Adversarial Evasion-Resilient Hardware Malware Detectors

Invited paper

Khaled N. Khasawneh
University of California, Riverside
kkhas001@ucr.edu

Nael B. Abu-Ghazaleh
University of California, Riverside
naelag@ucr.edu

Dmitry Ponomarev
Binghamton University
dima@cs.binghamton.edu

Lei Yu
Binghamton University
lyu@cs.binghamton.edu

ABSTRACT

Machine learning offers tantalizing possibilities in computing and autonomous systems: data driven components and systems are trained to learn their environment and offer decisions comparable or surpassing those of humans. However, adversaries can learn the behavior of classifiers and construct adversarial examples that cause them to make wrong decisions, with potentially disastrous consequences. We explore this space in the context of Hardware Malware Detectors (HMDs), which have recently been proposed as a defense against the proliferation of malware. These detectors use low-level features, that can be collected by the hardware performance monitoring units on modern CPUs to detect malware as a computational anomaly. An adversary can reverse engineer existing HMDs effectively and use the reverse engineered model to create malware that evades detection. To address this critical problem, we developed evasion-resilient detectors that leverage recent results in adversarial machine learning to provide a theoretically quantifiable advantage in resilience to reverse engineering and evasion. Specifically, these detectors use multiple base detectors and switch between them stochastically, providing protection against reverse engineering and therefore evasion. The detectors rely on diversity of the baseline classifiers and their evasion advantage correlates with how often they disagree. Thus, it's critical to study how correlated the decisions from different baseline detectors are: a characteristic called transferability. We study transferability across different classifier algorithms and internal settings discovering that non-differentiable algorithms make the best candidates for operation in adversarial settings.

ACM Reference Format:
Khaled N. Khasawneh, Nael B. Abu-Ghazaleh, Dmitry Ponomarev, and Lei Yu. 2018. Adversarial Evasion-Resilient Hardware Malware Detectors, Invited paper. In *IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN (ICCAD '18), November 5–8, 2018, San Diego, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3240765.3243463

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICCAD '18, November 5–8, 2018, San Diego, CA, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5950-4/18/11…$15.00
https://doi.org/10.1145/3240765.3243463

1 INTRODUCTION

This section should clearly state the problem being addressed and explain the reasons for seeking a solution to this problem.

Computing systems continue to be the target of attacks by increasingly sophisticated and motivated adversaries. Systems are compromised through exploits and vulnerabilities that enable the attackers to deploy malware with different goals including stealing private information or creating botnets [14]. Although significant effort continues to be directed at making systems more secure, the goal of preventing all vulnerabilities is likely to continue to be elusive. Moreover, even if a system is completely secure, attacks that target users, for example, to trick them to execute malicious code, are also difficult to prevent.

Thus, it is critical to invest in detection techniques that allow us to rapidly detect the presence of malware to limit its damage. Unfortunately, detecting malware after it compromises a system is also becoming increasingly more complicated. Typical techniques for malware detection [3, 7, 8, 26] have both coverage limitations and introduce substantial overhead (e.g., 10x slowdown for information flow tracking is typical in software [25]). These difficulties limit malware detection to static signature-based virus scanning tools [4] which have known limitations [15], allowing the attackers to bypass them and remain undetected.

In response to these trends, Hardware Malware Detectors (HMDs) have recently been proposed to make systems more malware-resistant. Several studies have shown that malware can be classified based on low-level hardware features such as instruction mixes, memory reference patterns, and architectural state information such as cache miss rates and branch prediction rates [2, 10, 13, 17, 20, 23]. In addition, the SnapDragon processor from Qualcomm appears to be using HMDs for online malware detection, although the technical details are not published [22]. HMDs can offer a substantial advantage to defenders because it is always on and has little impact on performance or power [17].

Should HMDs become widely deployed, it is natural to expect that attackers will attempt to evade detection; this is an adversarial machine learning setting [18, 24]. Our prior work [12] explored whether attackers can adapt malware to continue to operate while avoiding detection by HMDs. We find that this is indeed the case: attackers can reverse engineer a detector and use the reverse engineered model to evade detection. We also explored whether retraining can help and found that some classifiers can be retrained, but malware can continue to reverse engineer and evade detection. Moreover, after several cycles of evasion and retraining, classifiers may no longer be able to separate the evolving malware from normal programs.

To address these issues, we explored the principle of evasion resilient hardware malware detectors (RHMDs). Specifically, RHMDs consist of several different baseline detectors. Instead of using all detectors together (i.e., ensemble learning), RHMD stochastically switches between the detectors, using only one at a time. This random switching between detectors makes it difficult (in a theoretically quantifiable way, based on Probably Approximately Correct, or PAC, theory [24]) to reverse engineer the detector, and therefore more difficult to evade it.

Central to the ability of RHMDs to provide guaranteed protection is an assumption that the underlying detectors are heterogeneous. It is important to examine this assumption especially under adversarial examples. Specifically, transferability [19] is the property of how well a two different classifiers correlate with each with respect to adversarial examples. If transferability is high, a malicious evasive malware can simultaneously hide from the different detectors, and the RHMD construct does not sufficiently provide protection. In fact, the protection provided by RHMD correlates to how often the baseline detectors disagree. Thus, this paper explores transferability between HMDs.

We explore transferability across different learning algorithms, as well as with respect to the parameters used in each algorithm (e.g., using different detection periods or different features), as well when both the learning algorithm and the parameters change. We discover that transferability is lowest for algorithms with non-differentiable classification boundaries such as decision trees. In contrast, both logistic regression and neural networks are highly transferable among each other, but not to non-differentiable DTs. Based on these results, we conclude by offering recommendations for constructing RHMDs.

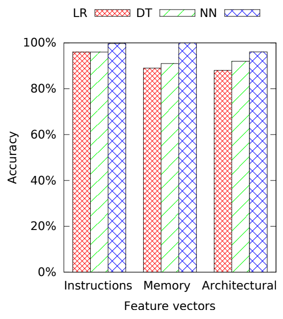
2 EVASION-RESILIENT HARDWARE MALWARE DETECTORS (RHMDS)

We start from prior works [2, 10, 13, 17] that established that HMDs can effectively detect malware (without evasion). To motivate our transferability study, we have to first introduce and provide insights into the structure of RHMDs: Hardware Malware Detectors that are resilient to evasion [12]. We structure this section around answering a set of successive questions.

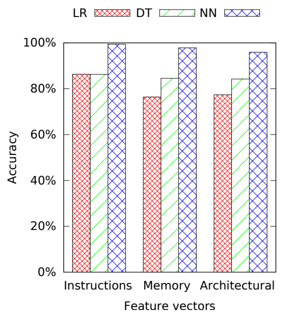
Question 1: Can an adversary reverse engineer an HMD?

By reverse engineering an HMD, an adversary can extract the detection model that the HMD uses. Recent results in adversarial classification imply that arbitrarily complex but deterministic classifiers can be reverse-engineered [24]. We confirmed this results by showing that we can efficiently reverse engineer a number of detectors under realistic assumptions. Specifically,Figure 1 shows the accuracy of reverse engineering classifiers built using Logistic Regression (left figure) or Neural Networks (right figure) using different features (instruction mixes, memory address distributions or architectural features such as cache misses) and different learning algorithms.

Question 2: Having a model of the detector, can the attacker create malware that will evade detection?



(a) Logistic regression



(b) Neural networks

Figure 1: Reverse-engineering efficiency

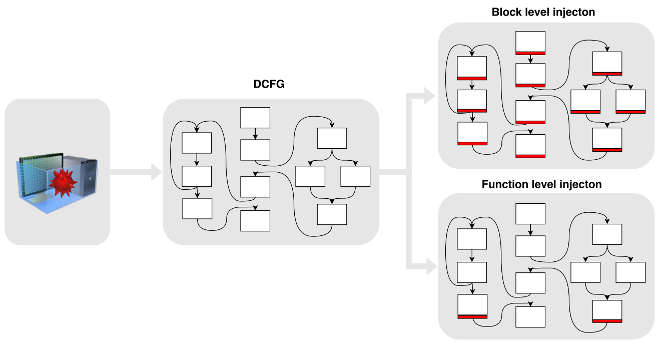


Figure 2: Methodology for generating evasive malware

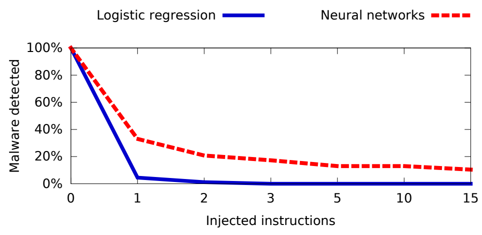
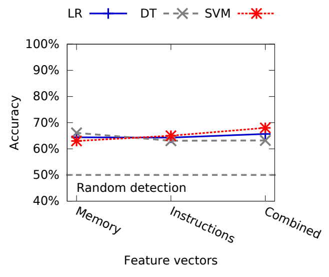


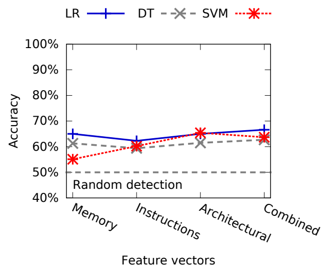
Figure 3: Malware detection accuracy under attack

We create evasive malware by starting from normal malware and injecting instructions at different granularity (Figure 2): at the basic block or at the function level. Specifically, when we inject at the basic block level, we inject instructions at every basic block. We chose this approach because we wanted to experiment with real malware which implies that we do not have the source code, and in most instances, the binary resists decompilation because of obfuscation and packing. The injected instructions are selected based on the reverse engineered model to maximize evasion likelihood based on the reverse engineered model of the detectors.

For Logistic Regression (linear model) and Neural Networks (non-linear model), we showed that existing HMDs can be rendered ineffective using simple modifications to the malware binary (Figure 3). In practice, malware writers have even larger leeway in implementing evasive strategies; because of difficulties in de-compiling



(a) Two feature vectors



(b) Three feature vectors

Figure 4: RHMD reverse engineering

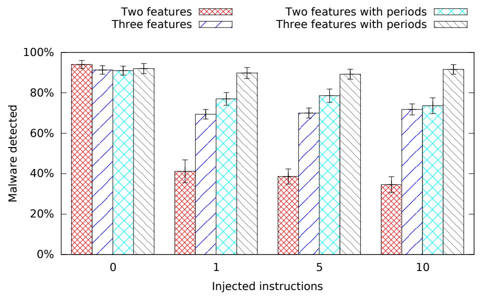


Figure 5: RHMD evasion resilience

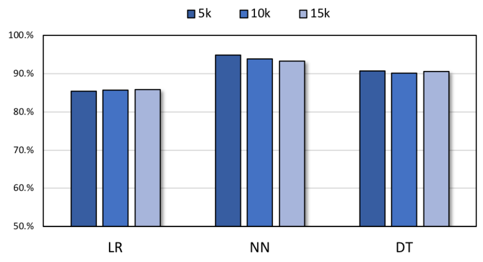


Figure 6: Evaluation of detection accuracy for all detectors; the detectors were built using the three machine learning algorithms and across all collection periods

obfuscated malware, we had to rely only on adding instructions. For example, malware writers with source code can also implement rewriting strategies that reduce the number of instructions from categories with a large positive weight.

Question 3: Does retraining HMDs help?

We explore whether malware can evade detection even if the detector is retrained with some samples of the evasive malware. If retraining is possible, this opens the door to potentially periodically updating detectors as malware evolves. We show that for the simple evasion strategies that can fool a given detector, retraining a linear detector is ineffective unless we sacrifice the detection performance on normal malware; if the training set is biased towards evasive malware, the detector can no longer detect non-evasive malware. On the other hand, if we do not, then evasive malware detection remains poor (and normal malware detection still suffers). Interestingly, more sophisticated detectors (non-linear) can be successfully retrained, but the attacker is still able to reverse engineer the retrained detector and evade it again. The detection performance eventually degrades after several cycles of evade and retrain.

Question 4: Can we make HMDs robust to evasion?

After showing that the current generation of HMDs is vulnerable to evasion attacks, we explore whether new HMDs can be constructed that are robust to evasion. In particular, we propose a new resilient HMD organization that uses multiple diverse detectors and switches between them unpredictably. We show that detectors built in this fashion are resilient to both reverse engineering (Figure 4) and evasion (Figure 5). This resilience increases with the number and diversity of the individual detectors. In addition, we study implementation complexity of such classifiers in hardware

3 TRANSFERABILITY

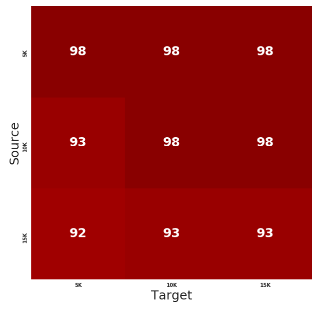
We showed that we can reverse engineer detectors that use different features, different detection periods, and different learning algorithms. Based on the reverse engineered detectors, we were able to craft evasive malware efficiently. Transferability [19] is a property where one classifier (Classifier A) behaves similarly to another (Classifier B) when they have different architectures (learning algorithm, features, and other parameters) and even when they are trained with a different training set.

Specifically, transferability is of interest with respect to adversarial examples where it has important implications: for example, it implies that adversarial examples developed against one classifier can be effective in fooling another which is the basis for black box attacks (such as the ones we used). Transferability also impacts our RHMD defense. Specifically, if our baseline classifiers are highly transferable, this means that the agree often and therefore have low diversity with respect to adversarial examples making them easy t evade. Thus, understanding transferability properties is critical to our choice of baseline detectors for RHMD.

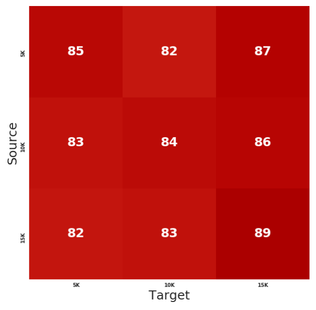
To understand the transferability properties in this application space, in this section we study both intra-algorithm (same classification algorithm, with different parameters) and cross-algorithm transferability and their implications on evasion attacks. In addition, we use these results to understand the behavior of RHMD configurations, and how to best construct them.

3.1 Experimental Setup

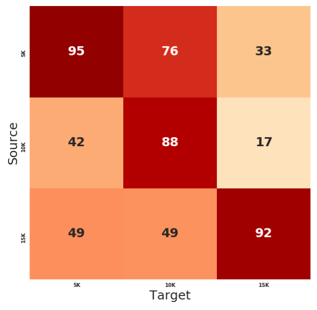
Dataset: we use the same dataset and data collection technique that we used in our prior work [12]. This dataset contains 3000 malware and 554 regular program (all for Windows Desktop machines). The dynamic trace was collected by running both regular programs and malwares on a Windows 7 virtual machine using Pin instrumentation tool [1]. The collected trace duration for each executed program was 5000 system calls or 15 million of committed instructions, starting after a warm-up period, whichever is reached first. The dataset was divided into 60% *training* to train the detectors, and 40% *testing* (used to create evasive malware and test their evasion transferability). Furthermore, the *training* and *testing* datasets traces was prepared using three different collection periods (5K, 10K, and 15K). Collection period refers to the size of the instruction window used to collect the classification features.



(a) Logistic regression (LR)



(b) Neural networks (NN)



(c) Decision Tree (DT)

Figure 7: Intra-algorithm transferability results for three machine learning algorithms. Cell (i, j) reports the intra-algorithm transferability between detector i (source) and detector j (target), i.e. the percentage of evasive malware produced using detector i that can evade detection of detector j.

Table 1: Machine learning algorithms that was used for the transferability study

ML Technique	Differentiable Model	Linear Model
LR	Yes	Log-linear
NN	Yes	No
DT	No	No

Algorithms: for this study, we selected three machine learning techniques: neural networks (NN), logistic regression (LR), and decision trees (DT). As shown in Table 1, we selected NN for its state-of-the-art performance, LR for its simplicity, and DTs for their non-differentiability. To train all detectors we have used scikit-Learn [21]. NN is implemented as a multi-layer perceptron (MLP), which has a hierarchy of a single hidden layer that has a number of neurons equal to the number of features in the feature vector. Additionally, we use the `tanh` function as the activation function. LR and DT are trained using scikit-Learn default parameters. For each machine learning algorithm, we train three detectors using the three detection periods that form the *training* dataset. Figure 6, reports the accuracy results for all the detectors that were trained and used in this study for the *testing* dataset.

Evasion techniques: for each algorithm we construct evasive malware samples specifically to evade the detection of the HMDs based on the algorithm. For example, since LR uses a linear separation boundary, we inject instructions that can move the malware fastest across this boundary. For LR and NN we have used the same evasion techniques described in [12]. For DTs, to select what to inject in order to craft an evasive malware, we look at the tree and we try to find the shortest path that leads to a regular program decision. Since in our methodology, we can only add instructions to the malware, we need to select a path that can be taken only by increasing the number of instructions to effect the decisions along that path. We inject from this selection of instruction to cause the decision to proceed along the chosen path.

Since the evasive strategy is highly specific to the features, although we would like to study transferability with respect to feature selection, the transferability is extremely low in our current set up. Specifically, our evasion strategy inserts instructions that affect the specific feature assumed but have no effect that correlates to the other features. For example, the insertion policy to affect instruction mixes has negligible effects on architectural features (for example, cache miss or branch prediction events), or memory features (memory accesses distributions). We believe this is a limitation of our evasion policy.

3.2 Intra-algorithm transferability

We measure intra-algorithm transferability between detector i (source) and detector j (target), both trained using the same algorithm, using the same or different collection periods, as the proportion of evasive malware samples created using detector i that can evade detector j. If this number is high, that means that adversarial examples are transferable across the detectors.

We study intra-algorithm transferability for the three machine learning algorithms across different collection periods. Figures 7a-7c shows the intra-algorithm rates for each of the three machine learning algorithms used in our study. In the figures, the rates on the diagonal (i,i) represent the proportion of the evasive malwares that were able to evade the detection by the detector i which was used to create the evasive malware. On the other hand, the off-diagonal rates (i,j) represents the proportion of the evasive malwares that were able to evade detection of detector j and were created using detector i. The first observation that can be inferred from these figures is that all algorithms are vulnerable to intra-algorithm transferability at a non-negligible manner. Figure 7a, shows that LR is the most vulnerable as evasive malwares across all detectors can transfer with a rate larger than 92%. For NN, Figure 7b shows that are also vulnerable with a transferability rate of at least 81%. On the other hand, for the DT, Figure 7c, shows that the diagonal stand out more, which means that this algorithm is to some extent more robust to evasion attacks transferability. The robustness of DTs could simply stem from their non-differentiability. The conclusion is that differentiable algorithm like NN and LR are more vulnerable to intra-algorithm transferability attacks than non-differentiable algorithms like DT.

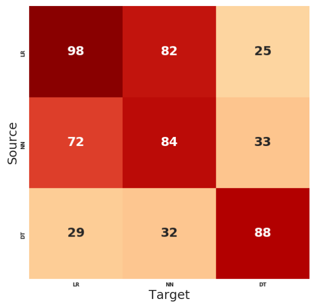


Figure 8: Cross-algorithm transferability for three classification algorithms. Each cell (i, j) reports the cross-algorithm transferability between detector i (source) and detector j (target) which is the percentage of evasive malware generated using detector i that can evade detection of detector j.

3.3 Cross-algorithm transferability

We use the same methodology to measure cross-algorithm transferability. We study cross-algorithm transferability across the three machine learning algorithms using a setting with 10K collection period for each. Figure 8, shows the resulting cross-algorithm transferability matrix. Recall that the rows indicate the detector that was used to create the evasive malware while the columns indicate the detector that was used to classify the evasive malware. The results show that LR and NN exhibit high cross-algorithm transferability; LR to NN cross-algorithm transferability rate is 82.33% and NN to LR cross-algorithm transferability rate is 71.69%. What is interesting is that the cross-algorithm transferability when we craft evasive malware using DT is low; DT to LR cross-algorithm transferability rate is 28.6% and DT to NN cross-algorithm transferability rate is %31.89. Additionally, DT was the most resilient detector to cross-algorithm transferability attacks with a cross-algorithm transferability rates of 24.53% and 32.55% from LR and NN respectively.

3.4 Combined transferability

We also investigate whether transferability occurs across classifiers with both different classification algorithms and classification periods (Figure 9). The same observations from before hold: the non-differentiable classification algorithm exhibits low transferability both intra and inter-algorithm, as well as in combination. In contrast, both LR and NN exhibit high transferability between each other, even when the classification period also changes.

4 IMPLICATIONS ON RHMD

In this section, we discuss some of the implications of the transferability results both on attacks on HMDs, as well as on the RHMD defense. With respect to the attack, high transferability to the destination classifier implies that black box attacks become easier. Specifically, we can train any model and expect to have adversarially engineered examples that fool the model to transfer to fool the target classifier. Thus, to raise the barrier for black box attacks, we recommend using a low transferability target such as decision trees. We note that the choice and configuration of the detector does not

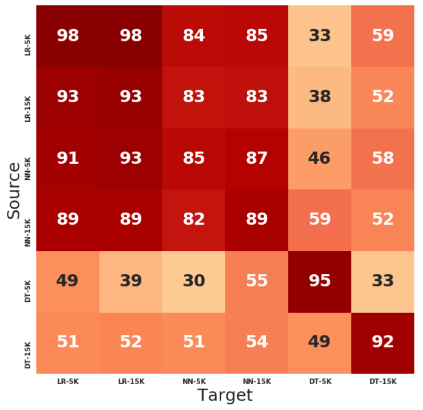


Figure 9: Combined transferability across classifiers and change of classification period. Similar conclusions hold: transferability to and from DT is low and between LR and NN is high, even with different classificaiton periods.

provide perfect resilience since the attacker can test many hypotheses on the model architecture and configuration until they obtain a high transferability model. In contrast, we believe that RHMD provides a theoretically significant advantage in the difficulty of reverse engineering that originates from the unpredictability of the responses from the different baseline detectors.

We believe that transferability also influences RHMDs. Intuitively, if the baseline detectors are highly transferable it is likely that an evasive malware could evade each, essentially indicating weak diversity in their decisions. In order to construct more robust RHMDs, we consider transferability impact along 3 axes: resilience to evasion attacks, accuracy, and implementation overhead.

Resilience to evasion: intra-algorithm transferability results show that differentiable algorithm like NN and LR are more vulnerable to intra-algorithm transferability attacks than non-differentiable algorithms like DT. From RHMD perspective, this suggests that using base detectors using differentiable algorithms and diversified using different collection periods will not help make RHMD more resilient since no matter which base detector the RHMD chooses randomly for detection, there is a high probability that the evasive sample created for any detector can transfer across all detectors. On the other hand, non-differentiable algorithms have low transferability intra-algorithm, making it possible to use different detector mixes without suffering high transferability within an RHMD.

In addition, the cross-algorithm transferability results show that probability of cross-algorithm transferability across LR and NN is very high. However, this was not the case for DT with both LR and NN. Therefore, this suggests that having the base detectors of the RHMD formed from NN and LR only will not help in makings RHMD more resilience due to high transferability. On the other hand, having a mix of differentiable and nondifferentiable algorithms to train the base detectors of the RHMD would help on making the RHMD more resilient to evasion attacks since the probability of cross-algorithm transferability is not high.

Accuracy: in terms of accuracy, the NN detectors showed the best accuracy across all detection periods (Figure 6). Moreover, it showed moderate intra-algorithm and cross-algorithm transferability. Therefore, having NN base detectors would allow the RHMD to have the best accuracy, while providing moderate transferability compared to the other detectors (LR and DT).

Overhead: LR detectors are the lightest in terms of overhead. However, they have the lowest accuracy and the highest transferability compared to the other detectors (NN and DT).

We caution that the transferability results and therefore some of our conclusions are dependent on the evasion approach; other evasion techniques (for example, those that rewrite the malware, or those that attempt to evade in multiple feature spaces concurrently) may result in different transferability properties. We explored only a subset of the sources of diversity, excluding, importantly, differences in training sets, and differences in the classification features, due to limitations of our evasion methodology.

5 CONCLUDING REMARKS

We considered the problem of adversarial evasion resilient hardware malware detectors (RHMDs). Hardware Malware Detectors (HMDs) is a proposed approach for malware detection where we classify the behavior of programs in low-level feature spaces (e.g., architectural signals) to identify malware as a computational anomaly. In prior work, we showed that HMDs can be easily evaded through a black box attack where we reverse engineer the detector, and use the reverse engineered model to easily evade it. We also showed that retraining the detector eventually fails to capture the evolving behavior of malware. As a result, RHMDs offer evasion resilience by using multiple baseline detectors and switching between them stochastically, offering theoretically quantifiable advantage in reverse engineering accuracy. With low reverse engineering accuracy, evasion through a black box attack becomes impractical.

In this paper, we study the problem of transferability between detectors in the context of this application. Transferability refers to how well adversarial examples developed to evade one detector (example the reverse engineered model) serve to evade another (e.g., the target detector). This property has substantial implications on the attack as well as the defense offered by RHMD. We characterized transferability both within the same learning algorithm (but with different configuration parameters), as well as across learning algorithms. We discover that non-differentiable classifiers exhibit low transferability both within and across algorithms. In contrast, differentiable classifiers transfer with high probability to other differentiable algorithms, as well as to other configurations within the same algorithm. Based on these results, we recommend that non-differentiable algorithms such as DTs be used in the mix of RHMD baseline detectors.

It is interesting to explore whether the classifier captures fundamental properties of malware. While this is a difficult question for machine learning in general, it is clear that some attacks have a specific and difficult to transform computational footprints. Examples of such attacks include covert-channel attacks [5, 16], side-channel attacks [6, 9] and ransomware [11].

ACKNOWLEDGEMENTS

The work in this paper is partially supported by National Science Foundation grants CNS-1619322 and CNS-1617915.

REFERENCES

[1] C.Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. Reddi, and K. Hazelwood. 2005. Pin: building customized program analysis tools with dynamic instrumentation. In *Proc. PLDI.*

[2] John Demme, Matthew Maycock, Jared Schmitz, Adrian Tang, Adam Waksman, Simha Sethumadhavan, and Salvatore Stolfo. 2013. On the feasibility of online malware detection with performance counters. *ACM SIGARCH Computer Architecture News* 41, 3 (2013), 559–570.

[3] Artem Dinaburg, Paul Royal, Monirul Sharif, and Wenke Lee. 2008. Ether: malware analysis via hardware virtualization extensions. In *Proceedings of the 15th ACM conference on Computer and communications security (CCS).* 51–62.

[4] Manuel Egele, Theodoor Scholte, Engin Kirda, and Christopher Kruegel. 2012. A survey on automated dynamic malware-analysis techniques and tools. *ACM Computing Surveys (CSUR)* 44, 2 (2012).

[5] D. Evtyushkin, D. Ponomarev, and N. Abu-Ghazaleh. 2016. Understanding and mitigating covert channels through branch predictors. *ACM TACO* 13, 1 (2016), 10.

[6] D. Evtyushkin, R. Riley, N. Abu-Ghazaleh, and D. Ponomarev. 2018. BranchScope: A New Side-Channel Attack on Directional Branch Predictor. In *Proc. ASPLOS.* 693–707.

[7] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. 2003. Terra: a virtual machine-based platform for trusted computing. In *Proc.SOSP.*

[8] Guofei Gu, Phillip Porras, Vinod Yegneswaran, Martin Fong, and Wenke Lee. 2007. BotHunter: Detecting Malware Infection Through IDS-driven Dialog Correlation. In *Proceedings of 16th USENIX Security Symposium.*

[9] M. Kayaalp, N. Abu-Ghazaleh, D. Ponomarev, and A. Jaleel. 2016. A high-resolution side-channel attack on last-level cache. In *Proc. DAC.*

[10] M. Kazdagli, V. J. Reddi, and M. Tiwari. 2016. Quantifying and improving the efficiency of hardware-based mobile malware detectors. (2016), 1–13.

[11] Amin Kharraz, William Robertson, Davide Balzarotti, Leyla Bilge, and Engin Kirda. 2015. Cutting the gordian knot: A look under the hood of ransomware attacks. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment.* 3–24.

[12] K. Khasawneh, N. Abu-Ghazaleh, D. Ponomarev, and L. Yu. 2017. RHMD: Evasion-resilient Hardware Malware Detectors. In *Proceedings of Micro.* 315–327.

[13] K. Khasawneh, M. Ozsoy, C. Donovick, N. Abu-Ghazaleh, and D. Ponomarev. 2015. Ensemble Learning for Low-Level Hardware-Supported Malware Detection. In *Proc. RAID.* 3–25.

[14] G. McGraw and G. Morrisett. 2000. Attacking Malicious Code: Report to the InfoSec Research Council. *IEEE Software* 17, 5 (Sept. 2000), 33–41.

[15] A. Moser, C. Kruegel, and E. Kirda. 2007. Limits of Static Analysis of Malware Detection. In *Proc. ACSAC.* 421–430.

[16] H. Naghibijouybari, K. Khasawneh, and N. Abu-Ghazaleh. 2017. Constructing and characterizing covert channels on GPGPUs. In *Proc. Micro.* 354–366.

[17] M. Ozsoy, C. Donovick, I. Gorelik, N. Abu-Ghazaleh, and D. Ponomarev. 2015. Malware-aware processors: A framework for efficient online malware detection. In *High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on.* IEEE, 651–661.

[18] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2016. Practical black-box attacks against deep learning systems using adversarial examples. *arXiv preprint arXiv:1602.02697* (2016).

[19] Nicolas Papernot, Patrick D. McDaniel, and Ian J. Goodfellow. 2016. Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples. *CoRR* abs/1605.07277 (2016). http://arxiv.org/abs/1605.07277

[20] N. Patel, A. Sasan, and H. Homayoun. 2017. Analyzing Hardware Based Malware Detectors. In *Proceedings of the 54th Annual Design Automation Conference 2017.*

[21] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research* 12, Oct (2011), 2825–2830.

[22] Qualcomm 2016. Qualcomm Smart Protect Technology. (2016). Last Accessed July 2016 from https://www.qualcomm.com/products/snapdragon/security/smart-protect.

[23] Adrian Tang, Simha Sethumadhavan, and Salvatore J Stolfo. 2014. Unsupervised anomaly-based malware detection using hardware features. In *International Symposium on Recent Advances in Intrusion Detection (RAID).* 109–129.

[24] Y. Vorobeychik and B. Li. 2014. Optimal Randomized Classification in Adversarial Settings. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014).*

[25] Guanhua Yan, Nathan Brown, and Deguang Kong. 2013. Exploring discriminatory features for automated malware classification. In *Detection of Intrusions and Malware, and Vulnerability Assessment.* Springer, 41–61.

[26] Heng Yin, Dawn Song, Manuel Egele, Christopher Kruegel, and Engin Kirda. 2007. Panorama: capturing system-wide information flow for malware detection and analysis. In *Proceedings of the 14th ACM conference on Computer and communications security (CCS).* 116–127.