



[DATA ARTICLE TEMPLATE V.19 (DECEMBER 2024)]

ARTICLE INFORMATION

Article title

A Dataset of Windows Malware Execution Traces

Authors

Razvan Raducu, Alain Villagrasa-Labrador, Ricardo J. Rodríguez, and Pedro Álvarez*.

Affiliations

Computer Science and Systems Engineering Department, Engineering Research Institute of Aragón (I3A), University of Zaragoza, Spain.

Corresponding author's email address and Twitter handle

alvaper@unizar.es

Keywords

malware dynamic analysis; behavioral execution trace; system calls; Windows API;

Abstract

Malware continues to be a major cybersecurity concern, with increasing volume and sophistication making effective detection methods essential. Behavior-based approaches rely on high-quality execution trace data to analyze how malicious software interacts with systems during runtime. Publicly available datasets often lack sufficient detail, contain limited family diversity, or provide only simplified API call sequences. In this paper, we present a dataset that addresses this gap by offering a large collection of richly detailed Windows malware execution traces generated in controlled environments. It has been generated through automated dynamic analysis, executing the malware samples in a controlled virtualized environment, specifically, in the CAPEv2 Sandbox on Windows 10 virtual machines. The raw sandbox analysis reports have been then processed using the MALVADA framework, a modular Python-based pipeline that filters, structures, labels, and standardizes execution traces. The resulting dataset consists of 31,844 JSON execution trace files where each trace contains static metadata, dynamic behavioral information, and labelling fields. The dataset is suitable for reuse in multiple research contexts, including the development and benchmarking of malware detection methods, behavioral clustering, dynamic analysis of malicious software, and automated labelling studies. Its standardized JSON structure facilitates integration with existing data analysis and machine learning pipelines, as well as combination with other datasets for extended studies.

SPECIFICATIONS TABLE

| | |
|---------|-----------------------------------|
| Subject | Computer Sciences |
|---------|-----------------------------------|

| | |
|---------------------------------|--|
| Specific subject area | Cybersecurity, Malware analysis and recognition |
| Type of data | JSON format |
| Data collection | The dataset was generated executing malware samples in a virtualized controlled environment and monitoring the exhibited activity. Specifically, KVM was used to deploy several virtual machines (VMs) running Windows 10 x64, and the CAPEv2 sandbox to automate sample submission, execution and activity report generation. Each of these reports contain key events occurred during the sample execution: created processes, sequence of API invocations, used system resources, network communications, etc. Subsequently, the MALVADA tool was used to curate, filter and label these malware reports. The labelling was based on the functionality of the AVClass tool. |
| Data source location | The analyzed malware samples were obtained from several publicly available malware repositories like VX-underground, Malware Bazaar, VirusShare, theZoo, and MalShare. |
| Data accessibility | Repository name: WinMET (Windows Malware Execution Traces) Dataset Data identification number: [10.5281/zenodo.12647555] Direct URL to data: https://doi.org/10.5281/zenodo.12647555 |
| Related research article | Razvan Raducu, Alain Villagrasa-Labrador, Ricardo J. Rodríguez, Pedro Álvarez, <i>MALVADA: A framework for generating datasets of malware execution traces</i> , SoftwareX, Volume 30, 2025, 102082, ISSN 2352-7110, https://doi.org/10.1016/j.softx.2025.102082 . [1] |

33 VALUE OF THE DATA

- 34
- 35
- 36
- 37
- 38
- 39
- 40
- 41
- 42
- 43
- 44
- 45
- The Windows operating system is one of the main targets for malware [2], as it is the most used OS worldwide [3]. This dataset contains comprehensive behavioral information of approximately 32,000 execution traces of Windows malware samples. Each trace includes detailed contextual data such as API call sequences with arguments and return values, process trees representing spawned processes, accessed files and registry keys, synchronization objects, and network communications. This level of detail goes beyond simplified datasets that are available for the scientific community which typically only provide API names or identifiers.
 - This dataset has been generated using a reproducible process which is based on an ecosystem of open source and publicly available tools. The process is structured as a modular pipeline with clearly defined steps, including malware execution, report generation, incorrect report detection, duplicate filtering, sensitive data anonymization, trace generation in JSON format, malware family labeling, and statistical reporting. The reproducibility of this pipeline facilitates the replication of



46 the dataset creation process by other researchers, allowing the execution of new malware samples
47 and the generation of additional execution traces to enrich and update the dataset.

- 48 • Traces of the dataset have been labelled with metadata describing their corresponding malware
49 family. CAPE [4] and AVClass [5] have been used in the labelling procedure. These labelled traces
50 are particularly suitable for malware recognition and detection based on Artificial Intelligence (AI)
51 techniques. The dataset's scale (approx. 32,000 samples) and diversity of families and behaviors
52 support a variety of data-driven research tasks, including feature extraction, supervised and
53 unsupervised learning, and benchmarking of classification models. Additionally, JSON formatting
54 simplifies preprocessing for sequence modeling, graph-based learning, and statistical analysis.
- 55 • The dataset can be used by researchers in cybersecurity, data science, and software engineering.
56 Its detailed traces allow dynamic malware analysis, API sequence mining, labeling standardization,
57 and visualization of malware behavior. The standardized JSON format facilitates integration into
58 analytical pipelines, visualization tools, and machine learning workflows without requiring
59 complex conversions. The dataset includes both behavioral data and metadata, enabling its use in
60 multiple research contexts such as behavioral clustering, anomaly detection, attack
61 reconstruction, and tool development. The open and structured nature of the dataset supports
62 collaboration between different research areas and institutions.

63 BACKGROUND

64 The growing sophistication and frequency of cyberattacks have intensified the need for advanced
65 mechanisms able to prevent intrusions and accurately detect the type of malware responsible for
66 them. The development and evaluation of such mechanisms strongly depend on the availability of
67 high-quality datasets. However, existing malware execution datasets are often limited in size and the
68 type of contextual information that include. Consequently, there is a clear need for new datasets that
69 capture detailed behavioral traces of malware and are openly available to the research community.
70 Furthermore, the maintainability and extensibility of these datasets should be guaranteed over time
71 to support the continuous research progress and more robust and reliable detection strategies capable
72 of effectively responding malware evolution.

73 The theoretical background of this work is grounded in behavior-based malware analysis, which
74 examines sequences of API calls and system interactions to characterize and differentiate malicious
75 programs. Methodologically, malware samples are executed in a controlled sandbox environment to
76 monitor and capture their execution behavior. This behavior is subsequently structured in traces that
77 include metadata related to the characterization and typology of malware. These resulting traces are
78 the starting point to understand, interpret and analyze malware execution behavior.

79 DATA DESCRIPTION

80 The dataset is hosted on Zenodo [6], an open-access and multi-disciplinary online repository. It
81 comprises execution trace data in JSON format and files containing malware family labels. The traces
82 are compressed into several archive volumes. **Table 1** describes the organization of the dataset and
83 the supplemental files.

84

Table 1. Organization of the dataset.

| File / Volume | Type / Format | Contents / Purpose |
|--|---|---|
| WinMET_volume_1.7z ... WinMET_volume_5.7z | 7-zip archives (password protected: "infected") | Each volume contains a subset of the execution trace JSON files (total across volumes: 31,844 JSON traces). |
| avclass_report_to_label_mapping.json | JSON | Mapping of each execution trace to the label assigned by AVClass. |
| cape_report_to_label_mapping.json | JSON | Mapping of each execution trace to the label assigned by the CAPE sandbox labeling algorithm. |
| reports_consensus_label.json | JSON | For each execution trace, provides both its CAPE and AVClass labels. |

85

86 The traces are split into 5 volumes, each containing approximately 6,369 JSON files (except the final
87 one with 6,368). Compressed size per volume is about 2.5 GB, uncompressed size per volume is about
88 154 GB. Total compressed size: ≈ 13 GB; total uncompressed size: ≈ 750 GB for all volumes combined.
89 Each JSON file in the volumes is an execution trace. A trace contains (among other keys) metadata
90 about the executed sample (identification hashes, imported DLL or information about the executable
91 structure), the dynamic execution behavior (processes, sequence of API calls, parameters and results
92 of those API invocations, accessed resources, etc.), and labeling information (results returned by
93 vendors and the malware family labels determined from those results). **Table 2** lists the most relevant
94 JSON entries for each execution trace.

95

Table 2. Most important JSON keys of an execution trace.

| JSON Key | Description |
|-----------------------------------|---|
| detections | Array containing antivirus detection results and related metadata, derived from sandbox analysis and VirusTotal. |
| target | Object containing metadata about the analyzed malware file, including cryptographic hashes, PE structure, imported functions, strings, and VirusTotal scan results. |
| target.file.md5 / sha256 / ssdeep | Cryptographic hashes (MD5, SHA-256, SSDEEP) used to uniquely identify the malware sample and detect duplicates. |

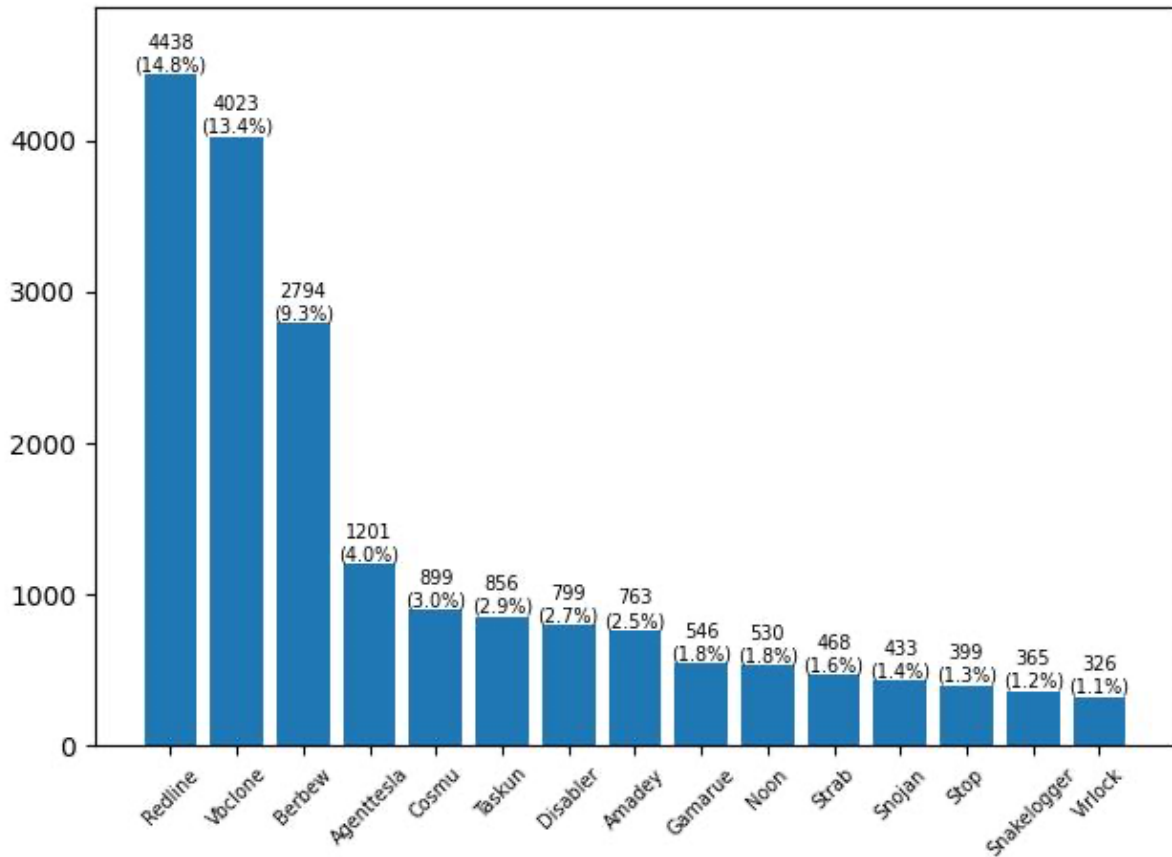
| | |
|------------------------------------|---|
| target.file.imports | List of imported libraries (DLLs) and their functions, including addresses and names, extracted from the PE header. |
| target.file.pe | Information about the Portable Executable (PE) structure, including resources and sections. |
| target.file.strings | Strings extracted from the binary, which can provide indicators of functionality or obfuscation. |
| target.file.virusotal | Object containing VirusTotal scan information, including scan ID, number of positives, total engines used, and vendor signature results. |
| dropped | Array of files or artifacts dropped or created by the malware during execution. |
| behavior | Object containing detailed dynamic analysis data gathered during execution, including processes, API calls, process trees, and summaries of accessed resources. |
| behavior.processes | Array of processes spawned during execution. Each process entry includes a unique ID, parent ID, and a list of API calls invoked. |
| behavior.processes.process_id | Numerical identifier of the process analyzed within the trace. |
| behavior.processes.calls | List of individual API calls made by a process during execution, including categories, names, arguments, and return values. |
| behavior.processes.calls.category | Broad category of the API call (e.g., filesystem, network, registry, synchronization). |
| behavior.processes.calls.api | Name of the specific API function invoked. |
| behavior.processes.calls.arguments | Array of argument name–value pairs passed to the API function during execution. |
| behavior.processes.calls.return | Return value from the API call, captured at runtime. |
| behavior.processes.tree | Hierarchical structure showing parent–child relationships among processes spawned during execution. |

| | |
|-------------------------|--|
| behavior.summary | Aggregated summary of system resources accessed by the sample, including files, registry keys, mutexes, executed commands, and other artifacts. |
| avclass_detection | Malware family label assigned by the AVClass tool, derived from VirusTotal vendor signatures. |
| Additional fields (...) | Depending on the sandbox configuration, additional metadata may be included, such as network activity, synchronization primitives, or extended analysis artifacts. |

96

97 The dataset includes a large and diverse set of malware families, as identified by both the AVClass and
 98 CAPE labeling algorithms. **Figure 1** and **Figure 2** show the top 15 malware family labels assigned by
 99 AVClass and CAPE, respectively, ranked by the number of samples associated with each family. The
 100 percentages shown in the figures are calculated relative to the total number of labeled samples,
 101 excluding those assigned the “(n/a)” label, which is assigned samples that could not be identified as
 102 any malware family by some of the algorithms. These unlabeled samples are part of the dataset but
 103 are not represented in the figures.

104 A comparison of both figures shows discrepancies in the families recognized by both labelling
 105 algorithms. AVClass infers malware families from the consensus of multiple antivirus engines, while
 106 CAPE identifies them based on dynamic behavior analysis. Therefore, these approaches apply different
 107 methodologies that often lead to variation in labelling. Nevertheless, these variations highlight the
 108 value of combining both perspectives, integrating a static (based on antivirus) and dynamic (on
 109 execution behavior) labelling. The result is reliable characterization of malware samples.



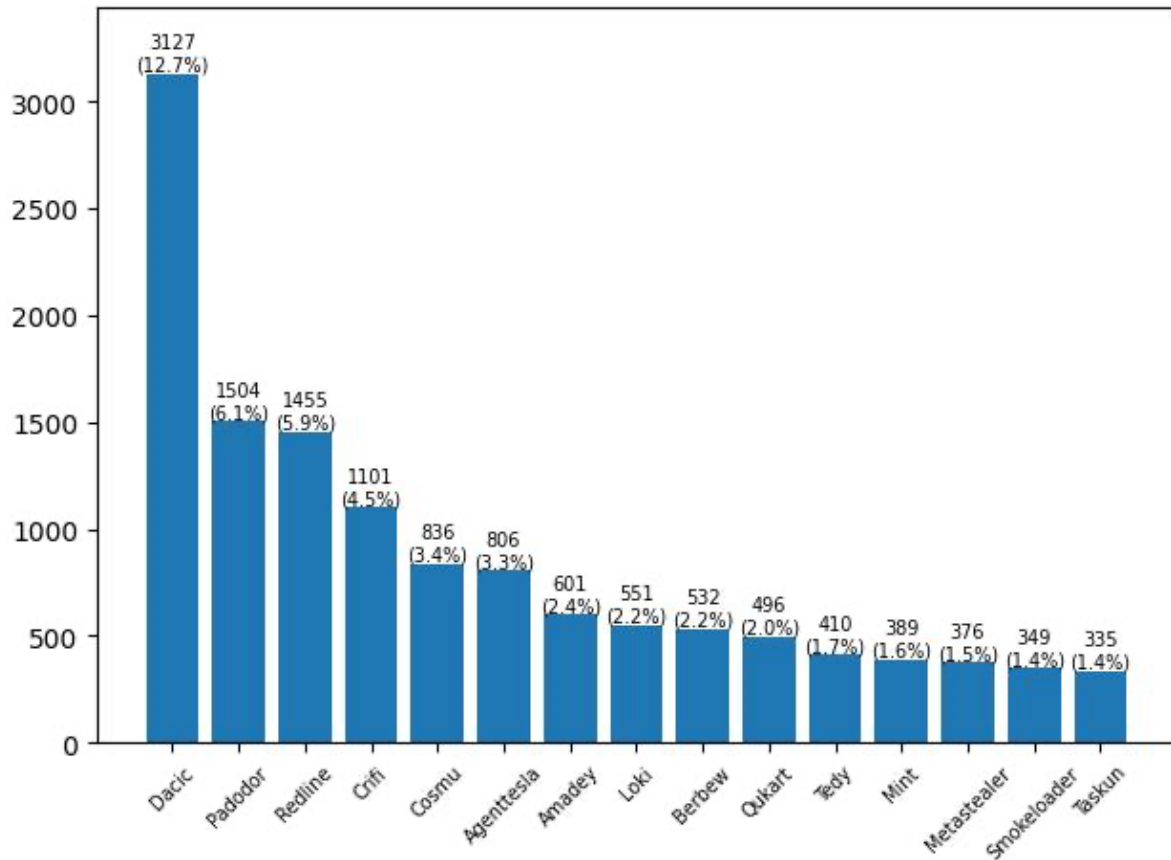
110

111

Figure 1. Distribution of AVClass labels (percent w.r.t. total amount of execution traces).

112

113 i



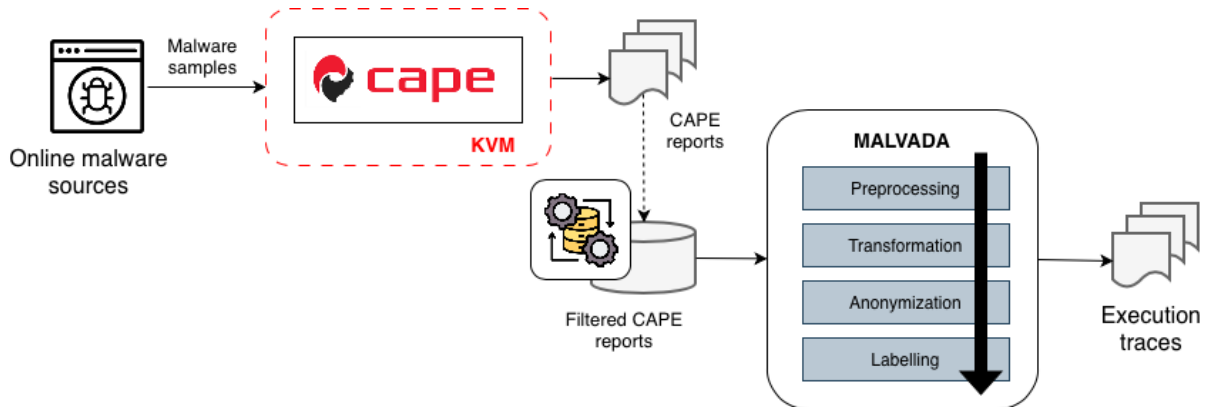
114

115 **Figure 2. Distribution of CAPE labels (percent w.r.t. total amount of execution traces).**

116 **EXPERIMENTAL DESIGN, MATERIALS AND METHODS**

117 The dataset was acquired by executing Windows malware samples in a controlled virtualized
 118 environment and collecting their execution traces. Those samples were obtained from several publicly
 119 available malware repositories such as *VX-underground*, *Malware Bazaar*, *VirusShare*, *theZoo*, and
 120 *MalShare*.

121 The process of execution trace generation consisted of three main stages that are represented in **Figure**
 122 **3**: (1) sandbox execution of malware samples using an enhanced version of the CAPEv2 Sandbox
 123 platform, (2) collection and preprocessing of dynamic analysis reports generated by CAPE, and (3)
 124 automatic processing and labeling of reports using the MALVADA framework to produce standardized
 125 JSON execution traces and associated family labels.



126

127

Figure 3. Data acquisition process.

128 *Environment for the execution of malware samples*

129 A controlled malware analysis environment was deployed on a dedicated Ubuntu 22.04 LTS host using
 130 KVM virtualization. Samples were executed within Windows 10 x64 virtual machines (VMs). Although
 131 Windows 10 was used at data-collection time, the setup is fully portable to Windows 11 VMs,
 132 facilitating future dataset updates and methodological continuity. While minor differences in system
 133 internals and API implementations exist between Windows 10 and Windows 11, these do not
 134 substantially affect the structure of the collected execution traces, as the CAPEv2 monitoring hooks
 135 target stable Windows API layers. Each VM was instrumented with a modified version of CAPEv2
 136 Sandbox to monitor and log system activity generated by malware during runtime. We used the CAPE
 137 Hook Generator [7] tool to generate C-language hook skeletons, which were then inserted into CAPE's
 138 monitoring module (capemon) and recompiled. These hooks targeted additional APIs related to file
 139 and registry operations, network communications, memory usage and process injection,
 140 synchronization objects (e.g., mutexes, semaphores). These API invocations are necessary for
 141 understanding the execution behavior of malware, specifically, those related to the use of system
 142 resources and shared objects. The virtual machines executed samples in isolation to prevent network
 143 interference or contamination between analyses. Incorrect executions due to crashes or connectivity
 144 errors were discarded automatically.

145 The result of sample executions was a collection of CAPE reports. These reports were stored in an
 146 internal data server to be preprocessed. This preprocessing mainly consisted of applying a set of filters
 147 to guarantee that the samples were executed correctly. Execution suspected of exhibiting evasion
 148 behaviors were discarded from the repository of reports.

149 *Trace generation with MALVADA*

150 The MALVADA tool was used to process CAPE reports in order to extract key data for understanding
 151 the malware execution behavior. This tool generates a detailed execution trace for each report which
 152 includes: the process tree, the API call sequence of each process, the parameters and results of those
 153 calls, accessed operating system resources, network communications, and used synchronization
 154 objects, among others. Besides, two labels are assigned to each trace for determining the malware



155 family of the corresponding sample, generated by the labelling algorithms of CAPE and AVClass,
156 respectively.

157 Internally, MALVADA was designed as a modular pipeline able to process efficiently large-scale
158 collections of execution reports. The stages of the pipeline provide functionality to filter incorrect or
159 duplicated reports, to transform reports into execution traces (in JSON format) by extracting and
160 structuring relevant data about the execution behavior and context, to anonymize sensitive
161 information, and to generate the malware family labels. Besides, the tool generates different statistics
162 about the processing of the reports and the composition of the final dataset of traces.

163 MALVADA was programmed in Python. It works with minimal user intervention and offers a variety of
164 configuration parameters related to its deployment and the report processing (a detailed description
165 of these parameters can be found in [1]).

166 LIMITATIONS

167 The dataset is limited to execution traces generated in a controlled sandbox environment using a
168 specific configuration of CAPEv2 Sandbox on Windows 10 virtual machines. As a result, those behaviors
169 that rely on a certain type of user interaction besides the default mouse movement, system-specific
170 configurations, or evasion techniques targeting sandboxes may not be fully captured.

171 Although the dataset contains over 30,000 traces, it does not cover the complete spectrum of existing
172 malware families and variants, and the distribution of samples across families is uneven. Some families
173 are represented by many samples, while others are less frequent.

174 The labelling of samples depends on CAPE and AVClass algorithms, which are based on vendors
175 integrated into VirusTotal. These labels may contain inconsistencies or inaccuracies due to differences
176 in vendor naming conventions and detection capabilities.

177 Finally, the dataset focuses exclusively on Windows malware and does not include benign software
178 traces or malware targeting other platforms, which limits its representativeness for cross-platform
179 behavioral studies.

180 ETHICS STATEMENT

181 The authors have read and follow the ethical requirements for publication in Data in Brief and
182 confirming that the current work does not involve human subjects, animal experiments, or any data
183 collected from social media platforms.

184 CRedit AUTHOR STATEMENT

185 **Razvan Raducu:** Conceptualization, Data curation, Investigation, Methodology, Resources, Software,
186 Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing; **Alain**
187 **Villagrasa-Labrador:** Data curation, Investigation, Resources, Software, Validation, Writing – review &
188 editing; **Ricardo J Rodríguez:** Funding acquisition, Investigation, Project administration, Writing –
189 review & editing; **Pedro Álvarez:** Conceptualization, Data curation, Funding acquisition, Investigation,
190 Project administration, Supervision, Validation, Visualization, Writing – review & editing.



191 **ACKNOWLEDGEMENTS**

192 This work was supported in part by grant PID2023-151467OA-I00 (CRAPER), funded by
193 MICIU/AEI/10.13039/501100011033 and by ERDF/EU, by grant TED2021-131115A-I00 (MIMFA),
194 funded by MCIN/AEI/10.13039/501100011033, by the Recovery, Transformation and Resilience Plan
195 funds, financed by the European Union (Next Generation), by the Spanish National Cybersecurity
196 Institute (INCIBE) under “Proyecto Estratégico de Ciberseguridad -- CIBERSEGURIDAD EINA UNIZAR”,
197 and under “Cátedra Internacional de Ciberseguridad UNIZAR”, and by the University, Industry and
198 Innovation Department of the Aragonese Government, Spain, under “Programa de Proyectos
199 Estratégicos de Grupos de Investigación” (DisCo research group, ref. T21-23R). The work of Razvan
200 Raducu was supported by the Government of Aragon, Spain, through the Diputación General de
201 Aragón (DGA) Predoctoral Grant, during 2021–2025.

202 **DECLARATION OF COMPETING INTERESTS**

- 203 • The authors declare that they have no known competing financial interests or personal
204 relationships that could have appeared to influence the work reported in this paper.

205 **REFERENCES**

206 [
207

[1] R. Raducu, A. Villagrasa-Labrador, R. J. Rodríguez and P. Álvarez, “MALVADA: A framework for generating datasets of malware execution traces,” *SoftwareX*, vol. 30, 2025.

[2] AV-ATLAS, “Malware & PUA,” 2025. [Online]. Available: <https://portal.av-atlas.org/malware>; [accessed October 7, 2025].

[3] GlobalStats, “Desktop Operating System Market Share Worldwide,” 2025. [Online]. Available: <https://gs.statcounter.com/os-market-share/desktop/worldwide>; [accessed October 7, 2025].

[4] K. O'Reilly, “CAPE: Malware Configuration And Payload Extraction,” [Online]. Available: <https://github.com/kevoreilly/CAPEv2>; [accessed October 7, 2025].

[5] S. Sebastián and J. Caballero, “AVclass2: Massive Malware Tag Extraction from AV Labels,” in *Annual Computer Security Applications Conference (ACSAC)*, Austin, USA, 2020.

[6] R. Raducu, A. Villagrasa-Labrador, R. J. Rodríguez and P. Álvarez, “WinMET Dataset,” 2025. [Online]. Available: <https://doi.org/10.5281/zenodo.12647555>; [accessed October 7, 2025].

[7] R. Raducu, R. J. Rodríguez and P. Álvarez, “CAPEv2 (capemon) hook(s) generator,” 2024. [Online]. Available: <https://github.com/reverseame/cape-hook-generator>; [accessed October 7, 2025].

208

209]

