BODMAS: An Open Dataset for Learning based Temporal Analysis of PE Malware

Limin Yang[*], Arridhana Ciptadi[†], Ihar Laziuk[†], Ali Ahmadzadeh[†], Gang Wang[*]
[*]University of Illinois at Urbana-Champaign [†]Blue Hexagon
liminy2@illinois.edu, {arri, ihar, ali}@bluehexagon.ai, gangw@illinois.edu

Abstract—**We describe and release an open PE malware dataset called** BODMAS **to facilitate research efforts in machine learning based malware analysis. By closely examining existing open PE malware datasets, we identified two missing capabilities (i.e., recent/timestamped malware samples, and well-curated family information), which have limited researchers' ability to study pressing issues such as concept drift and malware family evolution. For these reasons, we release a new dataset to fill in the gaps. The** BODMAS **dataset contains 57,293 malware samples and 77,142 benign samples collected from August 2019 to September 2020, with carefully curated family information (581 families). We also perform a preliminary analysis to illustrate the impact of concept drift and discuss how this dataset can help to facilitate existing and future research efforts.**

I. Introduction

Today, machine learning models (including deep neural networks) are broadly applied in malware analysis tasks, by researchers [30], [5], [11], [6] and antivirus vendors [1].

In this field of work, it is highly desirable to have public datasets and open benchmarks. On one hand, these datasets will be instrumental to facilitate new works to resolve open challenges (e.g., adversarial machine learning, interpretation techniques [28], [10]). On the other hand, public benchmarks and datasets can help researchers to easily compare their models and keep track of the progress as a community.

However, creating open malware datasets is highly challenging. For example, the authors of [5] have discussed many of such challenges including legal restrictions, costs and difficulty of labeling malware samples, and potential security liabilities. In addition to these factors, another key challenge is the *dynamic evolving* nature of malware (as well as benign software) [20]. As new malware families and variants appear over time, they constantly introduce changes to the underlying data distribution. As a result, there is a constant need for releasing new datasets and benchmarks over time.

Over the past decade, there were only a handful of open *PE malware* datasets released to the research community [30]. Notable examples include Microsoft Malware Classification Challenge dataset [24], Ember [5], UCSB Packed Malware dataset [2], and a recent SOREL-20M dataset [11]. We have summarized their key characteristics in Table I.

Our Dataset: BODMAS. While existing datasets have been instrumental to researchers to develop, test, and compare machine learning models, we have identified *two missing elements* in existing datasets, which has limited researchers' ability to perform *temporal analysis* on malware classifiers for malware detection and family attribution. First, most datasets mentioned above contain malware samples that appeared between 2017 to 2019. The data is slightly outdated to study recent malware behaviors. Second, most existing datasets do not contain well-curated family information. This limits researchers' ability to test learning-based family attribution methods and analyze family evolution patterns.

For these reasons, we compile a new dataset, called BODMAS, to complement existing datasets. Our dataset contains 57,293 malware samples and 77,142 benign samples (134,435 in total). The malware is randomly sampled *each month* from a security company's internal malware database, from August 29, 2019, to September 30, 2020 (one year). For each sample, we include both the original PE binary as well as a pre-extracted feature vector that shares the same format with existing datasets such as Ember [5] and SOREL-20M [11]. Researchers could easily combine our dataset with existing ones to use them together. More importantly, our dataset provides well-curated family labels (curated by security analysts) covering 581 malware families. The family label information is much richer than existing datasets (e.g., the Microsoft dataset [24] only has 9 families).

Preliminary Analysis. In this paper, we use our dataset (and existing datasets) to perform a preliminary analysis on the impact of *concept drift* (where the testing set distribution shifts away from the training set [8]) on binary malware classifiers and multi-class family attribution methods. We illustrate the impact of concept drift on different learning tasks. In particular, we highlight the challenges introduced by the arrival of *previously unseen malware families*, which have contributed to increasing false negatives of binary malware classifiers and crippled malware family classifiers in an "open-world" setting. In the end, we discuss the open questions related to our observations and how BODMAS could help to facilitate future research in our community.

Contributions. Our contributions are:

- First, we worked with a security company to release an open PE malware dataset that contains recent (2019–2020), timestamped malware and benign samples with well-curated family information[1].
- Second, using this dataset, we performed a preliminary analysis of concept drift in binary malware classifiers and experimented with some mitigation strategies.

[1]The dataset is available here: https://whyisyoung.github.io/BODMAS/

Dataset	Malware Time	Family	# Families	# Samples	# Benign	# Malware	Malware Binaries	Feature Vectors
Microsoft	N/A (Before 2015)	●	9	10,868	0	10,868	◑	○
Ember	01/2017–12/2018	◑	N/A	2,050,000	750,000	800,000	○	●
UCSB-Packed	01/2017*–03/2018	○	N/A	341,445	109,030	232,415	●	○
SOREL-20M	01/2017–04/2019	○	N/A	19,724,997	9,762,177	9,962,820	●	●
BODMAS (Our)	**08/2019–09/2020**	●	**581**	134,435	77,142	57,293	●	●

TABLE I: Public PE malware datasets. ○="not available"; ◑="partially available", ●="available". For Ember, we have combined Ember2017 and Ember2018 and removed the duplicated samples. In addition to benign and malware samples, there are 500,000 unlabeled samples in the Ember dataset. *The vast majority of malware samples in UCSB-Packed fall within 2017–2018 (97.36%). Only a small portion (2.64%) of malware samples in a "wild-set" appeared before 2017.

- Third, we illustrated the challenges of malware family attribution in an open-world setting over time, and discussed open challenges.

II. Background and Related Work

Machine Learning for Malware Analysis. Machine learning (ML) has been applied to malware detection (classifying malware from benign files) and malware categorization (classifying malware into different families) [30], [14], [9], [31]. Features used by ML models can be extracted by static analysis (i.e., analyzing the file statically) [25], [21], [13], [17], [5], [11] and dynamic analysis (i.e., observing run-time behavior by executing the file) [22], [6], [23], [3], [32]. Static analysis is more efficient, but its performance could be affected when a binary is packed or obfuscated [2]. Dynamic analysis is costly in terms of computing resources (e.g., sandboxes) and analysis time. More importantly, today's malware can probe the system environment to detect the presence of sandbox and debugging tools, and then stay dormant to evade detection [19]. In this paper, we primarily focus on static features considering their high efficiency to support large-scale analysis.

Benchmarks and Datasets. Compared to other ML application domains (e.g., computer vision and natural language processing), it is more difficult to create and share open malware datasets. In addition to challenges such as legal restrictions, cost/difficulty of labeling malware samples, potential security liabilities, a more important challenge is the dynamic evolution of malware [5]. Due to such dynamic changes, there is a constant need for updating the datasets and benchmarks.

There are a few noticeable efforts to release PE malware datasets. We summarize their key statistics and available data fields in Table I. Microsoft Malware Classification Challenge dataset [24] was released in 2015 which only contains 10K malware samples. Note that the malware samples only contain the hexadecimal representation of the binary but have no header information. Also, it does not contain any benign files. Ember [5] was released in 2017 and then updated in 2018. Compared to the Microsoft dataset, Ember is much bigger and also includes benign files (feature vectors). Later in 2020, researchers from UCSB studied the impact of packed malware on static classifiers, and released a dataset that contains different types of packed malware samples [2]. Very recently in December 2020, SOREL-20M [11] was released, a dataset that was orders of magnitude bigger than existing ones.

Why A New Dataset? By inspecting the existing datasets in Table I, we identified *two major missing capabilities of existing datasets*, and thus compile a new dataset to complement existing ones.

First, most existing datasets contain malware samples that appeared between 2017 to 2019 (including the most recently released SOREL-20M [11]). The datasets can be slightly outdated to study recent malware behaviors. As such, we want to release a new malware dataset that covers malware samples that appeared more recently from August 2019 to September 2020. Combined our dataset with existing datasets such as Ember and SOREL-20M, researchers could have malware samples span over three years to study malware evolution and potential concept drift of classifiers.

Second, most existing datasets do not contain well-curated family information. This has limited researchers' ability to study family-related problems. For example, the Microsoft dataset only contains 9 malware families which is a very small number. Datasets such as SOREL-20M and UCSB do not contain family labels. For Ember, while it was released primarily for binary malware classification, the dataset indeed contains some family tags. However, a close inspection shows these family tags are not curated. For instance, a popular malware family tag in Ember is called "high" (8,417 samples), which appears to be incorrectly parsed from VirusTotal reports. The original field in the reports is "Malicious (High Confidence)," which is not a real family name. For these reasons, we want to include more malware families and well-curated family labels in our new dataset to fill in the gaps.

III. Dataset Description

Our dataset includes 57,293 malware samples and 77,142 benign samples (134,435 in total). The malware samples are randomly sampled *each month* from a security company's internal malware database. We performed the data collection from August 29, 2019, to September 30, 2020. The benign samples were collected from January 1, 2007, to September 30, 2020. Benign samples are also selected from the security company's database in order to represent benign PE binary distribution in real world traffic. We name the dataset `BODMAS`.

For each malware sample, we include its SHA-256 hash, the original PE binary, and a pre-extracted feature vector. For each benign sample, we include its SHA-256 hash, and a pre-extracted feature vector. Like existing datasets (e.g., Ember, SOREL-20M), It is noted that due to copyright considerations

Phase	# Samples		Ember-GBDT (17–18)		UCSB-GBDT (17–18)		SOREL-GBDT (17–19)		SOREL-DNN (17–19)	
	Benign	Malware	FPR	F_1	FPR	F_1	FPR	F_1	FPR	F_1
Validation	-	-	0.10%	98.62%	0.10%	92.14%	0.10%	98.79%	0.10%	98.01%
Test-10/19	3,925	4,549	0.00%	94.87%	0.03%	71.13%	0.09%	97.67%	0.31%	94.79%
Test-11/19	3,718	2,494	0.00%	95.83%	0.02%	81.04%	0.05%	98.09%	0.40%	96.20%
Test-12/19	6,120	4,039	0.01%	96.64%	0.06%	84.85%	0.24%	98.30%	0.45%	96.79%
Test-01/20	5,926	4,510	0.18%	93.69%	0.12%	78.04%	2.14%	96.31%	2.27%	95.41%
Test-02/20	3,703	4,269	0.07%	93.43%	0.33%	68.35%	4.82%	95.73%	6.68%	93.23%
Test-03/20	3,577	4,990	0.01%	95.75%	0.01%	75.30%	0.13%	98.14%	0.35%	95.98%
Test-04/20	5,201	4,640	0.00%	96.98%	0.02%	80.82%	0.14%	98.90%	0.26%	97.30%
Test-05/20	6,121	5,449	0.00%	97.50%	0.05%	85.69%	0.13%	98.64%	0.29%	96.03%
Test-06/20	8,182	4,217	0.01%	97.76%	0.04%	83.18%	0.22%	98.94%	0.43%	96.74%
Test-07/20	6,392	4,995	0.01%	96.38%	0.03%	66.20%	0.07%	98.68%	0.33%	93.86%
Test-08/20	2,514	3,823	0.01%	92.93%	0.02%	47.19%	0.06%	95.99%	0.10%	85.85%
Test-09/20	4,198	4,577	0.02%	92.14%	0.03%	55.97%	0.08%	95.70%	0.13%	82.88%

TABLE II: Testing the binary classifiers on each month of our BODMAS dataset. The testing F_1 is typically lower than the validation F_1, indicating concept drift. We also include the number of testing samples in each month. Note that the validation sets are provided by each of the classifiers' original datasets, the sizes of which vary and thus are omitted from this table.

benign sample binaries are not included in the dataset. Our feature vectors follow the same format of Ember [5] and SOREL-20M [11]. In this way, researchers have the option to analyze the original malware binaries. They can also use our feature vectors that are compatible with existing datasets.

In this dataset, we provide the ground-truth label ("malware" or "benign"), curated malware family information, and the *first-seen* time of a sample based on VirusTotal reports [1]. The family label is obtained mostly by analyzing verdicts from multiple antivirus vendors with in-house scripts (similar as AVClass [26]). The in-house scripts are constantly curated and updated by our threat team. A small portion (about 1%) of malware were labeled via manual analysis of the binaries. In total, the dataset covers 581 malware families.

These malware samples are from a diverse set of malware categories (14 categories in total). The most prevalent categories are Trojan (29,972 samples), Worm (16,697 samples), Backdoor (7,331 samples), Downloader (1,031 samples), and Ransomware (821 samples).

IV. Concept Drift in Binary Classification

With this dataset, we now perform a preliminary analysis to study the impact of concept drift on malware classifiers. We first focus on *binary* malware classifiers in this current section. We will perform the corresponding analysis for *multi-class* malware family classifiers later in Section V.

A. Concept Drift Across Different Datasets

To examine potential concept drift, an intuitive way is to apply classifiers trained on existing (older) datasets to our (newer) dataset. Presumably, the performance of a classifier trained on older datasets would have some degradation when evaluated on newer data due to distribution changes.

Experiment Setup. For Ember and UCSB datasets, we randomly split the dataset for training and validation (80:20), and train a Gradient Boosted Decision Tree (GBDT) classifier. We use the same hyper-parameters adopted in Ember's GBDT implementation on Github [5]. For SOREL-20M, training a classifier from scratch requires extensive computational resources. Instead, we use their pre-trained GBDT model and their deep neural network (DNN) model. We leverage their validation set to tune the threshold of the false positive rate.

The testing sets are exclusively provided by our BODMAS dataset. To show the trend over time, we divide our dataset into 12 subsets based on time (one set for each month). The size of each set varies from 6,212 to 12,399 samples. Similar to previous works [5], [2], [11], we calculate the false positive rate (FPR) to represent the number of benign samples misclassified as "malicious". We also report the commonly used F_1 score to compare the overall performance of each classifier. To make sure the classifiers are practical, we always control the false positive rate to a small number during training (e.g., below 0.1%). As such, we do not report metrics such as AUC (Area under the ROC Curve) that take consideration of high-false-positive settings.

Impact of Concept Drift. We train each classifier with 5 random seeds (SOREL-20M also provided 5 pre-trained models with 5 seeds). Then we report the average results. To simulate a realistic scenario, we tune the classifiers to control their false positive rate under 0.1% using their validation sets. Table II shows the validation performance as well as the testing performance over time.

For all these classifiers, we observe that all the classifiers can achieve high F_1 scores (under a 0.1% validation FPR) on their own validation sets. When these classifiers are applied to our testing sets (more recent data), most of them can maintain the target FPR (except for a few occasional months). However, the overall F_1 is generally lower than the validation F_1, indicating potential concept drift. The higher FPR of certain months also indicates potential drift of benign samples.

Not too surprisingly, the classifier trained on the UCSB dataset shows the largest degradation. The reason is this is a specialized dataset containing mostly packed/obfuscated malware, and thus may not generalize well to our testing samples. Comparing the GBDT classifiers trained on SOREL-20M and Ember, we find the SOREL-20M classifier (trained on a much larger dataset) sustains better over time. Surprisingly, the DNN trained on SOREL-20M performs slightly worse than the GBDT classifier. Note that both classifiers are pre-trained by the authors of SOREL-20M [11]. It is possible

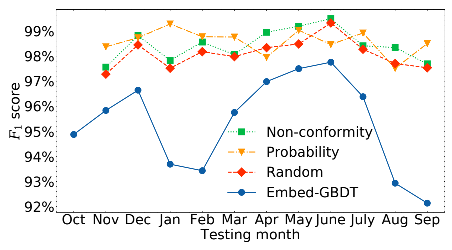


Fig. 1: The classifier is re-trained each month by cumulatively adding 1% of labeled data from each month to the original Ember dataset. We tested three sampling strategies to select new samples for labeling.

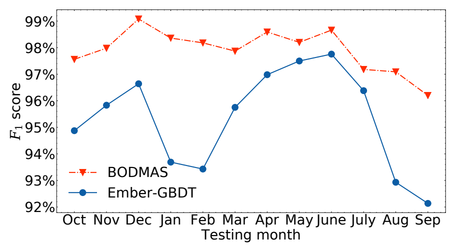


Fig. 2: Training with the first month of `BODMAS` (September 2019) vs. the original Ember baseline.

DNN can perform better with extra efforts of fine-tuning. Due to the recourse and time constraints, we directly used their classifiers and did not fine-tune them on the 20 million samples. Across all the classifiers, we observe the last two testing months (August and September in 2020) have the worst performance. Overall, the results confirm the impacts of concept drift: classifiers trained on older data have degraded performance on more recent data.

B. Mitigation Strategy 1: Incremental Retraining

To improve the classifier performance against concept drift, we experiment with several directions. The first direction is to label new samples and periodically retrain the classifier.

Experiment Setup. We test this strategy using the following experiment. We choose the original Ember classifier trained in Section IV-A as the *Baseline*. We did not use SOREL classifiers as baselines because they are significantly more expensive to retrain due to the size of their original training set. Then for each testing month, we retrain the classifier by adding 1% of newly labeled samples from the *previous month*. We start this process from November 2019. For example, at the beginning of November 2019, we sample 1% of data from the previous month (October 2019) to assign labels. Then we add the newly labeled samples to the training set to retrain the classifier. This classifier is then tested on the data of November 2019. We continue this process by *cumulatively* adding 1% newly labeled samples each month and retraining the classifier. We have tested using 0.5% new labels, and the conclusion stays the same (results are omitted for brevity).

Regarding the sampling strategies, we consider three different methods: *random*, *probability* [12], and *non-conformity* [16]. Random sampling means we select samples (from the previous month) at random. The prediction probability method ranks unlabeled samples based on the current classifier's output probability (low probability samples are ranked higher). Non-conformity metrics measure how much a given sample deviates from the current data distribution in a given class (malware/benign). Samples with higher non-conformity are selected first.

Effectiveness of Incremental Retraining. The results are shown in Figure 1. Using the original Ember classifier as the baseline, we use the different sampling strategies to perform periodic retraining each month. For each strategy, we report the average results over 5 different random seeds. First, for the baseline, we observe that its performance has some ups and downs, which is consistent with that in Table II. Overall, the performance is worse than its validation performance. Second, with retraining, we can see clear improvements in the classifier performance over time in comparison with the baseline. By labeling just 1% of the samples each month, all the F_1 scores constantly surpass 97%. Comparing different sampling methods, we find their performances are close (non-conformity slightly outperforms the random strategy).

We also experimented with an even lower sampling rate, i.e., 0.5%. Under a 0.5% sampling rate, all three methods can boost the F_1 score to 96% or higher for most of the months. Overall, the results confirm that periodic retraining is a viable strategy to mitigate the impact of concept drift.

C. Mitigation Strategy 2: Training with New Data

To mitigate the impact of concept drift, another strategy is to train a new model using the more recent data alone. On one hand, this approach could eliminate the impact of outdated samples. On the other hand, training a new classifier from scratch may require more extensive data labeling.

Experimental Setup. As a quick experiment, we use a single month of data from our `BODMAS` to train a GBDT classifier. More specifically, we take all the samples that appeared before October 1st, 2019 in `BODMAS` (roughly one-month of malware samples) for the classifier training. Same as before, we split the data into training and validation sets with an 80:20 ratio. During training, we control the FPR on the validation set to be no higher than 0.1%. We then apply the classifier to the rest of the months in `BODMAS`. For comparison, we use the Ember classifier trained in Section IV-A as the *Baseline*.

Performance of the Newly Trained Classifier. Figure 2 shows the results. The newly trained classifier improves the F_1 score for all the testing months, compared with the original classifier trained on Ember dataset. This confirms the benefit of labeling new data to train a classifier from scratch. In addition, we can still observe a slight *downward trend* of the new classifier (which is trained on the first month of data of `BODMAS`), indicating that concept drift still has an impact.

Sources of Errors. Although the newly trained classifier has a higher F_1 score, there are still errors. We next break down the sources of errors in Table III. Recall that we used the validation set to control the FPR to 0.1%. During the testing time, we

Testing month	FPR	FNR	Existing Family FNR	Unseen Family FNR
10/19	0.00%	4.77%	3.39%	43.04%
11/19	0.00%	3.97%	2.67%	35.35%
12/19	0.08%	1.71%	1.44%	16.67%
01/20	0.19%	2.99%	2.16%	26.97%
02/20	0.51%	3.14%	2.37%	26.28%
03/20	0.00%	4.17%	3.63%	19.76%
04/20	0.04%	2.74%	2.45%	8.05%
05/20	0.02%	3.52%	2.66%	9.38%
06/20	0.05%	2.56%	2.32%	6.27%
07/20	0.00%	5.49%	5.16%	6.80%
08/20	0.00%	5.65%	4.76%	15.56%
09/20	0.10%	7.23%	5.78%	16.35%

TABLE III: Breakdown of the sources of false negatives.

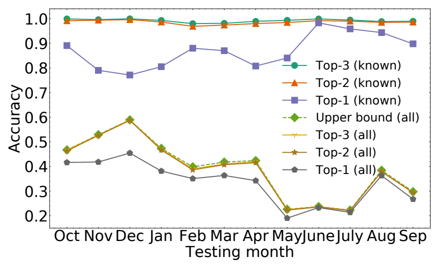


Fig. 3: Top-K accuracy on all testing data, and Top-K accuracy on known families only. The classifier is trained with $N = 10$ families in the training set. The "upper bound (all)" line represents the maximum possible accuracy of this classifier on each testing month.

observe that the FPR remains satisfying (i.e., no higher than 0.1%) for most of the months (except for January and February in 2020). In the meantime, the false negative rate (FNR) is much higher, ranging from 1.71% to 7.23%.

To further understand the source of FNR, we take advantage of the *malware family* information in our `BODMAS` dataset. For each testing month, we categorize the malware samples into *existing malware families* (i.e., families that already exist in the training set) and *unseen families* (i.e., newly appeared families that do not exist in the training set). Table III shows the false negative rate (FNR) for both types of families. We observe that existing families indeed produce false negatives (e.g., new malware variants from a known family), but the FNR is kept within 6% across all 12 months. In comparison, unseen families have much higher FNRs (as high as 43.04%). This indicates that malware samples from new families are more likely to be misclassified (as benign) by the classifier.

V. Concept Drift in Malware Family Attribution

Next, we move to the *malware family attribution* problem. Traditionally, malware family attribution is done largely based on signatures and rules (manually crafted by security analysts). However, these signatures are often brittle, which can be easily bypassed by slightly modified malware (i.e., variants from the same family). Machine learning, with a better ability to generalize, is considered a promising alternative to help with malware family attribution.

Close-world vs. Open-world. Malware attribution is commonly formatted as a multi-class classification problem. Given a dataset of N malware families, we split it into training and testing sets. Both training and testing sets will contain samples from *all N families*. In this *close-world* setting (where all the testing families also appear in the training set), it is usually easy to train an accurate classifier [29].

Malware family attribution is a much more challenging problem when we change the setup to the *open world*. First, the number of malware families N will be large and keeps increasing over time. A large N will immediately increase the difficulty of training an accurate multi-class classifier. Second, the classifier will encounter malware samples from previously unseen families (families that do not exist in the training set). During the testing time, attributing such unseen-family samples to any existing families will produce errors. Instead, the classifier should have the ability to recognize that the sample does not belong to any known families.This requires the classifier to also solve an out-of-distribution (OOD) detection problem.

Experiment Setup. We use our `BODMAS` dataset to run a preliminary experiment for malware family attribution. As we discussed in Section II, existing datasets lack well-curated family information, and thus do not support our analysis. In this experiment, we train a malware family classifier under a closed-world setting (commonly assumed), and then test this classifier in an open-world setting (reality).

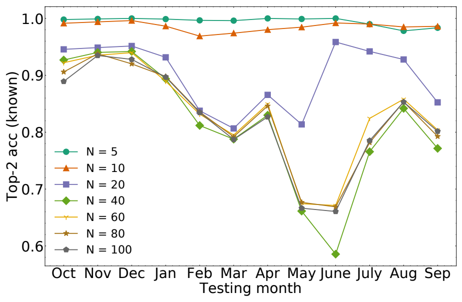
The experiment setup is similar to that of Section IV-C. We use the malware samples in the first month of `BODMAS` as the training set to train a multi-class GBDT classifier. Then we test the classifier in the remaining 12 months. Note this experiment only considers malware data for family attribution.

To examine the impact of N (the number of malware families in the training set), we trained a series of classifiers by varying N (N = 5, 10, 20, 40, 60, 80, 100), using the same hyper-parameters as Section IV-C.

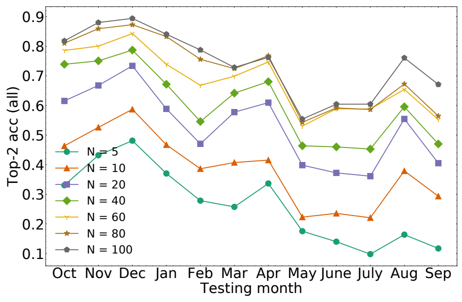
Evaluation Metric. For multi-class classification problems, classification *accuracy* is a commonly used evaluation metric. Like other application domains (e.g., image classification), we not only consider the top-1 accuracy but also the top-K accuracy in our experiments. Top-K accuracy measures the likelihood that the top-K predicted families contain a testing sample's true family. In practice, a high top-K accuracy has its values. For example, certain malware families still share common behaviors (or kill chain), which may require similar interventions. Even if a classifier cannot perfectly attribute the malware sample to the exact family, as long as the classifier narrows it down to a small number of possible families, it can help the defender to take meaningful actions.

Impact of Previously Unseen Testing Families. Recall that we train a close-world classifier and test it in an open-world setting. As an example, we first set $N = 10$ (10 training families), and then test it with all the testing data in the following 12 months. The results are shown in Figure 3.

In Figure 3, we report two types of accuracy. First, we report the classification accuracy on *all* the testing data, which includes malware families that are known in the training set (i.e., the 10 training families) and families that are previously



(a) Top-2 accuracy on known families only.



(b) Top-2 accuracy on all testing families.

Fig. 4: The impact of the number of known malware families N in the training data.

unseen in the training set. Then, we also report the accuracy on the *known* families only.

We have three observations from Figure 3. First, the classifier performs reasonably well on *known families*. For example, the top-2 and top-3 accuracy are all above 98%. The top-1 accuracy is lower but is still around 80% or higher. Second, the classifier performs significantly worse if we consider *all* the families in the testing set. The top-2 and top-3 accuracy is around 20% to 60%. The errors are contributed largely by the previously unseen families in the testing set from each month. Figure 3 also shows a green line denoted as "upper bound (all)". This line represents the maximum accuracy that this classifier could reach (even if it predicted all known families perfectly). We observe that the top-2 (all) accuracy is already close to this line. Overall, the result illustrates that the previously unseen families significantly degrade the performance of a close-world classifier.

Impact of Number of Training Families. To understand the impact of N (the number of families in the training set), we further plot Figure 4. We vary N and report the top-2 accuracy for different classifiers. Figure 4a shows the top-2 accuracy on *known* training families only, and Figure 4b shows the top-2 accuracy on *all* the testing families.

Figure 4a illustrates the impact of N on classifier training. As we increase the number of training families (N), the testing performance on these families decreases. This confirms that a large number of families will increase the training difficulty of a multi-class classifier. For example, after N gets to 40 training families, the top-2 accuracy on these families is already dropped to 70% or lower for certain months. In practice, it could be challenging to further scale up the number of families for such a classifier.

For Figure 4a, we did a further inspection of the drop around May–June 2021. We find the drop in likely caused by a family called "sfone". This family is under-trained as it only has 52 samples in the training set (September 2020). However, in May and June, there was a temporary bursty arrival of "sfone" malware (2,491 samples) which may contain new variants.

Figure 4b shows the top-2 accuracy on all of the testing families. When we increase N, the overall testing accuracy is getting better. This is because a larger N means more families become "known" to the classifier during training. As such, the impact of unseen families during testing is reduced. Another important observation is that the overall testing accuracy has a clear downward trend (i.e., the pattern of concept drift). Again, this is because malware samples from unseen families are accumulating over time. Many security products attempt to address this problem by performing frequent (e.g., daily) model updates. While that can alleviate the problem to some degree, it still does not completely solve it. To address the concept drift, we argue that an "open-world" classifier should be trained to actively handle samples from previously unknown families (i.e., OOD detection [33], [12], [18], [15]).

VI. Discussion and Conclusion

We release an open PE malware dataset BODMAS to the research community. Our dataset complements existing PE malware datasets, while offering new capabilities. First, we include well-curated family information for a large number of malware families (581). Second, we include more recent, timestamped malware and benign samples (over one year period) to support temporal analysis of malware classifiers. Using this dataset, we performed preliminary analysis on the impact of concept drift on both binary malware classifiers as well as multi-class malware family classifiers.

Our analysis reveals many open challenges when we put malware classifiers on the *time axis*. For binary classifiers, the decision boundaries shift over time due to the arrival of new malware families and malware mutation in known families. For malware family attribution, we illustrate the challenges in the "open-world" setting introduced by the scale of the classification and malware family evolution. Finally, related to malware families, we also lack tools to effectively model the relationships of malware families (e.g., similarities, hierarchical structures) to inform practical actions. Our dataset can help to support more extensive future works to address these problems.

The analysis in this paper is preliminary. We only examined concept drift using empirical malware samples. Such organic concept drift can be potentially amplified by *adversarial machine learning* attacks [4], [28], [7]. For example, there are recent works that jointly study adversarial machine learning and OOD detection [27], but their experiments are limited to image datasets. We believe our dataset can help to facilitate future research efforts along with these directions.

References

[1] VirusTotal. https://www.virustotal.com/. Accessed 2021.

[2] Hojjat Aghakhani, Fabio Gritti, Francesco Mecca, Martina Lindorfer, Stefano Ortolani, Davide Balzarotti, Giovanni Vigna, and Christopher Kruegel. When malware is packin'heat; limits of machine learning classifiers based on static analysis features. In *Proc. of NDSS*, 2020.

[3] Blake Anderson, Daniel Quist, Joshua Neil, Curtis Storlie, and Terran Lane. Graph-based malware detection using dynamic analysis. *Journal in computer Virology*, 2011.

[4] Hyrum S Anderson, Anant Kharkar, Bobby Filar, David Evans, and Phil Roth. Learning to evade static pe machine learning malware models via reinforcement learning. *arXiv preprint arXiv:1801.08917*, 2018.

[5] Hyrum S Anderson and Phil Roth. Ember: an open dataset for training static pe malware machine learning models. *arXiv preprint arXiv:1804.04637*, 2018.

[6] Ulrich Bayer, Paolo Milani Comparetti, Clemens Hlauschek, Christopher Kruegel, and Engin Kirda. Scalable, behavior-based malware clustering. In *Proc. of NDSS*, 2009.

[7] Yizheng Chen, Shiqi Wang, Dongdong She, and Suman Jana. On training robust PDF malware classifiers. In *Proc. of USENIX Security*, 2020.

[8] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 2014.

[9] Mariano Graziano, Davide Canali, Leyla Bilge, Andrea Lanzi, Elaine Shi, Davide Balzarotti, Marten van Dijk, Michael Bailey, Srinivas Devadas, Mingyan Liu, et al. Needles in a haystack: Mining information from public dynamic analysis sandboxes for malware intelligence. In *Proc. of USENIX Security*, 2015.

[10] Weijie Han, Jingfeng Xue, Yong Wang, Lu Huang, Zixiao Kong, and Limin Mao. Maldae: Detecting and explaining malware based on correlation and fusion of static and dynamic characteristics. *Computers & Security*, 2019.

[11] Richard Harang and Ethan M Rudd. SOREL-20M: A Large Scale Benchmark Dataset for Malicious PE Detection. *arXiv preprint arXiv:2012.07634*, 2020.

[12] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *Proc. of ICLR*, 2017.

[13] Xin Hu, Kang G Shin, Sandeep Bhatkar, and Kent Griffin. Mutantx-s: Scalable malware clustering based on static features. In *Proc. of USENIX ATC*, 2013.

[14] Rafiqul Islam, Ronghua Tian, Lynn M Batten, and Steve Versteeg. Classification of malware based on integrated static and dynamic features. *Journal of Network and Computer Applications*, 2013.

[15] Taewon Jeong and Heeyoung Kim. OOD-MAML: Meta-learning for few-shot out-of-distribution detection and classification. In *Proc. of NeurIPS*, 2020.

[16] Roberto Jordaney, Kumar Sharad, Santanu K Dash, Zhi Wang, Davide Papini, Ilia Nouretdinov, and Lorenzo Cavallaro. Transcend: Detecting concept drift in malware classification models. In *Proc. of USENIX Security*, 2017.

[17] Deguang Kong and Guanhua Yan. Discriminant malware distance learning on structural information for automated malware classification. In *Proc. of KDD*, 2013.

[18] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *Proc. of ICLR*, 2018.

[19] Martina Lindorfer, Clemens Kolbitsch, and Paolo Milani Comparetti. Detecting environment-sensitive malware. In Robin Sommer, Davide Balzarotti, and Gregor Maier, editors, *Proc. of RAID*, 2011.

[20] Feargus Pendlebury, Fabio Pierazzi, Roberto Jordaney, Johannes Kinder, and Lorenzo Cavallaro. {TESSERACT}: Eliminating experimental bias in malware classification across space and time. In *Proc. of USENIX Security*, 2019.

[21] Roberto Perdisci, Andrea Lanzi, and Wenke Lee. Mcboost: Boosting scalability in malware collection and analysis using statistical classification of executables. In *Proc. of ACSAC*, 2008.

[22] Konrad Rieck, Thorsten Holz, Carsten Willems, Patrick Düssel, and Pavel Laskov. Learning and classification of malware behavior. In *Proc. of DIMVA*, 2008.

[23] Konrad Rieck, Philipp Trinius, Carsten Willems, and Thorsten Holz. Automatic analysis of malware behavior using machine learning. *Journal of Computer Security*, 2011.

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

[25] Matthew G Schultz, Eleazar Eskin, F Zadok, and Salvatore J Stolfo. Data mining methods for detection of new malicious executables. In *Proc. of IEEE S&P*, 2001.

[26] Marcos Sebastián, Richard Rivera, Platon Kotzias, and Juan Caballero. Avclass: A tool for massive malware labeling. In *Proc. of RAID*, 2016.

[27] Vikash Sehwag, Arjun Nitin Bhagoji, Liwei Song, Chawin Sitawarin, Daniel Cullina, Mung Chiang, and Prateek Mittal. Better the devil you know: An analysis of evasion attacks using out-of-distribution adversarial examples. *arXiv preprint arXiv:1905.01726*, 2019.

[28] Giorgio Severi, Jim Meyer, Scott Coull, and Alina Oprea. Exploring backdoor poisoning attacks against malware classifiers. *arXiv preprint arXiv:2003.01031*, 2020.

[29] Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *Proc. of IEEE S&P*, 2010.

[30] Daniele Ucci, Leonardo Aniello, and Roberto Baldoni. Survey of machine learning techniques for malware analysis. *Computers & Security*, 2019.

[31] Phani Vadrevu and Roberto Perdisci. Maxs: Scaling malware execution with sequential multi-hypothesis testing. In *Proc. of AsiaCCS*, 2016.

[32] Tobias Wüchner, Martín Ochoa, and Alexander Pretschner. Robust and effective malware detection through quantitative data flow graph metrics. In *Proc. of DIMVA*, 2015.

[33] Limin Yang, Wenbo Guo, Qingying Hao, Arridhana Ciptadi, Ali Ahmadzadeh, Xinyu Xing, and Gang Wang. CADE: Detecting and explaining concept drift samples for security applications. In *Proc. of USENIX Security*, 2021.