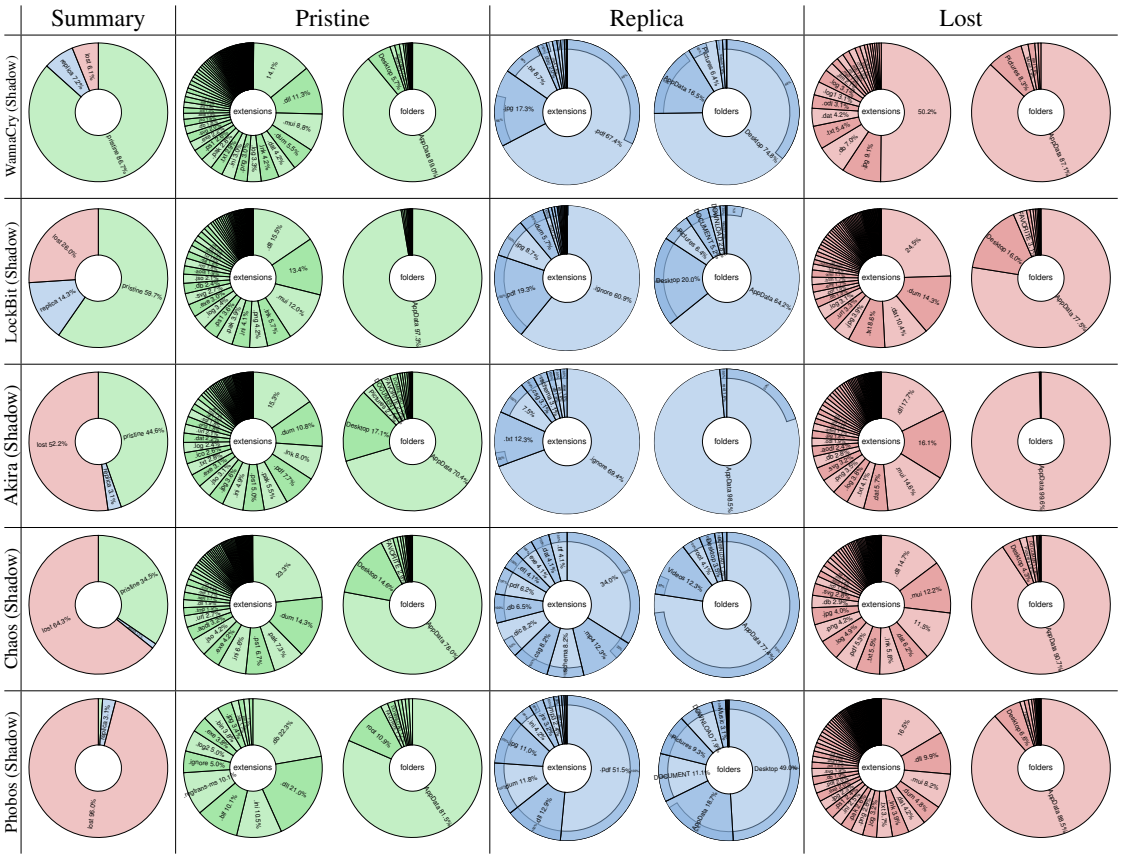

Figure 8: Doughnut-chart visualisation of Ranflood's contrast profiles against WannaCry, LockBit, Akira, Chaos, and Phobos.

deviation of ca. 20% (across pristine, lost, and replica files).

For the individual ransomware samples, 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 through I/O access contention. The breakdown of the WannaCry replicas in Figure 8 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 ransomware's attacks. Notably, the percentage of lost files for Akira and Chaos increased with respect to the figures shown in Figure 5, going from 49.8% to 52.2% and from 57% to 64.3%, respectively. While we see these slight increases as a natural consequence of variability in experiments, the fact that there is no visible contrast action performed by Ranflood hints that the tool probably did not run correctly in those experiments. We conjecture that these ransomware could prevent the correct execution of Ranflood, but, to verify that hypothesis, one would need e.g., information on the processes running during the attack. This result inspires future work on adding process logging among the metrics recorded by SAFARI's IT toolset, to collect additional data for analysing the behaviour of ransomware and mitigation tools. 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".

File-System-Tree Profiles. We conclude our analysis of the contrast profiles with the file-system-tree profile vi-

sualisation of the same configurations discussed in the previous paragraph: the Shadow-Copy 30-second delay contrast for WannaCry, LockBit, and Phobos, reported in Figure 9.

Starting from WannaCry, we note that Ranflood substantially hindered the performance of the ransomware, which generated both far fewer replicas and lost files than in its attack profile (cf. Figure 6). In particular, the main remaining replicas are user files found under the "Desktop" folder. Similarly, the main replicas found under the LockBit contrast profile appear under the user files – primarily "Desktop", where we also find the main lost user files. As mentioned for the analysis of the doughnut charts, Ranflood seems not to have run in either the Akira or Chaos experiments, as witnessed by the high similarity between the tree visualisations of both ransomware with respect to their corresponding tree plots in Figure 6. Finally, Phobos' attack is essentially mitigated through replicas, visible through the dark-red shading of the "Desktop" folder.

Format Spectrum Profiles. Figure 10 shows the format spectrum of the considered five ransomware families under active Ranflood mitigation. Reading the figure in comparison with the baseline spectrum (cf. Figure 7) reveals how Ranflood mitigates the attack profile for each family, and whether the countermeasure shifts the balance in favour of pristine files, promotes replica flooding, or leaves the spectrum essentially unchanged.

Globally, Ranflood's effect is visible as a systematic increase in the blue channel (replicas) – for the families sensitive to the countermeasure – and a corresponding redistribution of red and green that varies substantially between families.

WannaCry's profile retains an overwhelmingly green spectrum. Its global lost proportion decreases slightly while the pristine proportion rises. Unexpectedly, the blue replica channel contracts – a surprising reduction that inverts the direction one might expect from a tool that introduces additional replica files. The category-level profile clarifies this finding: across all format groups the spectra remain predominantly green with very low red and blue contributions; only in Documents does a mild blue component remain visible, consistent with Ranflood targeting its flood activity towards common document formats. The most likely explanation is that WannaCry already targets so few extensions in the unmitigated scenario that Ranflood's action falls predominantly outside the ransomware's selective extension list – the ransomware either skips decoys or encrypts only a marginal fraction thereof, producing neither a red nor a blue signal above the baseline.

LockBit shows the clearest evidence of Ranflood effectiveness in the comparison: the global lost proportion falls substantially while the pristine proportion rises. The blue replica channel more than doubles, indicating that a sizable fraction of the files in the spectrum are now Ranflood-generated replicas that have resisted encryption. In particular, in the unmitigated baseline (Figure 7), Documents and Database files are red-dominant, whereas in the mitigated spectrum they shift to blue and mixed-green, respectively. Images, Archives, and Code/Scripts also transition from partial red to predominantly green, with blue contributions visible throughout. By contrast, the remaining categories – Executable/Library, Log/Diagnostic, and Backup/Temp – are similar in the two spectra, confirming that Ranflood's contribution is focused precisely where LockBit was most destructive.

Akira's spectrum is almost indistinguishable from its baseline counterpart. This observation further solidifies our conjecture that Ranflood did not run properly to mitigate this ransomware's attacks. Indeed, if Ranflood were successfully diverting the ransomware's encryption load onto decoy files, pristine files should be better preserved. Moreover, the unmitigated and mitigated spectra are structurally similar in their category-level patterning, whereas we would have expected a shift comparable to that observed for LockBit – e.g., in Document files.

The observation is similar for Chaos. In particular, we note some minor changes towards the red end of the spectrum, indicating a marginally more aggressive attack stance; these are most likely attributable to natural experimental noise rather than any counterproductive mitigation action.

While Phobos remains the most aggressive among the considered ransomware samples, the new blue stripes present in the mitigated spectrum confirm Ranflood's counteraction. Nevertheless, Ranflood's replicas constitute a limited fraction of the total targeted file set, suggesting that the ransomware's encryption speed outpaces the tool's flooding rate.

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-

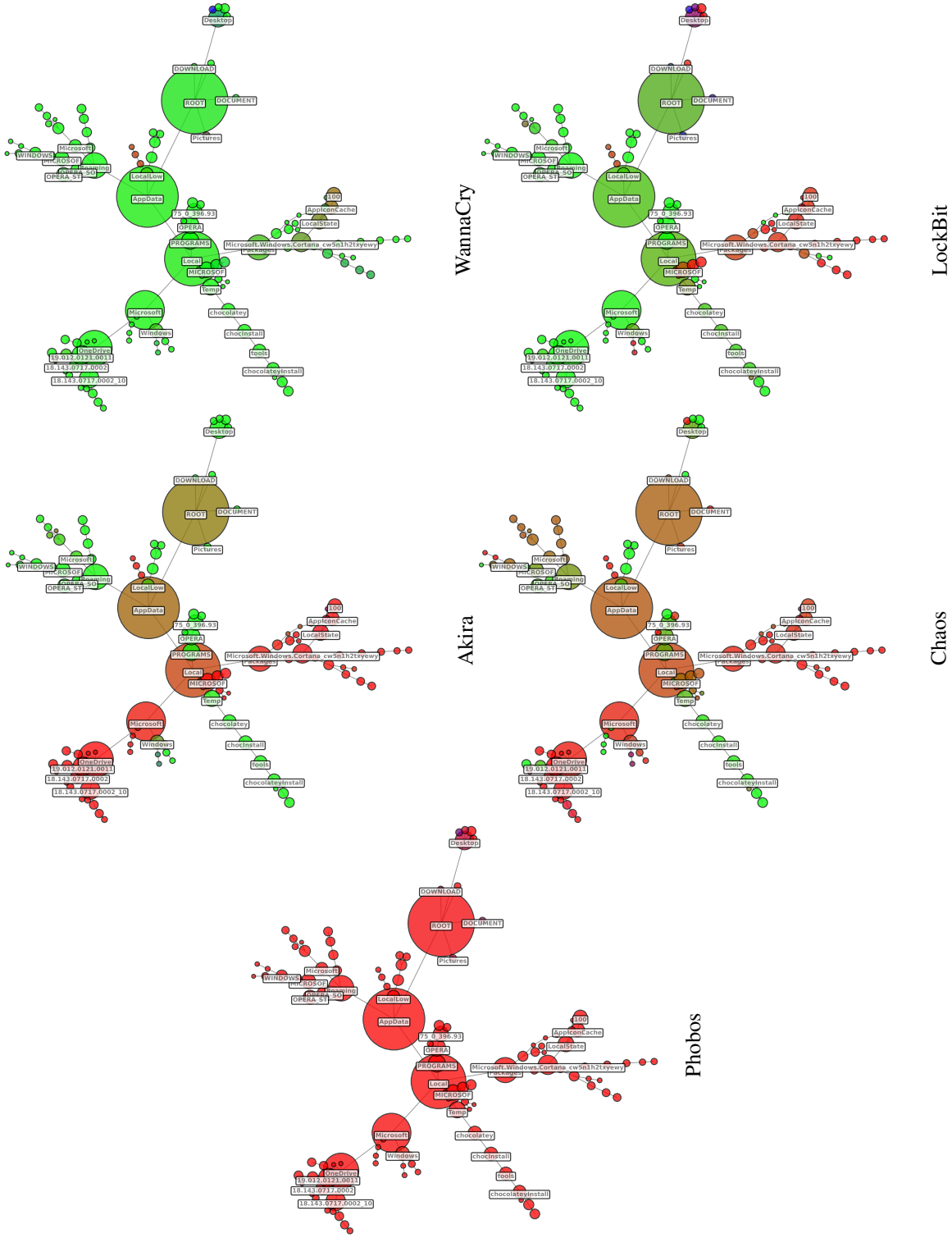

Figure 9: File-system-tree-chart visualisation of Ranflood's contrast profiles against WannaCry, LockBit, Akira, Chaos, and Phobos.

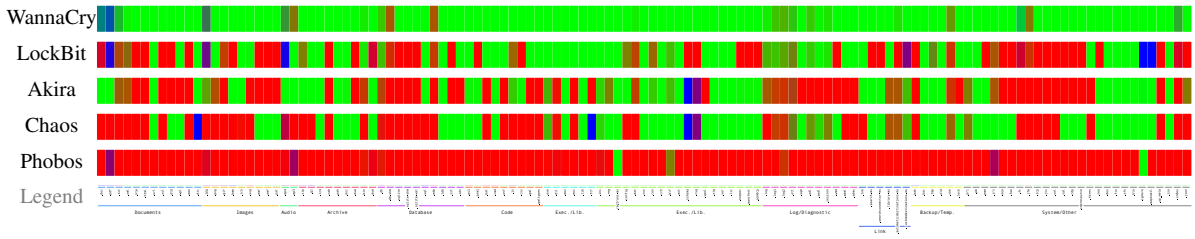

Figure 10: Spectrum visualisation of Ranflood's contrast profiles against WannaCry, LockBit, Akira, Chaos, and Phobos.

source framework that automates ransomware (and countermeasure) testing in realistic environments. Built with modern technologies, it integrates Infrastructure-as-Code and OS-agnostic Task Automation for reproducible and consistent experiment setups across diverse hardware. We designed SAFARI's architecture after a comprehensive evaluation of the available models, platforms, and tools that could be used to implement its conceptual building blocks, choosing what we consider the most suitable options, and developing new components where needed. In particular, we devoted a significant effort to devising an effective interface for the visualisation of results collected in tests, enabling operators to quickly access both the overall key indicators and the underlying details. 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 seven renowned ransomware strains (Ryuk, Vipasana, WannaCry, LockBit, Akira, Chaos, and Phobos) and evaluating the effectiveness of a ransomware mitigation tool (Ranflood) against five 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-critical files such as those in the "Desktop", "Documents", and "Pictures" folders. Akira, Chaos, and Phobos appear to be the most aggressive strains, encrypting most of the 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 [14], 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.

A further direction concerns the observability of running processes during experiments. As witnessed by the Akira and Chaos mitigation results, Ranflood appears not to have executed correctly in those scenarios, yet SAFARI currently provides no mechanism to investigate what happens at the process level during an attack – which executables are active, how they interact, and what inter-process relationships emerge at run time. Addressing this gap would allow investigators to verify whether a countermeasure launched as intended, to diagnose interference between the ransomware and the contrast tool, and to collect richer behavioural evidence beyond the file-system footprint. Thus, we plan to extend SAFARI's Investigation Tools layer with process-level monitoring capabilities – e.g., via lightweight tracing of system calls or process-creation events – retaining the non-interfering, static analysis philosophy that underpins the current design.

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] A. Liska, T. Gallo, Ransomware: Defending Against Digital Extortion, 1st Edition, O'Reilly

Media, Inc., 2016.

[2] R. Richardson, M. North, Ransomware: Evolution, mitigation and prevention, International Management Review 13 (1) (2017) 10–21,101. URL https://www.proquest.com/scholarly-journals/ransomware-evolution-mitigation-prevention/docview/1881414570/se-2

[3] H. Oz, A. Aris, A. Levi, A. S. Uluagac, A survey on ransomware: Evolution, taxonomy, and defense solutions, ACM Comput. Surv. 54 (11s) (Sep. 2022). doi:10.1145/3514229.

[4] J. Hernandez-Castro, A. Cartwright, E. Cartwright, An economic analysis of ransomware and its welfare consequences, RSOS 7 (3) (2020) 190023. doi:10.1098/rsos.190023.

[5] T. Meurs, M. Junger, E. Tews, A. Abhishta, Ransomware: How attacker's effort, victim characteristics and context influence ransom requested, payment and financial loss, in: APWG eCrime, IEEE, 2022, pp. 1–13. doi:10.1109/ECRIME57793.2022.10142138.

[6] P. H. Meland, Y. F. F. Bayoumy, G. Sindre, The ransomware-as-a-service economy within the darknet, Comput. Secur. 92 (2020) 101762. doi:10.1016/J.COSE.2020.101762.

[7] A. Y. Connolly, H. Borrion, Reducing ransomware crime: Analysis of victims' payment decisions, Comput. Secur. 119 (2022) 102760. doi:10.1016/J.COSE.2022.102760.

[8] Chainalysis Team, Ransomware payments exceed $1 billion in 2023, hitting record high after 2022 decline, https://www.chainalysis.com/blog/ransomware-2024/, [Online; accessed Apr. 2026] (Feb. 2024).

[9] Chainalysis Team, 2024 crypto crime mid-year update part 1: Cybercrime climbs as stolen funds surge, https://www.chainalysis.com/blog/2024-crypto-crime-mid-year-update-part-1/, [Online; accessed Apr. 2026] (Aug. 2024).

[10] Federal Bureau of Investigation, 2025 IC3 annual report, https://www.ic3.gov/AnnualReport/Reports/2025_IC3Report.pdf, [Online; accessed Apr. 2026] (2026).

[11] B. A. S. Al-rimy, M. A. Maarof, S. Z. M. Shaid, Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions, Computers & Security 74 (2018) 144–166. doi:10.1016/j.cose.2018.01.001.

[12] T. McIntosh, et al., Ransomware mitigation in the modern era: A comprehensive review, research challenges, and future directions, ACM Comput. Surv. 54 (9) (Oct. 2021). doi:10.1145/3479393.

[13] C. Beaman, A. Barkworth, T. D. Akande, S. Hakak, M. K. Khan, Ransomware: Recent advances, analysis, challenges and future research directions, Computers & Security 111 (2021) 102490. doi:10.1016/j.cose.2021.102490.

[14] T. McIntosh, T. Susnjak, T. Liu, D. Xu, P. Watters, D. Liu, Y. Hao, A. Ng, M. Halgamuge, Ransomware reloaded: Re-examining its trend, research and mitigation in the era of data exfiltration, ACM Comput. Surv. (Aug. 2024). doi:10.1145/3691340.

[15] A. Dey, E. Totel, B. Costé, DAEMON: dynamic auto-encoders for contextualised anomaly detection applied to security monitoring, in: W. Meng, S. Fischer-Hübner, C. D. Jensen (Eds.), ICT Systems Security and Privacy Protection - 37th IFIP TC 11 International Conference, SEC 2022, Copenhagen, Denmark, June 13-15, 2022, Proceedings, IFIP Advances in Information and Communication Technology, Springer, 2022, pp. 53–69. doi:10.1007/978-3-031-06975-8_4.

[16] D. Berardi, S. Giallorenzo, A. Melis, S. Melloni, L. Onori, M. Prandini, Data flooding against ransomware: Concepts and implementations, Computers & Security (2023) 103295doi:10.1016/j.cose.2023.103295.

[17] D. Berardi, S. Giallorenzo, A. Melis, S. Melloni, M. Prandini, Ranflood: A mitigation tool based on the principles of data flooding against ransomware, SoftwareX 25 (2024) 101605. doi:10.1016/j.softx.2023.101605.

[18] T. Compagnucci, F. Callegati, S. Giallorenzo, A. Melis, S. Melloni, A. Vannini, SAFARI: A scalable air-gapped framework for automated ransomware investigation, in: L. N. Zlatolas, K. Rannenberg, T. Welzer, J. García-Alfaro (Eds.), ICT Systems Security and Privacy Protection - 40th IFIP International Conference, SEC 2025, Maribor, Slovenia, May 21-23, 2025, Proceedings, Part

I, Vol. 745 of IFIP Advances in Information and Communication Technology, Springer, 2025, pp. 210–223. doi:10.1007/978-3-031-92882-6_15.

[19] A. De Benedictis, F. Flammini, N. Mazzocca, A. Somma, F. Vitale, Digital twins for anomaly detection in the industrial internet of things: Conceptual architecture and proof-of-concept, IEEE TII 19 (12) (2023) 11553–11563. doi:10.1109/TII.2023.3246983.

[20] M. Masi, G. P. Sellitto, H. Aranha, T. Pavleska, Securing critical infrastructures with a cybersecurity digital twin, Software and Systems Modeling 22 (2) (2023) 689–707.

[21] H. Jiang, et al., Pandora: A cyber range environment for the safe testing and deployment of autonomous cyber attack tools, in: SSCC, Springer, 2021, pp. 1–20.

[22] N. d'Ambrosio, G. Capodagli, G. Perrone, S. P. Romano, Scass: Breaking into scada systems security, Comp. Sec. 151 (2025) 104315. doi:10.1016/j.cose.2025.104315.

[23] N. K. Kandasamy, S. Venugopalan, T. K. Wong, N. J. Leu, An electric power digital twin for cyber security testing, research and education, Computers and Electrical Engineering 101 (2022) 108061. doi:10.1016/j.compeleceng.2022.108061.

[24] A. Pokhrel, V. Katta, R. Colomo-Palacios, Digital twin for cybersecurity incident prediction: A multivocal literature review, ICSEW'20, ACM, New York, NY, USA, 2020, pp. 671–678. doi:10.1145/3387940.3392199.

[25] R.-V. Mahmoud, M. Anagnostopoulos, J. M. Pedersen, Detecting cyber attacks through measurements: Learnings from a cyber range, IEEE IMM 25 (6) (2022) 31–36. doi:10.1109/MIM.2022.9847127.

[26] A. Kharraz, S. Arshad, C. Mulliner, W. K. Robertson, E. Kirda, UNVEIL: A large-scale, automated approach to detecting ransomware, in: T. Holz, S. Savage (Eds.), USENIX Security Symposium, USENIX Association, 2016, pp. 757–772.

[27] C. J. W. Chew, V. Kumar, P. Patros, R. Malik, Real-time system call-based ransomware detection, International Journal of Information Security 23 (2024) 1839–1858. doi:10.1007/s10207-024-00819-x.

[28] S. Gulmez, A. Gorgulu Kakisim, I. Sogukpinar, Xran: Explainable deep learning-based ransomware detection using dynamic analysis, Computers & Security 138 (2024) 103703. doi:10.1016/j.cose.2024.103703.

[29] C. C. Moreira, D. C. Moreira, C. Sales, A comprehensive analysis combining structural features for detection of new ransomware families, Journal of Information Security and Applications 81 (2024) 103716. doi:10.1016/j.jisa.2024.103716.

[30] D. Li, W. Shi, N. Lu, S.-S. Lee, S. Lee, ARdetector: android ransomware detection framework, The Journal of Supercomputing 80 (2024) 7557–7584. doi:10.1007/s11227-023-05741-y.

[31] A. A. Ahmed, A. Shaahid, F. Alnasser, S. Alfaddagh, S. Binagag, D. Alqahtani, Android ransomware detection using supervised machine learning techniques based on traffic analysis, Sensors 24 (1) (2024) 189. doi:10.3390/s24010189.

[32] F. Callegati, S. Giallorenzo, A. Melis, S. Melloni, M. Prandini, A. Vannini, Investigating operational technology attacks as code, Empirical Software Engineering 30 (6) 158. doi:10.1007/s10664-025-10713-2.

[33] MITRE Corporation, MITRE Caldera: Automated adversary emulation platform, https://caldera.mitre.org, accessed: March 2026 (2024).

[34] HashiCorp, Terraform documentation, https://developer.hashicorp.com/terraform/docs, accessed: March 2026 (2024).

[35] The OpenTofu Project, Opentofu documentation, https://opentofu.org/docs/, accessed: March 2026 (2024).

[36] Pulumi Corporation, Pulumi documentation, https://www.pulumi.com/docs/, accessed: March 2026 (2024).

[37] Amazon Web Services, Aws cloudformation user guide, https://docs.aws.amazon.com/cloudformation/, accessed: March 2026 (2024).

[38] Amazon Web Services, Aws cloud development kit (cdk) documentation, https://docs.aws.amazon.com/cdk/, accessed: March 2026 (2024).

[39] Microsoft, Azure resource manager documentation, https://learn.microsoft.com/azure/azure-resource-manager/, accessed: March 2026 (2024).

[40] Microsoft, Bicep documentation, https://learn.microsoft.com/azure/azure-resource-manager/bicep/, accessed: March 2026 (2024).

[41] The Crossplane Authors, Crossplane documentation, https://docs.crossplane.io/, accessed: March 2026 (2024).

[42] T. Delaet, W. Joosen, B. V. Brabant, A survey of system configuration tools, in: R. van Drunen (Ed.), Uncovering the Secrets of System Administration: Proceedings of the 24th Large Installation System Administration Conference, LISA 2010, San Jose, CA, USA, November 7-12, 2010, USENIX Association, 2010.
URL https://www.usenix.org/conference/lisa10/survey-system-configuration-tools

[43] VMware, Inc., Salt Project: Event-driven automation and configuration management, https://docs.saltproject.io, accessed: March 2026 (2024).

[44] Puppet, Inc., Puppet infrastructure automation documentation, https://www.puppet.com/docs, accessed: March 2026 (2024).

[45] Progress Chef, Chef Infra: Infrastructure automation documentation, https://docs.chef.io, accessed: March 2026 (2024).

[46] J. Stauffer, Fabric: Simple, Pythonic remote execution and deployment, https://www.fabfile.org, accessed: March 2026 (2024).

[47] Red Hat, Inc., Ansible collaborative documentation, https://www.redhat.com/en/ansible-collaborative, accessed: March 2026 (2024).

[48] S. R. Davies, R. Macfarlane, W. J. Buchanan, Evaluation of live forensic techniques in ransomware attack mitigation, Forensic Science International: Digital Investigation 33 (2020) 300979. doi:https://doi.org/10.1016/j.fsidi.2020.300979.

[49] P. Bajpai, R. Enbody, Memory forensics against ransomware, in: 2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), 2020, pp. 1–8. doi:10.1109/CyberSecurity49315.2020.9138853.

[50] T. R. McIntosh, J. Jang-Jaccard, P. A. Watters, Large scale behavioral analysis of ransomware attacks, in: L. Cheng, A. C. S. Leung, S. Ozawa (Eds.), Neural Information Processing, Springer International Publishing, Cham, 2018, pp. 217–229.

[51] A. Kharraz, W. Robertson, D. Balzarotti, L. Bilge, E. Kirda, Cutting the gordian knot: A look under the hood of ransomware attacks, in: M. Almgren, V. Gulisano, F. Maggi (Eds.), Detection of Intrusions and Malware, and Vulnerability Assessment, Springer International Publishing, Cham, 2015, pp. 3–24.

[52] J. Dafoe, N. Chen, B. Chen, Z. Wang, Enabling per-file data recovery from ransomware attacks via file system forensics and flash translation layer data extraction, Cybersecurity 7 (1) (2024) 75. doi:10.1186/s42400-024-00287-9.
URL https://doi.org/10.1186/s42400-024-00287-9

[53] P. K. Manadhata, J. M. Wing, An attack surface metric, IEEE Trans. Software Eng. 37 (3) (2011) 371–386. doi:10.1109/TSE.2010.60.

[54] C. Theisen, H. Sohn, D. Tripp, L. A. Williams, BP: profiling vulnerabilities on the attack surface, in: 2018 IEEE Cybersecurity Development, SecDev 2018, Cambridge, MA, USA, September 30 - October 2, 2018, IEEE Computer Society, 2018, pp. 110–119. doi:10.1109/SECDEV.2018.00022.

[55] T. M. J. Fruchterman, E. M. Reingold, Graph drawing by force-directed placement, Softw. Pract. Exp. 21 (11) (1991) 1129–1164. doi:10.1002/SPE.4380211102.

[56] A. A. Hagberg, D. A. Schult, P. J. Swart, Exploring network structure, dynamics, and function using networkx, Python in Science Conference (2008). doi:10.25080/TCWV9851.

[57] OpenWrt Team, Openwrt, https://openwrt.org/, [Online; accessed Sept. 2024].

[58] H. Oz, A. Aris, A. Levi, A. S. Uluagac, A survey on ransomware: Evolution, taxonomy, and defense solutions, ACM Comput. Surv. 54 (11s) (Sep. 2022). doi:10.1145/3514229.

[59] A. Maigida, S. Abdulhamid, M. Olalere, K. Alhassan, H. Chiroma, E. Dada, Systematic literature review and metadata analysis of ransomware attacks and detection mechanisms, Journal of Reliable Intelligent Environments 5 (07 2019). doi:10.1007/s40860-019-00080-3.

[60] M. Halsey, Windows 10 File Structure in Depth, Apress, Berkeley, CA, 2016, pp. 449–457. doi:10.1007/978-1-4842-0925-7_27.

[61] N. Scaife, H. Carter, P. Traynor, K. R. B. Butler, Cryptolock (and drop it): Stopping ransomware attacks on user data, in: IEEE ICDCS, 2016, pp. 303–312. doi:10.1109/ICDCS.2016.46.

[62] Kaspersky, Ransomware attacks and types (2021).

[63] Kroll, Data exfiltration ransomware attacks (2021).

[64] C. Rossow, C. J. Dietrich, C. Grier, C. Kreibich, V. Paxson, N. Pohlmann, H. Bos, M. Van Steen, Prudent practices for designing malware experiments: Status quo and outlook, in: IEEE S&P, IEEE, 2012, pp. 65–79.

[65] H. Zhang, L. Zhao, A. Yu, L. Cai, D. Meng, Ranker: Early ransomware detection through kernel-level behavioral analysis, IEEE Transactions on Information Forensics and Security 19 (2024) 6113–6127. doi:10.1109/TIFS.2024.3410511.

[66] A. Continella, A. Guagnelli, G. Zingaro, G. De Pasquale, A. Barenghi, S. Zanero, F. Maggi, Shieldfs: a self-healing, ransomware-aware filesystem, in: Proceedings of the 32nd Annual Conference on Computer Security Applications, ACSAC '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 336–347. doi:10.1145/2991079.2991110.
URL https://doi.org/10.1145/2991079.2991110

[67] A. Kharraz, E. Kirda, Redemption: Real-time protection against ransomware at end-hosts, in: M. Dacier, M. Bailey, M. Polychronakis, M. Antonakakis (Eds.), Research in Attacks, Intrusions, and Defenses, Springer International Publishing, Cham, 2017, pp. 98–119.

[68] J. von der Assen, C. Feng, A. Huertas Celdrán, R. Oleš, G. Bovet, B. Stiller, Guardfs: A file system for integrated detection and mitigation of linux-based ransomware, Journal of Information Security and Applications 93 (2025) 104078. doi:https://doi.org/10.1016/j.jisa.2025.104078.
URL https://www.sciencedirect.com/science/article/pii/S2214212625001152

[69] S. K. Shaukat, V. J. Ribeiro, Ransomwall: A layered defense system against cryptographic ransomware attacks using machine learning, in: 2018 10th International Conference on Communication Systems & Networks (COMSNETS), 2018, pp. 356–363. doi:10.1109/COMSNETS.2018.8328219.