Received 13 March 2024, accepted 4 May 2024, date of publication 13 May 2024, date of current version 21 May 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3400167

RESEARCH ARTICLE

Redefining Malware Sandboxing: Enhancing Analysis Through Sysmon and ELK Integration

RASMI-VLAD MAHMOUD[1], MARIOS ANAGNOSTOPOULOS[1], SERGIO PASTRANA[2], AND JENS MYRUP PEDERSEN[1]

[1]Cyber Security Group, CMI Section, Department of Electronic Systems, Aalborg University, 2450 Copenhagen, Denmark
[2]Computer Security (COSEC) Lab, Department of Computer Science, Escuela Politécnica Superior, Universidad Carlos III de Madrid, 28911 Leganes, Spain

Corresponding author: Marios Anagnostopoulos (mariosa@es.aau.dk)

This work was supported by OTTO MØNSTEDS FOND.

ABSTRACT In cybersecurity, adversaries employ a myriad of tactics to evade detection and breach defenses. Malware remains a formidable weapon in their arsenal. To counter this threat, researchers unceasingly pursue dynamic analysis, which aims to comprehend and thwart established malware strains. This paper introduces an innovative methodology for dynamic malware analysis while critically evaluating prevailing technologies and their limitations. The proposed approach hinges on harnessing the capabilities of an open-source Security Information and Event Management (SIEM) toolset, namely the Elastic-Stack. This toolset is utilized to capture, structure, and analyze the behavioral patterns of malware without relying on any pre-existing sandbox framework. This augmentation facilitates a profound understanding of the activities exhibited by malware samples. With the help of the proposed ecosystem, we compile the *AAU_MalData* dataset that encompasses distinctly the benign and malicious behavior. Specifically, we analyzed the behavior of the 2,800 malware within a realistic network topology and systematically collected Windows event logs, which serve as a comprehensive record of the malware's actions. These event logs are precious as they are organized in a timestamped format, providing a chronological list of system activities, such as event descriptions, process and file details, and registry modifications. These are pivotal in comprehending the malware's functionality and repercussions on the compromised system. Furthermore, by incorporating the MITRE ATT&CK framework, we leveraged the event logs to correlate the malware's mode of operation, delve into its Command and Control operations, and investigate its persistence mechanisms, enabling a structured and practical approach to malware analysis. The AAU_MalData dataset, organized in a JSON format data structure, is offered to the research community first as a proof of concept to demonstrate the toolset's feasibility for dynamic malware analysis and second as a potential training ground for anti-malware mechanisms based on host and network Indicators of Compromise (IoCs).

INDEX TERMS Dynamic malware analysis, sandbox, Elasticsearch, MITRE ATT&CK, Sysmon.

I. INTRODUCTION

Contemporary cyberattacks exhibit advanced sophistication, employing diverse techniques and methodologies to achieve their desired impact [1]. Among the arsenal of tools leveraged in such attacks, malware is a widely employed method strategically designed to disrupt the target's infrastructure and engage in covert surveillance. Notably, these malicious software are not limited to specific platforms or operating systems (OS), as they are developed to operate on multiple environments [2].

The pervasiveness of malware attacks and the proliferation of malicious software is exhibiting a substantial surge, demonstrating a notable increase of 22.9% in recent years [3].

The associate editor coordinating the review of this manuscript and approving it for publication was Kashif Saleem.

Previous research identified these trends as alarming threats that pose significant risks to computer users. The scale of the issue is exemplified by the astonishing amount of approximately one billion infected files reported in January 2021 [4], and even worse, forecasting a further escalation in the following years.

It is essential to acknowledge that malicious software not only threatens individual end-users but also extends to diverse entities, including governmental organizations, businesses across various sectors, Fortune 500 companies, financial institutions, industrial systems, medical and healthcare services, educational institutions, and even law enforcement agencies [5]. Thus, the detrimental consequences of malware attacks are not limited exclusively to direct financial losses, such as those resulting from ransomware incidents. They can also significantly impact a company's brand reputation and customer base or even the security of critical infrastructures, exacerbating the overall consequences affected entities and society face.

Given the urgency and criticality of the issue, there is a pressing need to proactively detect and mitigate such malicious attempts and comprehend their mode of operation before they can carry out their intended harm in an operational system [6]. Consequently, thorough analysis processes of malware samples must be pursued to understand their characteristics and functionalities. Three primary analysis approaches, namely static, dynamic, and hybrid methods, are commonly employed in this domain. Furthermore, it is imperative to investigate the continuous evolution of evasion tactics employed by malware authors, as extensively elucidated by [1]. These tactics can be classified into detection-dependent and detection-independent evasions.

Detection-dependent tactics involve techniques like fingerprinting, where various environmental indicators are scrutinized to detect virtual or emulated host environments. Reverse Turing Tests analyze user behavior to differentiate human-machine interactions from automated processes. Targeted approaches aim to identify specific environmental conditions required for malware activation. On the other hand, detection-independent evasion strategies aim to evade analysis environment detection by employing techniques agnostic to the target environment, such as stalling malware activities with time delays. Triggered-based techniques activate malicious behavior based on specific inputs from keyboards, system triggers, or network events [1]. Nonetheless, malware samples have also been developed to execute in virtual environments since production services are deployed there for efficiency, easy scalability, and ease of management.

While identifying known malware poses relatively trivial challenges, the main obstacle lies in accurately determining the nature and characteristics of zero-day malware. Typically, expert analysts play a pivotal role in assessing the nature of executable files. Analysts calculate signatures that investigation tools can employ if deemed malicious to facilitate future recognition and detection of similar malware specimens [6].

Our objective is twofold. Firstly, we aim to introduce a dynamic malware analysis framework that can bypass malware evasion techniques using solely Virtual Machines (VMs) without relying on existing sandbox technologies. Secondly, we propose to organize the generated information into a dataset that helps to advance cybersecurity research. Specifically, the proposed framework facilitates the analysts to gather host and network data over an extended period within a simulated, realistic enterprise environment. The system leverages Sysmon, Elasticsearch, and Kibana for data collection, storage, and visualization and offers substantial automation for analyzing malware samples. We acknowledge that ELK is a recommended tool for manual malware analysis [7]. However, it does not provide mechanisms to automatically execute numerous malware samples in parallel and collect the relevant data for further analysis, which is the key requirement in our research. Thus, we extend the ELK functionality and transform it into a framework tailored for dynamic malware analysis on a large scale.

In detail, Sysmon event logs offer detailed information concerning system and process behavior. They provide valuable insights into various Indicators of Compromise (IoCs), including processes, files, and registries, thereby enabling the analysis of operations' interdependencies and subsequent child processes. Furthermore, the framework enables the integration of MITRE ATT&CK techniques during the data collection phase, enhancing the information in these logs and facilitating comprehensive exploration and analysis of the security threats and attack vectors. Finally, to demonstrate the feasibility of the proposed framework for dynamic malware analysis, we compile and offer the research community the *AAU_MalData* dataset that includes distinctly benign and malicious behavior. To achieve that, we record the behavior of 2,800 malware for 25 minutes each within a realistic network topology and capture the events representative of their activities. The dataset is organized in a compact data structure, namely JSON format. Such datasets can be helpful to the security research community, for instance, for training anti-malware mechanisms based on host and network evidence.

In brief, the contributions of the paper at hand are summarized as:

- A dynamic malware analysis framework based solely on a virtualization environment to effectively bypass malware evasion techniques without relying on existing sandbox technologies.
- Utilization of Sysmon, Elasticsearch, and Kibana for data collection, storage, and visualization of host and network evidence.
- Leveraging Sysmon event logs to acquire detailed and meaningful insights into system and process behavior, facilitating the analysis of different events and interdependencies among operations.
- Enabling the integration of MITRE ATT&CK framework during data collection, thus enhancing the

information in the logs and supporting comprehensive exploration and analysis of security threats and attack vectors.
- Analysis of 2,800 malware with the help of the proposed framework enables the collection of host and network data over an extended period, resulting in the AAU_MalData dataset representing benign and malicious behavior. This dataset is offered publicly to the security research community to contribute to the anti-malware research.

The remaining sections of the paper are organized as follows: Section II provides an overview of the current state of the art. In contrast, Section III outlines the study's framework and the tools utilized. Details about the structure and collection of the AAU_MalData dataset are provided in Section IV, which also presents some findings. Section V explains how malware actions can be correlated using the MITRE ATT&CK framework. Finally, Section VI discusses the limitations of the approach, and the paper concludes with a summary of essential points in Section VII.

II. MALWARE ANALYSIS

Two significant types of malware analysis exist: *static analysis*, where the malware's code is reviewed without being executed, and *dynamic analysis*, which essentially involves executing the malware's code or scripts while monitoring its actions and behavior for subsequent analysis.

Static analysis, being practical, comes at a cost that is not scalable and cannot be applied to every sample. As a solution, automatic tools for malware analysis have been developed that perform different techniques, and at the end, a report is produced with the action taken by the malware. Typically, the report contains a heuristic classification concerning the benign or malicious nature of the sample [6]. Although analysis techniques have evolved drastically to detect advanced attacks and evasion techniques over the years, the malware authors are thriving in constructing their software to evade and trick detection mechanisms.

Dynamic analysis aims to discover the malicious activity without compromising the security platform [6]. In contrast to static analysis, dynamic analysis does not rely on analyzing source code and focuses on meaningful behavioral patterns. Therefore, it is not prone to standard evasive techniques, such as packing or obfuscation. However, it is liable to malware hiding or not fully performing its action if the malware detects that it is running inside an analysis environment. Analyzing the malware behavior in a controlled environment allows monitoring the program's operation on run-time and acquiring the process and the resources used or accessed by the malware during its execution, like files or registry keys. Such a controlled environment can be VMs, simulators, emulators, or isolated machines (sandboxes) equipped with the necessary monitoring tools. Various dynamic analysis techniques include function call tracking, function parameter measuring, and information flow tracking.

Several online or on-premises automated tools exist that can produce an in-depth analysis of malware behavior and its actions. Most existing tools and frameworks can be grouped within two categories: inside-the-box and out-the-box [8]. **Inside-the-Box** or Sandbox, refers to a malware analysis methodology that relies on utilizing pre-installed tools within the analysis host or establishing communication between the infected tools framework and a virtual network. On the other hand, **Out-the-box** refers to a new guest OS where the analysis tools are installed and the environment is separated correctly. For instance, Virtual Machine Introspection (VMI) is an out-the-box type of analysis, where the suspicious binary is run on a VM and the analysis in another one or hypervisor [8]. VMI can monitor multiple VMs simultaneously in real-time and facilitates isolation. Therefore, kernels are not exposed, as in the case of the in-the-box approach.

In more detail, a sandbox environment is specifically engineered for the execution of unknown files, providing a secure way to assess a malware's functionality without posing any potential threat to the attached computing system. Such environments predominantly comprise virtualized replicas of authentic systems meticulously designed to mimic or accurately replicate the characteristics of the systems they emulate. This ensures that the examined files undergo execution within a controlled and protected setting. The sandbox platforms play an essential role in providing an isolated environment where the behavior of executable files can be closely examined, and data related to their post-execution activities can be systematically documented and subjected to an in-depth analysis [9].

Cuckoo sandbox [10] is a popular inside-the-box technology for analysts and researchers in the industry to perform dynamic malware studies. Cuckoo creates a virtual environment using different virtualization technologies, like Virtualbox, VMWare, or Xen, to collect malware behavior data. Cuckoo operates based on an analysis manager responsible for submitting the malware samples and collecting the reports, a virtual network, and an analysis guest system that will be infected with the malware. The analysis process is automated through the agent that is installed on the guest system. The Cuckoo host interacts with the guest machines via a virtual network, while the users can submit and control samples via the GUI or command line through the manager. Cuckoo's drawback is represented by the in-guest agent (python agent) that needs to be installed on the analysis VM, since it can be fingerprinted by the malware, which limits the effectiveness of the analysis framework [1].

The most representative tool for VMI is Drakvuf [11], capable of tracing Kernel and user-level malware by observing process execution, file operations, and system calls as kernel functions at the hypervisor level. Drakvuf injection technique allows it to monitor the instructions written into the VM memory at the location of interest and, therefore, can detect kernel rootkits and limits malware capability to perform evasive techniques. Drakvuf has excellent potential as a modern and open-source project maintained by the CERT of Poland. However, it cannot collect network data from the environment where the malware is analyzed.

As it is evident from the previous discussion, there are multiple types of platforms [12] that can provide insights into how malware samples are operating by monitoring their behavior, either as they are attached to the analysis framework or by binding them to the memory of the device. However, some of these tools tend to be cumbersome in their configuration, or they have a high subscription cost. Furthermore, [13] underscores the significance of privacy considerations, highlighting that not all samples can be submitted to cloud-based analysis frameworks. This limitation is particularly consistent in the case of malware samples, where the sensitive nature of the data and the potential risks associated with cloud-based analysis further restrict their submission.

Furthermore, the virtual environments where malware is analyzed differ from the real ones, so malicious code can detect that it is running in an analysis platform and limit its activity or not be triggered at all. For instance, the malware may probe for an active Internet connection or remain dormant and avoid conducting its malicious purposes for a period of time after the infection to stall the analysis process. It should also be noted that sandbox technologies are susceptible to evasion techniques due to the potential presence of artifacts from the analysis tools or other virtualized components. Frequently, these environments may also incorporate debugging tools, which are commonly employed for software analysis purposes. Moreover, many of these technologies rely on time-based analysis, assuming that malware will initiate malicious actions shortly after infection. This approach renders them less effective for extensive analysis or malware's tactic to remain dormant during its initial phase, a phenomenon referred to as "stalling". Similarly, "out-the-box" sandbox technologies cannot capture and record network activity; thus, they rely exclusively on kernel-level tracing mechanisms, which results in a shortcoming in monitoring and recording Command and Control (C&C) activities [14]. To verify these limitations, Chen et al. [15] demonstrated that merely 2% of the considered malware samples exhibited malicious activity when they were examined in an analysis environment.

To overcome these limitations, we present a framework for analyzing malware samples in the following section. This system relies on Elasticsearch as its foundation for data storage and indexing, facilitating rapid search and analysis. Kibana also serves as a user-friendly interface for extracting valuable insights, while Sysmon generates logs containing critical security and operational data from system events.

This way, we eliminate the requirement of utilizing a sandbox environment and overcome the limitations posed by such techniques. In addition, we deploy several anti-evasion techniques to restrict the malware capability for fingerprinting the analysis environment.

III. PROPOSED FRAMEWORK

This section presents our approach to malware behavior analysis and activity collection. The desired outcome is a sandbox-independent infrastructure using VMs as guests and a managing system for automatic configuration and monitoring of malware execution. This way, the proposed framework can be utilized to analyze multiple samples in parallel and at scale.

A. ARCHITECTURE

The proposed architecture provides a framework, where malware can be executed in isolation, while still creating an environment that can be similar to a real one. Therefore, the architecture is based on VMs, respecting the inside-the-box model for isolation and containment that interconnected via a virtual network. The guest machine is a pre-configured Windows 10 OS that will be infected with the malware samples for analysis, as detailed in the next section. The analysis component is build upon Elastic-Stack, an open-source and popular set of tools in industry, proven effective for network measurements [16]. Figure 1 illustrates the overall architecture of the proposed framework. On the left side, it is depicted the Windows VM that undertakes the role of the target machine for the samples, while on the right side is the analysis framework along with the malware farm, where the samples reside. This setup can be replicated into n parallel blocks for analysis, with the only restriction the amount of the available resources (e.g., RAM memory) for the VM execution.

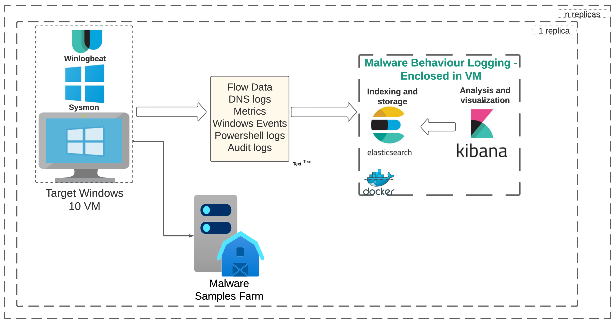


FIGURE 1. Architecture of the sandbox environment.

B. WORKFLOW OF MALWARE ANALYSIS

The platform aims to trigger the malware samples to infect and execute their malicious operation in a controlled and isolated environment, concretely a desktop machine, and record their behavior. Therefore, the workflow is divided into five phases as follows:

1) PHASE 1 — PREPARATORY STEP

The analysis environment is initially prepared as illustrated in Figure 2. The procedure sets the required VMs once and saves their initial state, so it can be used as it is for every new analysis task and at scale. In detail:

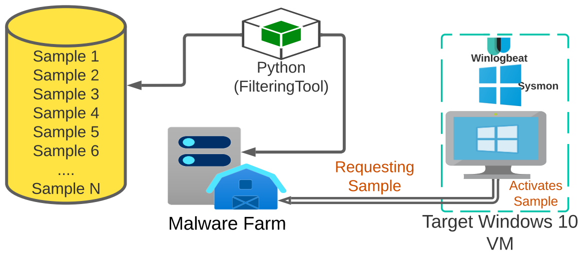


FIGURE 2. **Preparation of the environment before analysis.**

- **Preparation of the malware (triage and filtering)**: During this step, malware samples are collected from VxUnderground [17], an online platform known by its extensive collection of malware samples, functioning as a comprehensive repository and resource hub dedicated to the academic exploration of advanced computer security, malware analysis, and cyber threats. The procedure begins with the filtering of the malware samples, involving an inspection of the PE (Portable Executable) information contained within the file header. It is important to emphasize that this file header is specifically employed by executable files exclusive to the Windows operating system. In more detail, the *Python Filtering Tool* systematically selects the malware, removing any samples that lack a PE header. Simultaneously, the tool assesses the compatibility of the identified samples with the 64-bit architecture.
  In our context, the primary focus is identifying and excluding samples lacking executability. Once such samples are successfully identified, they are stored at the Malware Sample Farm for further analysis and processing.
- **Windows Environment Hardening**: As previously mentioned, malware employs evasion techniques, especially when dissected in virtual environments. While determining the specific actions required for individual samples to activate requires extensive manual effort, this methodology prioritizes addressing the prevalent evasion techniques. Therefore, in preparing the guest Windows Destkop 10 OS VM, we systematically eliminate any artifacts susceptible to fingerprinting, such as VirtualBox hardware components, substituting them with authentic device names. Additionally, for each analysis instance, novel MAC addresses are generated, distinct from the allocated space of VirtualBox. Furthermore, we employ PAFish [18], an open-source tool designed for evaluating the robustness of sandbox environments by simulating the evasion techniques employed by malware to fingerprint a sandbox. Concretely, the PAFish tool is executed to assess the efficacy of our configuration in thwarting these evasion attempts, thereby enabling an evaluation of the analysis environment's reliability. Subsequently, files and folders are strategically positioned in the system to emulate the behavior of a genuine user interacting with the computer.
  It is imperative to note that, to facilitate the execution of malware, all defensive mechanisms, including Windows Defender and the firewall, are intentionally disabled.

2) PHASE 2 – DYNAMIC ANALYSIS

The actual dynamic analysis of the samples takes place in this phase. For consistency, the part of the infrastructure that contains the Windows VM is spawned and destroyed for each analyzed sample. At the same time, the VM, where ELK resides, is kept intact during the analysis. As enablers for this analysis, ElasticSearch, Winlogbeat, and Sysmon are used; specifically, ElasticSearch stores the data, Winlogbeat ships the events, and Sysmon grabs the information from the Windows logs. As detailed in our previous work [16], such a setup can collect, organize, store, and ship various types of information. Particularly, as this work focuses on malware behavior analysis, the following activities are recorded during the analysis:

- Process creation (including full command line and hashes)
- Process termination
- File events (Creation, Deletion, Modifications)
- Driver/image loading
- Raw disk access
- Process memory access
- Registry access (create, modify, delete)
- Named pipes
- Network connections (DNS requests, Network Flows)

While this infrastructure still falls under most of the sandboxes' approach by having the analysis tools and target VM communicate over a virtual network, the primary difference and novelty comes with the usage of ELK as a method to monitor the malware behavior. The framework does not depend on recognized sandbox technologies susceptible to malware fingerprinting, as proposed by [19]. Instead, it exclusively utilizes ELK, a widely adopted cross-platform toolset for Security Information and Event Management (SIEM) operations. Furthermore, the framework's relevance extends beyond Windows machine examples as the infrastructure can be expanded by employing diverse operating systems for the target virtual machine. Therefore, the malware might be able to fingerprint that it is running inside a VM, but it cannot deduce, based on the appliances installed, that its behavior is logged. Indeed, deploying production services in virtual machines is currently a common practice in the industry due to the ease of management and cost reduction of virtualization technology.

- **Injecting and Running Malware**: As this infrastructure does not rely on a sandbox agent to control the analysis process, the injection of the malware sample is initiated by the Windows VM, meaning that each time the Windows machine starts, it automatically requests a new malware sample from the farm. Once a sample is grabbed, it is deleted from the farm, and stored on a specific location at the Guest VM, that handles its activation. In this context, the activation of the malware sample denotes the moment where the mechanism executes the specific sample in order to start recording its potential malicious behavior.

- **Collection of information**: The Windows VM is prepared beforehand with Sysmon to grab the events at the host level. Since Sysmon provides a modular architecture based on its configuration file, in our case, we employ a configuration file that can catch all types of events and thus provide rich information. At the same time, Powershell logs are collected alongside the events. On the other hand, regarding the network traffic, we opt only to record the network flows and DNS requests. To ingest all the information, we use similar to [16] Winlogbeat, a lightweight log shipper used to collect and forward Windows event logs and other log data to a centralized repository, and Packetbeat, a packet sniffer that supports a wide range of protocols.
- **Clean-up and Restart**: Monitoring the malware behavior lasts approximately 25 minutes. This specific timeframe is chosen to provide ample time for the infrastructure to execute effectively and circumvent any evasion tactics reliant on time-based mechanisms employed by the malware. It is important to note that certain malware samples may remain inactive beyond the designated timeframe. The precise duration of a sample's dormant period can only be ascertained through static analysis and reverse engineering.

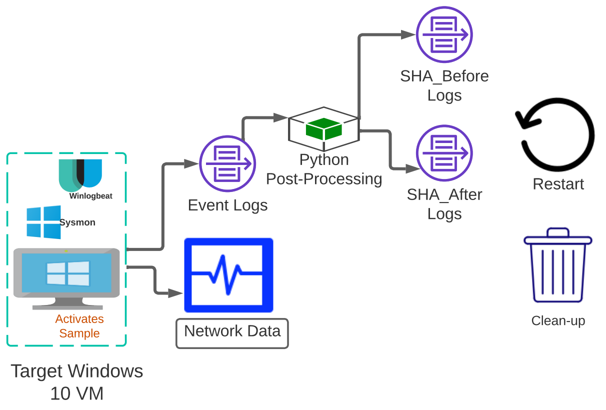


FIGURE 3. Workflow of the proposed framework iterated for each sample.

We aim to extend the observation period for the samples' analysis to gain a more comprehensive understanding of their potential activities. For this reason, the decision for 25 minutes is based on the trade-off between potentially providing enough time for the malware to execute its actions and not significantly delaying the entire analysis pipeline based on the available computational capabilities. Once this interval has elapsed, the Windows VM is terminated, and a new and cleaned instance is spawned. This way, we ensure no traces from the previous analysis can affect the new one. The overall process of the malware analysis is detailed in figure 3.

- **Creation of the dataset - Before & After Infection**: Here, the goal is to split the generated events into two subsets, one related to the "typical" operation of the Windows OS (i.e., before the triggering of the malware), with the latter including the events potentially generated by the malware (i.e., after the infection). This is achieved based on three timestamps:
If we consider T0 the timestamp that the target Windows VM is spawned, while T2 is the timestamp that the VM is destroyed, we can safely assume that all the Windows activity occurs during this duration. The exact timestamp (T1) that the malware is activated is determined based on a preconfigured flag within the logs that signals the execution of the malware binary. Thus, any subsequent events from this time are presumed to be related to the malware execution. Therefore, we initially query ELK to include all the activity between T2-T0. Afterward, we separate the events before and after the malware activation based on the T1 timestamp. The *Python Post-Processing* tool facilitates this procedure by executing queries and segregating events. It stores events occurring after the flag's appearance in the AAU_MalData_Contagious subset, while events preceding this occurrence are stored in the AAU_MalData_Pure subset. Additionally, the Python tool performs further refinements on the logs, involving removing metadata information and the selective extraction of events generated by the underlying operating system.

IV. ANALYSIS

In total, we download and experiment with 2,800 malware samples from VxUnderground [17], covering 89 Malware families and two classes, namely Trojans and Backdoors. These sample distinctions were derived from VxUnderground, where the samples are appropriately labeled.

Our objectives are firstly to provide proof that the proposed framework can be used for the dynamic analysis of malware samples in parallel and at scale and secondly to monitor and compile a dataset of their activity containing network and host-based logs. Such a dataset can be useful for future anti-malware research and training detection algorithms.

Overall, the analyzed samples exhibited sophisticated techniques to achieve persistence, evade detection, and control compromised systems. By comparing the event actions of Trojans and Backdoors samples, we can gain valuable insights into their behaviors. We categorize the type of the events according to Create Remote Thread, DNS query, Driver loaded, File created, File creation time changed, Process Create, Registry object added or deleted, and Registry value set. Then, we accumulated the total occurrence of the events for each category to understand which malware classes were more active in the designed framework. As displayed in Figure 4 the samples to create an equivalent number of activities with a slight increase for the case of Backdoors.

A. WINDOWS EVENTS DATA-SET

The analysis of the malware samples yields two distinct subsets, e.g., *AAU_MalData_Pure* and *AAU_MalData_Contagious*. The former encompasses everyday Windows OS activity events created prior to the triggering of the samples, while the latter comprises events predominantly generated by the malware samples. The generated logs undergo a data cleaning process, aiming to minimize their size and optimize their readiness for future research; for instance, meta information and events generated by legitimate Windows system processes are stripped out for the AAU_MalData_Contagious but maintained for the benign one. The resulting datasets incorporate various fields.

- **@timestamp** - Indicates the timestamp of the event.
- **ECS**: Refers to the Elasticsearch Common Schema utilized to standardize and organize the data.
- **Event**: Details the event type, corresponding code, and provider. Each Event in Sysmon is assigned a unique event number or type that helps to identify specific activities or behaviors associated with potential security threats. Knowing the event type and number can help filter out the events in a possible forensic investigation and also facilitate the creation of specific alerting rules.
- **Message**: Contains a general message related to the specific Event in a textual description.
- **Winlog**: Windows Event Logs (Winlogs) provides a chronological record of the system events, including actions performed by the malware. From the sequence of the events, a malware's behavior and activity can be deduced, like process injection, creation of persistence mechanisms, command-and-control communications, or attempts to evade detection. The details for each of these events are structured under the **Event_data** field, similar to Fig. 1 where it is listed as a snippet of processes created by a malware sample.

In turn, Event_data represents the specific data associated with each Windows Event log; this information is essential as it provides valuable insights into system activities and events.

- **CommandLine**: This field refers to the specific command line utilized to initiate the execution of the process. It contains the complete command, including arguments and parameters.
- **CurrentDirectory**: Denoting the current working directory at the process execution time, this field signifies the directory path within the file system where the process is executed. It helps to understand the context in which the process operates.
- **Image**: This field specifies the file path or location of the process executable file. It provides the exact location within the file system where the process executable is stored.
- **Hashes**: The Hashes field contains the hash values computed for the process executable file. Hash functions and techniques, such as MD5, SHA256, and IMPHASH, are applied to the file to produce a unique and fixed-length value. These hashes can be used to verify the integrity and authenticity of an executable file.
- **IntegrityLevel**: This field indicates the integrity level assigned to the process. Integrity levels are security mechanisms that enforce restrictions and permissions on processes, ensuring that they operate within predefined boundaries. There are four integrity levels: Low, Medium, High, and System. The System integrity level can only be acquired through OS processes.
- **LogonGuid** and **LogonId**: LogonGuid is a globally unique identifier assigned to a user logon session, while LogonId is a numeric identifier assigned by the OS to differentiate between different logon sessions. LogonGuid and LogonId are helpful in tracking and associating events and activities with specific user logon sessions, but they represent different identifiers.
- **OriginalFileName**: This field specifies the original name of the process executable file. It provides the filename, as it was initially determined during the development and compilation of the process.
- **ParentCommandLine**, **ParentImage**, **ParentProcessGuid**, **ParentProcessId**, **ParentUser**: The Parent keyword refers to the parent process that launches the current process. It provides crucial information for investigating the lineage and relationship between processes, enabling the identification of the specific parent process responsible for initiating the execution of the current process.
- **RuleName**: The RuleName field represents the rule's name associated with a particular event. It provides valuable information about predefined rules or criteria that were evaluated in the event context. Mapping these rule names to the MITRE ATT&CK framework can enhance the understanding of specific techniques employed during the event.

The produced dataset is publicly available to the security community [20]. The AAU_MalData_Contagious folder contains four data collections with the Backdoor malware samples' activity and eight Trojan data collections. All these events are recorded by monitoring the execution of the corresponding malware samples within the proposed framework.

The events produced by the malware samples are structured as shown in listing 2. They follow a key-value format, with the key being the sample's SHA256 hash value, while the value comprises all the events, namely *event_1; event_2; …;event_n*, generated by the malware, organized as described in Section IV-A. The AAU_MalData_Pure folder adheres to the same structure and encompasses routine activities, consisting of actions initiated by the OS system

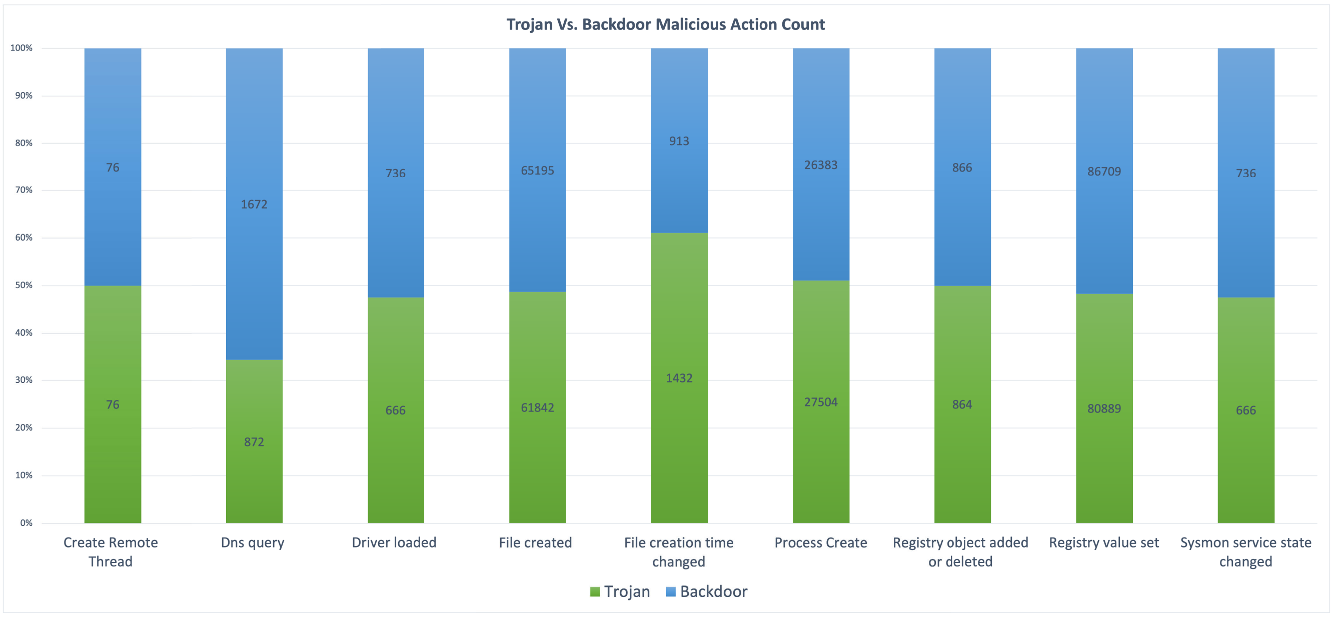


FIGURE 4. **Trojan vs. backdoor malicious action count.**

without any user or program interference. In addition, we synthesize the AAU_MaAlData_Synthetic dataset, by recording the activity typically generated by a user utilizing the same Windows VM machine. The generated events include legitimate activities related with office tasks, such as creating and editing file documents, opening multimedia files, e.g., music and video files, browsing to the Internet, installation or deletion of legitimate applications and other miscellaneous tasks. The intention of this dataset is to supplement the AAU_MalData_Pure dataset by introducing benign activity besides the passive mode events of the running OS, thus, facilitating a more explicit distinction between malicious and non-malicious activities.

B. DNS TRAFFIC ANALYSIS

In addition to the host level logs, the ELK toolset provides the capability to record the network activity of the malware samples in the form of network flows and DNS transactions. For brevity, we further detail the DNS query behavior of the analyzed malware samples. Such analysis can provide valuable insights, although it is challenging to draw generalized conclusions due to the unique characteristics of each sample. Despite the restricted network environment, namely no allowed access to the Internet, particular malware samples demonstrate a clear intention to establish communication with potential C&C servers. To further analyze this behavior, the distribution of the corresponding countries is calculated based on the Geo-location of the IP addresses associated with the domain names. This results in 50 countries; however, only countries with more than 20 occurrences are presented in Figure 5, while the rest are grouped under the ‘‘Others’’ category. Furthermore, we investigate if these domain names are already blocklisted in the VirusTotal (VT) DB. For the

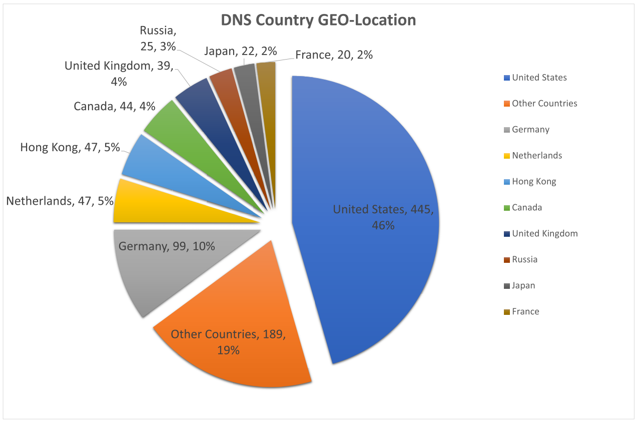


FIGURE 5. **Distribution of countries by the Geo-location of the IP associated with the domain.**

domain names flagged as malicious, we extract the type of activity associated with them based on the label assigned by VT.

The results reveal that almost half of the domain names correspond to IP addresses in the United States. Remarkably, a considerable percentage of the names associated with malware are not flagged as malicious by the VT analysis. However, these domains are confirmed to be requested by processes created by the malware samples. Specifically, out of the 1,292 unique domains, only 416 are classified as malicious by at least two security vendors on VT. The distribution of the malware categories, presented in Figure 6, sheds light on the types of activities associated with these domains, making it challenging to decisively conclude whether they are malicious or merely linked to benign activities.

```
{
  "winlog": {
    "event_data": {
      "CommandLine": "C:\\Windows\\SysWOW64\\WerFault.exe -u -p 4704 -s 1468",
      "Company": "Microsoft Corporation",
      "CurrentDirectory": "C:\\Windows\\system32\\",
      "Description": "Windows Problem Reporting",
      "FileVersion": "10.0.19041.1949 (WinBuild.160101.0800)",
      "Hashes":
        "MD5=C31336C1EFC2CCB44B4326EA793040F2,
         SHA256=CF361CB7148DB9FBC6F6A2D5AE97CCADB2C3B1F57E61C50B8CB59681D0AFE420,
         IMPHASH=C4254BB6DCC347D1C7D827F761FB0176",
      "Image": "C:\\Windows\\SysWOW64\\WerFault.exe",
      "IntegrityLevel": "Medium",
      "LogonGuid": "{a4145622-11dd-6382-d638-040000000000}",
      "LogonId": "0x438d6",
      "OriginalFileName": "WerFault.exe",
      "ParentCommandLine": "C:\\Users\\John Doe\\Downloads\\runbaby\\
                HEUR-Trojan-PSW.MSIL.Agensla.exe\" /s ",
      "ParentImage": "C:\\Users\\John Doe\\Downloads\\runbaby\\
                HEUR-Trojan-PSW.MSIL.Agensla.exe",
      "ParentProcessGuid": "{a4145622-12fc-6382-d700-000000007f00}",
      "ParentProcessId": "4704",
      "ParentUser": "LAB1\\John Doe",
      "ProcessGuid": "{a4145622-12fd-6382-dc00-000000007f00}",
      "ProcessId": "7240",
      "Product": "Microsoft Windows Operating System",
      "RuleName": "-",
      "TerminalSessionId": "1",
      "User": "LAB1\\John Doe",
      "UtcTime": "2022-11-26 13:22:05.736"
    }
  }
}
```

LISTING 1. Snippet of a malware sample action in Windows OS.

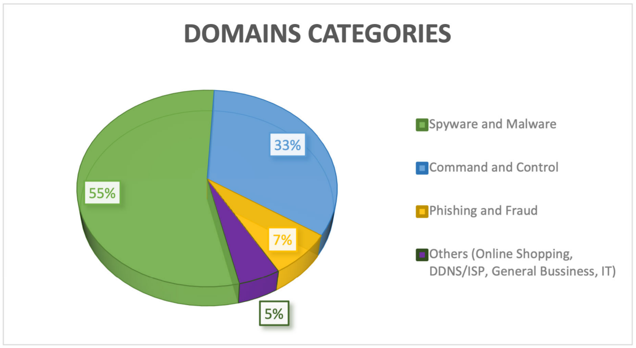


FIGURE 6. Categories drawn from VT for the malicious domains.

V. MITRE ATT&CK

Utilizing MITRE Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) framework for malware analysis offers a structured taxonomy that categorizes and comprehensively describes adversarial tactics and techniques as also documented by [21]. MITRE contains 12 tactics [22] that describe adversary objectives for their actions:

- **Initial Access** involves the methods adversaries employ to gain a foothold in a system.
- **Execution** occurs after gaining initial access, where adversaries may execute malicious code through a Command-Line Interface or Graphical User Interface or wait for the user to trigger the binary.
- **Persistence** is the effort to maintain access, even when users change passwords, by hijacking legitimate code on the victim system to move deeper into it.
- *Privilege Escalation* involves adversaries elevating their permissions within an enterprise system by exploiting vulnerabilities in applications and servers.

```
{
"Malware1_SHA256":[
   event_1:{"Malicious Activity",
            "Malicious Activity",
            "Malicious Activity",
       },
   event_2:{"Malicious Activity",
            "Malicious Activity",
            "Malicious Activity",
       },
       ...
   event_n:{"Malicious Activity",
            "Malicious Activity",
            "Malicious Activity",
       }
     ...
],
...
}
```

LISTING 2. Malicious activity JSON format sample.

- **Defense Evasion** is the act of clearing traces to avoid detection and bypass security controls, ensuring uninterrupted malicious activities.
- **Credential Access** entails capturing additional usernames and passwords through techniques like Bash History or Keychain to achieve objectives and maintain access.
- **Discovery** follows gaining access, where adversaries explore and gather information about vulnerabilities, stored data, and network resources through techniques like Network Service Scanning.
- **Lateral Movement** occurs when adversaries shift from one compromised user account to others within an office area, exploiting trusted internal accounts through techniques like Internal Spear-phishing.
- **Collection** involves adversaries gathering data to achieve malicious objectives, collected from compromised computers or peripheral devices using techniques like Data from Removable Media, with the next step being data exfiltration.
- **Command and Control** enables adversaries to control operations within an enterprise system remotely, turning compromised computers into controlled botnets.
- **Exfiltratio**n follows data collection, where adversaries package data using techniques like Data Compression to make exfiltration less conspicuous and capable of bypassing detection.
- **Impact** includes breaching confidentiality, degrading integrity, and limiting asset availability within an enterprise system.

While the absence of an internet connection presents constraints on the Execution of the samples, potentially forbidding the activation of certain samples, the authors opted against allowing internet connectivity due to security considerations.

This categorization can help an analyst investigate malware campaigns, understand their methods, and formulate more effective detection and mitigation strategies. Therefore, the framework could provide a systematic approach to formulating behavior and establishing the format for comparing similar malware classes.

For instance, Trojans and Backdoors represent two categories of malicious software with some standard functionalities. However, they aim for diverse objectives. These distinctions can be elucidated by leveraging the MITRE framework to associate their actions with specific Tactics, Techniques, and Procedures (TTPs), thus offering a comprehensive and standardized perspective on potential adversary actions within systems. As such, Figure 7 illustrates the operational behavior of these two malware classes, derived from the accumulated log files of the 2,800 analyzed malware samples and correlated with MITRE's ATT&CK matrices with the help of the MITRE ATT&CK Navigator [23].

Overall, Trojan malware masquerades as legitimate programs to deceive users into executing them, thereby granting unauthorized access to attackers. On the other hand, Backdoor malware are hidden entry points intentionally inserted by developers or attackers to bypass authentication mechanisms and gain remote access at a later stage. Consequently, it is valuable to compare their behavior with the use of the proposed platform. In this direction, the MITRE ATT&CK framework [24] is utilized to provide a comprehensive overview of their TTPs. A color coding scheme is applied to the generated matrices to distinguish between the techniques used by Trojans and Backdoors. In this scheme, **Yellow** is assigned to techniques that are exclusively utilized by **Trojans**, **Red** to techniques specific to **Backdoors**, while **Green** to techniques employed by both types of malware as depicted in Figure 7.

The examined samples demonstrate similar behaviors within the environment, albeit with some variations. Typically, the Backdoors exploit a crucial Windows OS component called Winlogon helper DLL to bypass user authentication. They operate during early system startup to ensure their presence remains active, covert, and seamlessly integrated with legitimate processes. This tactic enables them to evade detection, especially by employing techniques that load the backdoor after system reboots or updates. On the contrary, Trojans focus on establishing persistency within compromised systems by exploiting system boot or logon autostart locations. By leveraging these entry points, Trojans ensure automatic execution upon system startup. Additionally, Trojans employ deceptive tactics by masquerading as legitimate binary files and following the typical execution flow triggered by benign events. This camouflage technique allows them to blend in and avoid suspicions.

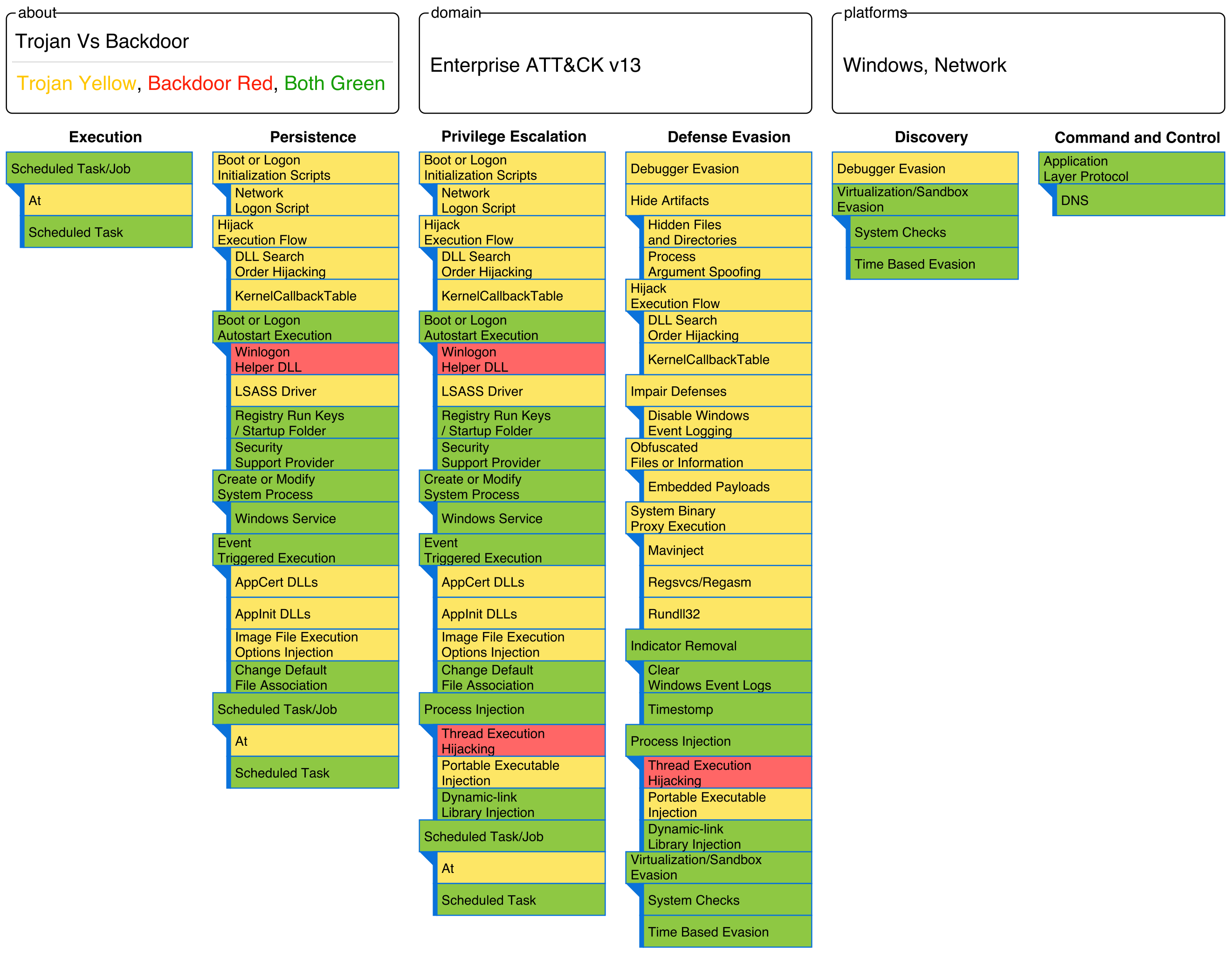

about
Trojan Vs Backdoor
Trojan Yellow, Backdoor Red, Both Green
domain
Enterprise ATT&CK v13
platforms
Windows, Network
Execution
Scheduled Task/Job
At
Scheduled Task
Persistence
Boot or Logon Initialization Scripts
Network Logon Script
Hijack Execution Flow
DLL Search Order Hijacking
KernelCallbackTable
Boot or Logon Autostart Execution
Winlogon Helper DLL
LSASS Driver
Registry Run Keys / Startup Folder
Security Support Provider
Create or Modify System Process
Windows Service
Event Triggered Execution
AppCert DLLs
AppInit DLLs
Image File Execution Options Injection
Change Default File Association
Scheduled Task/Job
At
Scheduled Task
Privilege Escalation
Boot or Logon Initialization Scripts
Network Logon Script
Hijack Execution Flow
DLL Search Order Hijacking
KernelCallbackTable
Boot or Logon Autostart Execution
Winlogon Helper DLL
LSASS Driver
Registry Run Keys / Startup Folder
Security Support Provider
Create or Modify System Process
Windows Service
Event Triggered Execution
AppCert DLLs
AppInit DLLs
Image File Execution Options Injection
Change Default File Association
Process Injection
Thread Execution Hijacking
Portable Executable Injection
Dynamic-link Library Injection
Scheduled Task/Job
At
Scheduled Task
Defense Evasion
Debugger Evasion
Hide Artifacts
Hidden Files and Directories
Process Argument Spoofing
Hijack Execution Flow
DLL Search Order Hijacking
KernelCallbackTable
Impair Defenses
Disable Windows Event Logging
Obfuscated Files or Information
Embedded Payloads
System Binary Proxy Execution
Mavinject
Regsvcs/Regasm
Rundll32
Indicator Removal
Clear Windows Event Logs
Timestomp
Process Injection
Thread Execution Hijacking
Portable Executable Injection
Dynamic-link Library Injection
Virtualization/Sandbox Evasion
System Checks
Time Based Evasion
Discovery
Debugger Evasion
Virtualization/Sandbox Evasion
System Checks
Time Based Evasion
Command and Control
Application Layer Protocol
DNS

FIGURE 7. **MITRE techniques employed by Trojans and Backdoors.**

Trojans and Backdoors employ defense evasion mechanisms as observed in the literature [1]. They execute various checks, including system-based and time-based checks, to determine if they operate within a sandbox environment. Furthermore, Trojans attempt to hide malicious artifacts, such as files, processes, and registry keys, to evade detection by security tools and analysts. In addition, they try to disable or modify antivirus software, firewalls, intrusion detection systems, and similar defensive mechanisms to remain undetected and maintain their presence. However, the outcomes of these activities are documented in the log files, serving as robust IoCs and providing evidence that the samples manifested techniques from their arsenal.

VI. LIMITATIONS

This study acknowledges several limitations inherent to its methodology and scope. Firstly, the sample selection process deliberately focused solely on Windows-based executable files, omitting other file types to minimize the risk of compromising the host system, a Linux operating system. However, this selection bias may have excluded potentially relevant file types that entail further analysis.

Secondly, the analysis environment was deliberately isolated from the Internet, which constrained the researchers' ability to observe the actual network activity, both outgoing and incoming traffic, between the samples and their C&C servers or other online resources. Consequently, it remains to be seen whether the observed malicious activities represent the full extent of the malware's capabilities or if the absence of further instructions from potential commands from these C&C servers limited them.

Third, we ignore whether malware that did not execute was due to the lack of triggering from external sources (e.g., opening a specific program or a time-based trigger) or due to actual evasion due to the environment being fingerprinted. Understanding the exact cause will require dedicated analysis, probably using static analysis tools, which would delay the overall pipeline.

Lastly, the computational resources required to conduct this scale analysis were substantial. The study required a significant allocation of hardware resources to accommodate the processing and storage demands of analyzing a large sample size. In scenarios where resource availability is limited, the scope and scale of the analysis may be compromised, potentially impacting the breadth and depth of the findings. We believe, however, that a production system with higher resources, e.g., from the security industry, can quickly implement our methodology and bypass such limitations.

Acknowledging these limitations is crucial, as it comprehensively explains the constraints inherent to our academic study's design and execution. These limitations offer valuable insights for future research endeavors, informing potential refinements and avenues for improvement in similar investigations.

VII. CONCLUSION

The perpetual struggle between attackers and defenders within the field of malware research remains a driving force behind progress. This paper has emphasized the importance of dynamic malware analysis in exploring novel attack patterns and techniques and in forensic investigations by extracting various IoCs to acquire more profound insights into system breaches. The work presented in the current paper has a twofold contribution: a sandbox-independent framework for malware dynamic analysis and a realistic dataset that captures the Windows event activities of 2,800 malware samples scrutinized within the aforementioned framework.

In particular, the proposed framework introduces an innovative approach to the dynamic analysis of malware, intending to trigger and monitor the malware's activity while evading the fingerprinting of pre-existing sandbox technologies by the malware. Researchers can systematically collect, organize, and enrich valuable information by analyzing the malware samples by utilizing Windows VMs as targets for infection and employing open-source tools such as Sysmon, ElasticSearch, and Kibana. Furthermore, the ability to segregate the malware activities into event datasets is demonstrated through the offering to the cybersecurity research community of the AAU_MalData dataset. The AAU_MalData dataset is divided into two subsets, AAU_MalData_Pure and AAU_MalData_Contagious; the first one corresponds to the regular operation of the Windows OS, while the latter to the analysis of 2,800 malware samples of Trojans and Backdoors type with the help of the proposed framework. Such dataset can be utilized, for instance, for the training of machine learning algorithms as suggested in [25] for advancing anti-malware research.

As the race between attackers and defenders in the malware landscape persists, the ongoing pursuit of research and innovation in dynamic analysis techniques will remain vital for maintaining an advantageous position against malicious actors.

ACKNOWLEDGMENT

The authors are grateful to OTTO MØNSTEDS FOND for their financial support for the author's research stay at the Universidad Carlos III de Madrid (UC3M), which has not only enriched the academic journey but has also strengthened international research collaboration.

REFERENCES

[1] A. Afianian, S. Niksefat, B. Sadeghiyan, and D. Baptiste, "Malware dynamic analysis evasion techniques: A survey," *ACM Comput. Surv.*, vol. 52, no. 6, pp. 1–28, Nov. 2020.

[2] A. Sharma, B. B. Gupta, A. K. Singh, and V. K. Saraswat, "Orchestration of APT malware evasive manoeuvers employed for eluding anti-virus and sandbox defense," *Comput. Secur.*, vol. 115, Apr. 2022, Art. no. 102627.

[3] F. A. Aboaoja, A. Zainal, F. A. Ghaleb, B. A. S. Al-rimy, T. A. E. Eisa, and A. A. H. Elnour, "Malware detection issues, challenges, and future directions: A survey," *Appl. Sci.*, vol. 12, no. 17, p. 8482, Aug. 2022.

[4] M. Asam, S. J. Hussain, M. Mohatram, S. H. Khan, T. Jamal, A. Zafar, A. Khan, M. U. Ali, and U. Zahoora, "Detection of exceptional malware variants using deep boosted feature spaces and machine learning," *Appl. Sci.*, vol. 11, no. 21, p. 10464, Nov. 2021.

[5] H. Oz, A. Aris, A. Levi, and A. S. Uluagac, "A survey on ransomware: Evolution, taxonomy, and defense solutions," *ACM Comput. Surv.*, vol. 54, no. 11s, pp. 1–37, Jan. 2022.

[6] O. Or-Meir, N. Nissim, Y. Elovici, and L. Rokach, "Dynamic malware analysis in the modern era—A state of the art survey," *ACM Comput. Surv.*, vol. 52, no. 5, pp. 1–48, Sep. 2020.

[7] A. Jewitt. (2021). *How to Build a Malware Analysis Sandbox With Elastic Security*. [Online]. Available: https://www.elastic.co/es/blog/how-to-build-a-malware-analysis-sandbox-with-elastic-security

[8] A. A. R. Melvin and G. J. W. Kathrine, "A quest for best: A detailed comparison between drakvuf-vmi-based and cuckoo sandbox-based technique for dynamic malware analysis," in *Intelligence in Big Data Technologies-Beyond the Hype*. Cham, Switzerland: Springer, 2021, pp. 275–290.

[9] M. Vasilescu, L. Gheorghe, and N. Tapus, "Practical malware analysis based on sandboxing," in *Proc. RoEduNet Conf. 13th Edition: Netw. Educ. Res. Joint Event RENAM 8th Conf.*, Sep. 2014, pp. 1–6.

[10] D. Oktavianto and I. Muhardianto, *Cuckoo Malware Analysis*. Birmingham, U.K.: Packt Publishing Ltd, 2013.

[11] S. Ilić, M. Gnjatović, B. Popović, and N. Maček, "A pilot comparative analysis of the cuckoo and drakvuf sandboxes: An end-user perspective," *Vojnotehnicki Glasnik*, vol. 70, no. 2, pp. 372–392, 2022.

[12] Ö. A. Aslan and R. Samet, "A comprehensive review on malware detection approaches," *IEEE Access*, vol. 8, pp. 6249–6271, 2020.

[13] K. Hamajima, D. Kotani, and Y. Okabe, "Partial outsourcing of malware dynamic analysis without disclosing file contents," in *Proc. IEEE 47th Annu. Comput., Softw., Appl. Conf. (COMPSAC)*, Jun. 2023, pp. 717–722.

[14] G. Kambourakis, M. Anagnostopoulos, W. Meng, and P. Zhou, *Botnets: Architectures, Countermeasures, Challenges*. Boca Raton, FL, USA: CRC Press, 2019.

[15] X. Chen, J. Andersen, Z. M. Mao, M. Bailey, and J. Nazario, "Towards an understanding of anti-virtualization and anti-debugging behavior in modern malware," in *Proc. IEEE Int. Conf. Dependable Syst. Netw. With FTCS DCC (DSN)*, Jun. 2008, pp. 177–186.

[16] R.-V. Mahmoud, M. Anagnostopoulos, and J. M. Pedersen, "Detecting cyber attacks through measurements: Learnings from a cyber range," *IEEE Instrum. Meas. Mag.*, vol. 25, no. 6, pp. 31–36, Sep. 2022.

[17] VXUnderground. (2022). *The Largest Collection of Malware Source Code, Samples, and Papers on the Internet*. Accessed: Nov. 27, 2022. [Online]. Available: https://www.vx-underground.org/#E:/root/Samples/Bazaar%20Collection

[18] INFOSEC. (2023). *Pafish (Paranoid Fish)*. Accessed: Jul. 10, 2023. [Online]. Available: https://resources.infosecinstitute.com/topic/pafish-paranoid-fish/

[19] A. Yokoyama, K. Ishii, R. Tanabe, Y. Papa, K. Yoshioka, T. Matsumoto, T. Kasama, D. Inoue, M. Brengel, and M. Backes, "SandPrint: Fingerprinting malware sandboxes to provide intelligence for sandbox evasion," in *Proc. Int. Symp. Attacks, Intrusions, Defenses*, Paris, France. Cham, Switzerland: Springer, Sep. 2016, pp. 165–187.

[20] R. V. Mahmoud. (2023). *Backdoor and Trojans Datasets of Malicious Activity*. Accessed: Jul. 15, 2023. [Online]. Available: https://github.com/rasmim/malwareDatasets

[21] M. F. Abdelwahed, M. M. Kamal, and S. G. Sayed, "Detecting malware activities with MalpMiner: A dynamic analysis approach," *IEEE Access*, vol. 1, pp. 84772–84784, 2023.

[22] W. Xiong, E. Legrand, O. Åberg, and R. Lagerström, "Cyber security threat modeling based on the MITRE enterprise ATT&CK matrix," *Softw. Syst. Model.*, vol. 21, no. 1, pp. 157–177, Feb. 2022.

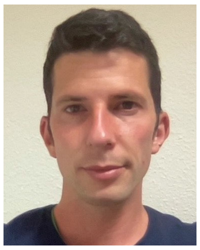
[23] MITRE. (2023). *MITRE ATT&CK Navigator*. Accessed: Jul. 15, 2023. [Online]. Available: https://mitre-attack.github.io/attack-navigator/,

[24] A. Georgiadou, S. Mouzakitis, and D. Askounis, "Assessing MITRE ATT&CK risk using a cyber-security culture framework," *Sensors*, vol. 21, no. 9, p. 3267, May 2021.

[25] K. Steverson, C. Carlin, J. Mullin, and M. Ahiskali, "Cyber intrusion detection using natural language processing on windows event logs," in *Proc. Int. Conf. Mil. Commun. Inf. Syst. (ICMCIS)*, May 2021, pp. 1–7.

RASMI-VLAD MAHMOUD received the M.Sc. degree in networks and distributed systems from Aalborg University, Aalborg, Denmark, where he is currently pursuing the Ph.D. degree. His research interests include cyber ranges, threat intelligence, and malware analysis.

MARIOS ANAGNOSTOPOULOS is currently an Assistant Professor of cyber security with Aalborg University, Copenhagen, Denmark. His research interests include network and computer security, specifically DNS security, denial of service attacks, botnets, malware analysis, and forensics.



SERGIO PASTRANA is currently an Associate Professor with the Universidad Carlos III de Madrid, Spain, where he teaches various courses on cybersecurity and cryptography. His research interests include different areas of security and privacy, including the measurement, and analysis of the socio-technical factors and human aspects of cybercrime.

JENS MYRUP PEDERSEN is currently a Professor of cyber security with Aalborg University, with his main research interest being network security. He is also the Head of the Cyber Security Research Group, Aalborg University, and the Cyber Security Masters Programme. In addition, he is active in talent development of young people. He has been involved in numerous Danish and European projects within cyber security, including projects focusing on sandboxing, virtualization environments for training and data generation, deception technologies, such as honeypots, and network traffic analysis applied to detection of malicious activities. In addition to his research, he has received multiple awards for his teaching.