Towards moving target defense for IoT malware detection

Ita Ryan
School of Computer Science and Information Technology
University College Cork
Cork, Ireland
iryan@ucc.ie

Luke Kurlandski
Rochester Institute of Technology
Rochester, USA
lk3591@g.rit.edu

Nate Mathews
Rochester Institute of Technology
Rochester, USA
nate.mathews@mail.rit.edu

Abstract

Machine learning (ML) techniques show promise in malware defense for the Internet of Things (IoT), but are vulnerable to tailored adversarial attacks. Moving Target Defense (MTD) is a security strategy that actively raises the cost to the attacker of a potential attack by changing the target's characteristics, preventing attackers from profiling the target. In this work we explore the potential for using MTD for IoT malware detection. Applying MTD to protect ML malware detection involves continuously changing the malware classification models, defeating attempts to profile the models. We research the state-of-the-art literature that uses an MTD-style strategy to increase ML model security. We identify two techniques: 'Naive MTD', which cycles between static models, and 'Full MTD', which refreshes models at runtime and is therefore more effective. Focusing on the studies in the ML literature that use Full MTD for adversarial robustness, we examine their approach, assessing features such as discard policy, decision-making and model updating schedule. We make a number of recommendations on development of a Full MTD strategy for ML IoT malware detection.

ACM Reference Format:
Ita Ryan, Luke Kurlandski, and Nate Mathews. 2026. Towards moving target defense for IoT malware detection. In *8th International Workshop on Software Engineering Research and Practices for the IoT (SERP4IoT '26), April 12–18, 2026, Rio de Janeiro, Brazil*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3786159.3788477

1 Introduction

IoT devices in commercial environments frequently reside on networks, and may be protected by anti-malware applications based on ML. However, ML models, once deployed, are vulnerable to adversarial attack, where malware is specially tailored to avoid detection. Techniques to counter such attacks include forcing the model to use more information in decision-making [50], certifying robustness [13], and using defensive distillation [33] or processing model inputs before training [53] to reduce perturbation sensitivity. Many such defenses have been shown to have vulnerabilities, and are insufficient on their own to protect ML-based systems [6].

A promising complimentary strategy is to accept that no defense is perfect, and use the cybersecurity technique of MTD to keep the defender's characteristics fluid. Individual IoT devices are sometimes obtained, studied and profiled by an attacker. Profiles of malware defense ML models obtained in this way can be leveraged to craft tailored malware designed to defeat the identical models on all similar devices. Using MTD to constantly change decision boundaries makes such attacks challenging [43].

In this paper we reviewed the literature on the use of MTD in ML domains. Based on that review, we identified two types of MTD: 'Naive MTD' and 'Full MTD'. **Naive MTD** cycles between detection models at runtime to prevent model profiling by attackers. However, the pool of detection models is static. Sophisticated attackers may eventually profile them all, and craft malware to defeat them. **Full MTD** periodically generates new detection models at runtime to refresh the pool. This is a more robust defense because hacked model profiles are invalidated when the pool is refreshed. In IoT, it can be used to ensure all devices run different models. We identified state-of-the-art (SOTA) Full MTD works and evaluated them in detail, listing key takeaways for future work protecting malware detection models in IoT environments.

There is little related work that surveys MTD for ML, and we found none focused solely on malware detection. Zhang et al. surveyed the use of artificial intelligence (AI) in MTD, and the use of MTD in AI-driven networks [56]. They touched briefly on malware but did not explore MTD in the malware space. Rashid and Such ran experiments to assess the effectiveness of several Naive and Full MTD tools designed to counter adversarial attack [36]. Focusing on the 'original' paper for each tool, they missed the opportunity to assess much-improved designs [42]. Their analysis did not include recent work in the malware domain [58].

The contributions of this paper are as follows:

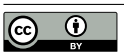
- Identifies two types of MTD for ML: Naive and Full
- Examines SOTA Full MTD papers in the ML space
- Explores current MTD for IoT
- Identifies key takeaways for IoT malware detection MTD

The rest of the paper is organized as follows. Section 2 describes the background for IoT, adversarial attack and robustness, and MTD. Section 3 presents the SOTA literature using MTD for adversarial robustness. Section 4 discusses lessons learned, identifying key takeaways for leveraging MTD to enhance ML defenses for IoT.

2 Background

2.1 Securing the Internet of Things (IoT)

The term *IoT* encapsulates the proliferation of peripheral devices on the modern Internet. Such devices have been leveraged *en masse* via malware, for example the Mozi botnet, to conduct distributed denial-of-service (DOS) attacks on high-profile targets [48]. IoT malware could also be deployed for other purposes, such as disabling power grids by infecting high-power devices [41]. Due to



SERP4IoT '26, Rio de Janeiro, Brazil

ACM ISBN 979-8-4007-2391-9/2026/04
https://doi.org/10.1145/3786159.3788477

inbuilt constraints (power, size, throughput, etc.), IoT devices are often less capable of defending themselves against intrusions than conventional computers [17]. The fusion of user and kernel space on some devices further complicates this process.

The first line of defense against IoT threats is a robust perimeter. Several works have proposed network monitoring strategies to restrict unauthorized communications to and from devices [38]. Unfortunately, simple allowlists that explicitly encode authorized communication endpoints can inhibit IoT devices' functionality [20].

Proposed solutions to making IoT devices more secure by default include guaranteeing control flow integrity [14] and sandboxing untrusted processes [57]. Software can be deployed on the device to block anomalous actions or verify a device's integrity, for example by identifying potential malware infections based on patterns in shellcode activity [24]. For smaller devices, these techniques may not be feasible. Lightweight '*roots of trust*' can check software state and verify run-time integrity [2], and projects such as ACFA [7] attempt to make them robust to infection. Anomaly detection [37] attempts to detect malicious activity masquerading as benign behaviour. Protocols have been developed to quarantine and eventually reclaim infected devices, and the first step is to identify such devices [44]. Following identification, devices are usually rebooted to restore their integrity [21].

Many works have proposed techniques to detect IoT malware using ML [55]. Active challenges faced in this domain include ensuring that models can run on small IoT devices [29], and detecting adversarial examples [15], which is discussed in the next section.

2.2 Adversarial Attack

'Adversarial examples' are inputs carefully crafted to fool an ML model [46]. A small, preferably imperceptible, perturbation is added to an input to cause its misclassification. A targeted attack produces an example that is classified (incorrectly) as a member of a specific class. An untargeted attack gives an incorrect classification. Early examples found the perturbation with minimal distortion that changed the classifier's prediction [46]. Later work simplified the optimization process, breaking most existing defenses (C&W) [6].

The Fast Gradient Sign Method (FGSM) optimizes to find the perturbation with maximal impact [16]. Its many variants are '*white-box*' attacks requiring unfettered access to the target model's gradients. '*Black-box*' attackers have limited feedback. If the target model's training data is available, a surrogate model can be trained and white-box adversarial examples generated. These often transfer to the target model [32]. The tendency of attacks to transfer across models is known as 'transferability'. Other black-box attacks take advantage of the model's confidence scores [9], or other observable features [8].

Unlike images, which may sustain changes without visible effect, malware is easily broken. Any adversarial malware generation attack must select a perturbation set [23] of functionality-preserving modifications that will not stop a piece of malware from running and completing its task. Early techniques were simple, for example adding 'no-op' API-calls [18], or injecting adversarial payloads at the end of files [22]. Recent more sophisticated techniques include rewiring control flow graphs [25]. After a perturbation set is defined, the attacker can apply a variety of optimization methods to produce the example. Due to the large, discrete nature of the malware perturbation space, gradient-free optimization methods such as Monte Carlo tree search [25] are commonly selected.

2.3 Adversarial Defense

Adversarial examples expose brittle decision boundaries in modern ML models, sparking a wide-ranging search for adversarial robustness: reduced susceptibility to adversarial examples. Techniques include Adversarial Training (AT) [28], purification defenses such as Feature Squeezing [53], baking robustness directly into the model [26], and certified defenses that establish adversarial robustness guarantees under specific threat models [52]. Another approach is to use an ensemble of models to spread risk: if models err differently, a unanimous or majority vote resists single-model attacks.

All have disadvantages. For example, stronger robustness guarantees usually mean higher compute cost or looser accuracy guarantees, while AT can be expensive depending on how the adversarial inputs are crafted and furthermore, typically reduces the classifier's natural accuracy [47]. AT can backfire in low-data regimes when it latches onto 'robust but useless features' [12]. Ensembling independently trained, vulnerable models often yields limited improvements in robustness due to the general transferability of adversarial examples [32], although ensembling can be more effective if the constituents of the ensemble are highly diverse [10].

These techniques have all been applied with success to the malware space. Adversarial training has been used to enhance malware classifiers that learn from handcrafted, tabular feature sets [3] as well as raw-byte classifiers that learn directly from executables [27].

2.4 MTD

The MTD cybersecurity strategy continuously and unpredictably alters an attack surface, increasing uncertainty and complexity for adversaries [51], invalidating meticulously-gathered intelligence, and significantly elevating the costs and resources required to mount successful attacks [30]. When used to defend IoT ML systems as described in this work, MTD entails *Shuffling* [11]: the randomization or dynamic rearrangement of system configurations and resources. The ML model configurations used at runtime are varied to confuse attackers. This may be done by selecting from a static, unchanging pool of models via a strategy such as game theory [35] (Naive MTD). Alternatively, new models may be generated at runtime, with old models completely discarded [4, 42, 43, 58] (Full MTD).

3 MTD for Adversarial Robustness

Adversarial attack relies on model profiling to identify useful attack strategies. MTD is ideally suited to counter this technique and provide ML malware detectors with adversarial robustness. MTD is also ideally suited to IoT defense. IoT developers face many challenges which make securing devices difficult [5]. Unlike well-defended systems residing in a back-end infrastructure, IoT devices may be out in the field where they can be captured by adversaries for profiling, or may be commercial devices that can be purchased and reverse engineered. For this threat model, varying malware detection per device means that compromising the detection model on one device does not compromise all others.

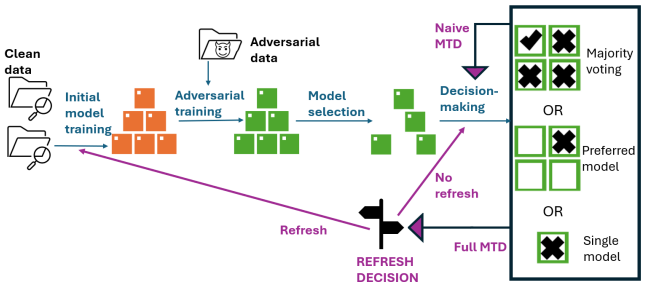


Figure 1: Building adversarially robust MTD models. Common steps are labelled in blue. Full v. naive MTD steps are in purple. Only Full MTD approaches include a refresh step.

3.1 Characterising MTD for Adversarial Robustness

Based on our analysis of SOTA research, in this work we identify two types of MTD for adversarial robustness: Naive MTD and Full MTD. The training and decision-making process for adversarially robust models using MTD can be seen in Figure 1. Naive MTD approaches do not include the refresh stage.

Naive MTD solutions select from a static pool of models.

They include all the variations of ensemble-style defense where the overall pool of models remains unchanged. Scheduling algorithms or heuristics are used to select between these static ML models [1, 19, 34, 40]. An example is Stratdef, a framework that allows the defender to make choices at every stage to tailor the range of models and the probability that each will be used [35]. In Stratdef, groups of models are selected for the expected 'threat model', a concept that here comprises attacker strength and attack intensity. Since deployment is expensive, live Stratdef instances are likely to run in environments that are expected to have a maximum-stress 'strong attacker, high intensity' threat model. A detailed analysis of how Stratdef assigns models at runtime is available in Appendix A of our online supplementary material [39]. It shows that in some configurations of the 'strong attacker, high intensity' setting, one model has a 0.985507 probability of being deployed at detection time, and there is only one alternative model. In other words, in one of its most likely deployment scenarios Stratdef will use one single model more than 98% of the time. For the remaining 1.5% of the time, it will only use one other model.

Although usually the number of active models is greater than two, all Naive MTD defenses cycle between static models which are never replaced. Sophisticated attacks can defeat the individual static models comprised in ensembles over time [54]. Therefore, we do not consider Naive MTD to be sufficiently robust to defend IoT malware detectors. It is not examined further in this paper.

Full MTD solutions automatically build and deploy new models periodically at runtime.

This minimizes the impact of ML model profiling. Refreshing frequently with entirely new models each time provides diversity to the MTD process and invalidates previous profiling. This study identifies and evaluates SOTA Full MTD research.

3.2 Method

We undertook a SOTA literature review along PRISMA review guidelines [31]. MTD is a term that is used primarily in the security domain, and might not be familiar to ML researchers. Therefore, we focused on the question '*What can we find on retraining of ML models for the purpose of defeating adversarial attacks?*'. Three researchers searched relevant databases for conference and journal papers, and did forward and backward citation searching of identified papers. A review process diagram is in Appendix C of the supplementary material [39]. From 122 papers of interest, we found only four papers using the Full MTD approach, regenerating and replacing models at runtime. They are listed in Table 1. Three are in the image domain [4, 42, 43], with one malware detector [58]. We discuss each of these papers in this section. There is a detailed description of the datasets they used in Appendix B [39].

3.3 Studies Using Full MTD

3.3.1 Morphence. Morphence [4] uses MTD to reduce an ML image classifier's vulnerability. An initial clean base model is trained on the dataset. As can be seen in Table 1, a variable number (n) of student models are then perturbed with random perturbations sampled from the Laplace distribution, and retrained. The intuition is that due to the differing perturbations the decision boundaries will vary between the students. However, if all n models used the same data for retraining they would be likely to re-converge. A unique dataset per model is required. Due to data scarcity, unique datasets are generated by applying distinct randomly-chosen transformations (such as rotation) to the original dataset.

To increase adversarial robustness, a subset $p<n$ of the models is then given adversarial training. This leaves (n-p) models trained on clean data, which should ensure there is no decrease in effectiveness in classification of clean data. (Adversarially-trained models normally lose accuracy on clean data.)

All models are trained to the required accuracy (see Table 2). At runtime, all n models classify each input. The 'scheduler' selects the classification with the highest confidence. The intuition is that models trained only on clean data (n-p) will have highest confidence for clean images, while adversarially-trained models ($p<n$) will have highest confidence for adversarial images.

After $Qmax$ queries have been received, all n models are replaced by a new set of n models. This supplies the MTD component of the defense. Model replacement is intended to foil attackers who are repeatedly querying models in order to build profiles of them. New models are built in the background to replace those that are being profiled. Since n models must be available at replacement time, replacement frequency is dependent on $Qmax$, query time (t), model generation time, and n.

Results: Considerable testing was done to compare full Morphence to one or more models with and without transformations, adversarial training, and MTD. The researchers found that Morphence improved detection of adversarial attack on both MNIST and CIFAR, with little (<1%) adverse effect on classification of clean data. In addition, Morphence limited the success of repeating successful attacks, due to the MTD component.

The researchers determined that the value of the hyperparameter λ, which controlled the noise level of the Laplace distribution used

Table 1: Overview of Papers Using State-of-the-Art MTD for Adversarial Robustness

Paper	Adversary	Initial training data source	Labelling strategy	Model(s)	Initial training
Morphence [4] Amich et al. 2021	Adversarial attack on computer vision	MNIST, CIFAR-10, per-model data augmentations (e.g. rotations, translations)	Uses the datasets' ground-truth class labels	MNIST: six-layer CNN trained via standard supervised learning. CIFAR-10: VGG-16 architecture, trained via supervised SGD.	Base model trained on clean data. n student models retrained from Laplace-perturbed base-model weights. Subset p<n given adversarial training
DeepMTD [43] Song et al. 2022	Adversarial attack on visual sensing in vehicles	MNIST, CIFAR-10, GTSRB	Uses the datasets' ground-truth class labels	9-layer CNN-A trained on MNIST using momentum-based SGD. More complex 9-layer CNN-B applied to CIFAR-10 and GTSRB.	n 'fork models' generated by retraining independently perturbed versions of base model.
Sardino [42] Song et al. 2023	As DeepMTD	As DeepMTD	As DeepMTD	As DeepMTD	Set of MLPs generates DNN weights with random number inputs to produce n 'fork models': 1.35m x faster than DeepMTD.
MTDroid [58] Zhou et al. 2024	Malware on Android	Subsets of Drebin, CICMalDroid 2020, and AndroZoo	Malicious: 5+ AV scanners alarm. Benign: no alarms	Ensemble - MLP, SVM, CNN, DT, LR, KNN, LSTM	Seven basic models each trained on distinct subset of the dataset to form initial model pool.

Table 2: Moving Target Defense Strategy for Reviewed Papers

Short name	Discard policy	Decision-making	Refresh retraining	Model updating
Morphence [4]	All models trained to required min accuracy. No discards	For each query, pick the student model with the highest confidence and return its prediction.	Same data and training as original	After query budget $Qmax$, active pool (n models) is retired, pre-generated new pool of n models is swapped in
DeepMTD [43]	All models trained to required min accuracy. No discards	1) if results differ within ensemble, detect adversarial input 2) if input is adversarial: a) refer to human OR b) find true classification via majority voting	Same data and training as original	Immediately after receipt of base model (e.g. when car is shipped). Can also be done periodically during low-compute times (e.g. when car is charging).
Sardino [42]	No accuracy check as it is too time-consuming	As DeepMTD	As DeepMTD	Ensemble renewed per query, with n models based on available compute time.
MTDroid [58]	Preferred sub-classifiers selected, remainder discarded	Majority voting by selected models in ensemble	Shuffles existing data with incoming samples to create new subsets	After a series of queries or too many failures, framework refreshes and retrains all basic models in the background

for random perturbations, and the values of n and p, affected the transferability of attacks. The use of transformations on training data to allow each student model to be retrained with a unique data subset was evaluated by comparing adversarial robustness between these student models and student models retrained on a common dataset. The comparison indicated that attack transferability was indeed reduced by this technique. However, attack transferability for highly transferable attacks such as FGSM remained high.

Remarks: This paper presents a very large number of results, some of which are difficult to interpret. For example, Fig. 3 in the paper indicates that, on CIFAR10, clean accuracy and C&W detection accuracy decline as p increases. No explanation is suggested for this. These experiments were run with p=0-4 and it is not clear how often they were run. The choice of lambda for Laplace distortion for CIFAR-10 also does not seem adequately explained.

The timing of model replacement may be vulnerable to attack, since it is based on a maximum number of queries. If the device comes under pressure, it is possible that n new models will not have been built at replacement time. This could be trivially handled by postponing the refresh so that if the new models are not ready, the old ones remain. However, it raises the question of whether the system is vulnerable to DOS attacks. If the system is overwhelmed with queries, new models might not be generated, the refresh could be postponed for a lengthy period and the MTD component might be effectively nullified. An optimal solution for this would be of interest since it is an issue that could arise with all Full MTD systems.

3.3.2 DeepMTD. DeepMTD uses MTD to improve detection and thwarting of adversarial attack for deep visual sensing in the context of road signs. The system is designed to accept a cropped image from a road-sign detector, and classify the road sign. The researchers train 9-level base CNNs on three well-known image datasets (MNIST, CIFAR-10 and GTSRB). They devise adversarial attacks showing that each model is vulnerable. They then assess whether having a group of models trained from the same data might be more effective. The idea is that immediately after the car or other device is shipped with a single model trained on the data, a group of n new models should be trained. New models should be retrained at regular intervals, and decisions on adversarial attack and the correct classification should be made by consensus. n=20 is used for examples and discussion throughout. Numerous possible variations are assessed. For example, training new models from scratch is more expensive than randomly perturbing weights and then retraining. Adversarial transferability is discussed but the authors state that the CNN models used have a lot of freedom and are not very subject to it. Results seem to bear this out. There is no retraining with adversarial examples. Due to the possible need to submit unclear results to a human for determination (presumably because

of restrictions around autonomous driving), two architectures are explored throughout. One allows for a human-in-the-middle, the other does not. For the same reason, detection of an adversarial example is considered a separate task from thwarting it, and statistics are given for both throughout. After considering attacks that leverage the base model, smart attackers who train against multiple models are envisaged, and the relevant results analysed.

The paper thoroughly explores the effect of operating-environment characteristics and evaluates DeepMTD on real world hardware configurations, providing data useful for IoT deployment.

Results: DeepMTD is not compared with other models, although consistency with other work is claimed. The base model's accuracy in training and validation on simple image dataset MNIST exceeds 99%. For the more complex images in CIFAR-10, validation accuracy is as low as 79.62%. Increasing the number of fork models n does not improve detection accuracy in the absence of attack. However, when under attack the defense rate improves as n increases, with a levelling out visible in DeepMTD's Fig. 13 as n approaches 20.

Remarks: This paper makes a praiseworthy attempt to establish that MTD can work on real-world hardware. However, generating new models for road sign recognition at n=20 takes approximately 45 minutes. Assessments of adversarial transferability, and the absence of adversarial data in the retrained models, suggest a need for further investigation of the resistance to adversarial attack. DeepMTD performs poorly in practical experiments by Rashid and Such [36], but it is unclear how often models are refreshed, if at all. Sardino, discussed next, is later work by the DeepMTD authors that tackles performance and adversarial robustness issues.

3.3.3 Sardino. Sardino builds on DeepMTD's road-sign classification work, with the same models, decision-making and attacker profile. The motivating difference is that Sardino uses a HyperNet / HyperGAN-like technique to regenerate models at runtime. This dramatically increases the model refresh speed and thus the possible refresh frequency. Meanwhile, Sardino attempts to retain clean accuracy, model diversity and adversarial robustness. It retains clean accuracy and model diversity by controlling cross-entropy loss and diversity loss. For adversarial robustness, rather than training with a specific adversarial attack type, Sardino employs adversarial learning, using a defender-attacker game between two models to train the model against adversarial attack. The attacker uses random numbers sampled from a normal distribution to generate perturbations which are added to the training data. The attacker and defender are jointly trained, each trying to defeat the other. The attacks generated by the attacker are non-deterministic; the idea is to harden Sardino against a range of attacks.

These techniques produce a base model, from which Sardino generates a 'HyperNet ensemble' of n models. Sardino refreshes the entire HyperNet ensemble at runtime between each image frame. A frame contains one or more traffic sign images. The weights of the models are variable, generated with each HyperNet refresh. The number of models in the ensemble (n) is also variable (max n= 100) and depends on desired processing throughput, the measured time taken by background activities, and the generation time per model.

Results: The authors use a test accuracy of 96.6% from a 2011 paper on GTSRB detection as a base accuracy rate [45] for classification accuracy evaluation. They show that single-model accuracies for Sardino models are between 94.6% and 96.9%, comparing reasonably well with the base. In a graph showing accuracy for increasing values of n (number of models), they show that when using the HyperNet ensemble with majority voting, accuracy passes the base at around n=50. At n=100, accuracy is 0.5% higher than the base. The HyperNet approach reduces the time to regenerate n models from approximately 45 minutes (DeepMTD) to the order of milliseconds (Sardino). This vastly improved refresh frequency provides increased adversarial robustness compared to DeepMTD. However, the accuracy of the refreshed models is not validated. In experiments, validating 100 models took 7.2 seconds, whereas unvalidated regeneration could be done in milliseconds.

Remarks: Per-task refresh is the logical endgame of an MTD strategy. Sardino takes a substantial step towards achieving this. In the real-time vehicle visual sensing system environment, swift per-frame regeneration and refresh were prioritised over model accuracy. The speed versus model accuracy compromise might not be necessary in every environment.

3.3.4 MTDroid. MTDroid [58] is designed to detect evasion attacks on Android ML anti-malware systems. Initially, a variety of model types (e.g. SVM, MLP) are trained. Each model type is trained on a different subset of the training data, to reduce the chance of attack transferability across model types. Then n 'N-variant' (functionally equivalent) sub-classifiers are trained from each base model. To provide adversarial inputs to the sub-classifiers, a mini-batch of data is randomly chosen from the clean data and 50% of the malware specimens in it are adversarially perturbed. The sub-classifiers are trained with the original clean data and these perturbed inputs.

Each of the m basic models in the pool has n variants incorporating adversarial perturbations. All mn models are candidates for inclusion in the classifier ensemble. Models are evaluated against the test set for effectiveness, i.e. the F1 score. For each model type, N-variants are ranked in descending F1 order. The highest-ranking are added to the final ensemble, which analyses input and classifies it based on majority voting. The ensemble contains at least one of each model type. After a series of queries or too many failures, the framework refreshes and retrains all models in the background, incorporating new input samples into the training data.

Results: Test results with n=1 to 10 show a rise in robustness but a fall-off in clean accuracy as n increases (Fig 3. [58]). MTDroid analysis suggests that it is better than SOTA for many characteristics. A scheduler is developed for retraining and replacing models.

Remarks: Some features seem unsuited to a production environment. For example, it is difficult to know how the number of failed samples would be evaluated at runtime; by definition, the system has failed to detect them. Furthermore, no strategy is given for labelling new input samples before adding them to training data. Finally, it is unclear whether the results of adversarial manipulations are assessed to ensure that they are valid functioning malware, and whether this would be done during the refresh phase.

3.4 MTD With IoT: EI-MTD

We identified only one study using MTD for adversarial robustness on IoT devices [34]. Qian et al.'s EI-MTD uses Naive MTD for edge intelligence adversarial robustness, and is tested on image processing. Development of the models begins by training a well-resourced

'teacher' model on a cloud data centre, including adversarial training for robustness. Then a number of smaller 'student' models are trained from the teacher using knowledge distillation. These are deployed to edge devices: one model per device. When an image requires classification, it is sent to a scheduling controller instead of being processed immediately on the device. The scheduler uses a Bayesian Stackelberg game to choose an optimal model (i.e. a different device) to send the image to for processing, ensuring that the adversary cannot tell which model will process the input.

The study attempts to address three main challenges in preventing adversaries from successfully pursuing black-box attacks on edge devices. The first, how to prevent successful targeting of specific models, is tackled by switching between models (MTD). The second, how to reduce attack transferability while retaining accuracy, is done by adding a differential regularization term during distillation to diversify student models. These two challenges are common to all anti-malware ML deployments, unlike the third: how to run on edge nodes with limited resources. This is addressed by training students with knowledge distillation to reduce the model size enough to run on an edge node while retaining accuracy.

Results: The study found that the teacher model after 15 training epochs reached approximately 83% clean top-5 accuracy (meaning the label was among the top 5 predicted classes) and approximately 74% adversarial top-5 accuracy. For the student models, while clean detection was 'a little lower' (from the graphs, less than 5% for top-1 in most cases) for the differential knowledge distillation than for normal training, there was an increase in accuracy of approximately 33% for top-1 and approximately 49% for top-5 on adversarial image detection. In addition, EI-MTD outperforms any static model for any adversarial attack rate, although there is a drop in performance for both EI-MTD and static models as the attack rate increases.

Remarks: By sending images back to a central controller for processing, EI-MTD loses the benefits of having models present on the edge. If the network overhead of sending images to be classified and retrieving results is incurred, it is difficult to see why the entire classifier does not reside on the cloud. Such a solution would still allow switching between models for MTD. It would remove the need to use knowledge distillation to create student models of reduced size, and could therefore result in a more robust solution.

4 Discussion

Our detailed analysis of the four SOTA papers using runtime refresh for MTD, and of Qian et al.'s implementation of Naive MTD on IoT, highlight several decision points which must be addressed by any Full MTD solution for IoT malware detection. We discuss these concerns below. Designers should consider the threat model for the planned deployment environment, as this will impact on which concerns should predominate. Deploying malware detection to IoT devices penalises scarce resources such as power consumption [17]. Adding MTD should cause as little extra overhead as possible. Key takeaways from the analysis are summarised in Table 3.

4.1 Datasets and Features Selection for IoT

The studies discussed use well-known image and malware datasets. IoT devices are heterogeneous. Outside widely-used IoT-adjacent platforms such as Android, labelled malware datasets are rare. Feature selection is also challenging. For example, the Android 'Drebin features' used for malware detection include permissions, which are not available on all IoT platforms. A successful malware detector for heterogeneous IoT would need to generalize malware functionality to a common representation, without losing sight of important differences between devices, operating systems and runtime environments. Verifying accuracy for such a detector requires labelled malware datasets for a range of different IoT platforms.

4.2 Adversarial Training (AT)

AT involves perturbing inputs to fool classifiers. Preserving images during perturbations is well understood. By contrast, preservation of malware functionality during perturbations is more difficult to achieve, but is key to avoid training classifiers with inaccurate labels. DeepMTD does not undertake AT, relying on limited transferability in its model architecture. Sardino, which uses the same architecture, introduces a game-like model of adversarial learning which is run on the base model. AT for IoT MTD must maximise adversarial robustness without adding excessive overhead to the refresh process, so doing AT once, on a base model, is preferred.

4.3 Discard policy

Approaches to selecting trained models for decision-making differ. Morphence trains all models to the required accuracy and runs them all on the sample at detection time. DeepMTD also trains all models to an acceptable accuracy. Therefore, neither have a discard policy. Sardino simply keeps all models, without verifying accuracy. MTDroid, the only malware detector, tests all new models against a test set and discards any which do not reach the required accuracy. An MTD implementation designed to run on heterogeneous IoT devices should base its discard policy on whether discarding or upskilling low-accuracy new models is most efficient.

4.4 Decision-Making Process

Morphence runs all models at detection time, and uses the classification from the model with the highest confidence. The intuition is that models trained on clean data will have the highest confidence for clean input, while adversarially-trained models will have the highest confidence for adversarial input. As discussed in section 3.3.1, results sometimes conflict with this intuition. DeepMTD and Sardino separate detection of adversarial input, and image classification. They detect adversarial input if model results vary randomly. For classification, they use majority voting. DeepMTD experiments with the optimal number of models to run. MTDroid runs all models in the ensemble, and uses majority voting for decision-making. Running all or multiple models has time and resource penalties. The decision-making strategy should be evaluated in the target environment.

4.5 Refresh Retraining

Most of the studies do not discuss the issue of concept drift, where training data becomes outdated or irrelevant over time, and use retraining for model updates that is identical to the original training. However, as pointed out in MTDroid, the MTD approach should lend itself to updating training data to tackle concept drift. MTDroid

Table 3: Key Takeaways for an IoT MTD Implementation

Aspect	Approach
IoT Datasets	Up-to-date labelled malware datasets are needed for a range of IoT platforms.
Adversarial training	AT must maximise robustness without excess refresh overhead. Base model AT training inherited by new models is preferred.
Discard policy	The discard policy must evaluate whether discarding or upskilling low-accuracy models is more efficient on the target device.
Decision-making	The decision-making strategy affects resource consumption. It must be carefully designed for the target environment.
Refresh retraining	A low-resource MTD approach that removes the need to retrain freshly-generated models is preferred.
Model updating	Model updating could come under pressure from DOS attacks. The design must protect IoT environments from this.
Architecture	Whether to centralise detection or keep it on the edge must be informed by deployment scenarios; both should be possible.
Resources	Several studies balance accuracy with efficiency. This choice should be configurable to align with variable device constraints.

attempts to supplement the training data with incoming samples to maintain high ongoing detection rates. While this may work in a test scenario where the true labels of incoming samples are known, it is unclear how it would translate to a live environment, where a misdiagnosis would not be flagged at runtime. DeepMTD uses perturbation and partial retraining. The extensive testing in the paper shows that while this saves time it provides lower detection accuracy than retraining from scratch. Sardino does not undertake any retraining, using the HyperNet approach to generate n new models at runtime. Ironically, this means a Sardino-like approach could be ideal for avoiding concept drift on IoT devices. Data for retraining, which is bulky and difficult to keep current, is not needed. Devices could be supplied with up-to-date base models when required.

4.6 Model Updating

Frequent model updates guard against model profiling, and are essential to the success of IoT MTD solutions. As described in section 3.3.1, the updating of Full MTD solutions could be under strain when a system is attacked. Without time to regenerate new models, the MTD component might be effectively disabled. Full MTD implementations should adopt appropriate mitigations for this issue, such as implementing model-by-model refresh. The problem could be mitigated by fast, Sardino-like refresh times.

4.7 Architecture

In IoT environments with resource-constrained devices, the question of whether to do detection on the edge or send binaries to a central, more-powerful processor is key. Qian et al. do both [34], sending items to a scheduler which then chooses an edge device to process them. This seems over-engineered. It would be more efficient to implement MTD at the scheduler location, process the file and send the result back to the edge device. Another approach could be to deploy less-powerful 'first-pass' models on the edge, subject to MTD, that could identify potential malware and send it to the cloud for further evaluation. Alternatively, a range of models requiring different resource levels could be developed, with strategies to limit any loss in detection accuracy in low-resource environments. Financial as well as resource cost should be considered [17].

4.8 Resources

Running ML models on resource-constrained IoT devices is challenging [49]. The DeepMTD paper [43] described the efficiency benefits of running models in serial mode with a configurable threshold for the number of models needed for a consensus decision. It also discussed in depth two types of implementation environment, and the effect of both high and low power modes. Such investigations are essential for each use-case, IoT profile and deployment environment. Execution time is high-priority in the real-time environment of vehicle visual sensing systems. For other edge devices, the priority could be accuracy, so that use of more models entailing greater detection time could be appropriate. A practical solution could allow the system to be configured to prioritise either speed or accuracy, depending on the local requirements.

5 Conclusion

In this work we explored potential considerations for using MTD to defend ML malware detectors on IoT devices. We are motivated by consideration of the threat model for IoT, where a field or commercially available device can often be secured by an adversary, allowing thorough profiling. We began by assessing existing literature on MTD for ML models, distinguishing between 'Naive' and 'Full' MTD. We undertook a detailed analysis of four studies that used state-of-the-art Full MTD in various domains, and a fifth that used Naive MTD in an IoT environment. We discussed key similarities and differences, highlighting key takeaways that can be used to protect comprehensive malware detection for IoT devices.

Acknowledgments

This publication has emanated from research conducted with the financial support of Research Ireland under Grant number 23/US/3939, and supported in part by the National Science Foundation under Award No. 2422241.

References

[1] George Abdel Messih, Tyler Cody, Peter Beling, and et al. 2024. RESONANT: Reinforcement Learning-based Moving Target Defense for Credit Card Fraud Detection. *Proc. of the 11th ACM Workshop on Adaptive and Autonomous Cyber Defense* (2024), 13–22.
[2] Tigist Abera, N Asokan, Lucas Davi, and et al. 2016. Things, trouble, trust: on building trust in IoT systems. In *Proc. of the 53rd Annual Design Automation Conf.*
[3] Abdullah Al-Dujaili, Alex Huang, Erik Hemberg, and Una-May O'Reilly. 2018. Adversarial deep learning for robust detection of binary encoded malware. In *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 76–82.
[4] Abderrahmen Amich and Birhanu Eshete. 2021. Morphence: Moving Target Defense Against Adversarial Examples. In *Proc. of the 37th Annual Comp. Secur. Applications Conf.* ACM, 61–75.
[5] Andy Baldrian and Joseph Hallett. 2024. RTFM: How Hard are IoT Platform Providers Making it for their Developers?. In *Proc. of the Sixth Workshop on CPS&IoT Security and Privacy*. ACM, 14–26.
[6] Nicholas Carlini and David Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symp. on Security and Privacy (SP)*. 39–57.

[7] Adam Caulfield, Norrathep Rattanavipanon, and Ivan De Oliveira Nunes. 2023. ACFA: Secure Runtime Auditing & Guaranteed Device Healing via Active Control Flow Attestation. In *32nd USENIX Secur. Symp.* 5827–5844.

[8] Jianbo Chen, Michael I Jordan, and Martin J Wainwright. 2020. Hopskipjumpattack: A query-efficient decision-based attack. In *2020 IEEE Symp. on Security and Privacy (SP)*. IEEE, 1277–1294.

[9] Pin-Yu Chen, Huan Zhang, Yash Sharma, and et al. 2017. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *ACM Workshop on Artificial Intelligence and Secur.*

[10] Xi Chen, Wei Huang, Ziwen Peng, and et al. 2024. Diversity supporting robustness: Enhancing adversarial robustness via differentiated ensemble predictions. *Computers & Security* 142 (2024), 103861.

[11] Jin-Hee Cho, Dilli P Sharma, Hooman Alavizadeh, and et al. 2020. Toward proactive, adaptive defense: A survey on moving target defense. *IEEE Commun. Surveys & Tutorials* 22, 1 (2020), 709–745.

[12] Jacob Clarysse, Julia Hörrmann, and Fanny Yang. 2023. Why adversarial training can hurt robust accuracy. In *The Eleventh Int. Conf. on Learning Representations.*

[13] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. 2019. Certified Adversarial Robustness via Randomized Smoothing. In *Proc. of the 36th Int. Conf. on Machine Learning (Proc. of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 1310–1320.

[14] Yufei Du, Zhuojia Shen, Komail Dharsee, and et al. 2022. Holistic Control-Flow protection on Real-Time embedded systems with kage. In *31st USENIX Secur. Symp.* 2281–2298.

[15] Bardia Esmaeili, Amin Azmoodeh, Ali Dehghantanha, and et al. 2023. A GNN-Based Adversarial Internet of Things Malware Detection Framework for Critical Infrastructure: Studying Gafgyt, Mirai, and Tsunami Campaigns. *IEEE Internet Things J.* (2023).

[16] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. *Int. Conf. on Learning Representations* (2015).

[17] Nikhil Krishna Gopalakrishna, Dharun Anandayuvaraj, Annan Detti, and et al. 2022. "If security is required": Engineering and Security Practices for Machine Learning-based IoT Devices. In *2022 IEEE/ACM 4th International Workshop on Software Engineering Research and Practices for the IoT (SERP4IoT).*

[18] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, and et al. 2016. Adversarial Perturbations Against Deep Neural Networks for Malware Classification. *arXiv* (2016).

[19] Ke He, Dan Dongseong Kim, and Muhammad Rizwan Asghar. 2025. MTD-AD: Moving Target Defense as Adversarial Defense. *IEEE Trans. on Dependable and Secure Computing* (2025), 1–14.

[20] Weijia He, Kevin Bryson, Ricardo Calderon, and et al. 2024. Can Allowlists Capture the Variability of Home IoT Device Network Behavior?. In *IEEE 9th European Symp. on Security and Privacy (EuroS&P)*. IEEE, 114–138.

[21] Manuel Huber, Stefan Hristozov, Simon Ott, and et al. 2020. The lazarus effect: Healing compromised devices in the internet of small things. In *Proc. of the 15th ACM Asia Conf. on Comput. and Commun. Secur.* 6–19.

[22] Felix Kreuk, Assi Barak, Shir Aviv-Reuven, and et al. 2018. Deceiving end-to-end deep learning malware detectors using adversarial examples. In *Workshop on Secur. in Machine Learning (co-located with NeurIPs).*

[23] Deqiang Li and Qianmu Li. 2020. Adversarial Deep Ensemble: Evasion Attacks And Defenses For Malware Detection. *IEEE Trans. on Information Forensics and Secur. (TIFS)* (2020).

[24] Hongda Li, Qiqing Huang, Fei Ding, and et al. 2022. Understanding and detecting remote infection on Linux-based IoT devices. In *Proc. of the 2022 ACM on Asia Conf. on Comp. and Commun. Secur.* 873–887.

[25] Xiang Ling, Zhiyu Wu, Bin Wang, and et al. 2024. A Wolf in Sheep's Clothing: Practical Black-box Adversarial Attacks for Evading Learning-based Windows Malware Detection in the Wild. In *USENIX Secur. Symp.*

[26] Deyin Liu, Lin Yuanbo Wu, Bo Li, and et al. 2024. Jacobian norm with Selective Input Gradient Regularization for interpretable adversarial defense. *Pattern Recognition* 145 (2024), 109902.

[27] Keane Lucas, Weiran Lin, Lujo Bauer, and et al. 2024. Training Robust ML-based Raw-Binary Malware Detectors in Hours, not Months. In *Proc. of the 2024 on ACM SIGSAC Conf. on Comp. and Commun. Secur.*

[28] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, and et al. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *Int. Conf. on Learning Representations.*

[29] Weina Niu, Yihang Wang, Xingyu Liu, and et al. 2023. GCDroid: Android malware detection based on graph compression with reachability relationship extraction for IoT devices. *IEEE Internet Things J.* 10, 13 (2023), 11343–11356.

[30] Hamed Okhravi, William W Streilein, and Kevin S Bauer. 2016. Moving target techniques: leveraging uncertainty for cyber defense. *Lincoln Laboratory Journal* 22, 1 (2016), 100–109.

[31] Matthew J Page, Joanne E McKenzie, Patrick M Bossuyt, and et al. 2021. The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. *BMJ* 372 (2021), n71.

[32] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, and et al. 2017. Practical black-box attacks against machine learning. In *ACM on Asia Conf. on Comp. and Commun. Secur.*

[33] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. In *2016 IEEE Symp. on Security and Privacy (SP)*. 582–597.

[34] Yaguan Qian, Yankai Guo, Qiqi Shao, Jiamin Wang, and et al. 2022. EI-MTD: Moving Target Defense for Edge Intelligence against Adversarial Attacks. *ACM Trans. Priv. Secur.* 25, 3, Article 23 (2022), 24 pages.

[35] Aqib Rashid and Jose Such. 2023. StratDef: Strategic defense against adversarial attacks in ML-based malware detection. *Computers & Security* 134 (2023), 103459.

[36] Aqib Rashid and Jose Such. 2025. Effectiveness of Moving Target Defenses for Adversarial Attacks in ML-Based Malware Detection. *IEEE Trans. on Dependable and Secure Computing* (2025), 1–16.

[37] Phillip Rieger, Marco Chilese, Reham Mohamed, and et al. 2023. ARGUS: Context-Based detection of stealthy IoT infiltration attacks. In *32nd USENIX Secur. Symp.* 4301–4318.

[38] Jonas Röckl, Nils Bernsdorf, and Tilo Müller. 2024. TeeFilter: High-Assurance Network Filtering Engine for High-End IoT and Edge Devices based on TEEs. In *Proc. of the 19th ACM Asia Conf. on Comp. and Commun. Secur.* 1568–1583.

[39] Ita Ryan, Luke Kurlandski, and Nate Mathews. 2025. Supplementary material for SERP4IoT 2026. https://figshare.com/s/c7f52254abc558426a72

[40] Sailik Sengupta, Tathagata Chakraborti, and Subbarao Kambhampati. 2019. MTDeep: Boosting the Security of Deep Neural Nets Against Adversarial Attacks with Moving Target Defense. In *Decision and Game Theory for Security*. Springer, Cham, 479–491.

[41] Saleh Soltan, Prateek Mittal, and H Vincent Poor. 2018. BlackIoT: IoT botnet of high wattage devices can disrupt the power grid. In *27th USENIX Secur. Symp.*

[42] Qun Song, Zhenyu Yan, Wenjie Luo, and Rui Tan. 2023. Sardino: Ultra-Fast Dynamic Ensemble for Secure Visual Sensing at Mobile Edge. In *Proc. of the 2022 Int. Conf. on Embedded Wireless Systems and Networks*. ACM, 24–35.

[43] Qun Song, Zhenyu Yan, and Rui Tan. 2022. DeepMTD: Moving Target Defense for Deep Visual Sensing against Adversarial Examples. *ACM Trans. on Sensor Networks* 18, 1 (2022), 1–32.

[44] Keshav Sood, Mohammad Reza Nosouhi, Neeraj Kumar, and et al. 2021. Accurate detection of IoT sensor behaviors in legitimate, faulty and compromised scenarios. *IEEE Trans. on Dependable and Secure Computing* 20, 1 (2021), 288–300.

[45] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and et al. 2011. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *The 2011 Int. Joint Conf. on Neural Networks*. 1453–1460.

[46] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, and et al. 2014. Intriguing properties of neural networks. In *Int. Conf. on Learning Representations (ICLR).*

[47] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, and et al. 2019. Robustness May Be at Odds with Accuracy. In *Int. Conf. on Learning Representations.*

[48] Teng-Fei Tu, Jia-Wei Qin, Hua Zhang, and et al. 2022. A comprehensive study of Mozi botnet. *Int. Journal of Intelligent Systems* (2022).

[49] Gias Uddin. 2021. Security and Machine Learning Adoption in IoT: A Preliminary Study of IoT Developer Discussions. In *2021 IEEE/ACM 3rd International Workshop on Software Engineering Research and Practices for the IoT (SERP4IoT)*. 36–43.

[50] Shuo Wang, Hongsheng Hu, Jiamin Chang, Benjamin Zi Hao Zhao, and Minhui Xue. 2024. LACMUS: Latent Concept Masking for General Robustness Enhancement of DNNs. In *2024 IEEE Symp. on Security and Privacy (SP)*. 2977–2995.

[51] Bryan C Ward, Steven R Gomez, Richard Skowyra, and et al. 2018. Survey of cyber moving targets second edition. *MIT Lincoln Laboratory Lexington United States, Tech. Rep* (2018).

[52] Eric Wong and Zico Kolter. 2018. Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope. In *Proc. of the 35th Int. Conf. on Machine Learning*, Vol. 80. PMLR, 5286–5295.

[53] Weilin Xu, David Evans, and Yanjun Qi. 2017. Feature squeezing: Detecting adversarial examples in deep neural networks. In *Network and Distributed Systems Symp. (NDSS).*

[54] Yunrui Yu, Xitong Gao, and Cheng-Zhong Xu. 2022. MORA: Improving Ensemble Robustness Evaluation with Model-Reweighing Attack. arXiv:2211.08008 [cs.LG] https://arxiv.org/abs/2211.08008

[55] Baoguo Yuan, Junfeng Wang, Peng Wu, and et al. 2021. IoT malware classification based on lightweight convolutional neural networks. *IEEE Internet Things J.* 9, 5 (2021), 3770–3783.

[56] Tao Zhang, Fanyu Kong, Dongshang Deng, and et al. 2025. Moving Target Defense Meets Artificial-Intelligence-Driven Network: A Comprehensive Survey. *IEEE Internet Things J.* 12, 10 (2025), 13384–13397.

[57] Xia Zhou, Yujie Bu, Meng Xu, and et al. 2024. uBOX: A Lightweight and Hardware-assisted Sandbox for Multicore Embedded Systems. *IEEE Trans. on Dependable and Secure Computing* (2024).

[58] Yuyang Zhou, Guang Cheng, Shui Yu, Zongyao Chen, and Yujia Hu. 2024. MTDroid: A Moving Target Defense-Based Android Malware Detector Against Evasion Attacks. *IEEE Trans. on Inf. Forensics and Secur.* 19 (2024), 6377–6392.