

# MalDICT: Benchmark Datasets on Malware Behaviors, Platforms, Exploitation, and Packers

Robert J. Joyce<sup>1,2,3</sup>, Edward Raff<sup>1,2</sup>, Charles Nicholas<sup>3</sup> and James Holt<sup>1</sup>

<sup>1</sup>Laboratory for Physical Sciences

<sup>2</sup>Booz Allen Hamilton

<sup>3</sup>University of Maryland Baltimore County

## Abstract

Existing research on malware classification focuses almost exclusively on two tasks: distinguishing between malicious and benign files and classifying malware by family. However, malware can be categorized according to many other types of attributes, and the ability to identify these attributes in newly-emerging malware using machine learning could provide significant value to analysts. In particular, we have identified four tasks which are under-represented in prior work: classification by behaviors that malware exhibit, platforms that malware run on, vulnerabilities that malware exploit, and packers that malware are packed with. To obtain labels for training and evaluating ML classifiers on these tasks, we created an antivirus (AV) tagging tool called ClarAVy. ClarAVy's sophisticated AV label parser distinguishes itself from prior AV-based taggers, with the ability to accurately parse 882 different AV label formats used by 90 different AV products. We are releasing benchmark datasets for each of these four classification tasks, tagged using ClarAVy and comprising nearly 5.5 million malicious files in total. Our malware behavior dataset includes 75 distinct tags - nearly 7× more than the only prior benchmark dataset with behavioral tags. To our knowledge, we are the first to release datasets with malware platform, exploitation, and packer tags.

## Keywords

Malware, Benchmark Dataset, Antivirus

## 1. Introduction

The malware ecosystem is both massive and diverse. Novel malware emerges regularly and existing malware is continually being updated to add functionality or improve evasion [1]. Analyzing malware by hand is slow and requires expert domain knowledge, so machine learning and other forms of automation are relied upon as a supplement [2, 3]. As a result, there has been significant research effort towards improving malware classification using machine learning. Existing work almost exclusively focuses on two classification problems: malware detection (detecting whether a file is malicious or benign) and malware family classification (determining the malware family that a malicious file belongs to) [4]. To our knowledge, SOREL is the only malware benchmark dataset that is currently available to the public and provides labeled data for a different classification problem than the two listed above. SOREL contains ≈10 million malicious files but is labeled according to just 11 behavioral tags [5]. In the wild, malware

---

CAMLIS'23: Conference on Applied Machine Learning for Information Security, October 19–20, 2023, Arlington, VA

✉ joyce8@umbc.edu (R. J. Joyce); raff.edward@gmail.com (E. Raff); nicholas@umbc.edu (C. Nicholas);

holt@lps.umd.edu (J. Holt)



© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).



CEUR Workshop Proceedings (CEUR-WS.org)

exhibits a far greater variety of behaviors, and there are other attributes by which malware can be classified that are entirely unexplored. The Malicia dataset includes 11,363 malware samples tagged according to 172 distinct exploits, but it is nine years old as of the time of writing and is no longer publicly distributed [6].

We have assembled a collection of four benchmark datasets named MalDICT (Malware Datasets for Infrequent Classification Tasks), with each dataset supporting a different, under-represented malware classification task. MalDICT is being published in the hope that it will encourage increased awareness and study of these tasks. We are also publishing benchmark results after training two standard malware classifiers on each of these datasets. This will enable researchers to compare the performance of their own models against a baseline and against each other. The four benchmark datasets within MalDICT are:

1. **MalDICT-Behavior:** 4,317,241 files tagged according 75 common malware categories and malicious behaviors.
2. **MalDICT-Platform:** 963,492 files tagged according to 43 common file formats, operating systems, and programming languages.
3. **MalDICT-Vulnerability:** 173,886 files tagged according to 128 common vulnerabilities exploited by malware.
4. **MalDICT-Packer:** 252,148 files tagged according to 79 common malware packers.

### 1.1. Antivirus Terminology

MalDICT was tagged by combining outputs from multiple different antivirus (AV) products. We developed a custom tool named ClarAVy for this, which we describe in Section 2. For the remainder of this section, we introduce terminology about AV products and survey related AV-based taggers.

When detecting a file as malware, an AV product will produce an output called an **AV label**. An example AV label is Trojan:Win32.Androm.abc. Each portion of the label describes a characteristic of the file that the AV detected as malicious, such as its behavior, file format, or family. In some cases AV labels may also include a threat group the malware is attributed to, a vulnerability the malware exploits, or the packer the file was packed with [7]. Scanning a malicious file with a collection of AV products generates an **AV scan report**.

```
1. TR/Andromeda.B
2. Trojan.Win32.Andromeda.xyz
3. Backdoor.Androm.99
4. Win32/Gamarue.1234
5. Trj.Gamarue!1.23W
6. W32.TrojanDownloader.Wauchos.A
7. Trojan.TR/Backdoor.Gen
8. Malware (ai Score=99)
9. BehavesLike:W32/Zbot-abc
```

Figure 1: Fictitious AV scan report for a file. Labels 1-6 correctly classify this file as belonging to the Andromeda family, which has the aliases “Androm”, “Gamarue”, and “Wauchos”. Label 7 does not assign the malware a family, but indicates that it belongs to the Trojan and Backdoor categories. Labels 8-9 are heuristics, and Label 9 incorrectly classifies the malware into the Zeus family.

An example AV scan report is shown in Figure 1. Note that the naming conventions and label formats used by each AV product are different. Also note the tokens with different spellings but identical meanings, such as W32/Win32, TR/Trj/Trojan, and Andromeda/Androm. We call these *token aliases*.

## 1.2. Related Work

AVClass is the seminal work on labeling malware using AV scan reports [8]. Given a report, the tool filters out duplicate AV labels, normalizes and tokenizes each label, filters out non-family tokens, and renames families that have known aliases. The most common remaining token becomes the family tag for the scan report. AVClass++ [9], Sumav [10], and AVMiner [11] are other tools which output malware family tags using AV scan data.

To our knowledge, the following AV-based taggers can assign non-family tags to malware. EUPHONY creates a graph with weighted edges between related reports, forms clusters from communities in the graph, and assigns labels based on the majority family, category, or file type in the cluster [12]. SMART distills AV scan reports into a multi-label representing the file’s behaviors [13]. However, it supports only 11 malicious behaviors. It is the tagging method used by the SOREL dataset. AVClass2 is an update to AVClass by its original creators [7, 8]. It identifies tokens in AV scan reports which indicate the family, category, file type, and notable behaviors of the malware. It is the only tool we surveyed which can identify tokens related to packers and vulnerabilities, but it relies on a hard-coded list of tokens to do so. Finally, García-Teodoro et al. [14] created a tool that outputs multi-labels corresponding to the counts of behavioral tokens in AV scan reports.

## 2. ClarAVy

The purpose of ClarAVy is to clarify the noisy outputs from a collection of AV products into simple, easy-to-interpret tags. It is the tool used for labeling the malware in MalDICT. Figure 2 shows the major stages of the ClarAVy architecture.

First, ClarAVy ingests a corpus of AV scan reports and tokenizes each label. A lexical category is assigned to most tokens (i.e., whether the token indicates a malware behavior, packer, etc.). Once all scan reports are processed, ClarAVy reviews any tokens with incomplete or ambiguous parsing and attempts to assign a global lexical category to them. Next, ClarAVy identifies tokens which are aliases of each other. Finally, ClarAVy re-processes all of the scan reports, this time using its newly-obtained information about lexical assignments and token aliases. For each scan report, ClarAVy outputs a token ranking and the lexical category each token was assigned to. In the remainder of this section, we provide technical details for each of these stages.

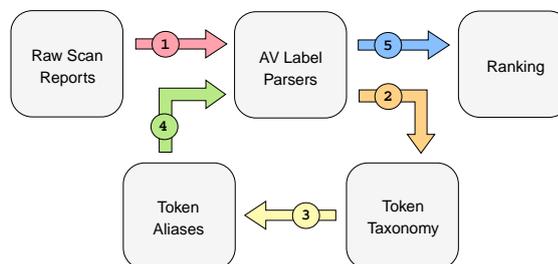


Figure 2: ClarAVy Architecture.

## 2.1. Token Taxonomy

A taxonomy of the different lexical categories which ClarAVy can assign to tokens is provided in Table 1. The FAM, PLAT, and BEH lexical categories are analogous to the "Family", "Platform", and "Type" fields in the CARO malware naming scheme. Some AV labels include additional information, such as the packer used to pack the file (PACK) or a vulnerability the malware exploits (VULN). Some AV labels may not be able to be fully parsed, necessitating the PRE and UNK lexical categories. The suffix of the AV label is assigned the SUF lexical category.

Table 1: Taxonomy of Tokens in AV Labels

BEH	The malware category or behavior
PLAT	The OS, file format, or programming language
VULNA	vulnerability exploited by the malware
PACK	The packer used to pack the file
FAM	The malware family that the file belongs to
SUF	A suffix token at the end of the AV label
PRE	Ambiguous, but not a FAM or SUF token
UNK	A token whose lexical category cannot be determined

## 2.2. AV Label Parsing

While parsing AV scan reports, ClarAVy identifies tokens within AV labels and attempts to assign each token to a lexical category in its taxonomy. ClarAVy tokenizes AV labels by splitting them on "delimiter" tokens, which are any tokens in the label that are non-alphanumeric. We call the sequence of delimiter tokens in an AV label its **delimiter format**. Most AV labels with the same delimiter format and from the same AV product have lexical categories in predictable locations. ClarAVy takes advantage of this property to make parsing simpler while also reducing ambiguity.

After identifying the delimiter format for an AV label, ClarAVy selects an appropriate parsing function. The parsing function attempts to assign a lexical category to each token in the label. Most of ClarAVy's parsing functions do this by applying regular expressions and boolean logic to the tokens. Figure 6 shows how the AV label `Exploit:Win32/MS08067.xyz` is parsed. This label is applied to malware which exploits the MS08-067 vulnerability. The delimiter format for this label is `TOK:TOK/TOK.TOK`, where TOK represents the locations that tokens may appear in the label. AV labels that have this delimiter format always have a CAT token in the first position, a TGT token in the second position, and a SUF token in the fourth

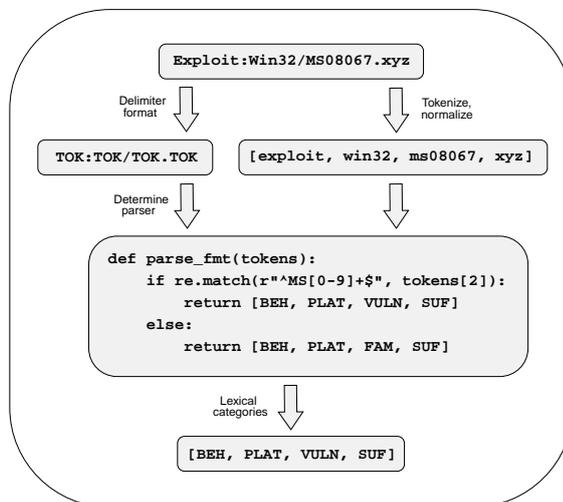


Figure 3: Parsing of the label "Exploit:Win32/MS08067.xyz". ClarAVy identifies the delimiter format of the label and selects a parsing function for it. This simple parsing function distinguishes between AV labels in this delimiter format that contain VULN or FAM tokens. The assigned lexical categories indicate that this AV label detects exploitation of the MS08-067 vulnerability in Windows.

position. A token in the third position of the label may either be a VULN or a FAM token, and the parsing function uses a regular expression to determine which lexical category should be assigned to it.

ClarAVy includes parsing functions for **882 different delimiter formats** across **90 AV products** - over 8,000 lines of Python code in total. A few AV products use only a single delimiter format, while others have dozens. Parsing functions range from trivial to complex, depending on how standardized the labels of an AV product are. We identified the most common delimiter formats used by each AV product to ensure maximal coverage. Then, we manually implemented and verified each parsing function to ensure that the lexical categories ClarAVy assigns to tokens are accurate.

### 2.2.1. Handling Parsing Ambiguity

In some cases, ClarAVy’s parsing functions cannot assign lexical categories to some tokens in an AV label. This is most often due to there being no programmatic way to distinguish tokens indicating behavior, platform, vulnerability, and/or packer from each other or from other generic tokens. The parsing function assigns these tokens the PRE lexical category to indicate that there is some ambiguity, but it is not a FAM or SUF token. More rarely, there are edge cases where tokens are truly ambiguous. The parsing function assigns the UNK lexical category to these tokens.

After all scan reports are parsed for the first time, ClarAVy attempts to determine the lexical category of each token that had some parsing ambiguity. Even if a token is assigned PRE or UNK by one parsing function, it may appear in other AV labels where it can be parsed correctly. If a token is unanimously assigned to a lexical category (not counting PRE and UNK), it is permanently assigned to that category when ClarAVy is used in the future.

ClarAVy is provided with a default wordlist that maps tokens to their lexical categories. This wordlist was generated by running ClarAVy on  $\approx 40$  million AV scan reports from VirusTotal [15]. We describe how we collected these scan reports in Section 3. Users can add to or alter this wordlist if they have different preferences. For example, we manually removed the “trojan” and “win32” tokens from the BEH and PLAT categories, respectively, since they are nearly ubiquitous in AV scan reports. In particular, AV products tend to use the “trojan” tag generically rather than for actual trojan malware [7].

### 2.3. Token Alias Resolution

During its next stage, ClarAVy attempts to identify tokens that have identical meaning. In FAM tokens, aliases may have very distinct spellings (e.g. Andromeda, Gamaue, and Wauchos in Figure 1). However, we observe that aliases for tokens in most other lexical categories generally have similar spellings. Our approach to token alias resolution uses a metric based on edit distance in addition to token co-occurrence percentage. We identify two different classes of token aliases, which we call **trivial aliases** and **parent-child aliases**.

### 2.3.1. Identifying Trivial Aliases

We say that a pair of tokens are trivial aliases if they share a lexical category and are nearly identical in spelling, where a single minor edit can transform one token into the other. For example, if one token can be transformed into a second token adding extra digit or character to the end (e.g. “backdoor” and “backdoor0”), ClarAVy considers the pair to be trivial aliases. Additionally, ClarAVy uses a small list of common substrings that are frequently observed at the beginning and end of tokens. If two tokens are identical except for the substring, it assigns them as aliases. Trivial aliases are very frequent in AV scan data, and this procedure is simple but highly effective.

### 2.3.2. Identifying Parent-Child Alias Candidates

ClarAVy also recognizes aliases from token pairs which have a “parent-child” relationship. Two conditions must apply to satisfy this relationship. First, the less common token (the child) must co-occur with the more common token (the parent) in a sufficient percentage of scan reports. Additionally, a score based on edit distance must be sufficiently high. We adapt metrics from Sebastián and Caballero [7] for computing co-occurrence percentage between tokens. Let the number of scan reports containing the child token be given by  $|t_i|$ , and let  $|(t_i, t_j)|$  be the number of scan reports containing both the child and parent token. The frequency that the child token co-occurs with the parent token is given by:

$$\text{co\_occur}(t_i, t_j) = \frac{|(t_i, t_j)|}{|t_i|}$$

A high co-occurrence percentage indicates that the child token may be related to the parent token, but other factors (such as spurious correlations between the outputs of different AV products) may cause dissimilar tokens to co-occur frequently. To reduce false positives, we also require that pair of tokens is similar in spelling. Let  $\text{len}(t)$  be the number of characters in token  $t$ . We define a custom edit score based on edit distance:

$$\text{edit\_score}(t_i, t_j) = 1 - \text{edit\_dist}(t_i, t_j) / \min(\text{len}(t_i), \text{len}(t_j))$$

Afterwards, we apply two heuristics to the edit score which we frequently observe in token aliases. If the shorter token is a substring in the longer token, or if the two tokens are anagrams, the edit score is capped at a minimum of 0.75. ClarAVy uses threshold parameters  $E$  (0.6 by default) and  $C$  (0.5 by default) to control parent-child aliasing. If  $\text{edit\_score}(t_i, t_j) \geq E$  and  $\text{co\_occur}(t_i, t_j) \times \text{edit\_score}(t_i, t_j) \geq C$ , then  $t_i$  has a parent-child relationship with  $t_j$ .

### 2.3.3. Resolving Parent-Child Aliases

Pairs of tokens with parent-child relationships are not immediately considered to be aliases. This is because a token may share a parent or child relationship with multiple other tokens. Algorithm 1 shows how ClarAVy identifies aliases from the set of tokens with parent-child relationships. Let  $T$  be a list of all known tokens within the same lexical category (e.g. all of the BEH tokens) sorted by token frequency, descending. At each iteration of the algorithm, the cur-

rent token  $t_i$  is treated as the **canonical name** for all of its aliases (i.e., all of its aliases will be renamed to the current token). A set of all “descendants” of the current token is created by recursively visiting child tokens. Then, each descendent token is assigned as an alias of the current token, provided that it has not been assigned a different alias already.

By default, the canonical name for a group of aliases is the most frequently-appearing token in the provided AV scan report dataset. ClarAVy comes with a text file which maps tokens to their canonical alias names. It was generated by using the previously-described alias resolution process on a dataset of  $\approx 40$  million AV scan reports from VirusTotal [15]. ClarAVy users can easily customize this mapping with their own alias pairs by editing the file. Furthermore, the canonical names in the alias mapping have priority over automatically-identified canonical names, allowing users to set their naming preferences.

## 2.4. Token Ranking

After assigning lexical categories to tokens and after resolving aliases, ClarAVy parses all scan reports a second time. This time, tokens with known aliases are replaced with their canonical names. Additionally, tokens which were previously assigned PRE or UNK may receive a more informative lexical category. For each AV scan report, ClarAVy outputs a ranking the BEH, PLAT, PACK, and VULN tokens in the report. Each token is given a score based on the number of times it appears in the scan report, adjusted for known correlations between AV products. PRE, SUF, and UNK tokens in the scan report are considered generic and discarded. Accurately ranking FAM tokens is much more challenging and is a target of our future work.

### 2.4.1. AV Product Correlations

The existence of correlations between AV products is well-known in the malware analysis industry. Leading causes include AV products sub-licensing their engines to others, AV products owned by the same company, and AV products “copying” another product’s detection results [8, 16]. There seem to be other factors contributing to these correlations as well, but they are poorly-understood [17]. We attempted to account for all major, publicly-known factors which would cause AV products in our dataset to produce correlated labels. To do this, we identified AV products which use very similar sets of delimiter formats in their labels. We manually confirmed each pair of correlated AV products that we wrote parsing rules for using publicly-

---

### Algorithm 1 Parent-Child Alias Resolution

---

**Require:** Sorted list of tokens  $T$

```

1: function ALIAS_RESOLVE( $T$ )
2:    $U \leftarrow \emptyset$ 
3:   for  $t_i \in T$  do
4:      $A \leftarrow \emptyset$ 
5:      $Q \leftarrow$  Queue
6:      $Q.$ enqueue( $t_i$ )
7:     while not  $Q.$ is_empty() do
8:        $t_j \leftarrow Q.$ dequeue()
9:       if  $t_j \notin U$  and  $t_j \notin A$  then
10:         $A \leftarrow A \cup t_j$ 
11:        for  $t_k \in t_j.$ children do
12:           $Q.$ enqueue( $t_k$ )
13:     for  $t_j \in A$  do
14:        $U \leftarrow U \cup t_j$ 
15:       if  $t_i \neq t_j$  then
16:         resolve_alias_pair( $t_i, t_j$ )

```

---

Figure 4: Algorithm for selecting aliases from parent-child candidates.

available information, shown in Figure 5. Like prior work, we observed that the main sources of correlation were due to AV products owned by the same company (e.g. McAfee and McAfee-GW-Edition) and AV products licensing their technology to others (e.g. ZoneAlarm previously used Kaspersky’s engine). In our dataset, there are 11 different AV products which use the BitDefender engine to varying degrees, often in combination with their own detection technologies. ALYac and Qihoo 360 use multiple other engines. We also noticed multiple instances of AV products being renamed or acquired by other companies (e.g. Commtouch was renamed to Cyren and aquired F-Prot).

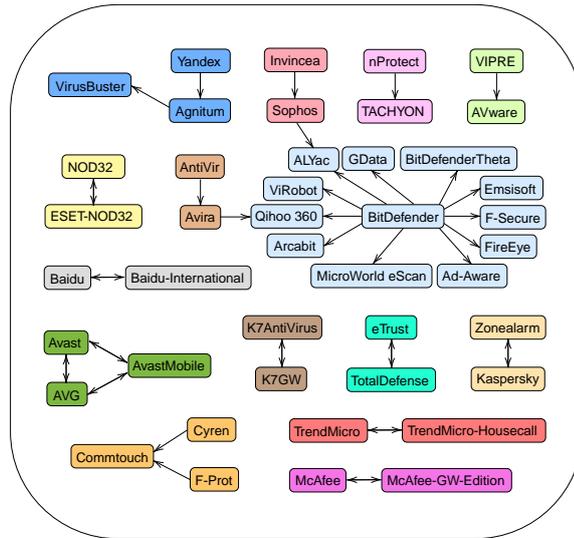


Figure 5: Publicly-known correlations in AV products.

#### 2.4.2. Token Scores

ClarAVy assigns a score to each token based on the number of times it appears in the scan report. This approach assumes that if multiple independent AV products output the same token, then the token is likely to be an accurate tag for the file. For this assumption to be valid, correlations between AV products must be accounted for. If two or more AV products with known correlations output the same token, ClarAVy combines them into a single “vote”.

A threshold parameter  $T$  is used to control the minimum token score allowed in the ranking that ClarAVy outputs. Tokens with fewer than  $T$  votes are excluded from the ranking. Higher values of  $T$  decrease the amount of noise in the outputs, but may also cause correct tokens to be omitted from the ranking.  $T$  can be set separately for each lexical category. By default  $T = 5$  for BEH and PLAT tokens and  $T = 1$  for VULN and PACK tokens. As we later show in Section 3.2, agreement of at least 5 independent AV products has a very low false positive rate. VULN and PACK tokens are much less frequent than BEH and PLAT tokens, but also much less noisy. Any threshold above  $T = 1$  for these lexical categories would cause a high false negative rate.

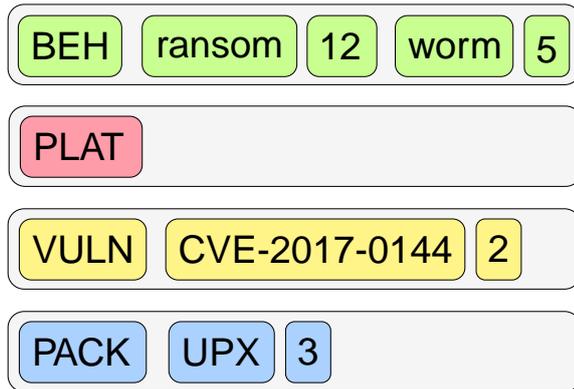


Figure 6: Example ClarAVy output for a malicious file. It is tagged as having ransomware and worm behavior. The file exploits the CVE-2017-0144 vulnerability and is packed with UPX. No platform tags are identified.

### 3. ClarAVy Validation

ClarAVy was developed with AV scan reports for 40,307,433 malicious files from chunks 0 through 465 of the VirusShare corpus [18]. We queried the VirusTotal API for these files between Feb. and Apr. 2023 to get these reports [15]. When developing each parsing function in ClarAVy, we randomly selected 10,000 AV labels with the corresponding delimiter format from this dataset. After creating a parsing function, we performed a brief

visual inspection of the resulting tokens and lexical assignments to ensure they were correct. After finishing the entire ClarAVy implementation, we ran it on these  $\approx 40$  million AV scan reports with default settings ( $T = 5$  for BEH and PLAT tokens and  $T = 1$  for VULN and PACK tokens). Then, we inspected lexical categories that ClarAVy assigned to each token and the alias mapping which it created. We manually verified both of these, correcting any errors if necessary. We identified 1,307 aliases for 92 malware behaviors, 194 aliases for 47 file-related tokens, and 53 aliases for 24 packers. The ClarAVy output included 134 distinct BEH tokens, 91 distinct PLAT tokens, 440 distinct VULN tokens, and 90 distinct PACK tokens. The ten most common tokens of each type are listed in Table 2.

Table 2: Top 10 Tokens Per Lexical Category

BEH	PLAT	VULN	PACK
virus	js	cve_2014_6332	nsis
downloader	script	cve_2010_2568	upx
riskware	html	cve_2017_17215	nsanti
adware	pe	cve_2017_11882	upack
dropper	vbs	cve_2010_0188	aspack
pua	hlllo	cve_2017_0199	themida
packed	msil	cve_2010_2586	nspack
worm	pdf	cve_2010_2586	pecompact
backdoor	multi	cve_2012_0507	fsg
redirector	android	cve_2012_0507	vmprotect

#### 3.1. Comparison to other AV-based taggers

ClarAVy’s comprehensive collection of parsing functions distinguishes it from other AV-based taggers. Most prior work uses hard-coded lists and/or heuristic methods for assigning tokens to lexical categories [8, 9, 7, 12]. For example, AVClass2 uses one parsing function per AV product for removing the suffix from AV labels [7]. Then, it uses hard-coded lists for assigning remaining tokens in the to lexical categories. AVClass2 supports updating these lists with related tokens, but it is not a default behavior and uses only co-occurrence statistics. These design choices lead to compounding errors in AVClass2’s outputs. Using the same method for suffix removal on all of an AV product’s labels may cause incorrect parsing, since the AV product likely has multiple delimiter formats. Using only hard-coded lists for assigning lexical categories will result in false negatives - especially if new tokens appear in future AV labels. With 882 parsing functions (averaging nearly 10 per supported AV product), ClarAVy assigns lexical categories to AV labels with greater fidelity. It can handle new AV labels, provided that their delimiter formats are supported.

#### 3.2. Evaluation Using the SOREL Dataset

We experimentally test ClarAVy’s ability to tag malware according to behavioral attributes. We do this using the SOREL dataset, which has 9,919,065 malicious PE files labeled according to 11

separate behavioral tags [5]. A file may have more than one tag if it displays multiple types of malicious behaviors. We queried the VirusTotal API for the malicious files in SOREL and were able to obtain AV scan reports for 7,294,655 of them. Then, we ran ClarAVy on these reports two times; once with  $T = 1$  for all lexical categories and once with  $T = 5$  for BEH and PLAT tokens. We also ran AVClass2 on these reports using default settings, except for an adjustment to its alias mapping which removes the alias between the "dropper" and "downloader"

tokens. This is because SOREL treats these as separate tags, but AVClass2 does not by default. It was also necessary to adjust the naming for some tags, since ClarAVy, AVClass2, and SOREL use slightly different terminology. We measured the per-class Precision, Recall, and F1-Measure for each of the 11 behavioral tags. Results are shown in Table 3 (with micro averaging) and Table 4 (with weighted averaging). ClarAVy with  $T = 1$  achieves the highest Recall and F1-measure, but has the lowest precision. AVClass effectively uses  $T = 2$ , since it discards any tokens which only receive a single vote. This allows it to reach a higher Precision than ClarAVy with  $T = 1$ , but the Recall and F1-Measure drop because some correct labels are discarded. ClarAVy with  $T = 5$  reaches an extremely high Precision but a low Recall and F1-Measure for the same reason. The very low false positive rate of ClarAVy with  $T = 5$  is a desirable property for an accurately-tagged dataset, and we judge the false negative rate to be of little impact.

### 3.3. Evaluation Using the MOTIF Dataset

Labeled malware data which can be used to evaluate ClarAVy is extremely limited. With the exception of SOREL, nearly all malware reference datasets either use benign/malicious labels or family labels [4]. SOREL only has 11 behavioral tags, does not have labels comparable to the PLAT, VULN, or PACK lexical categories that ClarAVy uses, and is itself dependent on AV scan data (due to using SMART as a source of labeling) [5, 13]. Therefore, it was necessary to find another way to evaluate ClarAVy’s outputs. To do this, we consider that malicious files belonging to the same family should be consistent regarding malware category, behavior, file format, and other factors. Although this is not always true (e.g., modular malware in the same family may have different components with specialized behavior, and in rare cases malware is written to target different platforms), this assumption generally holds. This allows us to evaluate how consistent ClarAVy’s outputs are with respect to malware family labels.

Suppose a dataset of malicious files  $M = \{m_1, m_2, \dots, m_n\}$ , where  $n$  is the number of files in the dataset. Let  $C_i \in M$  be the set of files that a malware tagging tool assigns tag  $i$ . For example,  $C_i$  could be the set of files that ClarAVy assigns the "ransomware" tag to. A malicious file may be assigned multiple tags. Then, let  $F = \{F_k\}_{1 \leq k \leq f}$  partition  $M$ , where  $F_k$  is the set of malicious files belonging to family  $k$ . Each file is assigned to exactly one family. Then, for

Table 3: SOREL Evaluation (Micro Avg.)

	ClarAVy (T=1)	ClarAVy (T=5)	AVClass2
Precision	.663	<b>.969</b>	.785
Recall	<b>.625</b>	.251	.483
F1-Measure	<b>.643</b>	.398	.598

Table 4: SOREL Evaluation (Weighted Avg.)

	ClarAVy (T=1)	ClarAVy (T=5)	AVClass2
Precision	.717	<b>.970</b>	.830
Recall	<b>.625</b>	.251	.483
F1-Measure	<b>.668</b>	.398	.610

each predicted label  $C_i$ , let  $D_i = \bigcup_{k=1}^f F_k$ , if  $\frac{|C_i \cap F_k|}{|F_k|} \geq 0.5$ . This constructs a set of malicious files  $D_i$  from malware families where at least 50% of files have tag  $i$  predicted. Using this, we can define metrics which are analogous to per-tag Precision and Recall:

$$\text{Precision}(C, D, i) = \frac{|C_i \cap D_i|}{|C_i \cap D_i| + |C_i \cap \bar{D}_i|} \quad \text{Recall}(C, D, i) = \frac{|C_i \cap D_i|}{|C_i \cap D_i| + |\bar{C}_i \cap D_i|}$$

Under these definitions, Precision measures the “noisiness” of a tag. It penalizes instances where a tag is assigned to a file, but where most files in its family are not associated with that tag. Conversely, Recall measures coverage of a tag within malware families. It penalizes instances where a family is likely to be associated with that tag, but there are files within that family where the tag is not assigned.

We acknowledge that there are flaws in this evaluation strategy. It is possible that families which are truly associated with a tag may be “missed” due to incorrect predictions. Furthermore, it does not necessarily confirm that predicted tags are correct (although we believe this is likely in most instances due to the high precision observed in our previous experiment using the SOREL dataset). However, in the absence of better-labeled data, we believe that this is a reasonable approach for measuring ClarAVy’s tagging consistency.

We then used ClarAVy and AVClass2 to tag VirusTotal reports for the MOTIF dataset. MOTIF contains 3,095 malware samples from 454 families, labeled with ground-truth confidence. ClarAVy and AVClass2 were run on default settings, and any tags in AVClass2’s family (FAM) or unknown (UNK) taxonomy were discarded because they are not output by ClarAVy. Additionally, the AVClass2 “windows” tag was discarded, since it appears in nearly all scan reports in MOTIF and ClarAVy treats it as generic. We computed Precision, Recall, and F1-Measure for each label using the method described above. Results are shown in Tables 5 and 6. ClarAVy clearly outperforms AVClass2 in this experiment. ClarAVy tags malware within the same family more consistently and there is less tagging noise.

Table 5: MOTIF Evaluation (Micro Avg.)

	ClarAVy	AVClass2
Precision	<b>.828</b>	.694
Recall	<b>.912</b>	.796
F1-Measure	<b>.868</b>	.741

Table 6: MOTIF Evaluation (Weighted Avg.)

	ClarAVy	AVClass2
Precision	<b>.880</b>	.723
Recall	<b>.912</b>	.796
F1-Measure	<b>.896</b>	.758

## 4. MalDICT Datasets

To build the MalDICT datasets, we ran ClarAVy on 40,307,433 VirusTotal reports for the malware in VirusShare chunks 0-465. We reviewed the tags that ClarAVy had assigned to these files and observed significant class imbalances. To account for this, we discarded tags which were too rare and down-sampled tags which were very common. BEH tags with less than 1,000

instances, PLAT tags with less than 500 instances, VULN tags with less than 100 instances, and PACK tags with less than 50 instances were not included. Tags which were too frequent were randomly down-sampled, so that they were no more than  $100\times$  more common than the minimum threshold in the training set and no more than  $25\times$  more common than the minimum threshold in the test set.

Depending on the lexical category, we selected between two different methods for dividing files into a training and test set. For MalDICT-Behavior and MalDICT-Platform, we selected from files in VirusShare chunks 0-315 for the training set and from VirusShare chunks 316-465 for the test set. Chunks 0-148 contain 131,072 files each, and the remaining chunks contain 65,536 files each. This supports an approximately 80% - 20% train-test split. More recent VirusShare chunks contain newer forms of malware that do not appear earlier in the dataset [18]. MalDICT-Behavior and MalDICT-Platform test sets contain malware added to the VirusShare corpus between July 2018 and Apr. 2023, while all of the malware in the training sets were added prior to July 2018. The first chunks were added to VirusShare in 2012, but we are aware of malware in VirusShare which was uploaded to VirusTotal in 2006 [18, 19]. Since new types of malware are continually being observed, This enables a temporal train-test split which simulates model performance on novel types of malware that do not appear in the training set. With up to nearly a five-year gap between the chunks in the training and test sets, MalDICT-Behavior and MalDICT-Platform can unveil whether a malware classifier is robust against out-of-distribution (OOD) data from a "future" time period.

The training and test sets for MalDICT-Vulnerability and MalDICT-Packer do not use a temporal split. This is because VULN and PACK tags are much less frequent, and we observed that multiple VULN and PACK tags only appear in the dataset over a short time interval. If we had used a temporal split, this would have resulted in a number of tags appearing in only the training set but not the test set or vice-versa. Instead, we used a stratified 80% - 20% train-test split to ensure even proportions of tags in the training and test sets.

#### 4.1. MalDICT Dataset Contents

Table 7 lists the number of files and number of unique tags in the four MalDICT datasets. Due to some files occurring in multiple datasets, MalDICT includes 5,457,778 unique malicious files in total. We are releasing the file hashes and ClarAVy token rankings for each of these files. Since they are a subset of the VirusShare corpus, the corresponding malicious files can be downloaded by any malware analyst who has been granted a VirusShare login [18]. Furthermore, we are releasing the disarmed executable and EMBER raw metadata for each PE file in MalDICT. Files were disarmed by zeroing out the OPTIONAL\_HEADER.Subsystem and FILE\_HEADER.Machine fields in their PE headers, which is the same method used by SOREL and MOTIF [5, 20].

Table 7: Contents of MalDICT Datasets

	Total Files	Train Set	Test Set	Tags
Behavior	4,317,241	3,744,022	573,219	75
Platform	963,492	738,264	225,228	43
Vulnerability	173,886	136,467	37,419	128
Packer	252,148	201,392	50,756	79

## 4.2. Sources of Bias in MalDICT

We now survey potential sources of bias in the MalDICT datasets. To counteract the questionable accuracy of individual AV labels, we chose to only include BEH and PLAT tags for which there is consensus between at least five uncorrelated AV products [21, 22]. We judge this to be necessary for tag accuracy, but we are aware that it may cause a selection bias [23]. Omissions or errors in AV labeling is in of itself another source of bias in our dataset [21, 22, 24]. However, there is no other source which can be reasonably used as a source of malware labels at this scale [25]. Finally, the methods we used for selecting files to include in MalDICT changed the tags and their distributions from what would be observed in the wild. We have already justified these design choices earlier in this section.

## 5. Baseline Models

We are releasing models trained on four MalDICT datasets. These models serve as measurements of baseline ML performance in this problem space. We selected MalConv2 and LightGBM as baseline models, since they are similar to those used by other notable datasets [26, 27, 28, 5, 20].

### 5.1. MalConv2 Baseline Model

Our first baseline model is MalConv2, a convolutional neural network that accepts raw file bytes as input [26]. MalConv2 was also used as a baseline model by the MOTIF dataset, and the original MalConv was used by the EMBER dataset [20, 28, 29, 26]. Our baseline MalConv2 model truncates any files greater than 1MB to 1MB in order to lessen GPU memory usage. The remaining hyperparameters were kept as the MalConv2 defaults. Then, we trained MalConv2 classifiers on

the four MalDICT training sets using eight NVIDIA RTX 6000 GPUs in parallel. The MalConv2 model for MalDICT-Behavior was trained for 33 epochs (approximately 24 hours), and the other three MalConv2 models were trained for 100 epochs each. When a file is provided as input to the baseline MalConv2 model, it outputs the probability of each tag being associated with that file. For the purposes of computing Precision, Recall, and F1-Measure, we consider an output greater than or equal to 0.5 as the threshold for predicting a tag. We used standard definitions of Precision, Recall, and F1-Measure for these results rather than our own definitions in Section 3.3. MalConv2 results on MalDICT are shown in Tables 8 and 9. MalConv2 displays good performance when classifying malware by vulnerability and packer. Performance is lower when classifying by behavior and by platform, and this is almost certainly due to the temporal train-test split present in MalDICT-Behavior and MalDICT-Platform, but not in MalDICT-Vulnerability or MalDICT-Packer.

Table 8: MalConv2 Evaluation (Micro Avg.)

	Behavior	Platform	Vulnerability	Packer
Precision	.651	.750	.926	.897
Recall	.492	.718	.888	.801
F1-Measure	.560	.733	.906	.846
ROC-AUC	.929	.965	.995	.987

Table 9: MalConv2 Evaluation (Weighted Avg.)

	Behavior	Platform	Vulnerability	Packer
Precision	.617	.772	.926	.892
Recall	.492	.718	.888	.801
F1-Measure	.512	.718	.903	.842
ROC-AUC	.896	.960	.995	.980

## 5.2. LightGBM Baseline Model

The EMBER feature vector format has become a de-facto standard for representing malware in the Windows Portable Executable (PE) file format [28]. Like the EMBER, SOREL, and MOTIF datasets, we use a LightGBM classifier trained on EMBER feature vectors as a baseline model. MalDICT includes malware that is not in the PE format, as well as files with corrupt or invalid PE header fields. EMBER vectors for these files could not be computed, so they were excluded from this experiment. Nearly all of the malware in MalDICT-Vulnerability are malicious scripts rather than PE files, so we did not train a LightGBM model on this data. In the remaining three MalDICT datasets, there were a small number of tags which contained little to no PE files, and they were also excluded. Since this is a multiclass, multilabel problem, we trained one-versus-Rest (OvR) LightGBM classifiers on each tag for 100 iterations each. Results are displayed in Tables 10 and 11.

The LightGBM classifier performed well on MalDICT-Platform and MalDICT-Packer, but was extremely poor at classifying malware in MalDICT-Behavior. MalConv2 performance on MalDICT-Behavior was substandard as well, but not to such an extent. MalConv2 and LightGBM both performed worse on MalDICT-Behavior than the SOREL dataset’s feed-forward neural network (FFNN) baseline classifier, which achieved ROC-AUC scores above 0.97 for all 11 of its behavioral tags [5]. We believe that the temporal train-test split and the increased number of tags in MalDICT-Behavior result in a more difficult classification problem than the SOREL dataset offers. Recall that the most recent malware in MalDICT-Behavior’s test set was added in April 2023, while the most recent malware in its training set was added in July 2018. This makes MalDICT-Behavior a true test on a malware classifier’s OOD performance. If a model performs well on this benchmark, practitioners can be assured that the model can generalize to malicious attributes that are present in malware far into the "future".

## 6. Conclusion

To our knowledge, MalDICT includes the first public malware datasets labeled according to platform, vulnerability, and packer. It also includes the most diverse public dataset of malware labeled by behavior, containing over 4.3 million malicious files and 75 distinct behavioral tags. We are releasing the file hashes and tags for the nearly 5.5 million malicious files in MalDICT. We are also releasing the EMBER raw features and disarmed executable files for all of the malware in MalDICT with the PE format. All of the malware in MalDICT can be obtained by researchers who have been granted to the VirusShare corpus.

Additionally, we are publishing ClarAVy, the tool that was used to accurately tag the malware

Table 10: LightGBM OvR Evaluation (Micro Avg.)

	Behavior	Platform	Packer
Precision	.177	.682	.783
Recall	.555	.953	.948
F1-Measure	.268	.795	.857
ROC-AUC	.897	.958	.992

Table 11: LightGBM OvR Evaluation (Weighted Avg.)

	Behavior	Platform	Packer
Precision	.363	.889	.844
Recall	.555	.953	.948
F1-Measure	.385	.911	.884
ROC-AUC	.805	.955	.991

in MalDICT. With support for 90 different AV products and 882 different AV label formats, ClarAVy offers more comprehensive parsing than any other AV-based malware tagging tool. ClarAVy can extract tags from tens of millions of AV scan reports, which can then be used to train production malware classifiers.

Our baseline classifier results indicate that there is significant room for improvement on all four tasks that MalDICT supports, especially malware behavior classification. The development of a classifier with strong performance on MalDICT-Behavior would represent a major success towards resisting concept drift over years of malware evolution. It is our hope that these contributions will facilitate and encourage further study of atypical malware classification tasks, fostering improved understanding and defense.

## References

- [1] S. Talukder, Z. Talukder, A survey on malware detection and analysis tools, *International Journal of Network Security & Its Applications (IJNSA)* Vol 12 (2020).
- [2] A. Mohaisen, O. Alrawi, Unveiling zeus: Automated classification of malware samples, in: *Proceedings of the 22nd International Conference on World Wide Web, WWW '13 Companion*, Association for Computing Machinery, New York, NY, USA, 2013, p. 829–832. URL: <https://doi.org/10.1145/2487788.2488056>. doi:10.1145/2487788.2488056.
- [3] D. Votipka, S. Rabin, K. Micinski, J. S. Foster, M. L. Mazurek, An Observational Investigation of Reverse Engineers' Process and Mental Models, in: *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019. doi:10.1145/3290607.3313040.
- [4] E. Raff, C. Nicholas, A survey of machine learning methods and challenges for windows malware classification, *CoRR abs/2006.09271* (2020). URL: <https://arxiv.org/abs/2006.09271>. arXiv:2006.09271.
- [5] R. E. Harang, E. M. Rudd, SOREL-20M: A large scale benchmark dataset for malicious PE detection, *CoRR abs/2012.07634* (2020). URL: <https://arxiv.org/abs/2012.07634>. arXiv:2012.07634.
- [6] Dataset - malicia project, ??? <http://malicia-project.com/dataset.html>, Last accessed on 2020-3-9.
- [7] S. Sebastián, J. Caballero, Avclass2: Massive malware tag extraction from av labels, in: *Annual Computer Security Applications Conference*, 2020, pp. 42–53.
- [8] M. Sebastián, R. Rivera, P. Kotzias, J. Caballero, Avclass: A tool for massive malware labeling, in: F. Monrose, M. Dacier, G. Blanc, J. Garcia-Alfaro (Eds.), *Research in Attacks, Intrusions, and Defenses*, Springer International Publishing, Cham, 2016, pp. 230–253.
- [9] Y. Kurogome, Avclass++: Yet another massive malware labeling tool, 2019. URL: <https://github.com/killvxk/avclassplusplus>, black Hat Europe.
- [10] S. Kim, W. Jung, K. Lee, H. Oh, E. T. Kim, Sumav: Fully automated malware labeling, *ICT Express* 8 (2022) 530–538.
- [11] L. Chen, Z. He, H. Wu, Y. Gong, B. Mao, Avminer: Expansible and semantic-preserving anti-virus labels mining method, in: *2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, IEEE, 2022, pp. 217–224.

- [12] M. Hurier, G. Suarez-Tangil, S. K. Dash, T. F. Bissyandé, Y. Le Traon, J. Klein, L. Cavallaro, Euphony: Harmonious unification of cacophonous anti-virus vendor labels for android malware, in: 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR), 2017, pp. 425–435. doi:10.1109/MSR.2017.57.
- [13] F. N. Ducau, E. M. Rudd, T. M. Heppner, A. Long, K. Berlin, Automatic malware description via attribute tagging and similarity embedding, arXiv preprint arXiv:1905.06262 (2019).
- [14] P. García-Teodoro, J. A. Gómez-Hernández, A. Abellán-Galera, Multi-labeling of complex, multi-behavioral malware samples, *Computers & Security* 121 (2022) 102845.
- [15] VirusTotal, ????, <https://www.virustotal.com/>, Last accessed on 2023-06-09.
- [16] A. Mohaisen, O. Alrawi, Av-meter: An evaluation of antivirus scans and labels, in: S. Dietrich (Ed.), *Detection of Intrusions and Malware, and Vulnerability Assessment - 11th International Conference, DIMVA 2014*, Egham, UK, July 10–11, 2014. Proceedings, volume 8550 of *Lecture Notes in Computer Science*, Springer, 2014, pp. 112–131. URL: [https://doi.org/10.1007/978-3-319-08509-8\\_7](https://doi.org/10.1007/978-3-319-08509-8_7). doi:10.1007/978-3-319-08509-8\_7.
- [17] R. J. Joyce, E. Raff, C. Nicholas, Rank-1 similarity matrix decomposition for modeling changes in antivirus consensus through time, 2021. arXiv:2201.00757.
- [18] Virusshare.com - because sharing is caring, ????, <https://virusshare.com/>, Last accessed on 2023-06-09.
- [19] R. J. Joyce, T. Patel, C. Nicholas, E. Raff, Avscan2vec: Feature learning on antivirus scan data for production-scale malware corpora, arXiv preprint arXiv:2306.06228 (2023).
- [20] R. J. Joyce, D. Amlani, C. Nicholas, E. Raff, Motif: A large malware reference dataset with ground truth family labels, 2021. arXiv:2111.15031.
- [21] M. Botacin, F. Ceschin, P. de Geus, A. Grégio, We need to talk about antiviruses: challenges & pitfalls of av evaluations, *Computers & Security* 95 (2020) 101859. URL: <http://www.sciencedirect.com/science/article/pii/S0167404820301310>. doi:<https://doi.org/10.1016/j.cose.2020.101859>.
- [22] A. Mohaisen, O. Alrawi, M. Larson, D. McPherson, Towards a methodical evaluation of antivirus scans and labels, in: Y. Kim, H. Lee, A. Perrig (Eds.), *Information Security Applications*, Springer International Publishing, Cham, 2014, pp. 231–241.
- [23] P. Li, L. Liu, D. Gao, M. K. Reiter, On challenges in evaluating malware clustering, in: S. Jha, R. Sommer, C. Kreibich (Eds.), *Recent Advances in Intrusion Detection*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 238–255.
- [24] A. Mohaisen, O. Alrawi, M. Mohaisen, Amal: High-fidelity, behavior-based automated malware analysis and classification, *Computers & Security* 52 (2015) 251 – 266. URL: <http://www.sciencedirect.com/science/article/pii/S0167404815000425>. doi:<https://doi.org/10.1016/j.cose.2015.04.001>.
- [25] A. Kantchelian, M. C. Tschantz, S. Afroz, B. Miller, V. Shankar, R. Bachwani, A. D. Joseph, J. D. Tygar, Better malware ground truth: Techniques for weighting anti-virus vendor labels, in: *ACM Workshop on Artificial Intelligence and Security*, 2015.
- [26] E. Raff, W. Fleshman, R. Zak, H. S. Anderson, B. Filar, M. McLean, Classifying sequences of extreme length with constant memory applied to malware detection, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 2021, pp. 9386–9394.
- [27] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, LightGBM: A Highly Efficient Gradient mohaisen2013 Decision Tree, in: I. Guyon, U. V. Luxburg, S. Bengio,

- H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30*, Curran Associates, Inc., 2017, pp. 3146–3154. URL: <http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf>.
- [28] H. S. Anderson, P. Roth, Ember: An open dataset for training static pe malware machine learning models, 2018. URL: <https://arxiv.org/abs/1804.04637>. doi:10.48550/ARXIV.1804.04637.
- [29] E. Raff, W. Fleshman, R. Zak, H. S. Anderson, B. Filar, M. McLean, Classifying sequences of extreme length with constant memory applied to malware detection, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 2021, pp. 9386–9394.