Machine Learning for Malware Detection: Beyond Accuracy Rates

Lucas Galante[1], **Marcus Botacin**[2], **André Grégio**[2], **Paulo Lício de Geus**[1]

[1] University of Campinas (Unicamp)

{galante, paulo}@lasca.ic.unicamp.br

[2]Federal University of Paraná (UFPR)

{mfbotacin, gregio}@inf.ufpr.br

Abstract. *Today's world is supported by connected, electronic systems, thus ensuring their secure operation is essential to our daily lives. A major threat to system's security is malware infections, which cause financial and image losses to corporate and end-users, thus motivating the development of malware detectors. In this scenario, Machine Learning (ML) has been demonstrated to be a powerful technique to develop classifiers able to distinguish malware from goodware samples. However, many ML research work on malware detection focus only on the final detection accuracy rate and overlook other important aspects of classifier's implementation and evaluation, such as feature extraction and parameter selection. In this paper, we shed light to these aspects to highlight the challenges and drawbacks of ML-based malware classifiers development. We trained 25 distinct classification models and applied them to 2,800 real x86, Linux ELF malware binaries. Our results shows that: (i) dynamic features outperforms static features when the same classifiers are considered; (ii) Discrete-bounded features present smaller accuracy variance over time in comparison to continuous features, at the cost of some time-localized accuracy loss; (iii) Datasets presenting distinct characteristics (e.g., temporal changes) impose generalization challenges to ML models; and (iv) Feature analysis can be used as feedback information for malware detection and infection prevention. We expect that our work could help other researchers when developing their ML-based malware classification solutions.*

1. Introduction

Today's world is supported by connected, electronic systems such that ensuring their secure operation is essential to our daily lives. A major threat to system's security is malware infections, which cause financial and image losses to corporate and end-users, thus motivating the development of malware detectors [Duncan 2019] [Stewart 2019].

In this scenario, Machine Learning (ML) has been demonstrated to be a powerful technique to develop classifiers able to distinguish malware from goodware samples [Imran et al. 2016] [Garcia and II 2016]. However, many ML research work on malware detection focus only on the final detection accuracy rate and overlook other important aspects of classifier's implementation and evaluation, such as feature extraction [Rezende et al. 2018] and parameter selection.

In this paper, we shed light to these aspects to highlight the challenges and drawbacks of ML-based malware classifiers development. We trained 25 distinct classification models and applied them to 2,800 real x86, Linux ELF malware binaries. Our models considered distinct types of features (e.g., static vs. dynamic, continuous vs. discrete, bounded vs. unbounded), thus motivating a discussion about their implications in ML models.

Our results shows that: (i) dynamic features (99.36% accuracy in the best case) outperforms static features (98.98% accuracy in the best case) when the same classifiers are considered; (ii) Discrete-bounded features (85.86% average accuracy rate) present smaller accuracy variance over time in comparison to continuous features (97.26% average accuracy rate), at the cost of some time-localized accuracy loss; (iii) Datasets presenting distinct characteristics (e.g., temporal changes) impose generalization challenges to ML models. Whereas the overall detection rate of all samples is 99%, each evaluated dataset achieved smaller detection rates when evaluated individually, due to the presence of characteristics which are observed in the overall model but not in the smaller subset; and (iv) Feature analysis can be used as feedback information for malware detection and infection prevention (e.g., network usage relates to $\approx$40% of malicious behaviour classification).

In summary, our contributions are the following: (i) we review the use of ML for malware detection and classification; (ii) we point common overlooked aspects in ML classifiers developments; (iii) we evaluate distinct ML classifier implementation decisions in practice; and (iv) we discuss how our evaluation can provide feedback information for malware detection and infection prevention.

This work is organized as follows: we present background information in Section 2; we present our assumptions, data collection and analysis methods in Section 3; we evaluate malware classifiers in Section 4; we discuss the impact of our discoveries in Section 5; we present related work in Section 6; finally, we draw our conclusions in Section 7.

2. Background

In this section, we present background information to support our developments.

Feature is a measurable property of the object file to be classified. Features can be classified as **discrete** (e.g., a given property is present or not) or **continuous** (e.g., how many occurrences of a given property were found within an object file). Features can also be classified as **bounded** (property values are limited to some range) or **unbounded**. Discrete features are also naturally bounded features. Finally, in the malware context, features can also be classified as **static** (extracted from a binary without executing it) or **dynamic** (extracted from a binary execution trace). In this work, we evaluate the impact of all these feature classes.

Supervised learning algorithms are those which rely on a full set of labeled data in advance to compare its outcomes. Malware experiments are often conducted using supervised algorithms trained with labeled goodware samples. Therefore, this strategy was also adopted in this work.

Accuracy is the ratio between the labels correctly attributed by the classifier and the ground-truth labeling data. Distinct classifier's accuracy can only be directly compared when their considered datasets are balanced (50% malware-50% goodware), an strategy adopted in this work.

Folding is an strategy to mitigate statistical variations within a dataset. A *k-folding* works by splitting a full dataset into smaller k subsets. One of these subsets is used for training

and the other for evaluation. This strategy is repeated k times. The final accuracy result is the average of all subsets accuracy. We applied folding to all presented experiments.

3. Methodology

In this section, we present the general methodology adopted for our experiments.

Dataset. We performed all experiments in the Linux platform, thus benefiting of an existing feature extraction solution [Galante et al. 2018]. To provide a comprehensive evaluation of Linux binaries and not bias our experiments with a single characteristic, we collected samples from distinct sources and periods of time. In total, we considered 2,800 unique malicious x86 ELF binaries identified by their MD5. The samples were crawled from `VirusShare`[1], a collection of samples of 2007, and from `VirusTotal`[2], during 2017. Additional dataset description was presented by [Galante et al. 2018]. We randomly selected an equivalent amount of benign samples for training procedures. We gathered ELF files from the `/bin` and `/usr/bin` directories of a fresh Ubuntu 16 LTS installation, thus we assume them as goodware.

Analysis Pipeline. All malware and goodware samples were submitted to static and dynamic analysis procedures for feature extraction. Dynamic analysis was conducted by executing the samples in a sandboxed environment by three minutes and logging all API and system calls via `strace` and `ltrace`. Upon feature extraction, features are consolidated in distinct ML models. The models are classified using three distinct classifiers: SVM, Random Forest and MLP. All classifiers were implemented in *python 2.7* via the *sklearn* library and all experiments were conducted in a 10-folding way.

Analyzed Features. To evaluate the impact of distinct feature classes and extraction procedures, we developed distinct models. Table 1 summarizes the considered features among all models. The discrete column represents boolean features (e.g., UPX packer presence was identified or not). The continuous column represent integer values (e.g., header indicates how many distinct headers are present in the binary).

Table 1. Malware Features classified according extraction method (static and dynamic) and representation (discrete or continuous).

Static				**Dynamic**	
Discrete		Continuous		Both	
Embedded files	Dissasembly fail	Size sections	# headers	fork syscall	/proc access
/home string	ptrace syscall	/home string	# .dynamic	ptrace syscall	/home access
/sys string	Network strings	/sys string	# sections	socket syscall	passwd access
Linkage	Header present	passwd string	# symbols	mmap syscal	permission denied
UPX	passwd string	# libs	# relocations	SIGTERM	
fork syscall	compiler string	Size sample	# debug section	SIGSEGV	

String features represent character patterns matched via regular expressions (e.g., network strings are `URL, email` and `IP`). Path features are locations for specific directories (e.g., `/home/username/` or `/etc/passwd`).

4. Evaluation & Results

In this section, we present accuracy results for multiple classifiers and models. The results are split according the considered dataset (`VirusTotal` and `VirusShare`), feature types (discrete or continuous) and extraction methods (static or dynamic).

[1] `https://virusshare.com/` [2] `https://www.virustotal.com/`

4.1. The importance of Selecting a Proper Classifier

Classification algorithms are supported by distinct assumptions, thus they produce different outcomes for the same datasets. In addition, each classifier presents its own tuning parameters, which can be adjusted according the classification task. Therefore, observing classifiers behavior is essential to select the best one for each task. In this work, we considered the behavior of three ML classifiers (SVM, RF, MLP) and varied their parameters such as to always achieve the best accuracy rates. We discovered that the SVM classifier achieves the best results when using the `rbf` kernel—Table 16 (98.54%)—and the `linear` kernel—Table 4 (98.62%)—, according the considered model. Similarly, RF best results can be achieved either by changing the number of estimator—Table 13 (90.94%)—or its depth—Table 11 (94.35%). Finally, MLP presents the best results when using the `adam` solver—Table 7 (83.77%)— and Table 22 (96.71%)—,according the considered scenario. Whereas this accuracy-focused classifier selection step is considered by most work in the academic literature, additional reasoning steps are often overlooked [Rezende et al. 2018]. To bridge this gap and allow understanding other classification boundaries, we following discuss additional important aspects.

Table 2. VirusTotal and VirusShare datasets. Random Forest classification of static continuous features.

Max Depth/ Estimators (#)	16	32	64
8	**99.17%**	99.06%	99.20%
16	99.13%	99.06%	99.09%
32	99.09%	99.13%	99.17%

4.2. The importance of a good Feature Extraction Procedure

From an ML algorithm point of view, features are understood only as a vector which is classified regardless of its source or interpretation. From a malware analysis point of view, however, features represent the behaviour of the sample it was extracted from. Thus, distinct feature extraction methods will lead to distinct accuracy results even for the same classifiers, since some methods are more prone to malware evasion (e.g., static analysis vulnerability to obfuscation) than others (e.g., dynamic analysis in transparent sandboxes). Unfortunately, many work on ML-based malware classification do not discuss the feature extraction procedure.

To understand the impact of relying on distinct feature extraction procedures, we submitted the same malware samples to static and dynamic analysis procedures and considered accuracy results for the same features models and classifiers. All models based in dynamic features—Table 15 (92.63%), Table 21 (94.98%), and Table 24 (99.36%)—presented higher accuracy rates than models based on static features—Table 3 (84.92%), Table 9 (90.99%), and Table 12 (98.98%). Therefore, we highlight the importance of considering feature extraction procedures in the development of malware classifiers.

4.3. The importance of the Evaluated Datasets

Most work highlight the fact that classifier's accuracy depend upon the dataset that a classifier is predicting. However, many work overlook that classifier's accuracy also depend upon the dataset on which the classifier was trained. Whereas this is not a major concern

Table 3. VirusTotal dataset. SVM classification of static continuous features.

Kernel/Iter(#)	1000	10000	100000
Poly	49.32%	49.74%	49.95%
Linear	73.87%	77.64%	80.94%
rbf	84.92%	**84.92%**	84.92%

Table 4. VirusShare dataset. SVM classification of static continuous features.

Kernel/ Iter (#)	1000	10000	100000
Poly	50.25%	52.20%	51.34%
Linear	**98.62%**	98.52%	98.10%
rbf	89.66%	89.50%	89.50%

Table 5. VirusTotal dataset. SVM classification of static discrete features.

Kernel/Iter(#)	1000	10000	100000
Poly	68.80%	68.80%	68.80%
Linear	85.24%	85.24%	85.24%
rbf	89.48%	**89.48%**	89.48%

Table 6. VirusShare dataset. SVM classification of static discrete features.

Kernel/ Iter (#)	1000	10000	100000
Poly	82.94%	78.67%	78.67%
Linear	84.48%	**84.48%**	84.48%
rbf	84.27%	84.27%	84.27%

for many applications which handle almost static data (e.g., object recognition in images), this is an important drawback for very dynamic scenarios, such as malware detection.

We evaluated that in practice by independently classifying the `VirusShare` and the `VirusTotal` datasets using the same set of benign apps. Table 26 (96.87%) shows that the `VirusShare` dataset presented higher accuracy rates than `VirusTotal` dataset—Table 25 (95.4%). Moreover, we also trained and predicted a classifier mixing all samples from both datasets. Table 2 shows that the mixed samples dataset presented better classification rates (99.17%) than individual datasets—Table 11 (94.35%). This result is explained by the fact that the `VirusShare` dataset dates 10 years prior to the `VirusTotal` dataset, thus the malware samples characteristics changed over time. When training a dataset using only one of the datasets, the classifier learns only the characteristics of that time. When mixing samples, however, the classifier learns characteristics of both periods and achieves a greater detection rate.

Unfortunately, many work do not consider distinct datasets in their evaluation or only present results mixing all datasets, thus not allowing one to understand the effects of time and multiple characteristics in classifier's evaluations.

4.4. The Analyst importance

ML classifiers present some advantages and drawbacks in comparison to human heuristics, but these are not often discussed. For instance, as a significant advantage, ML models can learn better than humans the separation between malware and goodware features. On the other hand, as a drawback, the selected separation line might be not meaningful for

Table 7. VirusTotal dataset. MLP classification of static continuous features.

Solver/ Alpha	1	100	1000
sgd	82.15%	80.00%	50.37%
adam	**83.77%**	76.60%	67.70%
lbfgs	67.75%	65.76%	68.69%

Table 8. VirusShare dataset. MLP classification of static continuous features.

Solver/ Alpha	1	100	1000
sgd	93.44%	95.39%	57.70%
adam	**96.85%**	95.29%	86.69%
lbfgs	83.41%	83.41%	81.84%

Table 9. VirusTotal dataset. MLP classification of static discrete features.

Solver/ Alpha	1	100	1000
sgd	85.86%	50.00%	50.00%
adam	90.68%	49.01%	48.17%
lbfgs	**90.99%**	84.45%	47.96%

Table 10. VirusShare dataset. MLP classification of static discrete features.

Solver/ Alpha	1	100	1000
sgd	85.18%	51.63%	48.29%
adam	84.76%	51.63%	51.63%
lbfgs	**85.86%**	83.46%	51.63%

Table 11. VirusTotal dataset. Random Forest classification of static continuous features.

Max Depth/ Estimators (#)	16	32	64
8	94.29%	**94.35%**	94.24%
16	94.24%	94.14%	94.08%
32	94.08%	94.14%	94.19%

Table 12. VirusShare dataset. Random Forest classification of static continuous features.

Max Depth/ Estimators (#)	16	32	64
8	98.91%	98.96%	**98.98%**
16	99.85%	98.88%	98.91%
32	98.80%	98.80%	98.50%

humans (e.g., a given random number of API calls is considered malicious) or strongly coupled to an specific dataset (e.g., overfitting).

To evaluate the impact of humans and machines selecting classification boundaries, we developed two classification models for all classifiers. The discrete model consists of boolean values selected for being meaningful to analysts (e.g., binary is packed or not) and the continuous model consists of integer values without immediate meaning for malware analysts (e.g., number of headers) but that can be clustered by the ML classifiers.

Table 8 (96.85%) show that the accuracy rates for continuous features are higher than those with discrete features— Table 10 (85.86%)—, which is expected due to the higher capabilities of machines. On the other hand, Table 17 (93.52%) and Table 18 (96.48%) of discrete features and Table 19 (92.68%) and Table 20 (98.87%) of continuous features show that the classification variance among datasets is smaller in discrete features: 3.1% for the former and 6.7% for the latter. It happens because the analyst's knowledge is not tied to any specific dataset, unlike ML boundaries. Therefore, we claim that the use of discrete or continuous features for ML models development should consider how diverse are the datasets to be classified.

4.5. What ML results teach us

In addition to effectively classifying binaries as malware, a good malware classifier should be able to provide feedback information for infection remediation and infection prediction, an step which is often overlooked by most approaches. We claim that the most significant features could be used to provide such feedback information. Table 27 and 28 present the feature importance rates for the Random Forest classifier.

We discovered that 40% of samples are statically detected due to the presence of network

Table 13. VirusTotal dataset. Random Forest classification of static discrete features.

Max Depth/ Estimators (#)	16	32	64
8	90.47%	89.69%	90.63%
16	91.05%	**90.94%**	90.89%
32	90.79%	90.94%	90.89%

Table 14. VirusShare dataset. Random Forest classification of static discrete features.

Max Depth/ Estimators (#)	16	32	64
8	85.86%	85.75%	85.80%
16	85.83%	**85.93%**	85.93%
32	85.91%	85.86%	85.91%

Table 15. VirusTotal dataset. SVM classification of dynamic continuous features.

Kernel/ Iter (#)	1000	10000	100000
Poly	49.92%	49.76%	50.71%
Linear	93.73%	86.51%	86.73%
rbf	92.63%	**92.63%**	92.63%

Table 16. VirusShare dataset. SVM classification of dynamic continuous features.

Kernel/ Iter (#)	1000	10000	100000
Poly	50.91%	54.05%	58.16%
Linear	97.97%	97.56%	80.35%
rbf	98.54%	**98.54%**	98.54%

Table 17. VirusTotal dataset. SVM classification of dynamic discrete features.

Kernel/ Iter (#)	1000	10000	100000
Poly	90.22%	90.22%	90.22%
Linear	93.52%	**93.52%**	93.52%
rbf	93.94%	93.94%	93.94%

Table 18. VirusShare dataset. SVM classification of dynamic discrete features.

Kernel/ Iter (#)	1000	10000	100000
Poly	79.68%	79.91%	79.91%
Linear	96.48%	**96.48%**	96.48%
rbf	96.35%	96.35%	96.35%

Table 19. VirusTotal dataset. MLP classification of dynamic continuous features.

Solver/ Alpha	1	100	1000
sgd	87.25%	95.35%	73.03%
adam	**92.68%**	92.21%	85.05%
lbfgs	79.87%	77.99%	67.02%

Table 20. VirusShare dataset. MLP classification of dynamic continuous features.

Solver/ Alpha	1	100	1000
sgd	98.87%	96.30%	49.11%
adam	98.66%	**98.87%**	98.66%
lbfgs	97.89%	89.21%	48.68%

Table 21. VirusTotal dataset. MLP classification of dynamic discrete features.

Solver/ Alpha	1	100	1000
sgd	94.41%	49.97%	47.83%
adam	**94.98%**	94.30%	47.83%
lbfgs	94.20%	49.97%	49.97%

Table 22. VirusShare dataset. MLP classification of dynamic discrete features.

Solver/ Alpha	1	100	1000
sgd	96.40%	50.01%	47.14%
adam	**96.71%**	96.17%	47.14%
lbfgs	96.22%	50.01%	50.01%

Table 23. VirusTotal dataset. Random Forest classification of dynamic continuous features.

Max Depth/ Estimators (#)	16	32	64
8	97.28%	97.49%	97.39%
16	97.65%	97.70%	**97.8%**
32	97.39%	97.60%	97.80%

Table 24. VirusShare dataset. Random Forest classification of dynamic continuous features.

Max Depth/ Estimators (#)	16	32	64
8	99.26%	99.26%	99.26%
16	99.15%	**99.36%**	99.28%
32	99.26%	99.26%	99.31%

Table 25. VirusTotal dataset. Random Forest classification of dynamic discrete features.

Max Depth/ Estimators (#)	16	32	64
8	94.82%	95.19%	95.19%
16	95.30%	95.19%	95.19%
32	95.19%	95.09%	**95.4%**

Table 26. VirusShare dataset. Random Forest classification of dynamic discrete features.

Max Depth/ Estimators (#)	16	32	64
8	96.81%	**96.87%**	96.81%
16	96.81%	96.76%	96.74%
32	96.84%	96.79%	96.74%

Table 27. Feature importance designated by Random Forest classifier. From prevalent to least relevant static features subset.

Static			
Discrete		Continuous	
Network strings	40%	Binary size	27%
UPX present	17%	# headers	16.70%
passwd strings	1.40%	# debug sections	0.20%

Table 28. Feature importance designated by Random Forest classifier. From prevalent to least relevant dynamic features subset.

Dynamic			
Discrete		Continuous	
mmap	50%	# mmap	68%
fork	6%	# fork	10.80%
SIGSEGV	10.60%	# SIGSEGV	1.30%

strings, thus indicating that samples have been exploiting network weaknesses to deploy their attacks. Such information indicates that the primary countermeasure to be deployed by system administrators is to harden their network defenses. Similarly, more than 50% of all samples were dynamically detected due to the mapping of suspicious memory addresses, thus indicating that OS hardening should focus on enhancing processes isolation.

Moreover, feature analysis also provides us feedback information for model enhancement and allows us to understand the root of malware detection. We discovered that whereas 6% of all samples are detected due to `fork` calls in the discrete model, the significance of this feature grew to 10% in the continuous model, thus indicating a better performance when this feature is modelled as a number of calls. This is explained by the fact that, whereas even benign samples might present some few `fork` calls, most malware samples heavily rely on creating new processes as a modular construction for payload distribution. Therefore, a high number of `fork` is more suspicious than *forking* at least once.

5. Discussion

In this section we discuss the implications of our findings for the development of future ML classifiers for malware detection.

Distinct Classifiers for distinct datasets. Our results show that accuracy rates vary according to used classifier, since each rely on distinct algorithm and parameters. To determine the best classifier for a given task and dataset, researchers should evaluate distinct classification algorithms and parameters. In most of our experiments, Random Forest has demonstrated to be the most suitable classifier for malware detection.

Feature Extraction affects model's results. Whereas ML classifiers operate the same way for all provided sets of features, the feature extraction procedures significantly affect classifier's accuracy. Whereas static feature extraction procedures require low computational resources and can be executed very fast, they can be bypassed by binary obfuscation, thus reducing classifier's accuracy. On the other hand, dynamic feature extraction procedures require higher computational resources and are time-consuming, but are not vulnerable to obfuscation. Therefore, models based on dynamic features present higher accuracy rates than models based on static features, even when the same classifiers and features are considered.

The effect of time ML models produce distinct outcomes according the considered datasets, not only due to distinct samples to be predicted, but also due to distinct training datasets. Malware samples are very dynamic pieces of software, and dataset from distinct periods of time present distinct characteristics, such as features prevalent in one period of time and not in another. Our evaluation showed that: on the one hand, mixing samples

from datasets of different time periods presented increased classifier's accuracy rates; on the other hand, individual dataset classification presented reduced accuracy rates. Therefore, we highlight the need of investigating the characteristics of the considered datasets when reporting classification results to avoid reporting biased results.

The role of Analyst's Expertise. ML models can be based on features selected by the analysts or in boundaries defined by the machine itself. Whereas the former has shown to achieve higher accuracy, it also presents higher variation among datasets. Therefore, researchers should determine datasets characteristics prior deciding which kind of feature modelling will be applied.

Features should provide feedback information. Whereas the ultimate goal of a malware classifier is to detect malware, we advocate for the need of understanding how classification decision work, thus allowing infection remediation and prevention. We showed that the evaluation of the most distinguishable features for malware classification might allow incident response and system hardening. For instance, we identified a prevalent use of network resources by malware samples, thus indicating that infection prevention professionals should focus on developing enhanced network defenses.

6. Related Work

In this section, we compare related research work in malware classification to better position our work among the academic literature.

The use of ML classifiers has become a popular approach to tackle the malware detection and classification problem, with many classifying models been developed and presented over time [Babaagba and Adesanya 2019, Kruczkowski and Szynkiewicz 2014]. However, the first work in the field limited their investigation to procedures to maximize the achieved accuracy rates via classifier's parameters selection and overlooked important aspects, which were only addressed by a second-generation of research work and solutions [Feizollah et al. 2015, Ahmadi et al. 2016]. These work highlighted the importance of feature selection for classifying `Android` and `Microsoft` malware samples, an investigation which is here extended to `Linux` malware samples. Whereas feature selection has been demonstrated to be an important step of ML classifiers development [Liangboonprakong and Sornil 2013], many recent research work on Deep Learning (DL) do not rely on feature extraction [Rezende et al. 2018]. We here advocate the need of considering properly-modelled features not only for increased accuracy rates but also to allow proper infection remediation and prevention. Finally, even when the aforementioned aspects are considered, datasets are often handled in an uniform way and the differences derived from dataset representing distinct periods of time [Menahem et al. 2013] are neglected. In this work, we highlight the need of understanding dataset characteristics to properly model and evaluate ML classifiers for malware detection.

7. Conclusion

In this paper, we investigated the challenges of ML-based malware classification beyond the final accuracy result. We trained 25 distinct classification models and applied them to 2,800 real Linux ELF malware binaries and discovered that: (i) dynamic features outperforms static features when the same classifiers are considered; (ii) Discrete-bounded features present smaller accuracy variance over time in comparison to continuous features, at the cost of some time-localized accuracy loss; (iii) Datasets presenting distinct characteristics (e.g., temporal changes) impose generalization challenges to ML models;

and (iv) Feature analysis can be used as feedback information for malware detection and infection prevention.

Reproducibility. The developed classification tool is open source and can be downloaded from: `https://github.com/marcusbotacin/ELF.Classifier`

Acknowledgement This work is supported by the Brazilian National Counsel of Technological and Scientific Development (CNPq, PIBIC 2016/2018, process 138239/2017-7).

References

Ahmadi, M., Ulyanov, D., Semenov, S., Trofimov, M., and Giacinto, G. (2016). Novel feature extraction, selection and fusion for effective malware family classification. ACM CODASPY.

Babaagba, K. O. and Adesanya, S. O. (2019). A study on the effect of feature selection on malware analysis using machine learning. ACM ICEIT 2019.

Duncan, B. (2019). Shade ransomware hits high-tech, wholesale, education sectors in u.s, japan, india, thailand, canada. `https://bit.ly/2X2beX5`.

Feizollah, A., Anuar, N. B., Salleh, R., and Wahab, A. W. A. (2015). A review on feature selection in mobile malware detection. *Digit. Investig.*, 13(C):22–37.

Galante, L. B., Botacin, M. F., Grégio, A. R. A., , and de Geus, P. L. (2018). Malicious linux binaries: A landscape. *XVIII SBSeg*.

Garcia, F. C. C. and II, F. P. M. (2016). Random forest for malware classification.

Imran, M., Afzal, M., and Qadir, M. A. (2016). Malware classification using dynamic features and hidden markov model. *Journal of Intelligent & Fuzzy Systems*.

Kruczkowski, M. and Szynkiewicz, E. N. (2014). Support vector machine for malware analysis and classification. WI-IAT. IEEE.

Liangboonprakong, C. and Sornil, O. (2013). Classification of malware families based on n-grams sequential pattern features. In *IEEE ICIEA*.

Menahem, E., Shabtai, A., and Levhar, A. (2013). Poster: Detecting malware through temporal function-based features. CCS '13. ACM.

Rezende, E., Ruppert, G., Carvalho, T., Theophilo, A., Ramos, F., and de Geus, P. (2018). Malicious software classification using vgg16 deep neural network's bottleneck features. In *ITNG*. Springer.

Stewart, R. (2019). New backdoor malware found infecting wordpress and joomla websites. `https://bit.ly/2QzWpbQ`.