RaDaR: A Real-World Dataset for AI Powered Run-time Detection of Cyber-Attacks

Sareena Karapoola
sareena@cse.iitm.ac.in
Indian Institute of Technology Madras
Chennai, India

Nikhilesh Singh
nik@cse.iitm.ac.in
Indian Institute of Technology Madras
Chennai, India

Chester Rebeiro
chester@iitm.ac.in
Indian Institute of Technology Madras
Chennai, India

V. Kamakoti
kama@cse.iitm.ac.in
Indian Institute of Technology Madras
Chennai, India

ABSTRACT

Artificial Intelligence-based techniques on malware run-time behavior have emerged as a promising tool in the arms race against sophisticated and stealthy cyber-attacks. While data of malware run-time features are critical for research and benchmark comparisons, unfortunately, there is a dearth of real-world datasets due to multiple challenges to their collection. The evasive nature of malware, its dependence on connected real-world conditions and active remote servers to execute, and its potential repercussions pose significant challenges for executing malware in laboratory settings. Consequently, prior open datasets rely on isolated virtual sandboxes to run malware, resulting in data that is not representative of malware behavior in the wild.

This paper presents RaDaR, an open real-world dataset for run-time behavioral analysis of Windows malware. RaDaR is collected by executing malware on a real-world testbed with Internet connectivity and in a timely manner, thus providing a close-to-real-world representation of malware behavior. To enable an unbiased comparison of different solutions and foster multiple verticals in malware research, RaDaR provides a multi-perspective data collection and labeling of malware activity. The multi-perspective collection provides a comprehensive view of malware activity across the network, operating system (OS), and hardware. On the other hand, the multi-perspective labeling provides four independent perspectives to analyze the same malware, including its methodology, objective, capabilities, and the information it exfiltrates. To date, RaDaR includes 7 million network packets, 11.3 million OS system call traces, and 3.3 million hardware events of 10, 434 malware samples having different methodologies (3 classes) and objectives (9 classes), spread across 30 well-known malware families.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

CCS CONCEPTS

• **Security and privacy → Malware and its mitigation**; **Network security**; **Operating systems security**; • **Computing methodologies → Machine learning**.

KEYWORDS

Artificial Intelligence for Cyber-Security, Datasets, Malware Analysis, Run-time behavior

ACM Reference Format:
Sareena Karapoola, Nikhilesh Singh, Chester Rebeiro, and V. Kamakoti. 2018. RaDaR: A Real-World Dataset for AI Powered Run-time Detection of Cyber-Attacks. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Cyber-attacks worldwide have increased at an alarming scale, affecting at least 60% of enterprises worldwide in 2021 [7]. The consequences of these attacks vary from data loss to reputation damage, business disruption, financial loss, extortion, and even sabotage of critical infrastructures. The instrumental tool that enables adversaries to execute such a wide range of offensive maneuvers is *malware*. Despite the decades-long research in malware detection, the increasing number of attacks and their sophistication indicates that the problem is far from solved.

Over the last two decades, there have been several attempts to use Artificial Intelligence (AI) for malware detection that rely on datasets of malware samples [3, 15, 17, 18, 20, 22, 33, 40, 42, 45, 46, 48]. However, most published works typically use private datasets. Each dataset captures different malware features such as static strings in the malware binaries or dynamic run-time behavior, including network communications, system calls invoked at the operating system (OS), or hardware events. Furthermore, the size of the datasets across different papers vary from as low as 500 samples [2, 9, 21] to 1.1 million malware samples [1, 39, 42, 45]. Given the diversity in the datasets, it is not easy to make a fair comparison across the different detection approaches.

A few organizations, such as Endgame [1] and Microsoft [39] have attempted to address the issue mentioned above by creating large-scale open malware datasets containing static features from malware binaries. These datasets facilitate static analysis that infers maliciousness using signatures extracted from the malware binaries,

for example, strings in the binaries. These datasets have inherently become standard benchmarks to compare different detection techniques. However, detection techniques based on static analysis are easily evaded by packing and obfuscating the binaries as is becoming popular in polymorphic and metamorphic malware [34, 35]. Dynamic analysis that executes malware and analyzes their run-time behavior has recently gained traction over static analysis due to its ability to counter packing and obfuscation [35]. Such dynamic analyses, powered with AI, are increasingly adopted for their ability to detect zero-day[1] malware. Primarily, AI enables the modeling of benign behavior and the identification of anomalies to facilitate the detection of such malware. Unfortunately, to date, there are no *real-world* datasets to compare different dynamic analysis techniques. This is primarily due to multiple orthogonal challenges in executing malware to create such a dataset.

- **C-1.** Malware execution typically requires *real-world environments*, failing which it can choose not to execute, remaining stealthy. Specifically, modern malware looks for triggers typically present in virtualized environments to detect and evade analysis.
- **C-2.** Malware execution heavily depends on its *communications to remote servers*, known as command-and-control (C&C) servers, that guide and instruct the malware on its subsequent actions. This requires an active Internet connection when the malware is executed.
- **C-3.** In many cases, the C&C servers associated with a malware sample are short-lived. They are pulled down within a few months after the malware is first reported [27], rendering later executions of the malware futile.
- **C-4.** While it is important to execute malware in connected environments, it can be catastrophic if the malicious impact is not *contained*.

Hence, executing malware in real-world and connected environments in a timely manner (before its short-lived C&C servers are unavailable), is essential to capture a *precise* representation of malware behavior, while ensuring the containment. Multiple prior works have attempted to build datasets of run-time behavior of malware [3, 15, 17, 18, 20, 22, 33, 40, 42, 45, 46, 48]. However, they are either not precise due to the use of virtualized and non-connected environments [17, 18, 20, 22, 33, 42, 45, 48], or are not open to the research community [3, 15, 18, 20, 40, 46, 48].

This paper presents RaDaR, an open and growing real-world dataset for run-time behavioral analysis of Windows malware. RaDaR dataset is precise as it is collected by executing malware samples in a timely manner (C-3), on a real-world testbed (C-1) with Internet connectivity (C-2), while containing their malicious impact (C-4). To ensure a real-world environment, the testbed employs a network of physical machines connected to the Internet to execute malware. For timely execution, RaDaR uses an automated framework that periodically downloads the latest samples from online repositories [47] and executes them on the testbed to collect the run-time trails. Thus, the framework ensures that RaDaR is regularly updated with newer malware samples. Finally, the framework uses a dedicated Internet connection and a two-level firewall, which allows the malware to operate while containing its spread.

[1] A new malware that is never reported before.

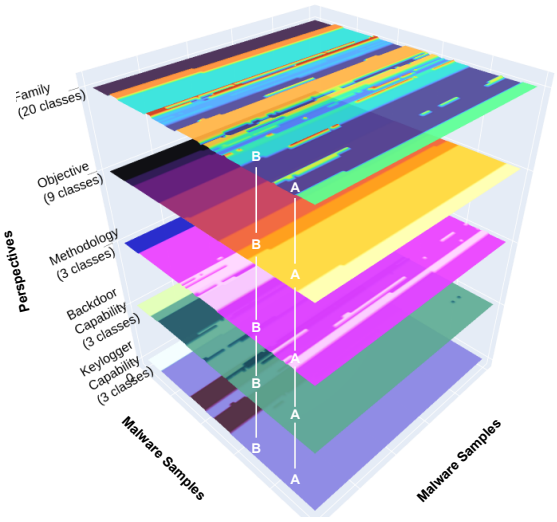

Figure 1: Independent perspectives of analyzing malware in RaDaR **and their class boundaries. Each surface is a grid of 10,000 points, where each point represents a malware sample, and its color indicates its class based on the perspective. For instance, the common corpus of 10,000 samples belongs to 20 malware families. However, they can be grouped into 9 classes based on their attack objective or 3 classes based on their methodology. These classes overlap. For instance, two malware samples, A and B of the same family, have different attack objectives and methodologies.**

RaDaR can foster different verticals in malware research with a multi-perspective data collection and labeling of malware behavior. A *multi-perspective collection*, with simultaneous capture of network, OS, and hardware behavior, enables a *fair comparison* of different solutions based on these trails. Such a comparison is infeasible today as the set of malware samples that these solutions use in their respective datasets are not consistent. Further, such a collection has the potential to enable multi-featured analyses, as malware classes differ in their functionality, leaving varying run-time trails at network, OS, and hardware.

Another critical aspect not addressed in prior works is the *multi-perspective labeling* of malware. The same malware can be labeled differently based on attributes such as its attack objective, methodology used to infect the victim, capabilities, the information they exfiltrate, or their family (i.e., code lineage). For instance, spyware and ransomware have different objectives but may share the same methodology for infection. Further, some malware may have few capabilities in addition to their main objective, such as stealthily logging user keystrokes. Hence, it is beneficial to analyze these malware attributes *independently* to draw clear class boundaries. Unlike prior works [15, 17, 20, 22, 33, 42, 46, 48], which propose a single perspective to label malware (e.g., family), we propose four independent perspectives, namely objective, methodology, additional capabilities, and the information exfiltrated by the malware. Figure 1 illustrates the class distribution and boundaries of 10,000 malware samples based on these perspectives. While the samples fall into 20 classes based on their family, they can be grouped into 9 classes

based on their objective[2]. As the class boundaries for the same set of samples vary widely based on the perspective, independent analysis of these perspectives is beneficial for effective detection. Such multi-perspective labeling enables designing specialized solutions and novel countermeasures, thus facilitating multiple verticals in malware research.

Following are the major contributions of this paper:

(1) RaDaR presents an open dataset of real-world behavior of malware samples from 2016 till date collected in a timely manner, with mechanisms for regular updates, providing multiple perspectives of malware behavior.
(2) RaDaR provides simultaneous capture of run-time malware behavior observable at network, OS, and hardware enabling multi-dimensional analysis and fair comparison of different solutions.
(3) We propose four independent perspectives to label malware, including its methodology, objectives, capabilities, and the type of information it exfiltrates.
(4) To date, RaDaR contains 2.7 terabytes of data with 7 million network packets, 11.3 million OS system call traces, and 3.3 million hardware events of 10, 434 malware samples having different methodologies (3 classes), objectives (9 classes), and spread across 30 well-known malware families.

Following is the organization of rest of the paper. Section 2 provides the necessary background for the paper. Section 3 presents the data collection framework. Section 4 discusses the multiple perspectives of malware behavior that RaDaR provides. Section 5 presents the RaDaR dataset and its class distributions. Section 6 presents our evaluations on the dataset. Section 7 presents the related work. Finally, Section 8 concludes the paper.

2 BACKGROUND

Malware is a program with malicious intent, which can vary widely in *objective* from popping up annoying advertisements (adware), downloading malicious applications (downloader), exfiltrating sensitive data (spyware), stealing financial credentials (banker), mining crypto-currencies (cryptominer), opening a stealthy access pathway for the attacker (backdoor), to sabotaging the entire system (ransomware). They can adopt different *methodologies* to enter the victim, such as being bundled with other legitimate software, spam emails that trick the users into downloading malicious files from infected removable drives, or by exploiting vulnerabilities.

For identification and analysis, security researchers use different taxonomies to label each malware sample. Malware is traditionally known based on its *type* as listed in Table 1 [36]. These names are based on their unique propagation methodologies (for e.g., trojan, virus) or their attack objective (for e.g., ransomware). Alternatively, they are known by their *family*, which is a collection of malware produced from the same code base and authors, and may have similar filename references, language, or C&C infrastructures. Typically, Anti-Virus (AV) companies also include in the label, a string that indicates the platform (Windows, Linux, etc.), type (Table 1), and family[16]. For instance, `Win64.Trojan.NukeSped.A_` sample is a 64-bit Windows executable trojan belonging to `NukeSped` family.

[2]The classes based on objective include benign applications, ransomware, spyware, backdoor, banker, cryptominer, deceptor, downloaders, and PUAs.

Table 1: Malware taxonomy based on its type or common-name [15, 17, 20, 35, 36, 46]

Class	Description
Trojan	A type of malware that downloads onto a computer disguised as a legitimate program.
Virus	A program that can copy itself and infect a computer without the knowledge of the user via infected removable drives.
Worm	A type of malware that typically exploits vulnerabilities to spread by making copies of itself from computer to computer
Bot	A self-propagating malware capable of infecting large number of hosts and taking complete control over a computer.
Spyware	A type of malware that infiltrates the victim and keeps gleaning sensitive information for an extended period.
Adware	A type of malware that pops-up annoying advertisements and inappropriate contents.
Downloader	A type of malware that downloads other malware on the victim.
Ransomware	A type of malware that can sabotage user files and extort a ransom from the user for restoration
Cryptominer	A type of malware that exploit computing resources of victim to mine cryptocurrencies
Backdoor	A type of malware that bypasses access control and grants an alternate covert pathway to resources at the victim.

Malware Analysis. Malware analysis is typically done in two ways. *Static analysis* examines malware binaries statically without executing them to extract signatures and imply its maliciousness. However, such static signatures can be easily thwarted by techniques that change the malware binary without affecting its functionality or run-time behavior. For instance, packing used in the popular polymorphic malware encrypts the contents of the binary, whereas obfuscation modifies the binary to create different copies of the same malware [34]. In contrast, dynamic analysis executes the malware and analyzes the run-time trails observable on the system stack. Consequently, it can counter the packing and obfuscation techniques that typically foil static analysis. Further, its potential to facilitate non-signature-based approaches that compare the run-time behavior of malware and benign applications makes it capable of detecting even zero-day malware.

Malware run-time behavioral trails are typically collected at network [3], OS [18], or hardware [40]. The network data capture all malware communications, including that to its C&C, whereas OS data captures its attempt to remain stealthy, achieve persistence, and execute its objective. More recently, researchers have explored the potential of micro-architectural events (e.g., number of cache misses) to detect malware [40]. These hardware events are measurable using hardware performance counters (HPC) [19] available in most modern microprocessors. Researchers rely on these behavioral trails to analyze and detect malware.

Evasion. While dynamic analysis is more powerful than static analysis against packing and obfuscation, modern malware have evolved to identify and evade even dynamic analysis environments.

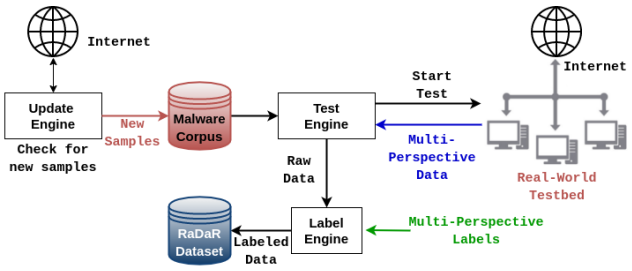

Figure 2: Automated Framework for Data Collection in RaDaR

Specifically, they look for artifacts (such as the presence of a virtual machine) to identify analysis environments and choose to remain dormant. To this end, the precise collection of malware behavior requires executing malware in real-world environments.

3 DATA COLLECTION FRAMEWORK

The trails captured by executing a malware sample are precise if they closely represent its real-world behavior. This section presents the framework used for collecting the RaDaR dataset and describes how it addresses the challenges C-1 to C-4 to ensure the precision of the collected trails.

Figure 2 describes the data collection framework, which has three engines, namely the update, test, and label engines. The update engine downloads the malware samples for analysis, whereas the test engine executes them on a real-world testbed to collect their run-time trails. Finally, the label-engine labels the collected trails based on different perspectives. Next, we discuss how the framework addresses the challenges in precisely collecting malware behavior.

Real-World Testbed (C-1). To prevent malware from evading analysis, the environment should be close to that of the real-world, which includes non-virtualized physical systems, preferably a heterogeneous network of devices, and Internet connectivity. The test engine employs a real-world testbed designed in our laboratory with a network of 512 different physical machines to execute malware. These machines include Windows and Linux desktops and single-board computers (Raspberry Pi and Intel Galileo boards). This heterogeneity of devices ensures that the testbed is likely to have sufficient triggers for malware execution similar to conditions in the wild.

Further, the execution of each sample can affect the system state, such as files, registry, and micro-architectural components. Since these modifications made by prior samples can affect subsequent analysis, we start every execution in a clean initial state of the system. To achieve this, the testbed has a quick state-reset mechanism that resets the machines to their initial states before executing every sample.

Internet Connectivity (C-2) and Containment (C-4). To provide connectivity while containing the malware impact, we use a dedicated Internet connection (ERNET [10]) for the framework that is isolated from our university network. Further, the testbed connects to the Internet via a two-level firewall that is placid on incoming communications while extensively scrutinizing all outgoing communications for malicious behaviors such as Denial-of-Service attempts, network scans, and spam emails. On detecting such malicious outgoing communications, the firewall blocks them. Thus, the testbed allows malware communications with their remote C&C servers while preventing the malicious impact from permeating outside the testbed.

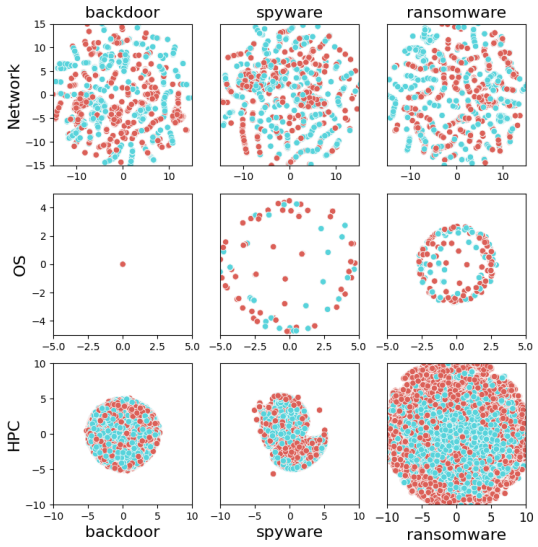

Figure 3: Distinguishability of backdoor, spyware, and ransomware run-time trails from that of benign applications at network, OS, and hardware. We capture these trails by executing 1000 samples of each class on the real-world testbed. The trails are pre-processed to extract 40 **features from the network (e.g., number of flows),** 9 **features from OS (e.g., write to file), and** 56 **features from the hardware (e.g., L1 cache misses) trails.**

Automated Timely Execution of Malware (C-3). The framework in Figure 2 is automated to collect behavioral data of malware in a timely manner when their short-lived C&C servers are likely to be active. The update engine periodically crawls public malware repositories for newly reported samples and downloads them to the malware corpus. The addition of new samples to the corpus triggers the test engine, which executes the latest samples from the corpus on the real-world testbed. The testbed collects the behavioral trails, which are later labeled and added to the RaDaR dataset. Thus the framework ensures a regular feed of new malware samples reported in the wild, which are analyzed immediately and updated to the RaDaR dataset.

4 MULTI-PERSPECTIVE ANALYSIS

RaDaR presents multiple perspectives of malware execution, including data collection observed at network, OS, and hardware, and different ways of labeling them. While multi-perspective collection provides a comprehensive view of malware activity in the system, multi-perspective labeling provides different perspectives to analyze the same malware. This section discusses the need for different perspectives to collect and label malware behavior.

4.1 Multi-Perspective Collection

Malware classes differ in objectives and functionalities and can leave varying trails in the network, OS, and hardware. Figure 3 presents a t-SNE visualization [49] of behavioral features that is

indicative of the distinguishability of backdoor, spyware, and ransomware trails from that of benign applications. As seen in the figure, some malware classes are more easily identifiable using one trail than the others. For instance, backdoor functionality involves consistent communications to its remote adversary to create a covert access pathway, leaving strong indicators in the network. Hence, backdoor behavior is most distinguishable from benign in the network compared to the OS and hardware. On the other hand, spyware functionality, which involves reading files and gleaning sensitive information, is most distinguishable in the OS and network compared to the hardware. Likewise, ransomware is most distinguishable in the hardware as it triggers distinct hardware micro-architectural events when it encrypts a large number of files in the target.

Prior works have extensively explored run-time trails for malware detection, typically using one of the network, OS, or hardware trails [3, 5, 15, 17, 18, 20, 22, 33, 42, 46, 48]. Each of these works presents highly acceptable results using the respective trails. However, a comparison of this large body of research to evaluate the capabilities of different trails is infeasible today, as every work uses execution trails of different samples collected in different timeframes and environments. A fair comparison of these solutions requires a *comprehensive view* of malware run-time activity in the system. While few datasets present a combination of network and OS trails [17, 20, 33, 42], we argue that the hardware perspective is also needed to build a comprehensive view of malware behavior, especially for classes like ransomware. Further, the differences in capabilities of run-time trails provide opportunities to explore more sophisticated ensemble-based approaches.

To facilitate multiple run-time perspectives, the testbed in Figure 2 *simultaneously* captures the network, OS, and hardware trails during malware execution. We use `tshark` [32] and Windows Process Monitor [31] to capture network and OS trails, respectively. On the other hand, we develop a customized Windows driver based on existing works to measure Hardware Performance Counters [13]. The network logs are collected at the gateway connecting the testbed to the Internet since all the network traffic is routed through it. The traffic can be attributed to different machines in the testbed based on the IP address. On the other hand, OS and hardware behavior are collected locally at the machine executing the malware, and are attributed to the malware based on its process identifier.

4.2 Multi-Perspective Labeling

Contemporary malware research typically labels malware based on type, family, or AV-labels (Refer Section 2) [15, 18, 20, 22, 33, 42, 43, 45, 46]. However, malware samples are diverse with multiple attributes, making it favorable to label malware with different perspectives. Further, some of these attributes may overlap, warranting an independent evaluation of each perspective.

Malware diversity and attributes. A malware sample can be characterized by a tuple of its attributes as ⟨methodology, objective, capabilities, family, information it exfiltrates⟩, each of which can vary widely as discussed next.

(1) Methodology. Malware can adopt different methodologies to infect the victim and propagate to other systems. Accordingly, they can be a virus, trojan, or worm. A virus is a malicious piece of code that attaches to a host program to get executed. It is transmitted from one computer to another through the host program. On the other hand, trojans and worms are standalone programs. While trojans require user interactions for activation and propagation, worms can self-activate and self-replicate via the network.

(2) Objective. Malware can have different attack objectives based on which it can be adware, downloader, spyware, banker, cryptominer, backdoor, botnet, or ransomware (Refer Section 2). Accordingly, they pose varying levels of risk to users. For instance, ransomware that sabotages the system is a high-risk malware, whereas adware that merely pops up user-annoying advertisements is a low-risk malware.

(3) Capabilities. Apart from the main objective, we observe that some malware may also have other capabilities. Some malware have *key-logging* capability to log user inputs, while others may have a hidden *backdoor* that opens an alternate access pathway for the attacker, in addition to their primary objective.

(4) Information Exfiltrated. We observe that every malware either steals or destroys some information of the target. The exfiltrated information typically includes one or more of the following | **(i)** System details (e.g., version of OS and system settings to identify analysis environments for evasion); **(ii)** User credentials; **(iii)** Keystrokes; **(iv)** Application passwords; **(v)** Details of email accounts; **(vi)** Clipboard and screenshots; **(vii)** Digital certificates; **(viii)** File-system contents; **(ix)** Process and hardware details; **(x)** Network-related details, including active ports and other systems in the network; **(xi)** Online activities of the user; and **(xii)** location and language.

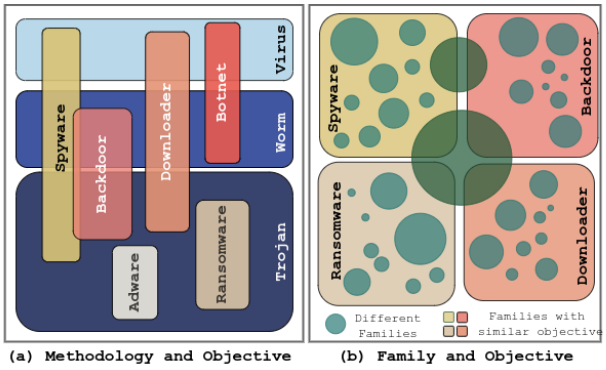

Figure 4: Overlapping malware attributes: (a) Methodology and Objective: The horizontal blue shaded bars (trojan, worm and virus) differ in the methodology, while the vertical bars (e.g. spyware, backdoor, and downloader) differ in the objective of the attack; (b) Family and Objective: The green circles indicate families of different sizes. The rectangular boxes indicate the families having the same objective.

Overlapping Attributes. While malware samples have different tuples characterizing them, they share some commonalities due to overlapping attributes. For instance, spyware that exfiltrates data could be implemented using any methodology (trojan, worm, or virus). Figure 4a illustrates this overlap with a distribution of

Table 2: Features observed at network, operating system and hardware.

	Category and Features
Network [3, 14, 38]	**Connection-based** (Destination IP and Port; Protocol; Total number of flows, packets per flow; Number of inbound and outbound packets; Average and standard deviation duration; Ratio of sizes of packets from originator and responder; Ratio of established states.); **TLS-based** (Ratio of TLS and non-TLS connection records; Ratio of TLS and SSL version in connection record; SNI and destination IP comparison; Ratio of self-signed certificates; Ratio of SSL records having SNI; Ratio of self-signed certificates; Average of certificate paths) **Certificate-based** (Average length of certificate keys; average and standard deviation of certificate validity; Validity of certificate period; number of certificates; number of domains in certificate; Ratio of certificate records and SSL records; the presence of Server Name Indication in subject alternative name) **HTTP-based** (URL path length; Number of URL query parameters; Filename length; Inter-arrival time; Number of URL flows; Number of downloaded and uploaded bytes; Number of files; Ratio of digits, Alphabets, Special characters, Upper/lower case, and Vowels in filename, URL, and Hostname respectively.)
OS [4, 6, 8]	**Registry** (Read, Query or Write to Windows Registry); **File** (Read a file, Create/Write to a file, Lock/Unlock a file); **Security** (Retrieve/change security descriptors of files, Add new CLSID2, Modify existing CLSID); **Process related** (Process Start, Process Exit, Load an image, Thread create, Thread exit); **Path-indicator** (Encoding of the path of the resource accessed, Prior knowledge of association of the filename, or directory path with malware); **Network** (TCP Send/Receive, UDP Send/Receive, Length of data); **Parameters and return values** (Length of data read/write,Desired Access rights for the requested resource, Shared Read, Write or Delete, Encoding of the options including Open Reparse Point, Synchronous I/O Alert etc, return values)
Hardware [40]	**Hardware Performance Counters (HPCs)** (54 HPCs such as: Core clock cycles; Instructions retired; Instruction length decoder stalls micro-operations from loop stream detector, decoders and micro-operation cache; resource stalls; Branches taken; Mispredicted branches; Register moves eliminated; Register moves elimination unsuccessful; L1-data cache misses; L1-data cache replacements Instruction-TLB miss, L2 requests, DTLB store/load misses leading to a walk; Hardware interrupts received; Instruction cache misses); Memory load LLC hits/misses.)

malware samples based on their methodology (blue shaded horizontal bars) and objective (vertical bars). The horizontal and vertical bars are disjoint individually among themselves, but together, they overlap and can significantly affect the accuracy of classification.

Similarly, Figure 4b illustrates the overlap between attributes of family (green circles) and objective (rectangular boxes). As evident, many families can share the same attack objective. For instance, `Corebot` [37], `Delf` [12], and `Formbook` [29] are all backdoor families. Further, a family may have malware samples of different objectives (i.e., circles overlapping two or more boxes). For instance, `Bladabindi` is a family of samples that can be a backdoor or spyware [30]. Figure 1 shows the significant variations between classes based on these attributes of a corpus of 10,000 malware samples. Thus, it is beneficial to analyze different malware attributes *independently* to draw clear class boundaries for effective classification.

Labeling in RaDaR**.** To facilitate multi-perspective analysis, RaDaR labels malware with four different independent attributes in addition to its family, namely, methodology, objective, two capabilities including keylogger and backdoor, and the information it exfiltrates. Each of these attributes presents different perspectives of malware and can aid in designing *specialized* solutions such as mechanisms to prevent malware infection (based on methodology) or attack mitigation (based on objective). Specifically, an objective-based perspective can enable multi-dimensional models to improve detection accuracy (refer Section 4.1) and customized responses, based on user tolerance to false positives. For example, users would prefer the termination of high-risk malware (such as ransomware) as soon as possible, to minimize the attack impact. On the other hand, they would not want the termination of low-risk classes like adware unless the detection is highly precise in order to minimize the false positives. Likewise, a capability-based perspective can help design specialized keylogger or backdoor detectors, whereas an information leak-based perspective can help implement appropriate data protection mechanisms.

In contrast, the single perspective of type, family or AV-labels in prior works [15, 18, 20, 22, 33, 42, 43, 45, 46], limits the scope of analyses possible on such datasets. Further, both type and family-based perspectives can result in fuzzy class boundaries, thus affecting the classification accuracy. Specifically, type-based perspective (Refer Table 1) mixes the attributes of methodology and objective (Figure 4a). On the other hand, different families can have similar functionalities and behavioral trails, while others may have distinctly behaving malware samples in the same family (Figure 4b). Such a characteristic of family-based perspective can be attributed to its definition, which is more indicative of code lineage and static features than run-time behavior. In contrast, AV-based labels often include specific meta-data in addition to type and family names, making them too specific for any generalization [41].

5 RADAR DATASET

The RaDaR dataset to date contains the behavior of 10,434 malware samples from 2016 obtained from Anti-Virus companies and public

malware repositories using the automated framework discussed in Section 3. In this section, we first describe the snapshots and features in the RaDaR dataset. We next present the class distributions of different perspectives in RaDaR .

Raw Behavioral Snapshots. As described in Section 2, the logs capture the time-series behavior of malware execution observable at network, OS, and hardware. Network logs contain the network packets from the machine executing the malware. In contrast, the OS logs capture all the system call traces of the malware, including its file, registry, process, and other operations. On the other hand, the hardware logs contain the values of hardware performance counters at a periodic interval of `100` ms. To date, RaDaR has 2.7 tera-bytes of data, including 7 million network packets, 11.3 million OS system call traces, and 3.3 million hardware events.

Features. RaDaR extracts a comprehensive set of features about malware execution as listed in Table 2. For the network data, we use the Zeek [51] tool to pre-process the packet-level logs into network *flow* summaries. A network flow comprises of all communications that share the same source and destination IP addresses and ports. In total RaDaR has 60K network flow summaries and 58 features. We use custom scripts to parse the OS and hardware logs. In this way, we extract 11 features for OS and 54 features for hardware. The data is converted to a matrix, where the rows are the behavioral snapshots and columns are the feature values. Thus, RaDaR has 3 matrices of order $60K \times 58$ for network, $11.3M \times 11$ for OS, and $3.3M \times 54$ for hardware, and each row in these matrices are labeled as per the perspectives.

Class Distributions. Table 3 shows the distribution of malware based on different perspectives in the RaDaR dataset. Most malware samples in the dataset belong to the trojan class (71%), which is similar to the distribution of malware in the real-world [44]. Similarly, RaDaR contains representative classes having different objectives, including banker, downloader, Potentially Unwanted Applications (PUA), deceptor, spyware, backdoor, ransomware, and cryptominers. While 20.3% of malware samples in RaDaR can log keystrokes of the users in addition to their primary objective, 19.1% of them have backdoor capabilities. A graphical representation of this distribution and the significant overlaps between malware classes across perspectives is shown in Figure 1.

Table 3 also provides the number of behavioral snapshots present for each class per perspective across network, OS, and hardware in RaDaR. As evident, the distribution of network, OS, and hardware snapshots may not match the distribution of malware in the dataset, as each malware differs in its activity across the three system components.

Table 4 lists the distributions of malware families in the dataset. RaDaR contains 30 families that can be grouped into 9 classes based on the objective of the malware. Table 5 shows the distribution of samples based on the information the malware collects and steals from the target. Most malware (> 51%) collect the system information to help it identify virtualized analysis environments. While 19% of malware log keystrokes, 6% and 3% of samples capture the screenshots and clipboard.

Table 3: Class distributions of different perspectives in RaDaR

Perspective	Class	%age	Number of Snapshots		
			Network	OS	Hardware
Methodology	Trojan	71.5%	36K	7.7M	2.4M
	Worm	18.6%	14K	2.6M	413K
Objective	Cryptominer	4.7%	992	293K	158K
	Banker	13.4%	4878	772K	51K
	Spyware	15.3%	11588	1.9M	59K
	Backdoor	12.9%	7845	1.5M	371K
	Ransomware	7.2%	2239	807K	182K
	PUA	10.3%	7152	2M	914K
	Downloader	18.5%	9277	1.7M	502K
	Deceptor	5.4%	4617	440K	478K
	Benign	9.9%	8964	1.9M	578K
Capabilities	Keylogger	20.3%	8618	1.8M	108K
	Non-Keylogger	69.7%	39K	8.1M	2.6M
	Backdoor	19.1%	15.1K	3M	412K
	Non-Backdoor	71%	33.4K	7.1M	2.3M

Table 4: Families in RaDaR

Objective		Family
Class	**Count**	**Class (#Count)**
Downloader	1991	Agent (1898), Chindo (69), Small (16), XeyoRAT(6), crossza(2)
Banker	1442	Emotet (1442)
PUA	1106	pua (1106)
Ransomware	770	Gandcrab(350), cyclone (305), Ryuk (40), Rapid (29), Ouroboros (26), Sigma (20)
Spyware	1639	Bladabindi(946),Agent (350), Vools (252), Buhtrap (77), Kryptik (14)
Backdoor	1388	Corebot (320), Formbook (250), Agent (441), Delf (377)
Deceptor	585	Deceptor(585)
Cryptominer	500	Coinminer(500)
Dropper	260	Agent (226), NukeSped (33), Delf (1)

Table 5: Distribution of information exfiltrated in RaDaR

Information	%	Information	%	Information	%
System	51.31	Accounts	2.45	Location	3.71
User	20.04	Keystrokes	19.83	Language	3.69
Network	7.00	Screenshots	6.29	Online trails	0.13
Filesystem	8.80	Passwords	3.17	Certificates	0.85
Hardware	10.67	Clipboard	3.19	Unknown	21.45
Process	2.40				

6 RESULTS

In this section, we present the results of our evaluation on RaDaR. For our experiments, we apply Principal Component Analysis (PCA) to reduce the feature space to 10 network features, 10 OS features, and 20 features from the hardware. For each perspective, we split the dataset in 70:15:15 ratio corresponding to train, validate and test sets, with an even distribution of classes.

Table 6: Results of the evaluations on RaDaR

Perspective	Class	Best F1-Score			Best Model		
		Network	OS	Hardware	Network	OS	Hardware
Methodology	Benign	0.998	**1**	1	RF	RF	LGBM
	Trojan	0.856	**0.999**	0.988	RF	RF	LGBM
	Worm	0.866	**0.999**	0.988	RF	RF	LGBM
Objective	Cryptominer	0.83	0.87	**0.94**	XGB-Binary	LGBM-Binary	XGB-Binary
	Banker	**0.9**	0.82	0.67			
	Spyware	0.89	**0.92**	0.7			
	Backdoor	**0.92**	0.8	0.75			
	Ransomware	0.86	0.7	**0.93**			
	PUA	**0.88**	0.68	0.56			
	Downloader	**0.99**	0.84	0.68			
	Deceptor	**0.99**	0.87	0.66			
Capabilities	Benign	0.98	0.999	**0.999**	XGB-OVR	XGB-OVR	XGB-OVR
	Non-Keylogger	0.892	0.994	**0.999**			
	Keylogger	0.875	0.999	**0.999**			
	Benign	0.998	**0.999**	0.8	LGBM-OVR	RF	XGB-OVR
	Non-Backdoor	0.930	**0.998**	0.98			
	Backdoor	0.935	**0.998**	0.96			
Family (e.g. Backdoor)	Corebot [37]	0.340	0.050	0.870	XGB-OVR	RF	XGB-OVR
	Delf [12]	0.390	0.153	0.982			
	Agent [11]	0.682	0.151	0.458			
	Formbook [29]	0.496	0.316	0.401			

RF: Random Forest XGB: XGBoost LGBM: LightGBM XGB-Binary/LGBM-Binary: Binary Models comparing benign and an objective class
OvR: One-versus-Rest OvO: One-versus-One

Methodology. To evaluate the detection of methodology, we train standard multi-class machine learning (ML) models including Decision Tree [23], k-Nearest Neighbours [25], Logistic regression [24], Random-Forest [26], XGBoost [50], and LightGBM [28]. For XGBoost and LightGBM, we consider both the one-versus-one (OvO) and one-versus-rest (OvR) configurations for multi-class classification [28, 50]. Table 6 presents the best F1-Score observed for detecting methodology using network, OS, or hardware trails. Methodology of a malware is best detected using OS and hardware features as compared to network. Intuitively, methodology deals with how malware infects a system and activates itself, and hence, OS and hardware trails have stronger indicators than the network trails. While Random-Forest offers the best F1-Score for detection at network and OS, LightGBM gives the best F1-Score at hardware.

Objective. We also observe that the detection F1-Score of objective using multi-class classifiers [23–26, 28, 50] was very low, which presents a wide scope for model improvements. In this regard, we next evaluate how different each objective class is from benign applications. To this end, we train specialized XGBoost binary models on each objective class and benign applications. Table 6 presents the results. Each malware class differs in the trails that best differentiate it from benign applications, as highlighted in the table. While the network trails are effective for most classes, OS trails are the best to detect spyware. Similarly, hardware trails are the most effective for detecting cryptominers and ransomware. We leave the exploration of complex models and ensemble-based approaches that can exploit these differences in run-time trails for future work.

Capabilities. We next evaluate detection of keylogging capability using standard multi-class models [23–26, 28, 50] on network, OS and hardware trails. The perspective has three classes namely, benign, non-keylogger, and keylogger. Interestingly, we find the keylogging capability is best detected using hardware and OS features, as shown in Table 6. Its detection F1-Score is the lowest in the network, as keylogging primarily involves intercepting the system calls to log the user keystrokes and does not involve any network activity.

Similar evaluation of backdoor capability using multi-class models [23–26, 28, 50] indicate that it is best detected with OS features, as compared to network and hardware (Table 6). This capability perspective has three classes, namely, benign, non-backdoor, and backdoor. The results are in contrast to the objective-based evaluation; wherein OS trails had lower detection F1-Score than network. We believe the models trained with the capability perspective are able to learn the traits of backdoor functionality better as compared to objective-based classes, which can have overlapping capabilities.

Family. Finally, we evaluate the relevance of family taxonomy for malware detection based on run-time behavior. As there are a large number of malware families (Table 4), we consider the example of backdoor families. Table 6 presents the results of detection F1-Score using standard multi-class classifiers on 4 backdoor families using the network, OS, and hardware trails. The results are sub-optimal. In essence, family is an indicator of code lineage and attribution and hence mainly useful for static analysis. In contrast, run-time behavior depends on malware functionalities, which is the same for all families of a particular class of malware and can affect classification.

Table 7: Comparison of prior works based on analysis environments and perspectives of data collection and labeling.

Dataset	Real World	Year of Samples	Year of Capture	Multi-Perspective Collection			Multi-Perspective Labeling									Open Dataset
				Network	OS	Hardware	Independent Attributes[#]	Binary	AV-label	Method	Objective	Family	Keylogger?	Backdoor?	Type of Information	
CAIDA [5]	✓	2001	2001	✓	-	-	✓[Ψ]	✓	-	✓[Ψ]	-	-	-	-	-	✓
[15]	✓	*	2010	-	✓	-	✗	✓	-	-	-	-	-	-	-	*
ISOT [33]	✗	*	2004-05, 2010, 2017	✓	✓	-	✓[Ψ]	✓	-	-	✓[Ψ]	✓	-	-	-	✓
CTU [22]	✗	*	2011	✓	-	-	✗	✓	✓	-	-	✓	-	-	-	✓
[46]	✓	2011	2011	✓	-	-	✗	✓	-	-	-	-	-	-	-	*
[20]	✗	2012-2013	2013	✓	✓	-	✗	✓	✓	-	-	-	-	-	-	*
ADFA [17]		*	2013	✓	✓	-	✗	✓	-	-	-	-	-	-	-	✓
UCI [45]	✗	2010-2014	2010-2014	-	✓	-	✗	✓	✓	-	-	-	-	-	-	✓
[18]	✗	*	2015	-	✓	-	-	✓	-	-	-	✓	-	-	-	*
[3]	✓	2015	2015	✓	-	-	-	✓	-	-	-	-	-	-	-	*
MalRec [42]	✗	2014-2016	2014-2016	✓	✓	-	-	✓	-	-	-	✓	-	-	-	✓
[40]	✓	*	2018	-	-	✓	-	✓	-	-	-	-	-	-	-	*
[48]	✗	*	2020	-	✓	-	-	✓	-	-	-	✓	-	-	-	*
RaDaR	✓	2016-2022[$]	2019-2022[$]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

✗ in Independent Attributes indicates the *type* taxonomy that mixes different attributes. Ψ Presents only one or two classes.
- Not Present * No information available or Not Open
$ Growing dataset till date.

7 RELATED WORK

Multiple prior works have proposed datasets of run-time behavior of Windows malware [3, 5, 15, 17, 18, 20, 22, 33, 40, 42, 45, 46, 48]. Table 7 compares them based on the environment they use to execute the malware, and the perspectives of data collection and labeling that they present.

Analysis Environments. Most works rely on virtual machines that are easily evaded by modern malware, and hence, are not representative of real-world behavior [17, 18, 20, 22, 33, 42, 45, 48]. On the other hand, the datasets generated in a timely manner under real-world conditions are not open [3, 15, 40, 46], or are least two decades old (2001) [5]. Such outdated datasets may not be relevant in the current malware landscape, as modern malware have evolved considerably. In contrast, the real-world testbed framework (refer Section 3) ensures a precise representation of malware behavior in the wild, while providing mechanisms to continually augment RaDaR with the latest malware samples (refer Section 5).

Data Collection. Most datasets lack a simultaneous capture of different trails of malware behavior, including hardware. They either present a single trail [3, 5, 15, 17, 18, 20, 22, 33, 40, 42, 46, 48], or a combination of network and OS trails [17, 20, 33, 42]. In contrast, the comprehensive view of malware activity across the system stack facilitates a fair comparison of different solutions and a multi-dimensional analysis of malware behavior as discussed in Sections 4.1 and 6.

Perspectives in Labeling. Prior datasets use the perspective of binary, type, family or AV-based strings to label malware [3, 5, 15, 17, 18, 20, 22, 33, 40, 42, 43, 45, 46, 48]. Binary datasets that classify samples into benign and malware are too restricted for any multi-dimensional analysis such as assessing risk, capabilities or forensics [3, 5, 15, 18, 20, 22, 33, 40, 42, 43, 46, 48]. On the other hand, the datasets based on type [15, 20, 45, 46] and family [18, 22, 33, 42] present only a single multi-class perspective, and can have fuzzy class boundaries due to overlapping attributes. While few open datasets present independent perspectives of objective or methodology of malware, they are limited to one or two classes (Ransomware and Botnet [33], Red Worm [5]), thus limiting the scope of analyses using such datasets. Finally, the AV-based perspective [41, 43] is too specific for any generalization, thus affecting the classification. In contrast, we present a dataset with four independent perspectives in addition to family: methodology, objective, additional capabilities including keylogging and backdoor, and the information it exfiltrates (Section 4.2). To the best of our knowledge, RaDaR is the first open dataset to capture precise malware behavior using real-world systems and provide diverse perspectives of its run-time activities.

8 CONCLUSION

This paper presents RaDaR, an open real-world dataset for malware behavioral analysis, with mechanisms to keep pace with the evolving malware landscape. RaDaR has multiple use cases for AI-based security research, including an unbiased comparison of detection approaches and the development of novel countermeasures incorporating multiple perspectives of malware execution. While the challenges in executing malware have resulted in datasets being largely private or restricted to the security researchers, we firmly believe that the open RaDaR dataset enables other communities, especially the data science researchers, to explore and analyze it.

REFERENCES

[1] Hyrum S. Anderson and Phil Roth. 2018. EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models. *CoRR* abs/1804.04637 (2018). arXiv:1804.04637 http://arxiv.org/abs/1804.04637

[2] Mohammad Bagher Bahador, Mahdi Abadi, and Asghar Tajoddin. 2014. HPCMalHunter: Behavioral malware detection using hardware performance counters and singular value decomposition. In *2014 4th International Conference on Computer and Knowledge Engineering (ICCKE)*. IEEE, 703–708.

[3] Karel Bartos, Michal Sofka, and Vojtech Franc. 2016. Optimized Invariant Representation of Network Traffic for Detecting Unseen Malware Variants. In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*. 807–822.

[4] Ulrich Bayer, Paolo Milani Comparetti, Clemens Hlauschek, Christopher Krügel, and Engin Kirda. 2009. Scalable, Behavior-Based Malware Clustering. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2009, San Diego, California, USA, 8th February - 11th February 2009*. The Internet Society.

[5] CAIDA. 2001. *CAIDA Analysis of Code-Red*. Retrieved March 10, 2022 from https://www.caida.org/archive/code-red/

[6] Davide Canali, Andrea Lanzi, Davide Balzarotti, Christopher Kruegel, Mihai Christodorescu, and Engin Kirda. 2012. A quantitative study of accuracy in system call-based malware detection. In *International Symposium on Software Testing and Analysis, ISSTA 2012, Minneapolis, MN, USA, July 15-20, 2012*, Mats Per Erik Heimdahl and Zhendong Su (Eds.). ACM, 122–132. https://doi.org/10.1145/2338965.2336768

[7] Sam Cook. 2022. Malware statistics and facts for 2022. Retrieved March 10, 2022 from "https://www.comparitech.com/antivirus/malware-statistics-facts/"

[8] Sanjeev Das, Yang Liu, Wei Zhang, and Mahinthan Chandramohan. 2016. Semantics-Based Online Malware Detection: Towards Efficient Real-Time Protection Against Malware. *IEEE Trans. Inf. Forensics Secur.* 11, 2 (2016), 289–302.

[9] John Demme, Matthew Maycock, Jared Schmitz, Adrian Tang, Adam Waksman, Simha Sethumadhavan, and Salvatore J. Stolfo. 2013. On the feasibility of online malware detection with performance counters. In *The 40th Annual International Symposium on Computer Architecture, ISCA'13, Tel-Aviv, Israel, June 23-27, 2013*. 559–570. https://doi.org/10.1145/2485922.2485970

[10] ERNET. 2022. *Education & Research Network*. Retrieved March 2, 2022 from https://ernet.in/

[11] F-Secure. 2021. *Trojan:W32/Agent*. Retrieved March 2, 2022 from https://www.f-secure.com/v-descs/agent.shtml

[12] F-Secure. 2021. *Trojan:W32/Delf*. Retrieved March 2, 2022 from https://www.f-secure.com/v-descs/trojan_w32_delf.shtml

[13] Agner Fog. 2022. *Software Optimization Resources*. Retrieved March 10, 2022 from https://www.agner.org/optimize

[14] Guofei Gu, Roberto Perdisci, Junjie Zhang, and Wenke Lee. 2008. BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection. In *Proceedings of the 17th USENIX Security Symposium, July 28-August 1, 2008, San Jose, CA, USA*, Paul C. van Oorschot (Ed.). USENIX Association, 139–154. http://www.usenix.org/events/sec08/tech/full_papers/gu/gu.pdf

[15] Hsien-De Huang, Tsung-Yen Chuang, Yi-Lang Tsai, and Chang-Shing Lee. 2010. Ontology-based intelligent system for malware behavioral analysis. In *FUZZ-IEEE 2010, IEEE International Conference on Fuzzy Systems, Barcelona, Spain, 18-23 July, 2010, Proceedings*. IEEE, 1–6. https://doi.org/10.1109/FUZZY.2010.5584325

[16] Médéric Hurier, Guillermo Suarez-Tangil, Santanu Kumar Dash, Tegawendé F. Bissyandé, Yves Le Traon, Jacques Klein, and Lorenzo Cavallaro. 2017. Euphony: harmonious unification of cacophonous anti-virus vendor labels for Android malware. In *Proceedings of the 14th International Conference on Mining Software Repositories, MSR 2017, Buenos Aires, Argentina, May 20-28, 2017*, Jesús M. González-Barahona, Abram Hindle, and Lin Tan (Eds.). IEEE Computer Society, 425–435. https://doi.org/10.1109/MSR.2017.57

[17] Impact. 2013. *IMPACT CyberTrust*. Retrieved March 2, 2022 from https://research.unsw.edu.au/projects/adfa-ids-datasets

[18] Mohammad Imran, Muhammad Tanvir Afzal, and Muhammad Abdul Qadir. 2015. Using hidden markov model for dynamic malware analysis: First impressions. In *2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*. IEEE, 816–821.

[19] Intel Corporation 2007. *Intel 64 and IA-32 Architectures Software Developer's Manual - Volume 3B*. Intel Corporation.

[20] Jae-wook Jang, Jiyoung Woo, Aziz Mohaisen, Jaesung Yun, and Huy Kang Kim. 2016. Mal-Netminer: Malware Classification Approach based on Social Network Analysis of System Call Graph. *CoRR* abs/1606.01971 (2016). arXiv:1606.01971 http://arxiv.org/abs/1606.01971

[21] Mikhail Kazdagli, Vijay Janapa Reddi, and Mohit Tiwari. 2016. Quantifying and improving the efficiency of hardware-based mobile malware detectors. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 1–13.

[22] Statosphere lab. 2013. *The CTU-13 Dataset*. Retrieved March 2, 2022 from https://www.stratosphereips.org/datasets-ctu13

[23] Scikit learn. 2021. *Decision Tree*. Retrieved March 2, 2022 from https://scikit-learn.org/stable/modules/tree.html

[24] Scikit learn. 2021. *Logistic Regression*. Retrieved March 2, 2022 from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[25] Scikit learn. 2021. *Nearest Neighbours*. Retrieved March 2, 2022 from https://scikit-learn.org/stable/modules/neighbors.html

[26] Scikit learn. 2021. *Random Forest Classifier*. Retrieved March 2, 2022 from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[27] Chaz Lever, Platon Kotzias, Davide Balzarotti, Juan Caballero, and Manos Antonakakis. 2017. A Lustrum of malware network communication: Evolution and insights. In *IEEE Symposium on Security and Privacy (SP), 2017*. IEEE, 788–804.

[28] LightGBM. 2022. *LightGBM*. Retrieved March 2, 2022 from https://lightgbm.readthedocs.io/en/latest/Python-Intro.html

[29] Malpedia. 2021. *Trojan:W32/Formbook*. Retrieved March 2, 2022 from https://malpedia.caad.fkie.fraunhofer.de/details/win.formbook

[30] Tomas Meskauskas. 2021. *What is BLADABINDI?* Retrieved March 2, 2022 from "https://www.pcrisk.com/removal-guides/18907-bladabindi-backdoor"

[31] Microsoft. 2022. *Process Monitor*. Retrieved March 2, 2022 from https://docs.microsoft.com/procmon

[32] Microsoft. 2022. *tshark*. Retrieved March 2, 2022 from https://www.wireshark.org

[33] University of Victoria. 2010. *ISOT Botnet and Ransomware Detection Datasets*. Retrieved March 2, 2022 from https://www.uvic.ca/ecs/ece/isot/datasets/botnet-ransomware/index.php

[34] Philip O'Kane, Sakir Sezer, and Kieran McLaughlin. 2011. Obfuscation: The Hidden Malware. *IEEE Security Privacy* 9, 5 (2011), 41–47. https://doi.org/10.1109/MSP.2011.98

[35] Ori Or-Meir, Nir Nissim, Yuval Elovici, and Lior Rokach. 2019. Dynamic malware analysis in the modern era—A state of the art survey. *ACM Computing Surveys (CSUR)* 52, 5 (2019), 1–48.

[36] Ori Or-Meir, Nir Nissim, Yuval Elovici, and Lior Rokach. 2019. Dynamic malware analysis in the modern era—A state of the art survey. *ACM Computing Surveys (CSUR)* 52, 5 (2019), 1–48.

[37] Danny Palmer. 2021. *CoreBot banking trojan malware returns after two-year break*. Retrieved March 2, 2022 from https://www.zdnet.com/article/corebot-banking-trojan-malware-returns-after-two-year-break/

[38] Roberto Perdisci, Wenke Lee, and Nick Feamster. 2010. Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces. In *Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2010, April 28-30, 2010, San Jose, CA, USA*. USENIX Association, 391–404. http://www.usenix.org/events/nsdi10/tech/full_papers/perdisci.pdf

[39] Royi Ronen, Marian Radu, Corina Feuerstein, Elad Yom-Tov, and Mansour Ahmadi. 2018. Microsoft malware classification challenge. *arXiv preprint arXiv:1802.10135* (2018).

[40] Hossein Sayadi, Nisarg Patel, Sai Manoj P. D., Avesta Sasan, Setareh Rafatirad, and Houman Homayoun. 2018. Ensemble learning for effective run-time hardware-based malware detection: a comprehensive analysis and classification. In *Proceedings of the 55th Annual Design Automation Conference, DAC 2018, San Francisco, CA, USA, June 24-29, 2018*. ACM, 1:1–1:6. https://doi.org/10.1145/3195970.3196047

[41] Marcos Sebastián, Richard Rivera, Platon Kotzias, and Juan Caballero. 2016. AVclass: A Tool for Massive Malware Labeling. In *Research in Attacks, Intrusions, and Defenses - 19th International Symposium, RAID 2016, Paris, France, September 19-21, 2016, Proceedings*. 230–253. https://doi.org/10.1007/978-3-319-45719-2_11

[42] Giorgio Severi, Tim Leek, and Brendan Dolan-Gavitt. 2018. Malrec: compact full-trace malware recording for retrospective deep analysis. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 3–23.

[43] Madhu K. Shankarapani, Kesav Kancherla, Subbu Ramamoorthy, Ram S. Movva, and Srinivas Mukkamala. 2010. Kernel machines for malware classification and similarity analysis. In *International Joint Conference on Neural Networks, IJCNN*

2010, Barcelona, Spain, 18-23 July, 2010. IEEE, 1–6.

[44] Statista. 2021. *Distribution of leading malware types worldwide in 2019.* Retrieved March 2, 2022 from https://www.statista.com/statistics/204434/most-frequent-malware-propagation-methods/

[45] UCI. 2017. *Dynamic Features of VirusShare Executables Data Set.* Retrieved March 2, 2022 from https://archive.ics.uci.edu/ml/datasets/Dynamic+Features+of+VirusShare+Executables

[46] Jorge Maestre Vidal, Ana Lucila Sandoval Orozco, and Luis Javier García-Villalba. 2017. Alert correlation framework for malware detection by anomaly-based packet payload analysis. *J. Netw. Comput. Appl.* 97 (2017), 11–22. https://doi.org/10.1016/j.jnca.2017.08.010

[47] VirusTotal. 2022. *VirusTotal.* Retrieved March 2, 2022 from https://www.virustotal.com/

[48] Qi Wang, Wajih Ul Hassan, Ding Li, Kangkook Jee, Xiao Yu, Kexuan Zou, Junghwan Rhee, Zhengzhang Chen, Wei Cheng, Carl A. Gunter, and Haifeng Chen. 2020. You Are What You Do: Hunting Stealthy Malware via Data Provenance Analysis. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society.

[49] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. 2016. How to use t-SNE effectively. *Distill* 1, 10 (2016).

[50] XGBoost. 2022. *XGBoost.* Retrieved March 2, 2022 from https://xgboost.readthedocs.io/en/stable/python/python_intro.html

[51] Zeek. 2022. *The Zeek Network Security Monitor.* Retrieved March 2, 2022 from https://www.zeek.org/