Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science

Master Thesis

Adversarial Machine Learning for Detecting Malicious Behavior in Network Security

Bc. Michal Najman

Supervised by
Mgr. Viliam Lisý, MSc., Ph.D.

Submitted in May, 2019

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Najman** Jméno: **Michal** Osobní číslo: **420162**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Umělá inteligence**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Adversarial Machine Learning for Detecting Malicious Behavior in Network Security

Název diplomové práce anglicky:

Adversarial Machine Learning for Detecting Malicious Behavior in Network Security

Pokyny pro vypracování:

Recent research has shown that existing machine learning models can be easily tricked to have high error rate using slight modification of inputs, called adversarial samples. The student will investigate this phenomenon in the domain of network security, propose, and evaluate principled countermeasures. He will:
1) review the related work on adversarial machine learning (AML);
2) identify the specific requirements of AML in network security applications;
3) formally define the problem of adversarial attacks to URL reputation systems;
4) design and implement a detector of attack to the reputation system;
5) propose a model of an attacker misusing the reputation system;
6) increase the robustness of the detector with respect to the attacker's model.

Seznam doporučené literatury:

Huang, L., Joseph, A.D., Nelson, B., Rubinstein, B.I. and Tygar, J.D., 2011, October. Adversarial machine learning. In Proceedings of the 4th ACM workshop on Security and artificial intelligence (pp. 43-58). ACM.
Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B. and Swami, A., 2016. Practical black-box attacks against deep learning systems using adversarial examples. arXiv preprint arXiv:1602.02697.
Barreno, M., Nelson, B., Sears, R., Joseph, A.D. and Tygar, J.D., 2006, March. Can machine learning be secure?. In Proceedings of the 2006 ACM Symposium on Information, computer and communications security (pp. 16-25). ACM.
Michael Brückner and Tobias Scheffer. 2011. Stackelberg games for adversarial prediction problems. In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '11). ACM, New York, NY, USA, 547-555.
Anish Athalye, Nicholas Carlini, David Wagner. 2018. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. ICML 2018

Jméno a pracoviště vedoucí(ho) diplomové práce:

Mgr. Viliam Lisý, MSc., Ph.D., centrum umělé inteligence FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **28.01.2019** Termín odevzdání diplomové práce: **24.05.2019**

Platnost zadání diplomové práce: **20.09.2020**

Mgr. Viliam Lisý, MSc., Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Abstract

Adversarial machine learning has two principal objectives: to design an attacker which is able to circumvent a detector; and to design a detector that is able to detect those attackers. We model the adversarial setting with game theory and propose that the solution of the game is in a Stackelberg equilibrium. To find the equilibrium, we start with the expected risk minimisation framework (ERM) and the game model from which we derive a bilevel optimisation task yielding an optimal detector. We then propose a learning algorithm that approximates a solution of this task. To support our theoretical findings, we solve a practical real-world problem of detecting attacks to a URL reputation service. A key part of our learning algorithm is the model of an attacker. We propose an attack algorithm to a URL reputation service that obfuscates the attacker's primary goal by generating covering activity with projected gradient descent and a fast gradient sign method. Using genuine data provided by Trend Micro Ltd., we show that an adversarial detector outperforms an anomaly detector at all false positive rates (1%, 0.1% a 0.01%) and successfully learns to detect unseen attacks carried out by our attacking algorithm.

Keywords: adversarial machine learning, game theory, machine learning, statistical learning, neural networks, network security

Abstrakt

Adversarialní strojové učení má v principu dva cíle: navrhnout útočníka, který je schopen obejít detektor; a detektor, který úspěšně detekuje dané útočníky. Tyto protichůdné motivy jsou v této práci modelovány pomocí teorie her a je předpokládáno, že řešení hry leží ve Stackelbergově rovnováze. Abychom tuto rovnováhu nalezli, ukážeme, že z minimalizace očekávaného rizika (ERM) a herního modelu lze odvodit dvouúrovňovou optimalizační úlohu, jejímž řešením je optimální detektor. Dále navrhneme účící algoritmus, který řeší tuto úlohu a jehož výstupem je aproximace (lokálně) optimálního detektoru. Prezentovanou teorii aplikujeme na realný problém útoků na reputační systém URL adres. Klíčovým prvkem námi navrhženého učícího algoritmu je model útočníka, proto navrhneme algoritmus útoků na reputační systému URL adres, který je schopen zamaskovat primární cíl útočníka generováním krycí aktivity. Útočící algoritmus je založen na projektovaném gradientním sestupu (PGD) a metodě znaménka gradientu (FGSM). Za použití legitimních dat od společnosti Trend Micro Ltd. ukazujeme, že námi navržený adversariální detektor překoná detektor anomálií na všech zkoumaných úrovních false positives (1%, 0.1% a 0.01%) a je úspěšně schopen detekovat nové útoky našeho útočícího algoritmu.

Klíčová slova: adversarialní strojové učení, teorie her, strojové učení, statistické učení, neuronové sítě, síťová bezpečnost

Declaration:

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, May 24, 2019

Michal Najman

Acknowledgements

I gratefully thank my supervisor Mgr. Viliam Lisý, MSc., Ph.D. for priceless advice and comments that dramatically improved this work. Also, I am thankful to him for helping me gain far deeper knowledge and understanding of game theory and algorithms associated with it. In addition, I thank my family for their endless support. Last but not least, I thank my beloved Bára who supported me both mentally and practically by reviewing this thesis.

Contents

Introduction

In recent years, computer science has seen an advent of powerful algorithms that are able to learn from examples. Even though the notion of learnable algorithms was recognised and studied in pioneering ages of the field already, its wide-range real-world applications were to be implemented only with the presence of big available data collections and vast memory and computational resources. Therefore, nowadays one meets the abundance of machine learning techniques used to solve various problems. The field spans from theoretical research to practical applications in areas such as medical diagnosis, financial predictions and, most importantly in case of this work, computer security.

Most of the applications follow a similar scenario: a problem is formalised following a standard machine learning paradigm; a vast data set is collected and a proper algorithm giving the best results is found forming a model of the problem. However, in some applications, once such a model is deployed to a complex real-world environment, one soon identifies the model performance's deteriorates due to the key aspects of the reality that have been omitted in the standard machine learning point of view.

An example is seen in computer vision. It was found that deep neural networks that reign competitions in image classification [1] are prone to so called adversarial images [2]. In particular, the state-of-the-art image classifiers based on deep neural networks score very well in terms of prediction accuracy when given genuine images. However, such a classifier can be fooled with an image that was purposely adjusted. To put it simply, what is seen as an unambiguous cat by a human observer can be confidently labelled as a dog by a classifier. For instance, this phenomenon challenges traffic sign classification used in autonomous vehicles because it has been shown that a few well-placed stickers are able to fool the classifier and make it misrecognise a yield sign for a main road sign [3].

To reflect such weakness, problems are reframed to a game-theoretic setting in which two autonomous rational players compete while following their mutually conflicting objectives [4–6]. The aforementioned example with images is, consequently, extended in the following way. One of the players acts as an image classifier and aims to maximise classification accuracy, whereas the other player, an attacker, perturbs the images to lower prediction confidence or, even better, to make the classifier misclassify the image.

Of course, the same is seen in computer security—the field defined by adversarial nature. Intruders desire to circumvent a detector by adjusting their attacks [7]; malware is developed by optimising an executable binary [8], and spams are improved statistically to avoid detection [9].

The aforementioned examples are instances of adversarial machine learning which is a field defined by two principal objectives: to design an attacker which is able to circumvent a classifier; and to design a classifier that is able to detect those attackers. In this work, we closely examine both aspects of adversarial machine learning and design

an attacker and a detector uniquely combining machine learning and game theory.

In contrast to classical statistical learning, an adversarial setting such as network security has three critical properties: firstly, only benign activity can be recorder; secondly, malicious activity responds to the presence of a detector and is optimised to meet the attacker's goal; and thirdly, a real-world detector is allowed to falsely misclassify only a limited portion of benign users.

To address those three properties, we start with the expected risk minimisation framework [10] and adjust it account for a strict false positive rate constraint. We then define a model of an attacker and a detector as two competing entities that play a Stackelberg game [11] and derive an optimisation task that builds upon statistical learning and game theory. Inspired by the state-of-the-art algorithms solving complex games [12, 13], we propose an algorithm that gives an approximate solution to the game optimisation task, that is the algorithm outputs an adversarial detector robust to potential attacks. A critical part of our approach is an attack algorithm which is used as an opponent in the detector's algorithm and the detector learns to detect its attacks. In contrast to standard classifiers, our adversarial detector is stochastic. This means that its output is a posterior class distribution rather than a most probable class as it is done with standard classifiers. The final label is then drawn from the detector's output.

We work with a real-world example to demonstrate our algorithms: a URL reputation service is usually used by anti-malware programs deployed at an end-user's device to warn the user that it is about to enter a malicious site. However, the reputation service gets misused by malicious actors who this way check wether a newly deployed malicious site of theirs has already been exposed. Using the proposed algorithms, we solve the task and design such a robust adversarial detector that is capable of recognising whether a user using this reputation service is benign or malicious solely based on URLs it queried the reputation service with and information in the corresponding HTTP requests. This is done based on real-world data provided by Trend Micro Ltd.

To support our claims, we empirically show that the same level of robustness, which is achieved by our detector, is not reached with an anomaly detector on the provided real-world data. In particular, at the false positive rates 1%, 0.1% and 0.01%, we show that the adversarial detector allows significantly lower portion of successful attacks. In addition, we show that our detector robustly detects attacks with more than 10 low-scored URLs per day. Last but not least, we present our detector labels a few samples in the provided benign dataset as malicious with high confidence. On closer inspection, we find that those users exhibit suspicious behaviour and are likely a genuine attacker or an infected computer.

Structure of Thesis: In Background (Sec. 1) and Related work (Sec. 2), we review related work on adversarial machine learning. In Problem Analysis (Sec. 3), we identify specific requirements of adversarial machine learning and formally propose a

solution to the problem of adversarial detection of malicious activity. In Game Definition (Sec. 4), we formally define a game in which a detector detects malicious users of a URL reputation system. Also, we propose two attack types: a good queries attack which performs straight-forward greedy attack and a gradient optimises attack's cost by composing obfuscation activity; and two detector types: an anomaly detector that omits the adversarial nature of the task and an adversarial detector that utilises it. In Experiments (Sec. 5), we empirically evaluate performance of proposed models and algorithms on real-world data provided by Trend Micro Ltd. In addition, we analyse the results and identify critical differences between proposed models.

1 Background

This section gives a brief introduction to the background of the thesis topic. We discuss risk minimisation, regularisation and define a Neyman-Pearson Task. Then we present important notions of game theory and neural networks that are essential to this work. A thorough study on related work and state-of-the-art solutions is given in the next section.

1.1 Risk Minimisation

A classifier $f \in \mathcal{F}$ is a mapping $f : \mathcal{X} \mapsto \mathcal{C}$ that determines which class $c \in \mathcal{C}$ a sample $x \in \mathcal{X}$ belongs to. For the purposes of this work, we only describe binary classification in the following pages, however, the task is, naturally, expandable to a general discrete set $\mathcal{C}$. In the classical risk theory, the classifier f is a subject to minimisation of expected risk $R(f)$ given a cost function $\ell : \mathcal{C} \times \mathcal{C} \mapsto \mathbb{R}$.

$$R(f) = \mathop{\mathbb{E}}_{x,c} \Big[\ell(f(x), c) \Big] \tag{1}$$

Formally, the Expected Risk Minimisation (ERM) is given by:

$$\min_{f \in \mathcal{F}} R(f) \tag{2}$$

Typically when working with binary classification, ℓ is considered a *1-0 loss* which assigns an equal cost of magnitude *1* for misclassifying objects and a zero cost for correct classification. The expected risk in this case accounts only for the rate of false positives and false negatives. If we employ *1-0 loss* into the expected risk, we arrive at the following form:

$$R(f) = \sum_{c \in \mathcal{C}} p(c) \int_{x : f(x) \neq c} p(x|c)\, dx \tag{3}$$

The integral can be considered a probability of classifying objects x to an incorrect class given a correct class c, ie. $f(x) \neq c$. Let us consider binary classification in which $\mathcal{C} = \{\mathsf{B}, \mathsf{M}\}$ where M stands for a positive class (M as a malicious class) and B for a negative class (B as a benign class). In the context of this work, the positive class refers to malicious activity, ie. activity that is desired to be uncovered, and the negative class covers benign, legitimate or normal behaviours. To conclude, the risk $R(f)$ can be rewritten as a mixture of two types of errors: the false positives rate and the false negatives rate.

$$R(f) = p(\mathsf{B}) \cdot \mathsf{FPR}(f) + p(\mathsf{M}) \cdot \mathsf{FNR}(f) \tag{4}$$

In practice, computing an expected risk often involves intractable integrals. Therefore, the risk is empirically estimated from observed samples. The empirical risk $\hat{R}(f)$ estimated from a set of training samples $T_m = \{(x_i, c_i)\}_{i=1}^m$ is defined as follows:

$$\hat{R}_{T_m}(f) = \frac{1}{m} \sum_{(x_i,c_i)\in T_m} L(f(x_i), c_i) \tag{5}$$

Vapnik [10] showed that with increasing m the empirical risk $\hat{R}_{T_m}(f)$ approaches $R(f)$.

1.2 Regularisation

When examining possible classifiers, we usually have a priori knowledge of certain classifier instances being more suitable than others. Hence, some classifiers f correspond to models that are more likely to be inadequate, and some are a priori preferred. The reasons may vary, but mostly one desires to decrease models complexity to avoid overfitting. To capture this knowledge, a regularisation term $\Omega : \mathcal{F} \mapsto \mathbb{R}$ penalising some classifiers f is often added to the risk.

1.3 Neyman-Pearson Task

The Neyman-Pearson Task [14] is a problem in which the priori class probabilities are unknown and thus only the false negative rate (FNR) is minimised while the false positive rate (FPR) is maintained lower than a given threshold.

$$\min_{f\in\mathcal{F}} \mathsf{FNR}(f) \qquad \text{s.t.} \quad \mathsf{FPR}(f) \leq \tau_0 \tag{6}$$

1.4 Game Theory

In the context of this work, let us consider a game of two players: a defender (denoted as -1) and an attacker (denoted as +1). A player $i \in \{\text{-}1, +1\}$ is associated with its action space $\mathcal{A}_i$. A player plays a pure strategy $a_i \in \mathcal{A}_i$ or a mixed strategy $\sigma_i \in \Delta(\mathcal{A}_i)$ which is a probability distribution over the player's actions space. Each player is expected to be a rational actor which carries out activity according to its risk. A player's risk is a function R_i that evaluates what risk is taken depending on players' strategies.

In a Stackelberg Game, the player -1 is a leader that commits to a strategy publicly while the player +1 is a follower who exploits the leader's public strategy. In other words, the follower (here an attacker) recognises the committed mixed strategy $\sigma_{\text{-}1}$ and selects a pure strategy a_{+1} that minimises its risk $R_{+1}(\sigma_{\text{-}1}, a_{+1}) = \mathbb{E}_{a_{\text{-}1}\sim\sigma_{\text{-}1}} R_{+1}(a_{\text{-}1}, a_{+1})$. An action a_{+1} that responds to $\sigma_{\text{-}1}$ minimises risk $R_{+1}(\sigma_{\text{-}1}, a_{+1})$ is called a best response.

$$\mathbf{BR}(\sigma_{\text{-}1}) = \operatorname*{argmin}_{a_{+1}\in\mathcal{A}_{+1}} R_{+1}(\sigma_{\text{-}1}, a_{+1}) \tag{7}$$

The defender is expected to be a rational actor too, therefore, it optimally crafts its mixed strategy. Knowing the attacker plays a best response $a^*_{+1} \in \mathbf{BR}(\sigma_{-1})$ it commits to play σ^*_{-1} that minimises its risk $R_{-1}(a^*_{+1}, \sigma_{-1}) = \mathbb{E}_{a_{-1}\sim\sigma_{-1}} R_{-1}(a^*_{+1}, a_{-1})$. The tuple $(\sigma^*_{-1}, a^*_{+1})$ is called a Stackelberg equilibrium (SE). Since the attacker can arbitrarily choose any $a^*_{+1} \in \mathbf{BR}(\sigma_{-1})$ (because all of them are optimal to the attacker), we define a strong Stackelberg equilibrium (SSE) in which the attacker breaks ties in favour of the defender, that is the attacker plays such $a^*_{+1} \in \mathbf{BR}(\sigma_{-1})$ which minimises the defender's risk $R_{-1}(a_{+1}, \sigma_{-1})$.

$$\min_{\sigma_{-1}, a_{+1}} R_{-1}(a_{+1}, \sigma_{-1}) \qquad \text{s.t.} \quad a_{+1} \in \mathbf{BR}(\sigma_{-1}) \tag{8}$$

1.5 Neural Networks

A neural network is a function approximator that consists of layered linear functions whose output is transformed with a non-linear activation. Literature proposes a great variety of neural networks architectures [1, 5, 6, 15–19]. In the context of this work, we define a feed forward neural network with fully-connected layers. A feed forward neural network takes an input x and transforms it with its layers one-by-one in a predefined sequence so that the output of the layer l is the input of the layer $l+1$. The last layer's output is the output of the neural network. We define a fully connected layer as:

$$y = f(W \cdot x + b) \tag{9}$$

where $x \in \mathbb{R}^N$ is the layer's input and $y \in \mathbb{R}^M$ is the layer's output. The matrix $W \in \mathbb{R}^{(M\times N)}$ and the vector $b \in \mathbb{R}^M$ are parameters of the layer. The function f is an activation of the layer.

Often, a rectified linear unit (ReLU) [15] is used as an activation.

$$f(x) = \max\{0, x\} \tag{10}$$

In this work, we use a scaled exponential linear unit (SeLU) [16] which is defined by the following function:

$$\mathsf{selu}(x) = \lambda \begin{cases} x & x > 0 \\ \alpha \cdot (e^x - 1) & \text{otherwise} \end{cases} \tag{11}$$

The authors of SeLU [16] propose the constants have the following values: $\alpha \approx 1.6733$ and $\lambda \approx 1.0507$.

In classification problems, a neural network $f(c, x)$ approximates the posteriori probability $p(c \,|\, x)$ where $c \in \mathcal{C}$ is a class and $x \in \mathcal{X}$ is an object which is to be labeled. To make a final decision about an input x, the class with highest probability is taken, i.e. $\operatorname{argmax}_{c\in\mathcal{C}} f(c, x)$. Usually, classification neural networks are trained by minimising

a cross-entropy loss. For a target distribution $\hat{y}$ and an estimated distribution y, the cross entropy is defined as follows:

$$l(\hat{y}, y) = \sum_{c \in \mathcal{C}} \hat{y}(c) \cdot \log(y(c)) \tag{12}$$

Since in classification problems each sample has one particular class assigned, the cross entropy loss changes. Given a sample x, its true class c and an estimator f, the loss is given by $log(f(c, x))$. With a mini-batch gradient descent, one can optimise the parameters of the neural network. Let θ be parameters of a neural network $f_\theta(c, x)$, then a gradient of the cross entropy loss can be estimated with m samples (called a mini batch) as follows:

$$\frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \log(f(c_i, x_i)) \tag{13}$$

The mini-batch gradient descent adjusts parameters of the neural net in each step t by drawing m samples (c_i, x_i) from the joint data distribution $p(c, x)$ and subtracting gradient of the loss scaled with a learning rate γ:

$$\theta^{t+1} = \theta^t - \gamma \cdot \frac{1}{m} \sum_{i}^{m} \nabla_\theta \log(f(c_i, x_i)) \tag{14}$$

2 Related Work

Examining adversarial aspects of various machine learning problems has currently been a popular topic. Mainly, this was triggered by Goodfellow et al. [2] who showed that neural networks are susceptible to adversarial examples. Since then many endeavours have been carried out to enhance neural networks or other machine learning algorithms by making them robust. Some tried to develop a provably robust classifier [20], while others reframed the classification problem to incorporate aspects of game theory [11]. Huang et al. [21] identify that there are several assumptions in the machine learning scheme that are often violated. For instance, they consider the data distribution is non-stationary. Barreno et al. [22] point out that some data may be generated by adversaries who play an instance of a deception game - that is they purposely adjust their actions to cover their true intention.

Despite most of the related work deals with image classification, efforts to utilise the same notions in computer security have been seen too [7, 8, 23]. Susceptibility to adversarial examples is, however, not the only weakness adversaries exploit, they also are able to modify future training datasets in their favour [24].

2.1 Adversarial Machine Learning

Lowd et al. [9] explore obfuscate strategies yielding spams that circumvent a spam filter. The authors consider attacks which are based on adding words to a spammy e-mail, while other modifications are not allowed. Three pools of words are defined: in the first attack random words from a dictionary are drawn; the second attack utilises common legitimate e-mail words; and in the third attack, words that are likely to appear in legitimate e-mails but are uncommon in spams are added.

To select the final set of words with the greatest effect from one of the three word pools, a black box threat model is used. In particular, the attacker repeatedly calls the detector to identify words, which make the detector label the spam as benign. As expected, the last pool of words mentioned outperforms the others. Moreover, this shows that additive changes to a malicious object are sufficient for obfuscating the detector (within this domain). The authors claim they are able to add words to spams in such a way the tested detection models do not detect 50% of them. We similarly design the good queries attack 4.4 in which we obfuscate user's activity by adding legitimate requests.

To reflect the successful attack algorithm, a defense strategy is proposed. It is shown that a robust detector which uncovers the adjusted spammy e-mails can be obtained by simply retraining the model on data now containing the attacks. However, the authors comment, a repeated obfuscation with a new set of effective words may again defeat the detector.

A similar notion is seen in more advanced classification models. For instance, deep neural networks are a popular class of classifiers nowadays for their performance in a

great range of fields. They were shown to outperform other methods in image classification (ImageNet Challenge [1]), natural language processing [19] and in many other fields. However, it was found that neural networks are susceptible to artificially crafted images. In particular, Goodfellow et al. [2] show an adversarial example may be labeled as an arbitrary class when accordingly adjusted. Moreover, despite the transformation of an input image is substantially bounded, for example by l_∞ norm, classifiers based on neural networks are prone to be circumvented anyway [3]. The susceptibility to adversarial samples follows the same observation in spam filtering – a good classifier is not necessarily robust to test time data manipulation.

As soon as it was recognised the neural networks contain built-in vulnerabilities which are exploitable, endeavours to improve the architecture were carried out. To address the weakness, some of the following work focus on a model definition and consider possible attacks already in the model design. This approach is summarised by Madry et al. [4] who study adversarial examples in image classification. The authors identify that expected risk minimisation (ERM) does not necessarily give models robust to adversarially crafted samples.

Their work extends the training framework based on ERM by a threat model in which each data point $x \in \mathbb{R}^N$ is assigned a set of perturbations $S(x) \subseteq \mathbb{R}^N$ that is available to the adversary. The authors work with $S_\epsilon(x)$ that contains perturbations bounded by l_∞, creating an ϵ-hyper-cube around each x:

$$S_\epsilon(x) = \{x' \in \mathbb{R}^N \mid l_\infty(x - x') \leq \epsilon\} \tag{15}$$

The norm l_∞ is used for simplicity and roughly represents human-undetectable image perturbations. Other approaches, however, consider more complex bounds that capture domain-specific constraints [25].

To fully relate to an adversarial setting, Madry et al. [4] propose that the adversary maximises the classifier's loss function L by modifying an image x to an adversarial example $x' \in S_\epsilon(x)$. This is further incorporated into the ERM framework, arriving at a saddle point problem:

$$\min_{f \in \mathcal{F}} \mathop{\mathbb{E}}_{x,c} \left[\max_{x' \in S_\epsilon(x)} L(f(x'), c) \right] \tag{16}$$

In other words, a solution to the problem gives an optimal robust classifier $f \in \mathcal{F}$ that is likely to classify all objects $x \in \mathbb{R}^N$ and their neighbourhood $S_\epsilon(x)$ correctly. We similarly compose a saddle point problem to solve adversarial problems, however, we assume the attacker's goal is general and its utility does not only correspond to classification accuracy.

The saddle point problem given above consists of two sub-problems: training the neural network and performing the inner maximisation. [4] approach the training part with Stochastic Gradient Descent (SGD) as it is commonly done in neural networks, while solving the inner maximisation task with Projected Gradient Descent (PGD) [17].

They conclude the ERM framework extended by this specific threat model gives a training method that is able to train neural networks in the adversarial setting and to produce classifiers robust to l_∞ bounded image perturbations. In addition, they find lower error is obtained with higher capacity models, suggesting that a robust model requires more parameters (eg. layers in neural networks).

To address the susceptibility to adversaries, several proposals of neural networks enhancements were submitted at ICLR 2018. However, seven out of nine were shown to be flawed due to following a similar ineffective scheme of masking the gradients [26].

In their paper, Athalye et al. [26] suggest there are three groups of gradient masking: first, a non-differentiable layer is inserted between the network layers; second, a classifier randomises its outputs; and third, a function transforms the input in such a way backward gradient explodes or vanishes. Showing that the submitted defensive methods follow the schemes, the authors succeeded in circumventing 7 of 9 proposed models. Concretely, they replaced or removed defensive non-differentiable components accordingly to estimate the gradient and crafted adversarial samples with PGD.

In our problem, the attacker's optimisation criterion is not differentiable because the search space is discrete and hierarchically composed. Inspired by this approach, we solve the problem similarly and parametrise non-differentiable elements of the criterion with an interpolating function. We then solve the attacker's optimisation with PGD.

In addition, Athalye et al. [26] suggest that randomisation of the classifier decision does not work for this only extends the iterations needed to acquire true gradient but does not increase robust aspects. We however use a stochastic detector whose decision is a realisation of a modelled posteriori probability – in other words, we too randomise the classifier's output. Importantly, we do so, because we model a mixed strategy which a detector as a player necessarily plays in an equilibrium. To support the claim, we show in Sec. 3.1.6 that a stochastic classifier is more general than a deterministic classifier and outperforms it.

A key aspect of adversarial machine learning is a definition to what extent the attacker knows private parameters of a defender. In a white box approach, the attacker has full access to a gradient, a structure or parameters of the classifier [11, 20, 22, 26]. In a black-box approach, the attacker gains access usually only to the output of the classifier and estimates other private setting from this output [27, 28]. In our game setting, we work with a white box thread model.

2.2 Provable robustness

Until now, all presented efforts to improve the neural networks susceptibility were approached empirically and usually without providing provable defenses [4]. A method that aims to give provable resistance to adversarial samples was proposed by Kotler et al. [20] who examine a novel network architecture that provably classifies all objects in a convex neighbourhood of a given image correctly. To achieve that, Kotler et al. [20] redefine a ReLU [15] in such a way it is not a function anymore but rather a set of

linear constrains yielding a convex polytope; i.e. a ReLU $y = \max\{0, x\}$ becomes:

$$y \geq x \tag{17}$$
$$y \geq 0 \tag{18}$$
$$y(l - u) \leq -ux + ul \tag{19}$$

where u and l are an upper, respectively lower bound of x. The bounds are unknown and need to be estimated for each ReLU.

With a convex relaxation of ReLU, image classification can be rewritten as a linear program with all components of the network now being linear. In the training process, the weights of the relaxed neural network are optimised so that the network correctly classifies not only the input image but also its convex embedding. More specifically, using a l_∞ norm a ϵ-neighbourhood of an input sample is embedded by a convex polytope and the network learns to disallow any adversarial samples in it.

Solving the optimisation problem in its LP form with a standard LP solver is not tractable due to a great number of variables needed to express state-of-the-art deep neural networks. However, the LP can be conveniently used to form an upper bound on robust classification accuracy. Now, this upper bound combined with the ReLU input bounds estimation becomes fully differentiable. The training process follows standard SGD [29] and gives a robust classifier that allows provably at most 6% error on MNIST [30]. In contrast, a classical neural architecture is vulnerable up to 80% error [20].

2.3 Optimising malware

In contrast to image classification, the space of inputs is usually discrete in computer security. An image can be represented as a vector in $[0, 1]^n$, while executable binaries span a very sparse subset of the binary space $\{0, 1\}^n$. Similarly, a set of executable source codes in a given programming language is a sparse subset of all character strings. Despite the theoretical difficulties several papers address the issue. [7] propose an obfuscate that optimises a malicious source code by applying some of the predefined modifications. The obfuscate method utilises the classifier's gradient to choose the most appropriate code modification. The set of plausible modifications is given beforehand and allows only additive changes. Although this significantly limits the attacker's action space, the authors claim reaching misclassification rates of up to 69%. [23] focus on static portable executables which they encode into a binary feature indicator vectors. Again, additive modifications are allowed only and malware is optimised with a bit gradient ascent. [8] take a different approach to malware optimisation and propose an agent which is trained with reinforcement learning. The agent is given a portable executable and its goal is to choose the most suitable modification of a piece of malware to lower the probability of detection.

2.4 Game-Theoretical Approach

As already shown, the problem of adversarial samples can be modelled as a game of two actors. However, Brückner et al. [11] propose a more general game model compared to those already mentioned. In particular, the authors define the players as a classifier and a data generator consisting of *all* actors generating data – that is the second player aggregately covers both benign and malicious actors.

This setting is explored using a game-theoretical point of view. The authors propose a Stackelberg prediction game in which a classifier, acting as a leader, and a data generator, acting as a follower, optimise their actions to meet their objectives. They argue the Stackelberg equilibrium is the most appropriate concept for trainable models, specifically compared to the Nash equilibrium. It is so, they claim, mainly because once a model is finalised and deployed, it is not changed anymore and thus the attacker can potentially learn all details of the model and adjust its actions to it.

In other words, the actions – the choice of model parameters and the test time data generation – are not carried out simultaneously, but instead the classifier commits to a specific parameters vector and the attacker utilises the information about the model and adjusts its attacking strategy accordingly. The later is modelled by a distribution shift at test time. The data generator transforms a probability of data p to a test time data probability $\dot{p}$ which maximises its objective function. In addition, the authors show that linear and kernel-based models together with suitable objective functions allow reformulating the problem to a quadratic program which yields the optimal model parameters.

We define our game very similarly to the Stackelberg prediction game. However, we, in contrast, consider the data generated by benign users are stationary; and malicious users only adjust their data in response to the detector. Also, we assume the classifier is complex and largely non-linear, which leads to problems that do not have analytical solutions. And finally, we consider the defender necessarily plays a mixed strategy in order to follow the Stackelberg's equilibrium.

Amin et al. [12] propose a gradient-based algorithm to solve a normal-form game by identifying a Stackelberg equilibrium. They assume that one player is a defender (playing a leader) and the second player is an attacker (a follower). They model a defender's mixed strategy with a parametrised distribution D_θ and update its parameters with a gradient descent. To estimate the expectation of the gradient, the authors use a Monte-Carlo method and sample random variables from corresponding distributions – for instance the action played by the defender is sampled from D_θ. With convenient parametrisation of D_θ, this approach provides an algorithm that solves a game in which the action spaces of both players are finite and discrete.

In this work, we follow this approach and similarly estimate the gradient of D_θ with a Monte-Carlo method. However, in contrast we deal with a game in which the defender's (or in our setting detector's) action space is infinite – it is in fact a family of possible classifiers and the mixed strategy is a probability distribution over all classifiers.

Lockhart et al. [13] propose a general-scheme algorithm that approximately solves an extensive-form game of two players where each is defined by its policy π_i. The algorithm consists of iterating over a two-step process of both players computing best responses to the policies from the previous step and updating the policies based on the now-generated best responses. They prove that if both players employ this algorithm and update their policies to minimise exploitability, the policies converge to a Nash equilibrium.

In spite of the fact that we aim to find a Stackelberg equilibrium, we take a similar approach and construct an algorithm that in each iteration consists of generating the attacker's best response and updating the defender's mixed strategy D_θ by minimising its expected risk.

2.5 Dataset poisoning

The Stackelberg Prediction Game assumes the model is fixed after deployment. In practice, however, engineers retrain the model on newly obtained data that might better represent their population. As this might be done periodically, the adversary shall take advantage of it and adjust its obfuscate strategy. Concretely, Rubinstein et al. [24] elaborate on poisoning anomaly detectors.

The poisoning obfuscate consists of purposely providing pre-crafted samples to the detector over a long period of time in belief, that the samples will create a blind spot in which all samples are considered benign by the detector. The authors assume that the input space is usually governed by a distribution of benign samples concentrated only in certain areas, leaving the rest for anomalous activity. Given a substantial amount of time, the adversary is gradually able to poison the detector by targeting the large empty parts of the input space and populating them with benign samples. In future retraining, the anomaly detector may mistakenly consider those re-populated areas a new phenomenon and label them benign. The attacker then simply crafts an obfuscate near to the poisoned areas of the input space.

The authors present that such an obfuscate is possible with an anomaly detector based on principal component analysis (PCA) which determines directions of the sample space with greatest variance. Replacing variance in PCA with median absolute deviation, which in contrary is a robust scale estimator, their model is robust to data set poisoning and successfully performs anomaly detection in backbone networks.

Despite dataset poisoning is an interesting problem, it is here mentioned to demonstrate other problems in adversarial machine learning that are not solved in this work.

3 Problem Analysis

In the present state of Internet, it is common for a site owner to run models classifying users or their behaviour. The task spans from user's interests specification to detecting deviating activity. Since such applications are becoming more popular, one may expect the users to modify their behaviours once they know they are being tracked and classified. Moreover, behaviour modification may very well be of rational nature, especially when a malicious user exploits loopholes or carries out lawless activity in order to pursuit its goal.

In other words, if there is a cost for being disclosed or seen as a certain category, the users will examine their actions to optimise for lower cost. As a result, machine learning models of any kind aiming to capture behaviours of those users necessarily need to have the adversary nature incorporated in their design.

In this work, we deal with a task of detecting malicious activity in network security. That is a problem that combines adversarial motivations of involved actors and principles of statistical learning to account for a distribution of observed activity. The goal of this section is to propose a theoretical background that gives an optimisation problem for finding a robust activity detector. In the following section, we then introduce an industrial real-world problem which is an instance of this task and propose a solution to it using the developed theory.

The straight-forward approach of solving the task of malicious activity detection would be to collect many examples of both kinds of user activity; that is to asses a dataset containing well-represented both malicious and benign users. This approach would follow the standard expected risk minimisation framework (ERM) and would give an activity classifier that minimises expected risk but omits the adversarial nature. However, one might arrive at difficulties during the construction of a balanced dataset for there is usually very few records of malicious activity, disproportionally less than the collection of normal, benign users. Also, and more importantly, the malicious actors modify their attack vectors once their method is exposed or they discover details concerning the detector.

Taking that into account, we consider the setting is a game of a classifier competing with a body of malicious users. This approach necessarily modifies the ERM framework and enhances it with game-theoretic notions.

Expected Risk Minimisation A malicious activity detection system is essentially a classifier that classifies users based on their behaviour. This in principle is a machine learning problem of finding a classifier $f \in \mathcal{F}$ minimising expectation of detection loss $\ell_{-1} : \mathcal{C} \times \mathcal{C} \mapsto \mathbb{R}$. The classifier is a mapping $f : \mathcal{X} \mapsto \mathcal{C}$ which takes vectors $x \in \mathcal{X}$ on its input and produces a decision $d \in \mathcal{C}$. However, the ground variable representing the object to be classified is a user's activity history $h \in \mathcal{H}$ which is translated to a corresponding feature vector by a feature map $\Phi : \mathcal{H} \mapsto \mathcal{X}$. All variables

and functions related strictly to a detector are subscripted with -1, whereas we use $+1$ in the attacker's case. This choice follows Brückner et al. [11]. ERM consists of identifying an optimal classifier f that minimises expectation of $\ell_{\text{-}1}$ over tuples of a feature vector and a class from $\mathcal{X} \times \mathcal{C}$. The expected risk can be formulated as a convex combination of risks conditioned on a class. Assuming there is two classes, i.e. $\mathcal{C} = \{\mathsf{B}, \mathsf{M}\}$, the expected risk can be rewritten as a combination of the risk attained on the malicious class and the risk attained on the benign class.

Definition 3.1. *Let the risk attained on a malicious class* M*,* $R_{\text{-}1}(f \,|\, \mathsf{M})$*, be the expectation of the loss conditioned on class* M*. Let the risk attained on a benign class* B*,* $R_{\text{-}1}(f \,|\, \mathsf{B})$*, be the expectation of the loss conditioned on class* B*.*

$$R_{\text{-}1}(f \,|\, \mathsf{M}) = \mathop{\mathbb{E}}_{x} \Big[\ell_{\text{-}1}(f(x), \mathsf{M}) \,|\, \mathsf{M} \Big]$$
$$R_{\text{-}1}(f \,|\, \mathsf{B}) = \mathop{\mathbb{E}}_{x} \Big[\ell_{\text{-}1}(f(x), \mathsf{M}) \,|\, \mathsf{B} \Big]$$

Definition 3.2 (Detector's Expected Risk Minimisation)**.** *In standard classification, the optimal classifier* f^* *is the solution of the following problem:*

$$\underset{f \in \mathcal{F}}{\textit{minimise}} \quad p(\mathsf{B}) \cdot R_{\text{-}1}(f \,|\, \mathsf{B}) + p(\mathsf{M}) \cdot R_{\text{-}1}(f \,|\, \mathsf{M})$$

3.1 Specifics of Adversarial Machine Learning

In this section, we examine adversarial machine learning in the domain of network security in general terms. The central task is to detect malicious users in the network without ideally affecting legitimate users.

The expectations in Def. 3.2 are usually estimated from a set of examples of each class. However, in the detection problem there are not enough examples of malicious behaviour and, in addition, this behaviour changes reflecting the current detector. This imposes two critical properties of adversarial machine learning:

- The priori class probabilities are not known.
- An individual attacker follows its private objective and (possibly rationally) chooses actions minimising its cost.

3.1.1 Property 1: Unknown Class Probabilities

To reflect the first property, we necessarily need to give up on ERM. We propose to redefine the detection problem to comply with the Neyman-Pearson Task (recall Sec. 1.3). The Neyman-Pearson task minimises the false negative rate (FNR) while keeping the false positive rate (FPR) below a certain level. This way the priori class probabilities are omitted. Inspired by this approach, we defined that the optimal detector minimises

a risk attained on malicious class $R_{\text{-}1}(f \mid \mathsf{M})$ and keeps the risk on benign class $R_{\text{-}1}(f \mid \mathsf{B})$ below a threshold. This reflects the desired strict constraint of real-world detection problems which is to identify an optimal detector but to avoid affecting benign users up to certain tolerance level. Conveniently, with a zero-one loss, the risk attained on malicious class $R_{\text{-}1}(f \mid \mathsf{M})$ becomes the false positive rate and similarly the benign class risk $R_{\text{-}1}(f \mid \mathsf{B})$ becomes the false negatives rate.

Definition 3.3 (Neyman-Pearson Detection Task)**.** *The Neyman-Pearson detection task minimises the expected loss conditioned on the malicious class while the expected loss conditioned on the benign class is maintained lower than a threshold* τ_0*.*

$$\begin{aligned} \underset{f \in \mathcal{F}}{\textit{minimise}} \quad & R_{\text{-}1}(f \mid \mathsf{M}) \\ \textit{subject to} \quad & R_{\text{-}1}(f \mid \mathsf{B}) \leq \tau_0 \end{aligned}$$

Using this formulation, prior class probabilities are omitted and, in addition, the task reflects the nature of security detection problems in which there is a hard constraint on false positives. In other words, we aim to find a classifier f that does not affect legitimate activity but given this constraint is the best detector of malicious activity.

3.1.2 Property 2: Adversarial Setting

By assuming an attacker is a rational actor that pursuits its goal, the setting of statistical learning changes to an adversarial game of two players: a detector and an attacker.

We sort sampled activity into two classes: benign (B) and malicious (M). The former is activity generated by legitimate users and the later is activity produced solely by an attacker in pursuit of its objectives.

To avoid detection, each individual attacker obfuscates its primary goal. However, if a good detector is deployed, obfuscation requires a large quantity of legitimate activity to make the final activity mask the primary goal entirely and avoid detection. The obfuscated activity of an attacker, in consequence, is recorder by the detector and stored as an activity history based on which the detector assigns a label to it.

3.1.3 Stackelberg Game

In practice, the detector is fixed after deployment and the choice of its particular form and parameters necessarily occurs before the deployment. This is a case of a Stackelberg game [11] in which the detector is a leader and the attacker is a follower. For theoretical properties (its existence for instance) we assume a strong Stackelberg equilibrium is played.

In a Stackelberg game, the follower plays a best response to the leader's public strategy and the leader optimises this strategy, accounting for the follower's best response. In such a setting, the leader optimally plays a mixed strategy while the follower plays

a pure strategy. This means the attacker obfuscates its activity optimally without a need of randomisation by playing a best response to detector's strategy.

As mentioned, the detector may necessarily randomise its actions to achieve the optimal cost. This translates to a detector playing a mixed strategy $\sigma(f) : \mathcal{F} \mapsto [0, 1]$ instead of a single particular f. To put it differently, the player detector posses a probability distribution over all possible classifiers $\sigma(f)$ and, according to it, randomises the choice of a particular classifier f and labels an input sample $x \in X$ as a class $d \in C$, $d = f(x)$. In conclusion, the detector decides a sample $x \in \mathcal{X}$ belongs to a class $d \in \mathcal{C}$ based on a probability $D_\sigma(d \,|\, x)$ that is constructed in accordance to $\sigma(f)$. The notion is captured in the following definition.

Definition 3.4. *A stochastic detector $D : \mathcal{X} \mapsto \mathcal{C}$ is a probability distribution $p(d \,|\, x)$ generating a decision $d \in \mathcal{C}$ conditioned on an observed sample $x \in \mathcal{X}$. D_σ is given by a detector playing a mixed strategy $\sigma : \mathcal{F} \mapsto [0, 1]$:*

$$D_\sigma(d|x) = \sum_{f:f(x)=d} \sigma(f)$$

3.1.4 Attacker

We model the attacker as a rational actor which plays the action minimising its expected costs. The particular form of costs and actions depends largely on the domain. Therefore, here we only present general notions defining the attacker and, later in Section 4.3, we propose the attacker's model that suits the running example of attacks to a URL reputation service.

According to [11], we propose all attackers follow the same objective and they differ only in their particular primary goal. That is, the activity obfuscation is practically the same task shared by all attackers and two attackers differ in what they aim to obfuscate. The final activity history of each of them is strictly a function of the primary goal.

For that reason, the model of the attacker considers a single-body aggregate player in which an individual attacker instance is thoroughly defined by its primary goal $g \in \mathcal{G}$. The common shared obfuscation function $\psi : \mathcal{G} \mapsto \mathcal{H}$ takes a primary goal g on its input and maps it to an activity history $h = \psi(g)$ that obfuscates the primary goal. Since we assume the attacker is required to meet its primary goal g, the outcomes of the obfuscation function are limited to contain activity related to meeting the primary goal, that is $\psi(g) \in S(g)$. The set $S(g)$ contains all activity histories that meets the primary goal g.

The attacker is a rational player which operates in a stochastic environment (goals have a prior distribution). This is captured by the risk of the attacker defined below.

Definition 3.5. *Let the primary goals $g \in \mathcal{G}$ be generated by a probability $p(g)$. Let Psi be a family of obfuscation functions $\psi : \mathcal{G} \mapsto \mathcal{H}$. Let $f \in \mathcal{F}$ be a classifier and*

$\Phi : \mathcal{H} \mapsto \mathcal{X}$ *a feature map. The attacker's risk* $R_{+1} : \Psi \times \mathcal{F} \mapsto \mathbb{R}$ *is given as the expectation of its loss* $\ell_{+1} : \mathcal{G} \times \Psi \times \mathcal{C} \mapsto \mathbb{R}$*. That is:*

$$R_{+1}(\psi, f) = \mathop{\mathbb{E}}_{g} \Big[\ell_{+1}(g, \psi, f(\Phi(\psi(g)))) \Big]$$

Relating to game theory, the obfuscation function ψ is an attacker's action and its best response to σ is given by minimising the attacker's expected risk $\mathbb{E}_{f\sim\sigma} R_{+1}(\psi, f)$.

Proposition 3.1 (Attacker's Best Response)**.** *The attacker's best response* $\boldsymbol{BR}(\sigma)$ *to a mixed strategy* σ *is a set of obfuscation functions* $\psi : \mathcal{G} \mapsto \mathcal{H}$ *that are the minimisers of the expectation of the attacker's loss* ℓ_{+1}*.*

$$\boldsymbol{BR}(\sigma) = \mathop{argmin}_{\psi} \mathop{\mathbb{E}}_{g,d} \Big[\ell_{+1}(g, \psi, d) \Big]$$

Proof.

$$\begin{aligned}
\mathbf{BR}(\sigma) &= \mathop{\mathrm{argmin}}_{\psi} \mathop{\mathbb{E}}_{f\sim\sigma} R_{+1}(\psi, f) && (20)\\
&= \mathop{\mathrm{argmin}}_{\psi} \mathop{\mathbb{E}}_{f,g} \Big[\ell_{+1}(g, \psi, f(\Phi(\psi(g)))) \Big] && (21)\\
&= \mathop{\mathrm{argmin}}_{\psi} \sum_{f} \sum_{g} \ell_{+1}(g, \psi, f \circ \Phi \circ \psi(g)) \cdot p(g) \cdot \sigma(f) && (22)\\
&= \mathop{\mathrm{argmin}}_{\psi} \sum_{g} \sum_{d} \sum_{f : f\circ\Phi\circ\psi(g) = d} \ell_{+1}(g, \psi, d) \cdot p(g) \cdot \sigma(f) && (23)\\
&= \mathop{\mathrm{argmin}}_{\psi} \sum_{g} \sum_{d} \ell_{+1}(g, \psi, d) \cdot p(g) \cdot \sum_{f : f\circ\Phi\circ\psi(g) = d} \sigma(f) && (24)\\
&= \mathop{\mathrm{argmin}}_{\psi} \sum_{g} \sum_{d} \ell_{+1}(g, \psi, d) \cdot p(g) \cdot D_{\sigma}(d \,|\, \Phi \circ \psi(g)) && (25)\\
&= \mathop{\mathrm{argmin}}_{\psi} \mathop{\mathbb{E}}_{g,d} \Big[\ell_{+1}(g, \psi, d) \Big] && (26)
\end{aligned}$$

□

3.1.5 Stochastic Detector

The detector's pure strategy consists of a particular detector f. However, as proposed above, its optimal strategy is generally mixed and the detector, therefore, randomises its final decision $d \in \mathcal{C}$.

A mixed strategy in case of the detector is a probability distribution $\sigma : \mathcal{F} \mapsto [0, 1]$ which assigns a probability to each particular detector f. The decision d representing the estimated class of a sample x is, consequently, a random variable whose probability distribution is the aggregate of probabilities $\sigma(f)$ for which $f(x) = d$. To capture that, we defined a decision distribution $D(d|x)$ in Def. 3.4.

In this work, we model $D(d|x)$ with a neural network which fruitfully allow us to bypass potentially infinite enumeration of detectors from $\mathcal{F}$. The detector's mixed strategy is, in conclusion, represented by the distribution $D_\theta(d|x)$ where θ is a parameters vector.

As proposed by Brückner et al. [11], the attacker's impact on the setting can be modelled by a distribution shift. However, in contrast to [11], in this work we assume only the malicious class activity is governed by adversarial objectives and benign activity is maintained unchanged irrespective of the detector's presence. Taking that into account, we define that the distribution of samples produced by attackers $p(x \mid \mathsf{M})$ is shifted in reaction to the presence of a deployed detector D and changes to $\dot{p}(x \mid \mathsf{M})$.

In a standard classification problem, we find f minimising the expected risk. In our adversarial setting, the detector is necessarily a distribution D that is a solution to the Neyman-Pearson Task with a non-stationary distribution of samples $\dot{p}(x \mid \mathsf{M})$.

Proposition 3.2. *Let the attacker play a best response $\boldsymbol{BR}(\sigma)$ to a mixed strategy σ, then the detector's risk of a mixed strategy σ attained on malicious activity, $R_{\text{-}1}(\sigma \mid \mathsf{M})$, is given by the best-case expectation of its loss attained on malicious activity.*

$$R_{\text{-}1}(\sigma \mid \mathsf{M}) = \mathop{\mathbb{E}}_{f}\Big[\, R_{\text{-}1}(f) \,\Big|\, \mathsf{M} \,\Big] = \min_{\psi \in \boldsymbol{BR}(\sigma)} \mathop{\mathbb{E}}_{q,d}\Big[\, \ell_{\text{-}1}(d, \mathsf{M}) \,\Big|\, \mathsf{M} \,\Big]$$

Similarly, the detector's risk of mixed strategy σ attained on benign activity, $R_{\text{-}1}(\sigma \mid \mathsf{B})$, is given by the expectation of its loss attained on benign activity.

$$R_{\text{-}1}(\sigma \mid \mathsf{B}) = \mathop{\mathbb{E}}_{f}\Big[\, R_{\text{-}1}(f) \,\Big|\, \mathsf{B} \,\Big] = \mathop{\mathbb{E}}_{h,d}\Big[\, \ell_{\text{-}1}(d, \mathsf{B}) \,\Big|\, \mathsf{B} \,\Big]$$

Proof. For the risk of a mixed strategy attained on malicious activity, it holds that:

$$\begin{aligned} R_{\text{-}1}(\sigma \mid \mathsf{M}) &= \mathop{\mathbb{E}}_{f}\Big[\, R_{\text{-}1}(f) \,\Big|\, \mathsf{M} \,\Big] && (27)\\ &= \mathop{\mathbb{E}}_{f,x}\Big[\, \ell_{\text{-}1}(f(x), \mathsf{M}) \,\Big|\, \mathsf{M} \,\Big] && (28)\\ &= \sum_{f}\sum_{x} \ell_{\text{-}1}(f(x), \mathsf{M}) \cdot \dot{p}(x \mid \mathsf{M}) \cdot \sigma(f) && (29) \end{aligned}$$

Consider a sample x is generated solely by the attacker (due to the M class in the conditional probability). We substitute x for $\Phi \circ \psi(g)$. Assuming a feature map $\Phi : \mathcal{H} \mapsto \mathcal{X}$ projects each h to one particular feature vector x and a malicious activity history h is given by a primary goal g obfuscated by a best response obfuscation function $\psi \in \mathbf{BR}(\sigma)$, the sum of probabilities $p(g)$ for which $\Phi \circ \psi(g) = x$ gives the non-stationary probability $\dot{p}(x \mid \mathsf{M})$.

$$\dot{p}(x \mid \mathsf{M}) = \sum_{h:\Phi(h)=x} \dot{p}(h \mid \mathsf{M}) \tag{30}$$

$$= \sum_{h:\Phi(h)=x} \sum_{g:\psi(g)=h} p(g) \tag{31}$$

$$= \sum_{g:\Phi\circ\psi(g)=x} p(g) \tag{32}$$

Using the substitution and considering the best-case, we arrive at:

$$R_{\text{-}1}(\sigma \mid \mathsf{M}) = \sum_f \sum_x \ell_{\text{-}1}(f(x), \mathsf{M}) \cdot \dot{p}(x \mid \mathsf{M}) \cdot \sigma(f) \tag{33}$$

$$= \min_{\psi \in \mathbf{BR}(\sigma)} \sum_f \sum_g \ell_{\text{-}1}(f \circ \Phi \circ \psi(g), \mathsf{M}) \cdot p(g) \cdot \sigma(f) \tag{34}$$

$$= \min_{\psi \in \mathbf{BR}(\sigma)} \sum_g \sum_d \ell_{\text{-}1}(d, \mathsf{M}) \cdot p(g) \cdot \sum_{f:f\circ\Phi\circ\psi(g)=d} \cdot \sigma(f) \tag{35}$$

$$= \min_{\psi \in \mathbf{BR}(\sigma)} \sum_g \sum_d \ell_{\text{-}1}(d, \mathsf{M}) \cdot p(g) \cdot D_\sigma(d \mid \Phi \circ \psi(g)) \tag{36}$$

$$= \min_{\psi \in \mathbf{BR}(\sigma)} \mathop{\mathbb{E}}_{q,d} \Big[\ell_{\text{-}1}(d, \mathsf{M}) \mid \mathsf{M} \Big] \tag{37}$$

Similarly for the risk of a mixed strategy attained on benign activity:

$$R_{\text{-}1}(\sigma \mid \mathsf{B}) = \mathop{\mathbb{E}}_f \Big[R_{\text{-}1}(f) \mid \mathsf{B} \Big] \tag{38}$$

$$= \mathop{\mathbb{E}}_{f,x} \Big[\ell_{\text{-}1}(f(x), \mathsf{B}) \mid \mathsf{B} \Big] \tag{39}$$

$$= \sum_f \sum_x \ell_{\text{-}1}(f(x), \mathsf{B}) \cdot p(x \mid \mathsf{B}) \sigma(f) \tag{40}$$

$$= \sum_f \sum_h \ell_{\text{-}1}(f \circ \Phi(h)), \mathsf{B}) \cdot p(h \mid \mathsf{B}) \cdot \sigma(f) \tag{41}$$

$$= \sum_h \sum_d \ell_{\text{-}1}(d, \mathsf{B}) \cdot p(h \mid \mathsf{B}) \cdot D_\sigma(d \mid \Phi(h)) \tag{42}$$

$$= \mathop{\mathbb{E}}_{h,d} \Big[\ell_{\text{-}1}(d, \mathsf{B}) \mid \mathsf{B} \Big] \tag{43}$$

□

Definition 3.6. *For simplicity, we interchangeably use σ and D_σ and D_θ as the detector's strategy. Thus:*

$$R_{\text{-}1}(D_\sigma \mid \cdot) = R_{\text{-}1}(\sigma \mid \cdot) = R_{\text{-}1}(D_\theta \mid \cdot)$$

Proposition 3.3 (Detector's Optimisation Problem)**.** *Let the detector minimise the expected risk attained on malicious activity, while maintaining the expected risk attained on benign activity upper-bounded by τ_0. Let the attacker minimise its expected risk. Then the stochastic detector D_θ parametrised by θ and the obfuscation function ψ which are the solution to the following bi-level optimisation problem are the Stackelberg equilibrium.*

$$\begin{aligned} \underset{\theta,\psi}{\textit{minimize}} \quad & \mathop{\mathbb{E}}_{q,d}\Big[\,\ell_{\text{-}1}(d,\mathsf{M}) \,\Big|\, \mathsf{M}\,\Big] \\ \textit{subject to} \quad & \mathop{\mathbb{E}}_{h,d}\Big[\,\ell_{\text{-}1}(d,\mathsf{B}) \,\Big|\, \mathsf{B}\,\Big] \leq \tau_0 \\ & \psi \in \underset{\psi'}{\textit{argmin}}\, \mathop{\mathbb{E}}_{g,d}\Big[\,\ell_{+1}(g,\psi',d)\,\Big] \end{aligned}$$

Proof. The proposition follows directly from the definitions and propositions above. □

3.1.6 On Stochasticity Importance

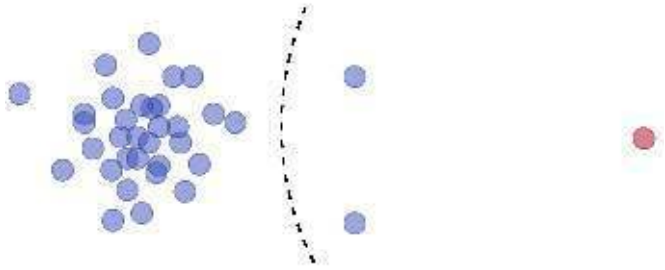
To understand the importance of stochasticity of a detector, let us present a toy example in which a deterministic detector has inevitably inferior performance to a stochastic detector. Consider an instance of a problem in Prop. 3.3 in which the false positive rate (FPR) constraint allows misclassifying only one benign sample. However, the distribution of the benign data has two outliers in the region where the attacker places its obfuscated sample. See visualisation in Fig. 1a which demonstrates this setting.

To achieve best performance, a deterministic detector shapes its decision line so that one of these outliers is well-classified while the other is misclassified. This solution meets the constraint. However, the attacker's the best response obfuscation function moves the malicious sample towards the one benign sample that is classified correctly. Had the detector chosen the other outlier to be well classified, the attacker would have adjusted and placed its attack to the now well classified benign sample. According to the Detector's optimisation problem, the optimal deterministic detector achieves zero detection rate. The final decision line is depicted in Fig. 1b.

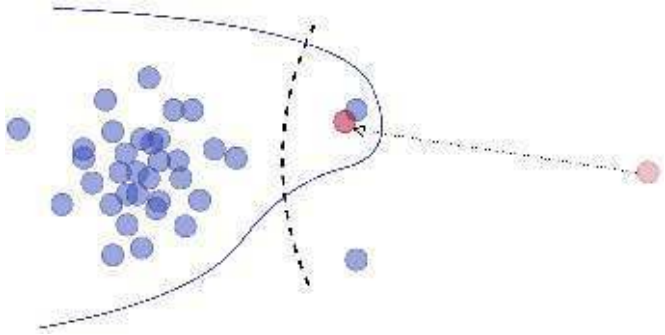
Now, consider a stochastic detector instead. To maximise detection rate while still meeting the FPR constraint, the detector adjusts the posteriori distribution in such a way that the two outliers are both covered with 50% probability of maliciousness. This meets the constraint on FPR and maximises detection rate since all possible obfuscations (in the region of possible obfuscations) are given also 50% detection probability. This is fully shown in Fig. 1c.

3.2 Assumption on Losses

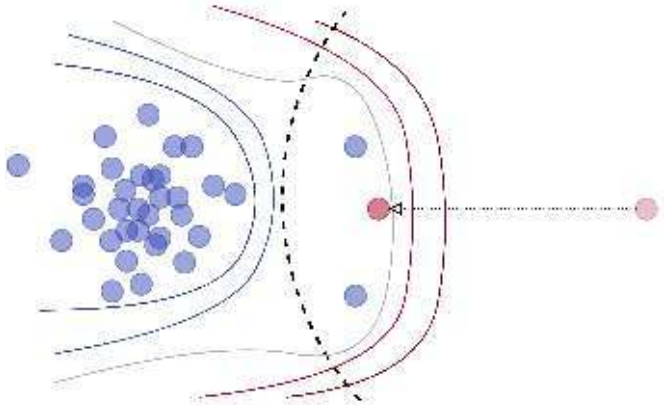
As it is common in ERM, we expect the detector's loss $\ell_{\text{-}1}$ is a zero-one loss. This simplifies the primary objective in the detector's optimisation problem.



(a) Problem setting



(b) Deterministic Detector



(c) Stochastic Detector

Figure 1: This figure shows a setting in which a deterministic detector underperforms a stochastic detector. Blue dots correspond to benign samples. Red dot is a malicious sample. Dashed semi-circle is bounds the region of possible obfuscation of the malicious sample. In Fig. 1b the blue line is a decision line of a deterministic detector. The arrow shows the obfuscation path from a primary sample to the obfuscated one. Fig. 1c depicts contours of a stochastic detector where blue-shaded lines outline areas of high benign-ness probability and red-shaded lines conversely high maliciousness probability. The shadow line is a 50% boundary. The stochastic detector gives more optimal solution to the detector's optimisation problem (Prop. 3.3)

Proposition 3.4. *Let the detector's loss ℓ_{-1} be a zero-one loss. Then the detector's risk attained on malicious activity $R_{-1}(\theta \,|\, \mathsf{M})$ is the expectation of the posteriori probability of a benign class conditioned on malicious activity. Similarly for the risk attained on benign activity $R_{-1}(\theta \,|\, \mathsf{M})$:*

$$R_{-1}(\theta \,|\, \mathsf{M}) = \min_{\psi \in \boldsymbol{BR}(\sigma)} \mathop{\mathbb{E}}_{g} \Big[D_\theta(\mathsf{B} \,|\, \Phi \circ \psi(g)) \,|\, \mathsf{M} \Big]$$
$$R_{-1}(\theta \,|\, \mathsf{B}) = \mathop{\mathbb{E}}_{h} \Big[D_\theta(\mathsf{M} \,|\, \Phi(h)) \,|\, \mathsf{B} \Big]$$

Proof. The proof is straight-forward.

$$R_{-1}(\theta \,|\, \mathsf{M}) = \min_{\psi \in \mathbf{BR}(\sigma)} \mathop{\mathbb{E}}_{q,d} \Big[\ell_{-1}(d, \mathsf{M}) \,|\, \mathsf{M} \Big] \tag{44}$$
$$= \min_{\psi \in \mathbf{BR}(\sigma)} \mathop{\mathbb{E}}_{q} \Big[\sum_{d} \ell_{-1}(d, \mathsf{M}) D_\theta(d \,|\, \Phi \circ \psi(q)) \,|\, \mathsf{M} \Big] \tag{45}$$
$$= \min_{\psi \in \mathbf{BR}(\sigma)} \mathop{\mathbb{E}}_{q} \Big[D_\theta(\mathsf{B} \,|\, \Phi \circ \psi(q)) \,|\, \mathsf{M} \Big] \tag{46}$$
$$R_{-1}(\theta \,|\, \mathsf{B}) = \mathop{\mathbb{E}}_{h,d} \Big[\ell_{-1}(d, \mathsf{B}) \,|\, \mathsf{B} \Big] \tag{47}$$
$$= \mathop{\mathbb{E}}_{h} \Big[\sum_{d} \ell_{-1}(d, \mathsf{B}) D_\theta(d \,|\, \Phi(h)) \,|\, \mathsf{B} \Big] \tag{48}$$
$$= \mathop{\mathbb{E}}_{h} \Big[D_\theta(\mathsf{M} \,|\, \Phi(h)) \,|\, \mathsf{B} \Big] \tag{49}$$

□

The posteriori probability of the stochastic detector $D_\theta(d \,|\, x)$ is explicitly modelled by neural network in this work. Thus we prefer the risk explicitly contains the term. However, this does not hold generally and in some cases it is more fruitful to estimate the risk as expectation of loss values (e.g. reinforcement learning).

The same trick which was used in case of the defender cannot by applied to the attacker. The attacker's loss $\ell_{+1} : \mathcal{G} \times \Psi \times \mathcal{C} \mapsto \mathbb{R}$ is more complex. Naturally, it consists of two components: a public and a private term. The public cost reflects the adversarial objective of escaping detection (e.g. detection probability). The private cost penalises the attacker for too costly obfuscation and is not necessarily adversarial to the detector's cost.

This also shows the game is a non-zero sum game as the private term in the attacker's loss does not have an adversarial equivalent in the detector's loss.

Following Brückner et al. [11], we defined the attacker as a shared body of attacker instances. However, if the attacker's loss is defined conveniently, the attacker's optimisation problem decomposes and it can be solved independently for each attacker's instance. The convenient form of the loss is shown Tab. 1.

d	$\ell_{+1}(g, \psi, d)$
B	$\Omega_{+1}(g, \psi(g))$
M	$\Omega_{+1}(g, \psi(g)) + L_0$

Table 1: Proposed Attacker's Loss

The motivation of this particular form of the loss is simple. If an attacker is detected it pays the amount L_0 for acquiring a new license or an account so that it is able to carry out further activity. However, the more complex activity histories it creates to obfuscate its primary goal, the more costly carrying out such activity is. This is represented by a private private cost $\Omega_{+1} : \mathcal{G} \times \mathcal{H} \mapsto \mathbb{R}$.

Recall that the obfuscation function ψ is constrained by the set of activity histories $S(g)$ such that $\psi(g) \in S(g)$, i.e. $\psi(g)$ can only create activity histories in $S(g)$. The set $S(g)$ defines activity histories that the attacker is able to construct from g.

Proposition 3.5. *Let the attacker's loss be defined by Tab. 1. Let the attacker's private cost be a function $\Omega_{+1} : \mathcal{G} \times \mathcal{H} \mapsto \mathbb{R}$. Then the attacker's best response problem of finding $\boldsymbol{BR}(\sigma)$ decomposes into identifying set of optimal obfuscations Ψ_g^* such that if $\psi^* \in \boldsymbol{BR}(\sigma)$ then $\psi^*(g) \in \Psi_g^*$. In other words, Ψ_g^* is the set of solutions to the following problem.*

$$\Psi_g^* = \underset{h \in S(g)}{argmin}\, L_0 \cdot D_\sigma(\mathsf{M} \,\big|\, \Phi(h)) + \Omega_{+1}(g, h)$$

Proof. Proposition 3.1 defines the attacker's best response problem in which the expectation is over variables g and d.

$$\psi^* \in \mathbf{BR}(\sigma) = \operatorname*{argmin}_{\psi} \mathop{\mathbb{E}}_{g,d} \Big[\ell_{+1}(g, \psi, d) \Big] \tag{50}$$

$$= \operatorname*{argmin}_{\psi} \mathop{\mathbb{E}}_{g} \mathop{\mathbb{E}}_{d} \Big[\ell_{+1}(g, \psi, d) \Big] \tag{51}$$

Let us substitute the loss ℓ_{+1} for its tabular form in Tab. 1. The inner expectation in Eq. (51) simplifies and becomes:

$$\mathop{\mathbb{E}}_{d} \Big[\ell_{+1}(g, \psi, d) \Big] = \Omega_{+1}(g, \psi(g)) \cdot D_\sigma(\mathsf{B} \,\big|\, \Phi \circ \psi(g)) + \tag{52}$$

$$+ (L_0 + \Omega_{+1}(g, \psi(g))) \cdot D_\sigma(\mathsf{M} \,\big|\, \Phi \circ \psi(g)) \tag{53}$$

$$= \Omega_{+1}(g, \psi(g)) \cdot (1 - D_\sigma(\mathsf{M} \,\big|\, \Phi \circ \psi(g))) + \tag{54}$$

$$+ (L_0 + \Omega_{+1}(g, \psi(g))) \cdot D_\sigma(\mathsf{M} \,\big|\, \Phi \circ \psi(g)) \tag{55}$$

$$= L_0 \cdot D_\sigma(\mathsf{M} \,\big|\, \Phi \circ \psi(g)) + \Omega_{+1}(g, \psi(g)) \tag{56}$$

This gives us a simplified of the best response problem:

$$\underset{\psi}{\operatorname{argmin}} \mathop{\mathbb{E}}_{g} \Big[L_0 \cdot D_\sigma(\mathsf{M} \,\big|\, \Phi \circ \psi(g)) + \Omega_{+1}(g, \psi(g)) \Big] \tag{57}$$

The best response problem now contains only the term $\psi(g)$ and, hence, the expectation can be decomposed. The criterion is minimised if we set $\psi(g)$ to h that minimises $L_0 \cdot D_\sigma(\mathsf{M} \,\big|\, \Phi(h)) + \Omega_{+1}(g, h)$. However, the obfuscation function $\psi(g)$ is constrained by $S(g)$. Taking $S(g)$ into account, we arrive at the following form.

$$\psi^*(g) \in \Psi_g^* = \underset{h \in S(g)}{\operatorname{argmin}} L_0 \cdot D_\sigma(\mathsf{M} \,\big|\, \Phi(h)) + \Omega_{+1}(g, h) \tag{58}$$

Ψ_g^* simply denotes the solution of the optimisation problem.

□

3.3 Approximate Best Response

The attacker's best response problem decomposes into separate optimisation problems for each primary goal g and the solution of each individual optimisation task is a set Ψ_g^*. The criterion of the task is, however, arbitrarily non-linear (especially in $D_\theta(d \,|\, x)$) and, therefore, finding the solution requires advanced approximative methods. For instance, take gradient descent that in principal identifies an approximation of a local minimum. Thus, in practice we are not able to identify Ψ_g^* or its elements accurately.

To address this challenge, we take inspiration in agent-based theory in which each player in a game is formulated as an agent with a policy π which represents its decision strategy. For instance, Lockhart et al. [13] or Amin et al. [12] use approximative methods to find a game's equilibrium while iteratively adjusting player's policies. Here, we deal with a Stackelberg game in which only one player optimises its strategy, the detector, and the other player plays a best response to it.

We propose to use an iterative attack algorithm π that takes a primary goal g as its input and outputs an activity history h^{obf} that obfuscates g.

Definition 3.7 (Attack Algorithm)**.** *Consider the separate attacker's problem in Prop. 3.5 to be a non-linear problem with a set of solutions Ψ_g^*. Then we define an attack algorithm π that takes a primary goal $g \in \mathcal{G}$ and a detector D_θ as its input and generates an activity history $h^{\mathsf{obf}} \in S(g)$ that approximates the element of Ψ_g^* which is favoured by the detector. In other words, π approximates the obfuscation function ψ^* which is in a strong Stackelberg equilibrium.*

Note that the attack algorithm $\pi(g, D_\theta)$ is an approximation of a best response obfuscation function $\psi^* \in \mathbf{BR}(D_\theta)$. This is particularly useful in the detector's optimisation problem because its bilevel form can be simplified with an attack algorithm.

Proposition 3.6. *Considering the losses in propositions 3.4 and 3.5 and an attack algorithm π as an approximation of the attacker's best response, then the detector's*

optimisation problem in Prop. 3.3 transforms to the following form, if π returns the best-case activity history in case of a tie.

$$\begin{aligned} \underset{\theta}{\textit{minimise}} \quad & \mathop{\mathbb{E}}_{g}\Big[\, D_\theta(\mathsf{B} \,\big|\, \Phi(\,\pi(g, D_\theta)\,)) \,\big|\, \mathsf{M} \,\Big] \\ \textit{subject to} \quad & \mathop{\mathbb{E}}_{h}\Big[\, D_\theta(\mathsf{M} \,\big|\, \Phi(h)) \,\big|\, \mathsf{B} \,\Big] \leq \tau_0 \end{aligned}$$

Notice that there is a practical problem with this formulation: in practice, $\pi(g)$ does not necessarily produce a best-case activity history. For example, if $\pi(g)$ is a gradient descent-based algorithm, it converges to a local minimum which is (1) not necessarily a global minimum and (2) is not necessarily the best-case activity history.

3.4 Detector's Learning Algorithm

In this section, we introduce an iterative training mechanism that finds a local minimum of the detector's approximative optimisation problem (Prop. 3.6).

The problem in Prop. 3.6 involves a constraint that we lift to the criterion using a Langrangian multiplicator. Inspired by Janisch et al. [31] and Suttle et al. [32], we transform the problem to a maxmin problem and find an approximate solution with gradient descent. The approach also builds on Lockhart et al. [13] who proposed the Exploitability Descent which iteratively adjusts players' policies based on a current best response.

Recall we defined the risks attained on malicious activity and benign activity respectively. With the definition of the attack algorithm (Def. 3.7) the risks have the following form.

$$R_{\text{-}1}(\theta \,\big|\, \mathsf{M}) = \mathop{\mathbb{E}}_{g}\Big[\, D_\theta(\mathsf{B} \,\big|\, \Phi(\,\pi(g, D_\theta)\,)) \,\big|\, \mathsf{M} \,\Big] \tag{59}$$

$$R_{\text{-}1}(\theta \,\big|\, \mathsf{B}) = \mathop{\mathbb{E}}_{h}\Big[\, D_\theta(\mathsf{M} \,\big|\, \Phi(h)) \,\big|\, \mathsf{B} \,\Big] \tag{60}$$

Lemma 3.7. *Let $L(\lambda, \theta) = R_{\text{-}1}(\theta \,\big|\, \mathsf{M}) + \lambda \cdot (R_{\text{-}1}(\theta \,\big|\, \mathsf{B}) - \tau_0)$ be the Langrangian of the task in Prop. 3.6. Then following statements hold true:*

- *The solutions of the task in Prop. 3.6 are also solutions of the task below with corresponding values of λ:*

$$\max_{\lambda \geq 0} \min_{\theta} L(\lambda, \theta)$$

- *The derivative of $L(\lambda, \theta)$ with respect to λ is:*

$$\nabla_\lambda L(\lambda, \theta) = R_{\text{-}1}(\theta \,\big|\, \mathsf{B}) - \tau_0$$

- *The derivative of $L(\lambda, \theta)$ with respect to θ is colinear with the following vector:*

$$\nabla_\theta L(\lambda, \theta) \propto p(\mathsf{M}) \cdot \nabla_\theta R_{\text{-}1}(\theta \,\big|\, \mathsf{M}) + p(\mathsf{B}) \cdot \nabla_\theta R_{\text{-}1}(\theta \,\big|\, \mathsf{B})$$

Proof. Using the Langrange multiplicators method, the Langrangian of the detector's optimisation problem (Prop. 3.6) becomes:

$$L(\lambda, \theta) = R_{\text{-}1}(\theta \,\big|\, \mathsf{M}) + \lambda \cdot (R_{\text{-}1}(\theta \,\big|\, \mathsf{B}) - \tau_0) \tag{61}$$

The solutions of the problem in Prop. 3.6 are also the solutions of the following problem:

$$\max_{\lambda \geq 0} \min_{\theta} L(\lambda, \theta) \tag{62}$$

The gradient of $L(\lambda, \theta)$ with respect to λ is straight-forward:

$$\nabla_\lambda L(\lambda, \theta) = R_{\text{-}1}(\theta \,\big|\, \mathsf{B}) - \tau_0 \tag{63}$$

However, notice that in terms of θ, the gradient unveils an interesting fact – that is, by taking the partial derivatives we assume that λ is fixed. Therefore, we essentially solve a minimisation problem with λ being a constant:

$$\min_{\theta} R_{\text{-}1}(\theta \,\big|\, \mathsf{M}) + \lambda \cdot (R_{\text{-}1}(\theta \,\big|\, \mathsf{B}) - \tau_0) \tag{64}$$

Since λ and τ_0 are constants, we can rearrange the problem (64) to the following form which is equivalent in terms of the optimal solution.

$$\min_{\theta} p(\mathsf{M}) \cdot R_{\text{-}1}(\theta \,\big|\, \mathsf{M}) + p(\mathsf{B}) \cdot R_{\text{-}1}(\theta \,\big|\, \mathsf{B}) \tag{65}$$

We dropped the constant term and defined $p(\mathsf{B}) = \frac{\lambda}{1+\lambda}$ and $p(\mathsf{M}) = \frac{1}{1+\lambda}$. Now, we take gradient of the criterion in (65) and arrive at:

$$\nabla_\theta L(\lambda, \theta) \propto p(\mathsf{M}) \cdot \nabla_\theta R_{\text{-}1}(\theta \,\big|\, \mathsf{M}) + p(\mathsf{B}) \cdot \nabla_\theta R_{\text{-}1}(\theta \,\big|\, \mathsf{B}) \tag{66}$$

Due to the rearrangement, the scale of $\nabla_\theta L(\lambda, \theta)$ differs from the criterion gradient. However, the vectors' directions are alike. □

3.4.1 Monte-Carlo Estimates of Gradient

The gradient of the risk attained on benign activity $\nabla_\theta R_{\text{-}1}(\theta \,\big|\, \mathsf{B})$ is the expectation of gradient of the posteriori probability $D_\theta(\mathsf{M} \,\big|\, x)$.

$$\nabla_\theta R_{\text{-}1}(\theta \,\big|\, \mathsf{B}) = \mathop{\mathbb{E}}_{h} \Big[\nabla_\theta D_\theta(\mathsf{M} \,\big|\, \Phi(h)) \,\big|\, \mathsf{B} \Big] \tag{67}$$

And the gradient of the risk attained on malicious activity $\nabla_\theta R_{-1}(\theta \,|\, \mathsf{M})$ is the expectation of the gradient of $D_\theta(\mathsf{B} \,|\, \pi(g, D_\theta))$.

$$\nabla_\theta R_{-1}(\theta \,|\, \mathsf{M}) = \mathop{\mathbb{E}}_{g}\Big[\, \nabla_\theta D_\theta(\mathsf{B} \,|\, \Phi(\pi(g, D_\theta))) \,|\, \mathsf{M} \Big] \tag{68}$$

Notice that equations (67) and (68) contain expectations over spaces $\mathcal{H}$ and $\mathcal{G}$. Since $\mathcal{H}$ and $\mathcal{G}$ are generally discrete and infinite, we estimate the expectations with a Monte-Carlo method. The approach of estimating gradient with a Monte-Carlo method is inspired by Amin et al. [12] who estimate policy gradient in similar manners.

This means we draw m random samples $\{h_i\}$ from the distribution $p(h \,|\, \mathsf{B})$ and m random samples $\{g_i\}$ from the distribution $p(g)$. Once we have m independent realisations of an activity history and a primary goal, we compute averages of terms in corresponding expectations which gives us an unbiased estimate of the expectations.

Averaging over $\{h_i\}$ gives the estimate of $\nabla_\lambda L(\lambda, \theta)$.

$$\nabla_\lambda L(\lambda, \theta) \approx \frac{1}{m}\sum_{i=1}^{m} D_\theta(\mathsf{M} \,|\, \Phi(h_i)) - \tau_0 \tag{69}$$

To get $\nabla_\theta L(\lambda, \theta)$ estimates, we define γ_B and γ_M as the estimates of the gradient of the risk attained on benign activity, and malicious activity respectively.

$$\gamma_\mathsf{B} = \frac{1}{m}\sum_{i=1}^{m} \nabla_\theta D_\theta(\mathsf{M} \,|\, \Phi(h_i)) \tag{70}$$

$$\gamma_\mathsf{M} = \frac{1}{m}\sum_{i=1}^{m} \nabla_\theta D_\theta(\mathsf{M} \,|\, \Phi(\pi(g_i, D_\theta))) \tag{71}$$

Having the gradient estimates γ_M and γ_B, the gradient of $L(\lambda, \theta)$ becomes:

$$\nabla_\theta L(\lambda, \theta) \sim p(\mathsf{M}) \cdot \gamma_\mathsf{M} + p(\mathsf{B}) \cdot \gamma_\mathsf{B} \tag{72}$$

Note the convenient property: the scale of $\nabla_\theta L(\lambda, \theta)$ is independent on λ which instead only trims the convex combination of estimates γ_M and γ_B. This significantly helps the learning algorithm's performance.

3.4.2 Learning Algorithm

To find a local minimum of the Langrangian $L(\lambda, \theta)$ we perform gradient descent for λ and, conversely, gradient ascent for θ. This is done iteratively in the scheme of the Exploitability Descent algorithm in [13].

In each iteration of their algorithm, Lockhart et al. [13] generate a best response of the adversary and then update the player's policy according to the best response. In our algorithm, we first sample the realisations $\{h_i\}$ and $\{g_i\}$. Then, using the algorithm π, we compute the obfuscated activity histories $\{h_i^{\mathsf{obf}}\}$. This step corresponds to finding

the optimal best-response function ψ^* and applying it to each primary goal g_i. Finally, we update λ and θ according to the aforementioned equations.

Algorithm 1: Detector's Learning Algorithm

Input: α_λ, α_θ, D_θ, π

Initialise $\theta^{(0)}$;
Initialise $\lambda^{(0)}$;
for $t = 1, 2, \dots T$ **do**
  draw m samples $\{h_i\}$ from $p(h \mid \mathsf{B})$;
  draw m samples $\{g_i\}$ from $p(g)$;
  compute $\{h_i^{\mathsf{obf}}\}$ from $\{g_i\}$ with $\pi(g, D_{\theta^{(t-1)}})$;
  $\lambda^{(t)} \leftarrow \lambda^{(t-1)} + \frac{\alpha_\lambda}{m} \sum_{i=1}^{m} D_\theta(\mathsf{M} \mid \Phi(h_i)) - \tau_0$;
  compute γ_{B} from $\{h_i\}$;
  compute γ_{M} from $\{h_i^{\mathsf{obf}}\}$;
  $p(\mathsf{B}) = \frac{\lambda^{(t-1)}}{1+\lambda^{(t-1)}}$;
  $p(\mathsf{M}) = \frac{1}{1+\lambda^{(t-1)}}$;
  $\theta^{(t)} \leftarrow \theta^{(t-1)} - \alpha_\theta(p(\mathsf{M}) \cdot \gamma_{\mathsf{M}} + p(\mathsf{B}) \cdot \gamma_{\mathsf{B}})$;
end
return D_θ

As mentioned, π in practice does not converge to the optimal best-case obfuscated activity history h^{obf} but may get stuck in a local minimum. In light of Alg. 1, this may not be a problem. As long as π is relatively stable and outputs reasonable activity histories, it can be considered as a player with non-optimal, yet consistent strategy which the detector learns to defend against. The better the approximation and convergence properties of π are, the more adequate the learning process of the detector becomes.

In contrast to benign activity, the malicious estimate γ_{M} in (71) encounters two problems. First, the distribution $p(g)$ is unknown and, second, $\nabla_\theta D_\theta(\mathsf{M} \mid \Phi(\pi(g_i, D_\theta))$ takes the gradient of a term that involves an algorithm.

The absence of $p(g)$ is critical. Nonetheless, it can be faithfully crafted having the knowledge of a particular domain. For example, in this work, we propose a set of reasonable primary goals and uniformly draw from them.

Taking the gradient of $D_\theta(\mathsf{M} \mid \Phi(\pi(g_i, D_\theta))$ with respect to θ is problematic because π is also dependent on θ. In [13], they simply ignore this dependence and consider only D_θ is a function of θ. Even though we theoretically and empirically worked with a novel way of differentiating the output of the detector using the implicit function theorem, our method suffered from convergence issues and it yet requires further research. In experiments, therefore, we adopt the approach from [13] and consider the output of π is not differentiable.

3.4.3 Information Available to Opponents

We assume the game is played in a white box setting from the attacker's point of view whereas the detector has now information about the attack algorithm π. That means the attacker has full knowledge of the detector's structure and parameters and it has access to gradients of the detector and the detector has only access to the final obfuscated activities and has no knowledge of the way the malicious samples are identified. For instance, if a gradient-based attacker is used, a detector, with the full knowledge of the attacker's algorithm, may purposely adjust gradient in areas where the attacker performs first steps of gradient descent and, in result, learn to defend attacks from such an attacker type. However, this is essentially a way of overfitting and our goal is to attain a detector that solves the detector's optimisation task in which the opponent is rational and able to obfuscate its primary goals optimally (in terms of a Stackelberg Game).

To conclude, we deal with a detection task in network security in which we aim to robustly detect malicious activity without limiting benign activity. An optimal detector for this problem is a stochastic binary classifier that can be approximated by solving the optimisation task in Prop. 3.6. To do so, we propose an algorithm 1 which in T iterations gives an approximation of an optimal detector.

3.5 Anomaly Detection

Note the problem in Prop. 3.6 requires a model of an attacker. We proposed such a model, however, we needed to come up with several assumptions concerning the attacker's motivation.

Let us now show a different approach which is related to unsupervised anomaly detection. We now assume no information about the malicious class is known. Thus we aim to identify a detector for which the risk attained on benign activity is exactly equal to a threshold τ_0.

Definition 3.8 (Anomaly Detection)**.** *Let the optimal anomaly detector be a distribution $D_\theta(d|x)$, solely parametrised by a vector θ, if it is a solution to the following problem:*

$$\begin{aligned} &\textit{find} && \theta \\ &\textit{such that} && \mathop{\mathbb{E}}_{h}\Big[\, D_\theta(\mathsf{M} \,\big|\, \Phi(h)) \,\big|\, \mathsf{B} \,\Big] = \tau_0 \end{aligned}$$

This formulation of the original problem is usually beneficial in that it requires fewer amounts of theoretical assumptions and the implementation is more straightforward. However, the lack of assumptions on the attacker's motivations causes there is no guarantee on the final shape of the detector. Performance of a particular anomaly detector instance is a term of sole empirical tests. We examine performance of a k nearest neighbours anomaly detector later in Experiments.

4 Game Definition

In the previous section, we proposed a malicious activity detection problem can be modelled as a game of two players: a detector and an adversary. The goal of the detector is to identify the best activity classifier, while the adversary seeks to optimally modify activity of malicious users in such a way they get misclassified by the classifier.

Below, we show such formulation can solve a real-world problem. We take a URL reputation service as a running example and formalise it and propose an algorithm that approximates the optimal solution.

In this work, we consider a network security company that runs a reputation service which returns rating of a queried URL. For example, if we query the service's API with www.google.com, the URL is rated with high score whereas the malicious URL www.malicious-url.com is rated poorly. This type of a service is usually deployed by network security companies to provide their security software with access to most up-to-date database of URL ratings.

The typical usage scenario is coined as follows. A client running on an end-user's device encounters the user is about to enter a website. To evaluate the danger level of the website, the client queries the API of the reputation service with the website URL. Accordingly, the client may show a warning message notifying the user of expected danger or carry out an appropriate action.

Usually, URL rating systems aim to identify various URL danger types. Here, we focus on one particular type of malicious misuse: malware producers that asses a set of URLs which are used as communication entry-points for deployed malware units. With one of these URLs, a unit of deployed malware is able to receive commands and adjust its actions. However, to maintain consistency and availability of its malware units, the malware producer must regularly check whether any of its URLs has been exposed – by querying the publicly available URL rating system.

We assume users access the service's API identified by a license and query it with HTTP requests. For the sake of simplicity, each request contains one URL whose reputation score is queried. The key element of an activity history, therefore, is the set of URLs the user has queried the service with. In particular, we record user's activity in a one-day time window. This means an activity history is a discrete object which contains different number of URLs for each user and captures a 24-hour activity record.

To conclude, the task is as follows: the computer security company desires to distinguish malicious users of the URL rating service from benign ones based on the URLs each user queries the service with.

4.1 Formal Definition

In this section, we formally define the running example of this work which is an attack to a reputation system.

The service is queried with a URL $u \in \mathbb{U}$ where $\mathbb{U}$ is a set of all URLs. The query

is a typical HTTP request with its attributes and the URL is the subject of the query. The service securely assigns each query to a user based on a license the user uses. Thus, we define an activity history $h \in \mathcal{H}$ as a collection of queries of the user. For example, if a user sends a sequence of queries for which we record a queried URL, an arrival timestamp, a source IP or possibly other information, this is recorded and integrally stored in a corresponding user's activity history h.

$$(u_1, t_1, \mathsf{IP}_1, \dots), (u_2, t_2, \mathsf{IP}_2, \dots), \dots, (u_k, t_k, \mathsf{IP}_k, \dots) \longrightarrow h \tag{73}$$

Recall that a user's activity history h represents the ground object based on which the detector classifies users. Note that the inner structure of h is discrete. This is problematic for attackers as there is no direct way of computing gradients with respect to h or its elements.

In the previous section, we defined a malicious user posses a primary goal that thoroughly defines its individual instance. In this example, a single malicious user is defined by a private set of primary URLs $U^{\mathsf{pr}} \subset \mathbb{U}$. The primary url set contains URLs which the malicious user necessarily employs to achieve its primary goal – that is to obtain the current reputation rating for each URL in U^{pr}. In consequence, the primary goal $g \in \mathcal{G}$ is defined entirely by the primary URLs.

$$g = U^{\mathsf{pr}} \tag{74}$$

Given its primary URLs, a malicious user queries the service with URLs U that may next to its primary URLs also contain legitimate queries which it uses to obfuscate its activity.

$$U^{\mathsf{pr}} \subseteq U \tag{75}$$

Recall we assumed the attacker is a rational player thus the particular content of U changes depending on the classifier. If there was no classifier and, therefore, the attacker was not motivated to adjust its behaviour, it would presumably query the service with U resembling primary URLs and perhaps containing just a little overhead, ie. $U \cong U^{\mathsf{pr}}$. No detector also means there is no need to strategies with the values of other HTTP request properties.

Nonetheless, once there actually is a classifier deployed, implying a cost for disclosure, the attacker rationally queries the service with additional legitimate URLs to obfuscate its primary goal. There are essentially to types of primary URLs U^{pr} obfuscation: adding legitimate queries and adjusting properties of each request. Note that the attacker is required to include all $u \in U^{\mathsf{pr}}$ which simplifies the problem. The allowed obfuscation methods limit the activity history derivable from a particular $\mathbb{U}^{\mathsf{pr}}$. That is, each primary URLs set U^{pr} induces a set of histories $S(U^{\mathsf{pr}}) \subset \mathcal{H}$ that contains

histories derivable from U^{pr} by obfuscation.

$$S(U^{\mathsf{pr}}) = \{h \in \mathcal{H} \,\big|\, U^{\mathsf{pr}} \subseteq \mathsf{urls\ in}\ h\} \tag{76}$$

We capture this with the obfuscation function $\psi : \mathcal{G} \mapsto \mathcal{H}$ which a malicious user employs to transform its original primary goal g to an obfuscated activity history h. Since a primary goal g is solely defined by a primary URLs set U^{pr}, we can redefine the obfuscation function for this use case to: $\psi : 2^{\mathbb{U}} \mapsto \mathcal{H}$. The obfuscation is naturally bounded by the aforementioned types, thus:

$$\psi(g) = \psi(U^{\mathsf{pr}}) \in S(U^{\mathsf{pr}}) \tag{77}$$

4.2 Features

The key component of the detector is a feature map $\Phi : \mathcal{H} \mapsto \mathcal{X}$ where $\mathcal{X} \subset \mathbb{R}^N$ because the activity history space $\mathcal{H}$ is generally a discrete non-numerical set and the detector $D(d \,|\, x)$ requires numerical inputs. In our running example, $\mathcal{H}$ is a space of all possible HTTP request sequences that query a URL reputation system. Therefore, we need to construct a feature map Φ that ideally reassembles numerical attributes which are helpful in distinguishing malicious samples from benign ones. At the same time, however, we aim to omit spurious features that only provide false or correlated evidence. In an adversarial setting, these are for example features which the attacker's loss function does not depend on. In extreme case, the attacker can arbitrarily adjust those features so that its activity is not detected and it causes zero additional costs.

The feature map used in this work comprises a histogram and a density of URL scores, total count of queries and a request time distribution. The first is certainly a good-quality feature, the second may become a partially spurious feature and the last is absolutely arbitrary to the attacker's model we proposed.

In URL scores histogram, we sort URLs in a given activity to bins according to their scores. Features represent observed frequencies in each bin. URL scores density is a normed frequency histogram, i.e. we take frequency histogram and normalise it so that the values sum up to one. Total count simply represents the number of obfuscating requests (i.e. without requests related to a primary goal). A request time distribution is again a normed frequency histogram of query times within a day in which requests were sent.

Intuitively, this feature map points to a straight-forward attack method: add legitimate URLs until obfuscation is achieved. We call this method a good queries attack and use it as a baseline attack (more details in Sec. 4.4).

The good queries attack however cannot properly distribute requests across time nor it can mix in URLs with different score values to mimic benign users activity. Therefore, we use a gradient attack that generates obfuscation activity based on a

criterion gradient. This method is certainly more complex and requires interpolating the discrete history space $\mathcal{H}$. More details in Sec. 4.5.

4.3 Attacker

In context of the reputation service presented in the previous sections, an attacker instance is a malicious actor that posses a set of primary URLs U^{pr} and aims to query the service to find out the reputation of each URL from U^{pr}.

Relating to the definition of the player attacker in Sec. 3.1.4, the attacker's goal is to identify an obfuscation function $\psi : 2^{\mathbb{U}} \mapsto \mathcal{H}$. However, with an assumption on a particular form of the attacker's loss, the attacker's task decomposes and instead of identifying ψ, the optimal goal obfuscation is a solution of the optimisation problem in Prop. 3.5. We further defined an obfuscation algorithm π which outputs an approximation of an optimal adversarial activity history.

That said, to reflect realistic attackers, we extend the attacker's operation space by a not-to-attack option. Such an option is needed because the solution to the problem in Prop. 3.5 gives an optimal activity history h^* even if the actual cost of carrying out this activity exceeds the cost of no activity by far. Taking this notion into account, we allow the attacker to give up on its primary goal and carry out no activity. This activity is denoted by a token $\mathsf{No\text{-}Activity}$. Accordingly, the codomain of an obfuscation function ψ, the attacker's loss and the decomposition of the optimisation problem adjust to this extension.

In effect, this means that given a primary URL set U^{pr}, the attacker solves the optimisation problem in Prop. 3.5 and checks whether the value of the solution is lower than the detection cost L_0. If it is lower, it carries out the optimal activity history. If the value of the optimum is larger than L_0 does not generate any activity ($\mathsf{No\text{-}Activity}$).

4.3.1 Attacker's Private Loss

The attacker's loss (as in Prop. 3.5) has two components: a public term and a private term. The public term is a single value L_0 that is paid if the attacker is detected. The private term $\Omega_{+1}(g, h)$ is undefined and relates to the specifics of the particular problem domain. Relating to the URL reputation system, we propose a private loss $\Omega_{+1}(U^{\mathsf{pr}}, U)$ which reflects only the number of queries the attacker produces to obfuscate its primary goal g, i.e. the attacker pays an amount L_u for each extra legitimate URL it uses as a disguise.

$$\Omega_{+1}(U^{\mathsf{pr}}, U) = L_u \cdot (|U| - |U^{\mathsf{pr}}|) \tag{78}$$

The particular value of L_u is again domain- and case-dependent. To find a reasonable value, we use a following reasoning: an activity history h that is labelled as 0% malicious (i.e. $D_\theta(\mathsf{M} \,|\, \Phi(h)) = 0$) costs exactly L_0 when it contains $\frac{L_0}{L_u}$ additional

obfuscation URLs. Below, in Sec. 4.6.2, we propose primary goals for this running example. Those primary goals contain from 1 to 40 URLs. We propose to limit the attacker to produce at most 2,000 additional URLs to construct an obfuscation activity that obfuscates a primary goal of at most 40 URLs. This gives a relation between L_u and L_0.

$$L_u = \frac{L_0}{2000} \tag{79}$$

4.4 Good Queries Attack

In Def. 3.7 we proposed that an approximative approach can be used to obfuscate primary goals. We take inspiration in Lowd et al. [9] and propose a base line algorithm that does not give an optimal solution but may carry out a successful attack. This attack is based on the assumptions that legitimate URLs very well obfuscate primary URLs U^{pr}. This means, we keep adding legitimate URLs to the resulting activity history as long as it decreases the attacker's optimisation criterion. The final activity history consists of primary URLs U^{pr} and the appropriate number of URLs from V. The remaining request parameters are set randomly.

Algorithm 2: Good Queries Attack

Input: $D_\theta(\mathsf{M} \,|\, x), U^{\mathsf{pr}} \subset \mathbb{U}$, legitimate URLs $V \subset \mathbb{U}$

$U \leftarrow U^{\mathsf{pr}}$;
while $D_\theta(\mathsf{M} \,|\, \Phi(U))$ *decreases* **do**
  arbitrarily select $u \in V$;
  $U \leftarrow U \cup \{u\}$;
end
return `CreateActivityHistory`(U)

4.5 Gradient Attack

In this section, we propose the gradient attack algorithm π (in accordance to an attack algorithm in Def. 3.7) that approximately obfuscates U^{pr} in T iterations by descending the criterion $L_0 \cdot D_\theta(\mathsf{M} \,|\, \Phi(h)) + \Omega_{+1}(U^{\mathsf{pr}}, h)$ along its gradient (as given in Prop. 3.5).

Notice the game of a detector D_θ and an attacker operates in three layers of spaces: internally the detector infers its decisions in a space $\mathcal{X} \subset \mathbb{R}^N$ but practically it does so utilising a feature map Φ with a discrete space $\mathcal{H}$ on its input. And thirdly, the attacker's algorithm π obfuscates primary goals from a discrete space $\mathcal{G}$. The spaces $\mathcal{X}$, $\mathcal{H}$ and $\mathcal{G}$ (or in our running example $\mathbb{R}^N$, requests space and URL space $\mathbb{U}$ respectively) are entirely distinct.

In the gradient attack, we want to take gradient of $D_\theta(\mathsf{M}|x)$ with respect to inputs and use it to optimally obfuscate the primary URLs U^{pr}. To do so, we introduce a space $K \subset \mathbb{R}^L$ that is an attack parameter space and a mapping $\varphi : 2^{\mathbb{U}} \times K \mapsto \mathbb{R}^N$ that,

using primary URLs $U^{\mathsf{pr}} \subset \mathbb{U}$ and an attack parameterisation $k \in K$, composes a feature vector $x \in \mathbb{R}^N$ such that the intermediate corresponding activity history h meets the constraints imposed by the set of derivable histories $S(U^{\mathsf{pr}})$. This is specifically useful in case of the attacker's optimisation criterion. Because if we substitute a feature map Φ for φ, we arrive this way at an optimisation task with a search space now being $K \subset \mathbb{R}^L$. Note that a particular form of φ is dependent on Φ.

Proposition 4.1. *Let $V \subset \mathbb{U}$ be a set of URLs and let $\varphi : 2^{\mathbb{U}} \times K \mapsto \mathbb{R}^N$ be an attack parametrisation function that is differentiable in $k = [k^A, k^B] \in K \subset \mathbb{R}^L$. The attacker's separated optimisation task becomes:*

$$\begin{aligned} \underset{k}{\textit{minimise}} \quad & L_0 \cdot D(\mathsf{M} \,\big|\, \varphi(U^{\mathsf{pr}}, k)) + \Omega_{+1}(U^{\mathsf{pr}}, k) \\ \textit{subject to} \quad & \sum_j k_j^B = 1 \\ & k_i^A \in \mathbb{N} \end{aligned}$$

The introduction of φ is inspired by Athalye et al. [26] who show that a non-differentiable layer in a neural net can be interpolated. They substitute such a layer for a differentiable one with similar properties and successfully compute the gradient.

4.5.1 Attack Parametrisation

Given a particular feature map Φ, it is critical to find K and φ that are ideally able to construct any $x \in \mathcal{X}$. This is understandably not always possible. With the feature map presented above, we therefore take the following to identify K and φ.

We construct a rich enough set $V \subset \mathbb{U}$ which contains URLs. We associate each $u_i \in V$ with a variable $k_i^A \in \mathbb{N}$ which denotes that the URL u_i shall be used k_i^A times in the activity history that obfuscates the primary URLs. This creates a mixture of ULRs that adjusts the score histogram and the total count of requests in the feature map.

In terms of the request time entropy in the feature map, we assume it is computed over bins representing a time interval. Thus, we associate each bin j with a variable $k_j^B \in [0, 1]$ that reflects a relative request mass in this bin. Naturally, the variables k_j^B are normalised: $\sum_j k_j^B = 1$. With such attack parametrisation we are able to compute gradient of the criterion with respect to k and construct activity histories in $S(U^{\mathsf{pr}})$ if we have a rich enough set V.

The attack parametrisation function then arranges requests according to $k = [k^A, k^B]$ drawing URLs from V and then computes a feature vector x as if it was done with a feature map Φ. Notice that we constructed φ_V to account for the derivable histories set $S(U^{\mathsf{pr}})$ (as defined in Eq. (76)).

Elements of V Ideally, we construct the set V so that it contains URLs that are independent in terms of their influence on a feature map. As mentioned, the feature

map Φ we use in this work constructs features based on a reputation scores histogram, a request count and request time entropy. The selection of V influences only the reputation scores histogram. Therefore, we construct V so that it contains URLs which each populates one bin of the reputation scores histogram. Such V creates a rich-enough mixture using which we are able to construct any activity history in $S(U^{\mathsf{pr}})$.

4.5.2 Gradient Attack Algorithm

Prop. 4.1 gives a non-linear optimisation with a differentiable criterion. However, the search space is constrained by $\sum_j k_j^B = 1$ and $k_i^A \in \mathbb{N}$. To solve this problem we use the projected gradient descent (PGD) [17] combined with the fast gradient sign method (FGSM) [2]. The attack algorithm π of the gradient attack is shown below (Alg. 3).

Algorithm 3: Gradient Attack Algorithm

Input: $D_\theta(\mathsf{M} \,\big|\, x), U^{\mathsf{pr}} \subset \mathbb{U}$, $V \subset \mathbb{U}$

$c(k) = L_0 \cdot D_\theta(\mathsf{M} \,\big|\, \varphi_V(U^{\mathsf{pr}}, k)) + \Omega_{+1}(U^{\mathsf{pr}}, k)$;

$k^{(0)} \leftarrow$ `InitK()`;

for $t = 1, 2, \ldots T$ **do**

  $k^{A,(t)} \leftarrow \texttt{Proj}^A(\nabla_{k^A} c(k))$;

  $k^{B,(t)} \leftarrow \texttt{Proj}^B(\nabla_{k^B} c(k), k^{A,(t)})$;

end

return `MakeActivityHistory`$(V, k^{(T)})$

Routine `InitK()` Initialisation of $k^{(0)}$ is critical because the gradient attack descents along a criterion's gradient and it turns out that setting $k^{A,(0)} = 0$, i.e. starting with solely U^{pr} does not converge very well. Thus we initialise $k_i^{A,(0)}$ uniformly randomly from $\{0, 1, \ldots, 2000\}$ and set $k^{B,(0)} = \frac{1}{\text{number of bins}}$ so that it starts with maximal entropy in request time distribution.

Routine $\texttt{Proj}^A(z)$ Input of this routine z is a gradient vetoer with respect to k^A. We take a sign of the gradient as in FGSM but we do not scale the gradient anyhow. We update k^A accordingly and then crop values below zero. This projection ensures $k_i^A \in \mathbb{N}$.

$$k_i^A \leftarrow \max\{0, \operatorname{sign}(z_i)\} \tag{80}$$

Routine $\texttt{Proj}^B(z, k^A)$ We maintain the scale of the input gradient z (i.e. the learning rate is set to 1.0), update k^B standardly, crop negative values and normalise with Z to sum up to one.

$$k_i^B \leftarrow \frac{\max\{0, k_i^B + z_i\}}{Z} \tag{81}$$

Notice, we know the current number of requests from k^A:

$$|U| = |U^{\mathsf{pr}}| + \sum_i k_i^A \tag{82}$$

As we defined it, k_i^B corresponds to relative frequency of requests sent in a time interval i. Since k^B is arbitrary distribution after the update, we adjust it to reflect the number of requests $|U|$. First, a time bin i gets $\texttt{floor}(k_i^B \cdot |U|)$ requests assigned. Flooring causes some requests were not assigned to a bin, thus we distribute the remaining requests randomly across bins – $\delta_i \in \{0, 1\}$ denotes whether a bin i gets assigned a remaining request. Finally, we use these assignments to compute the relative frequency k_i^B.

$$k_i^B \leftarrow \frac{\texttt{floor}(k_i^B \cdot |U|) + \delta_i}{|U|} \tag{83}$$

Routine `MakeActivityHistory(k)` First, we build the multi-set U by concatenating U^{pr} and V according to k^A.

$$U \leftarrow U^{\mathsf{pr}} \cup \{\underbrace{v_1, v_1, \ldots}_{k_1^A \text{ times}}, \underbrace{v_2, v_2, \ldots}_{k_2^A \text{ times}}, \ldots, \underbrace{v_{|V|}, v_{|V|}, \ldots}_{k_{|V|}^A \text{ times}}\} \tag{84}$$

Using the same procedure as in $\texttt{Proj}^B(z, k^A)$ we assign URLs from U to time bins. Finally, we create requests that each contains a URL $u \in U$ and is sent at the time associated with the bin that u belongs to. We return the activity history h which comprises these requests.

4.5.3 Imperfection of Gradient Attack

Descending along gradient is tricky, especially when projection is involved, as the descent may end up in a local minimum. The gradient attack algorithm solves the task of finding an optimal activity history $h \in \Psi_g^*$ and thus yielding h which is a local optimum is problematic. However, as we adopted a rather agent-driven view of the game (as in Prop. 3.6) we think of the gradient attack algorithm π_T as a feasible agent that does its best to solve the task. In spite of these imperfections, we then train the detector to play against such approximative adversaries.

In the following section we propose an algorithm that solves the detector's optimisation problem.

4.6 Detector

In this section, we introduce two types of a detector. The first type is an anomaly detector based on k nearest neighbours which solves the task in Def. 3.8. The second

type is a stochastic detector modelled with a neural network that solves the task in Prop. 3.6.

The output of both detectors is purposely stochastic. That is, they model the posteriori distribution $p(d \,|\, x)$ where $d \in \mathcal{C}$ is a decision and $x \in \mathcal{X}$ is a feature vector with a model $D_\theta(d \,|\, x)$. At test time, a realisation of a final label d is drawn from $D_\theta(d \,|\, x)$. At train time, the values of probability $D_\theta(d \,|\, x)$ are used in the training process.

4.6.1 Anomaly Detector

The task of detecting malicious behaviour can also be formulated as an anomaly detection problem (as in Def. 3.8). We collect examples of benign behaviour and then construct an anomaly detector whose false positive rate equals τ_0. This approach omits entirely the attacker's model and is based on an anomaly measure $d_k(x)$. We assume more anomalous, i.e. malicious, samples are prone to higher values of $d_k(x)$.

There are various types of anomaly detectors from which we pick one: k nearest neighbours (k-NNs). We use average euclidian distance to k nearest samples $P_k(x) \subset T^{\mathsf{B}}$ in the training set T^{B} as an anomaly measure $d_k(x)$.

$$d_k(x) = \frac{1}{k} \sum_{x' \in P_k(x)} ||x - x'||_2 \tag{85}$$

To comply with a stochastic detector definition, we use the anomaly measure $d_k(x)$ to derive the posteriori probability $D_\alpha(\mathsf{B} \,|\, x)$ as follows:

$$D_\alpha(\mathsf{B} \,|\, x) = \exp(-\frac{d_k(x)^2}{\alpha^2}) \tag{86}$$

The parameter $\alpha \in \mathbb{R}$ adjusts sensitivity to x and is equivalent in terms of the false positive rate to a threshold on the anomaly measure that is usually used with k-NNs. Thus, redefining k-NNs to be a stochastic anomaly detector is redundant in practice, however, we do it anyway as it is convenient for comparison purposes with a true stochastic detector.

The constraint on the false positive rate in Def. 3.8 suggests that our task is to find α for which the false positive rate equals τ_0. We use fast gradient sign method (FGSM) to find the optimal α by minimising the following problem on training samples $T_m = \{\Phi(h_i)\}_i^m = 1$:

$$\min_\alpha (\frac{1}{m} \sum_{x_i \in T_m} D_\alpha(\mathsf{M} \,|\, x_i) - \tau_0)^2 \tag{87}$$

The gradient attack in Alg. 1 requires the detector $D_\alpha(\mathsf{B} \,|\, x)$ to be differentiable in x. $D_\alpha(\mathsf{B} \,|\, x)$ is differentiable up to $d_k(x)$. $d_k(x)$ is not continuous in those x for

which $P_k(x)$ changes its elements. We estimate the gradient of $d_k(x)$ by simply taking derivative while keeping the set of k nearest neighbours $P_k(x)$ fixed.

$$\nabla_x d_k(x) = \frac{1}{k} \sum_{x' \in P_k(x)} \frac{x - x'}{||x - x'||_2} \tag{88}$$

In our experiments, we use $k = 5$ as this is empirically the best value.

4.6.2 Adversarial Detector

We model an adversarial stochastic detector D_θ with a neural network which takes a feature vector $x \in X$ on its input and infers a probability distribution $D_\theta(d|x)$. In test time, an actual decision d is drawn from the distribution $D_\theta(d|x)$. The detector's parameters θ correspond to the weights of the neural network.

We empirically arrived at a relatively shallow network consisting of five fully connected layers. Since the number of inputs N is relatively low (N 20), we assume this is a good trade-off between network's complexity and training time. To address the non-linear nature of features we start with the first two layers being wide with $10 \cdot N$ neurons. Then we narrow the net: the third layer has $5 \cdot N$, the fourth has $5 \cdot N$. Each layer is activated with a SeLU unit. The final layer has 1 output which is transformed with a logistic function (Eq. (89)) to be bounded by $[0, 1]$.

$$f(z) = \frac{1}{1 + \exp(-z)} \tag{89}$$

The output of the final layer's activation (i.e. logistic function) is intended to be an estimate of the posteriori probability $p(\mathsf{M} \mid x)$.

SeLU Activations and Regularisation Instead of classical ReLU, we use the SeLU (Eq. (11)) activation because of better properties of its gradient and its self-normalisation effect. During the process of learning we take gradient of $D_\theta(d|x)$ with respect to x to construct obfuscated activity histories. We found that near-optimal D_θ tends to adjust its weights so that initial steps of the adversarial optimisation are located in areas that are cropped but ReLU (i.e. the activations' inputs tend to be negative).

This is expected behaviour, however, as argued in [26], from the attacker's point of view this is easily bypass-able in a white-box attack. For instance, the attacker replaces all ReLUs by SeLUs. This does not change properties of the network dramatically but gives the attacker access to the gradient.

For that reason (and following the final advice of [26]) we assume the attacker would gain access to gradients anyway using this trick, thus we train the net to learn to defend even such attacks and use SeLUs already. The second reason to use a SeLU as activation comes from the original paper [16] in which the SeLU was introduced. The authors prove it has weights self-normalisation properties, that is, a SeLU is able

to replace batch-norm [33] in a fully-connected feed forward neural nets and allows to use deep architectures with many layers.

Training Sets T^{B} and T^{M} To identify the best parameters θ we use the detector's learning algorithm (Alg. 1). This algorithm estimates gradients from realisations of primary goals and benign activity histories. We draw activity histories $\{h_i\}$ from a training set T^{B} we collected to capture the distribution of benign activity histories. In case of the primary goals that shall be drawn from $p(g)$, we take a different approach because the distribution $p(g) = p(U^{\mathsf{pr}})$ is unknown. The key attributes of the feature map Φ are based on the reputation scores of the queried URLs. We construct primary URLs sets U^{pr} to reflect various ratios of already known bad-score URLs and not yet identified ones. This way we get a training set of primary urls T^{M}:

$$
\begin{aligned}
T^{\mathsf{M}} = \{ & \\
& \{\texttt{known malicious URL}\}, \\
& \{\texttt{uknown URL}\}, \\
& \{\texttt{known malicious URL}, \texttt{uknown URL}\}, \\
& \{\texttt{known malicious URL}, \texttt{known malicious URL}, \texttt{uknown URL}\}, \\
& \{\texttt{known malicious URL}, \texttt{uknown URL}, \texttt{uknown URL}\}, \\
& \ldots \\
\} &
\end{aligned}
$$

Implementation of Stochastic Detector We use pytorch [34] to implement the stochastic detector. However, due the specific requirements of the detector's learning algorithm (Alg 1) such as the inner attack optimisation or the outer λ double optimisation, we needed to implement the training process from scratch as the existing components of the pytorch framework does not fit the need. To compute gradients, we used the framework's autograd library, but gradient descent and the attack optimisation algorithm needed our custom implementation.

Handling No-Activity in Detector $T^{\mathsf{M}} = \{U_i^{\mathsf{pr}}\}$ makes up a faithful mixture of reasonable primary URLs sets. In our experiments, we use a set U_i^{pr} that contains at most 20 `uknown URL`s and 20 `known malicious URL`s. Since no prior preference over individual U_i^{pr} is assumed, we draw U_i^{pr} uniformly from T^{M}.

We allowed the attacker not to carry out any activity if obfuscation is too costly for it. This is captured by the No-Activity token which the attacker's algorithm π produces instead of an activity history h. Despite the detector's risk is derived assuming all primary goals are translated to some activity history, the introduction of No-Activity

does not cause principal problems as we can simply reformulate the equation for the non-stationary probability $\dot{p}(h \mid \mathsf{M})$:

$$\dot{p}(h \mid \mathsf{M}) = \sum_{U^{\mathsf{pr}}:\pi(U^{\mathsf{pr}},D_\theta)=h} p'(U^{\mathsf{pr}}) \tag{90}$$

where $p'(U^{\mathsf{pr}})$ is the probability of observing the primary URL set for which π does not yield No-Activity (i.e. $p'(U^{\mathsf{pr}})$ is $p(U^{\mathsf{pr}})$ normalised by the sum of $p(U^{\mathsf{pr}})$ that are not No-Activity). Consequently, during the Monte-Carlo estimation of the gradient, the estimate γ^{M} is computed from a set of obfuscated activity histories $\{h_i^{\mathsf{obf}}\}$. This set is generated by π from those $U_i^{\mathsf{pr}} \in T^{\mathsf{M}}$ that get obfuscated, i.e $\pi(U_i^{\mathsf{pr}}, D_\theta) \neq$ No-Activity:

$$\gamma^{\mathsf{M}} = \frac{1}{|\{h_i^{\mathsf{obf}}\}|} \sum_{j=1}^{|\{h_i^{\mathsf{obf}}\}|} \nabla_\theta D_\theta(\mathsf{M} \mid \Phi(h_j^{\mathsf{obf}})) \tag{91}$$

Similarity to Cross Entropy Using the Jensen's inequality, we can transform the criterion of θ minimisation (as in Eq. (65)) to a cross entropy loss. In Eq. (65) we essentially minimise $\mathbb{E}_{h,c}\, 1 - D_\theta(c \mid \Phi(h))$. If we remove constant terms and use Jensen's inequality, we arrive at a problem with equivalent solutions:

$$\min_\theta - \mathop{\mathbb{E}}_{h,c} \log(D_\theta(c \mid \Phi(h))) \tag{92}$$

If we estimate the expectation with m samples, the criterion becomes the cross entropy loss. This suggests that we practically solve the same task that is solved when training state-of-the-art neural classifiers. However, the key differences are: we use the algorithm π to create samples of a malicious class M and instead of the classical mini-batch gradient descent [35] we use Algorithm 1.

On Complexity of Stochastic Detector The similarity to cross entropy imposes important implications. Since we practically use the same loss function but model a mixed strategy σ instead of a single classifier f, the complexity of the detector D_θ needs to be much higher than the complexity of a classifier f. This also means, we shall expect training takes more time.

5 Experiments

In previous sections, we proposed a theoretical approach to solve adversarial detection problems. Then we introduced an industrial problem which we formally modelled in accordance to the proposed theory. The outcome is an adversarial detector which when properly trained is able to detect unseen malicious activity. To support the claim we conduct experiments on real-world data (provided by Trend Micro Ltd.) and compare the proposed adversarial detector with an anomaly detector, both being attacked by the good queries attack and the gradient attack. The experiments evaluate: (1) capability of a detector to meet the false positive rate constraint, (2) capability of a detector to detect attacks (measured by the successful attacks rate), (3) performance of the proposed attack algorithms. Further, we analyse the results and identify that: (1) the dataset of benign data contains a few highly suspicious samples, (2) the adversarial detector is highly robust to primary goals with more than 10 low-scored URLs and (3) even more powerful attackers than those at train time are detected at relatively large rates.

5.1 Dataset

The problem of detecting malicious activity in requests to a URL reputation service was proposed by the company Trend Micro Ltd. as a real world problem that is an instance of adversarial machine learning. Thus, we use the company's data to evaluate the proposed algorithms in this work. The dataset we were, gratefully, given contains information that is, nonetheless, private and cannot be made public. Therefore, we do not put the dataset online. However, we are able to include general information and statistics to preview the properties of the dataset.

The dataset consists of genuine real-world activity recorded at such a URL reputation service. Users are uniquely identifiable, thus we are able to make up activity histories of each user. The users are located in the Czech Republic at the time of recording based on the IP address location.

First, we clean data by removing requests that are broken or their information is incomplete. These count: a queried URL is not a valid URL or it is missing. Then we remove requests with a URL that is not a genuine accessible URL: that are, for instance, URLs containing `.arpa` or `.in_addr`. Then the activity is sorted to days and each queried URL is given a genuine reputation score returned by the reputation service. Then we collect these per-one-day per-user activity histories and remove those containing less then 10 queries (i.e. 10 requests per user per day) for we assume an activity this low is anomalous and including it would poisson the final data set.

The total number of samples after pre-processing is $54,970$ which split into a training set of 43 thousand samples and a testing set of 11 thousand samples (we use 80-20 ratio). The reputation scores that are returned by the service are processed so that they correspond to a probability of a particular URL being benign. We are given only

such values of the score so that the corresponding probability values are either 0.1, 0.5 or 0.9. The dataset contains only few malicious URLs (0.05%). The unrated URLs count 15.0% and the benign ones 84.95% of the URLs. Any future unknown (i.e. not included in the dataset) URL is considered unrated.

The distribution of URL use has very long tails. Fig. 2d shows roughly 50% of the URLs are used only once and 90% are used up to 10 times. On the other hand, the dataset contains URLs that the service was queried with over $1,000$ times.

An average sample comprises an activity history of 700 requests (per day). However, this distribution is fat-tailed. Most samples have around 1000 requests, yet there are samples with over $10,000$ requests and, on the other hand, samples with 10 requests. The histogram of the distribution is depicted in Fig. 2b.

In Fig. 2a, the request time distribution within a day is shown. Requests are sent mostly between 8am and 10pm with a peek between 6pm and 9pm. Also, there is a little drop around noon, suggesting this is a proper lunch time among recorded users. At night time from midnight until 7am, there is a significant drop in the number of sent requests. Ideally, to disguise as a benign user, an attacker shall follow this distribution and adjusts its attack and obfuscation accordingly.

5.2 Setting

To evaluate the training and attack algorithms we use the dataset provided by Trend Micro Ltd. In terms of attackers, we use the good queries attack algorithm (Sec. 4.4) and the gradient attack algorithm (Sec. 4.5). To compare various approaches to detection, we use an anomaly detector based on k-NN (Sec. 4.6.1) and an adversarial detector based on a neural network (Sec. 4.6.2).

Since the detector's learning algorithm is proposed to learn against an attacking adversary, it is reasonable to consider the detector based on a neural network shall be trained against both attacker types. However, theoretically and empirically the good queries attack is not as advanced as the gradient attack. Thus we only perform experiments in which the neural net detector is trained vs. the gradient attack. On the other hand, we evaluate attack performance for both attack types.

False Positive Rate We train the neural net detector against the gradient attack algorithm and show results for false positive rate threshold values τ_0 being 1%, 0.1% and 0.01% (10^{-2}, 10^{-3} and 10^{-4}). FPR of 0.01% on this size of a test set (to repeat, it counts 11 thousand samples) corresponds to 1 sample. This is already reaching limits of statistical evaluation and results with FPR set to values below 0.1% would ideally require bigger dataset size to reach greater significance.

Attacker's Loss The attacker losses are set to correspond to motivations given in Sec. 4.3.1. That is, we set the cost of being detected L_0 to 100 units and the cost of a request L_u to 0.05 units. The maximum attack cost is set to 99 (motivation is

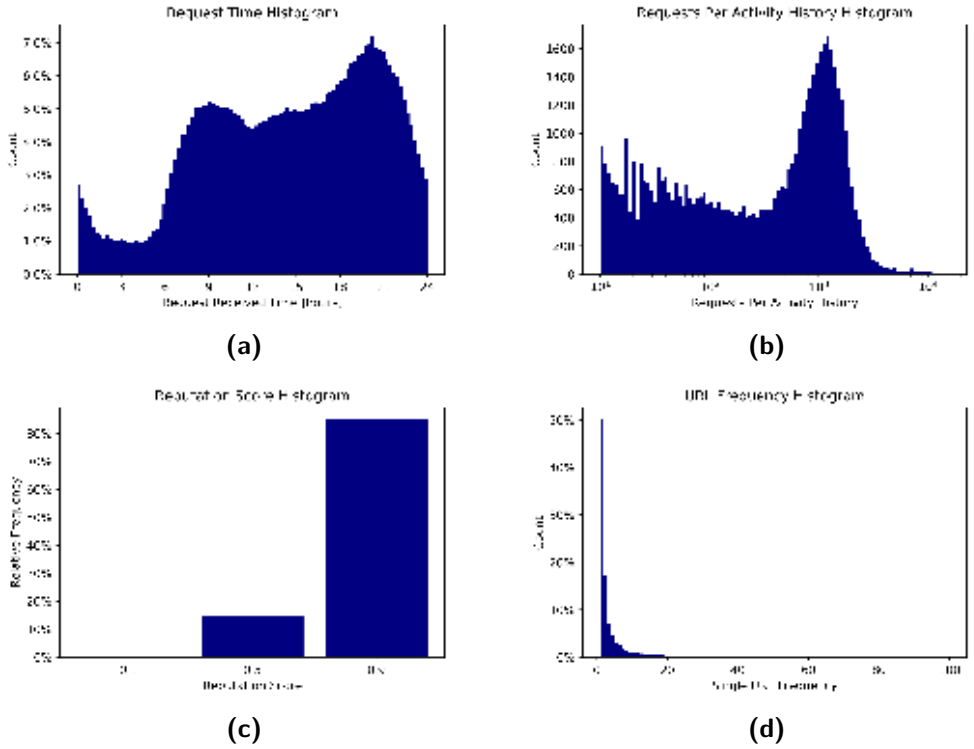


Figure 2: The histograms show distributions of activity captured in March, 2019 among users of a URL reputation service, located in the Czech Republic. The dataset is provided by Trend Micro Ltd. Fig. 2a depicts request day-time distribution, Fig. 2b shows the amount of requests that is sent in one day activity of a user. Fig. 2c shows the distribution of a URL reputation score which is associated with a URL query. Finally, Fig. 2d shows the repetitive nature of such a reputation service.

given in Sec. 4.3.1). We set L_u to 100 and the maximum attack cost to 99 to disallow the attacker to attack in extreme cases when its criterion closely reaches the no-attack threshold. This happens especially when the attacker's instance is in the area with very high confidence of malicious activity and it becomes rational no to carry out any obfuscation but to use only primary URLs U^{pr}. These cases are now labelled No-Activity .

Feature Map As already mentioned in Sec. 4.2, we use four types of features. The first feature map is a frequency histogram of URL reputation score values contained in an activity history. The dataset comprises only three distinct values of a score, thus we use three bins with edges at 0, 0.33, 0.66, 1. The second feature type is URL score density which is a normed frequency histogram. The third feature is a square root of a total count of requests per activity history. The distribution of requests counts is fat-tailed, thus we reduce the influence of large values by taking a square root. We also tried logarithm but square root seems to perform better. Lastly, we add a normalised

frequency histogram of request times to relate to time distribution of requests. We use 24 bins that each covers one hour. The full feature map counts 31 features in total. Data are normalised so that each input has empirical mean equal to 0.0 and variance equal to 1.0 on training data.

Performance Measures To evaluate the detector's performance, we use two main measures: the first is a false positive rate (FPR), the second is successful attacks rate (SAR). The false positive rate is the rate of misclassification on benign data which shall, for the optimal detector, equal to the threshold τ_0. Any deviation from the value τ_0, negative or positive, is a failure because such a detector either does not meet the FPR criterion or is too benevolent, suggesting there is a tighter one with better detection rate.

The successful attack rate is the ratio of undetected attacks and the total number of attacks. SAR corresponds, in fact, to a false negative rate and is the main criterion of the detector's optimisation task (Prop. 3.3). The obfuscation rate (OBR) is a subject of optimisation during the detector's learning (i.e. it is the false negatives rate). We define OBF as the ratio of undetected attacks and attacks performed (that is without No-Activity). The lower the successful attack rate and the obfuscation rate, the better the detector is. We also use other supporting measures: a No-Activity rate (NAR) and mean successful attack length (MAL). NAR gives a percentage of attacker instances that did not carry out any activity (and thus did not follow the goal). The mean successful attack length (MAL) gives an average number of additional URLs that were used in successful attacks (that is undetected attacks). To evaluate these measures, we use the as-if-deployed approach—this means realisations of the probability $D_\theta(d \,|\, x)$ are drawn to make the final decision d. This suggests that the measured numbers are in fact realisations of random variables and thus those statistics are random variables as well.

5.3 Detector Learning Procedure

Lambda A key difference of the detector's learning algorithm (Alg. 1) to standard classification learning schemes is the constraint on FPR which results in the variable λ as a control variable. Lemma 3.7 proposes it is better to think of λ as a priori class probability $p(\mathsf{M}) = \frac{1}{1+\lambda}$ which changes during the training procedure so that the constraint on FPR is met. We found that $p(\mathsf{M})$ very quickly converges to really low values (around ~ 0.95) so that the FPR constraint is met. To give little influence at least in the begging of learning to malicious data, we start training with $p(\mathsf{M}) = 0.5$ (i.e. $\lambda = 1$). Once the FPR constraint is met and $FPR \leq \tau_0$, malicious data start again to have larger influence on gradient and $p(\mathsf{M})$ increases again.

Learning Rate We use gradient ascent to find λ and gradient descent to find parameters θ. Since these are performed simultaneously but both correspond to different

aspects of the problem, we found that their learning rates shall differ which goes along with suggestions in [32]. We use a learning rate of magnitude 0.01 in case of θ and a learning rate of 5.0 in case of λ. This speeds up learning procedure especially in meeting the FPR constraint.

Batch Size In the process of gradient estimation, we generate m samples for each class to get gradients conditioned on class. We call m the batch size, although its meaning is different from the standard concept—usually, a batch size refers to the number of samples drawn in total from a training set during a gradient descent step but we actually draw m sample for each class (All introduced in Sec. 3.4.1 and Sec. 3.4.2). In terms of the value of the batch size, it turns out that the lower m the greater the chance all of the drawn malicious instances generate No-Activity which essentially causes zero gradient attained on a malicious class. On the other hand, our set of primary goals (primary URLs) T^{M} counts 360 samples which after splitting creates a train set of 288 primary goals. Since we want to employ the mini-batch gradient descent [35], m should be lower than the number of primary goals but reasonably large to suppress noise. To balance the two notions, we use $m = 100$.

Attacker's Optimisation The obfuscation algorithm π (Def. 3.7) takes a primary URL set U^{pr} and generates an obfuscated activity history h^{obf} (or No-Activity) in T steps. The gradient attack algorithm (Alg. 3) does so with a mixture of projected gradient descent (PGD) and fast gradient sign method (FGSM). As introduced in Sec. 4.5.2, we set the initial time distribution of activity k^B to be uniform and the initial number of obfuscation URLs k^A randomly to any of $\{0, 1, \ldots, 2000\}$. Naturally, the optimal number of steps T balances the quality of a solution and time needed to find it. We use $T = 400$ which turns out to be a reasonable balance.

5.4 Results

In this section, we show that the problem of detecting malicious behaviour is better solved with an adversarial detector which outperforms an anomaly detector. First, we show performance of the detectors in various settings and analyse their exploitability against two attack types. Secondly, we analyse the attacks performed against the best detector and show what kinds of attacks are are very likely to be detected.

5.4.1 Optimal Detector

To address the problem of detecting malicious users of a URL reputation service, we use two models of a detector: an anomaly detector based on k-NN and an adversarial stochastic detector based on a neural net. We show that the neural net trained against a model of an attacker outperforms the anomaly detector. We use three FPR thresholds,

$\tau_0 \in \{1\%, 0.1\%, 0.01\%\}$, and perform attacks with the good queries attack and the gradient attack.

False Positive Rate Threshold A key requirement of a detector in network security is that the false positive rate (FPR) is below a threshold τ_0 (as argued in Sec. 3.1.1). To validate our detectors are able to meet this constraint, we fit them on train data and measure FPR on test data. The adversarial detector is fitted against the gradient attack. The FPR results are shown in Tab. 2. All training sessions successfully converge below a desired threshold value. However, the anomaly detector does not meet the FPR constraint on a test set in case of $\tau_0 = 1\%$ and $\tau_0 = 0.1\%$, whereas the adversarial succeeds in all settings.

Note that the training FPR is usually below the desired threshold. Take for instance the anomaly detector that with $\tau_0 = 1\%$ gives a training FPR 0.85%. This is caused, we argue, by the distribution of benign data. The distribution contains a relatively large number of outliers that sparsely located far from the distribution. This causes a detector with fixed limited complexity (i.e. k in k-NN) is not capable of reaching the exact value of tau_0.

FPR Threshold τ_0	1%		0.1%		0.01%	
	k-NN	AdvDet	k-NN	AdvDet	k-NN	AdvDet
FPR on Train Data	0.58%	0.96%	0.09%	0.07 %	0.01%	0.01%
FPR on Test Data	1.03%	0.96%	0.11 %	0.08%	0.01 %	0.01%

Table 2: False Positive Rates on Test Data

FPR Thresh. τ_0	1%		0.1%		0.01%	
	k-NN	AdvDet	k-NN	AdvDet	k-NN	AdvDet
NAR	52%	57%	54%	32%	45%	1 %
OBR	25%	3%	85%	39%	95%	49%
SAR	12%	1 %	39%	27%	52%	48%

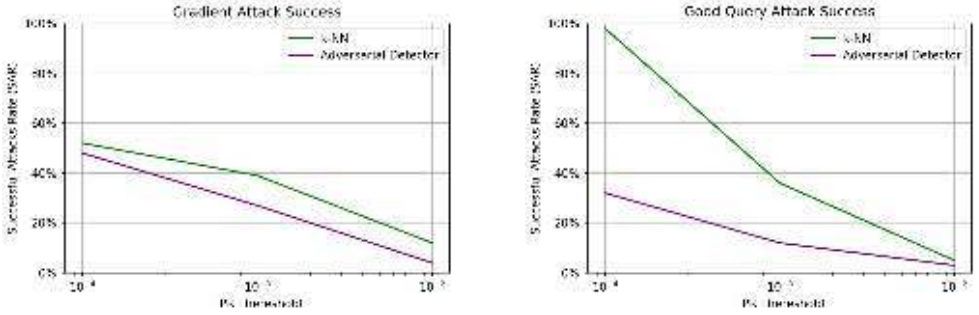
Table 3: Gradient Attack Results

FPR Thresh. τ_0	1%		0.1%		0.01%	
	k-NN	AdvDet	k-NN	AdvDet	k-NN	AdvDet
NAR	88%	94%	54%	4%	0%	0%
OBR	44%	50%	78%	13%	98%	32%
SAR	5%	3%	36%	12%	98%	32%

Table 4: Good Queries Attack Results

Exploitability by Gradient Attack We measure exploitability with a successful attack rate (SAR). The adversarial detector gives better performance than the anomaly detector in terms of SAR. The results of the gradient attack algorithm are shown in Tab. 3 and the trend is shown in Fig. 3a. With the FPR threshold at 1%, successful attack rate (SAR) is at 12% for the anomaly detector (k-NN) and 4% for the adversarial detector. But with lower thresholds, the difference narrows. At 0.1%, the anomaly detector at 39% and the adversarial detector allows SAR at 32%. With FPR threshold $\tau_0 = 0.01\%$, the adversarial detector (SAR 48%) outperforms the anomaly detector (SAR 52%). But both detectors reach almost 50% exploitability. To sum it up, the adversarial detector outperforms the anomaly detector, especially at 1% threshold.

We assume that the narrowing performance margin is also given by a low number of benign samples for this low FPR thresholds. As mentioned, for the 0.01% threshold, the detector achieves the desired constraint by misclassifying at most only 5 benign samples among the outliers in the train set. We conjecture that with a greater dataset, the adversarial detector achieves lower SAR values even with $\tau_0 = 0.01\%$.



(a) Successful Attack Rate (SAR) of Gradient Attack

(b) Successful Attack Rate (SAR) of Good Query Attack

Figure 3: Successful attack rate (SAR) as a function of the false positive rate (FPR). Note that, as the FPR threshold is increased, both detectors become more robust. At all FPR levels and with both attack types, the adversarial detector outperforms the anomaly detector.

Exploitability by Good Queries Attack The good queries attacks algorithm adds legitimate obfuscating requests to the final activity history as long as the cost of an attack decreases. In comparison to the gradient attack, it is weaker but more realistic to capture behaviour of a malicious user that rather intuitively obfuscates its primary goal. In Tab. 4 we show the results of the good queries attack against anomaly and adversarial detectors at various levels of FPR. The overall trend is depicted in Fig. 3b. It is clear that, same as with the gradient attack, the adversarial detector allows fewer attacks in at all FPR levels.

The difference is best seen at $\tau_0 = 0.1\%$ where the anomaly detector achieves a successful attack rate (SAR) of 36%, whereas the adversarial detector achieves 12%.

Note that, with $\tau_0 = 0.01\%$ and the anomaly detector, the good query attack (SAR 98%) performs surprisingly better than the gradient attack (SAR 52%). This is caused by the large NAR of 45% in the gradient attack while it is 0% with the good query attack – this means a large portion of attacks was not carried out due to, probably, too few iterations. However, if we consider the obfuscation rate (OBR) which is the percentage of successful undetected obfuscations out of actually performed attacks, both attacks are nearly similar with this detector.

To conclude, both detectors are robust against the good queries attack but the adversarial detector allows lower attack success rates at all FPR levels.

Suspicious Outliers in Dataset The set of benign activity T^{B} comprises data of real users of the company Trend Micro Ltd. During the training process of our detector, some benign samples tended to be classified as malicious with relatively high confidence (over 90%). A closer inspection revealed these samples truly contain requests with URLs that are suspicious. In fact, a single user was repeatedly labelled malicious in two of its sample (i.e. two independent days of activity). For instance, this user queried the service with a URL of a domain which, when visited, redirects to google.com if no URI path is given. But once a specific and long URI path is appended to the domain, it instead redirects several times to various other domains and gives an empty site in the end. Of course, this is far from identifying this particular user is a true malicious actor - it very well may have been an infected computer - but it shows that the detector correctly labels samples that contain suspicious activity and considers them outliers.

5.4.2 Attack Analysis

Primary Goal - No-Attack Dependency As argued, the problem of malicious activity detection is difficult in that only benign data are available at the time of training. In this work, we proposed a model of an attacker and, consequently, created a feasible set of primary goals. The training set of malicious data T^{M} contains primary URLs sets U_i^{pr} that we crafted purposely to represent various attackers. We generated U_i^{pr} to comprise URLs with a malicious reputation score and URLs with a yet unrated reputation score. (All introduced in Sec. 4.6.2)

We trained an adversarial detector and then performed attacks with the gradient attack algorithm. We found that there is a pattern in what primary goals tend to get obfuscated and what are turned into No-Activity . The relation is depicted in Fig. 4 where we plot each test set primary goal as a single point parametrised by the contents of the primary goal (i.e. the primary URL set U_i^{pr}). Primary goals that are turned to No-Activity are coloured in red and primary goals that are obfuscated and reassemble an attack are coloured blue. To outline the pattern, we estimate probability density for each group: No-Activity and attacks. The figure shows that primary URLs sets containing more than roughly 10 truly malicious URLs are more likely to become No-Activity . Whereas primary URLs sets with less than 10 truly malicious URLs are prone

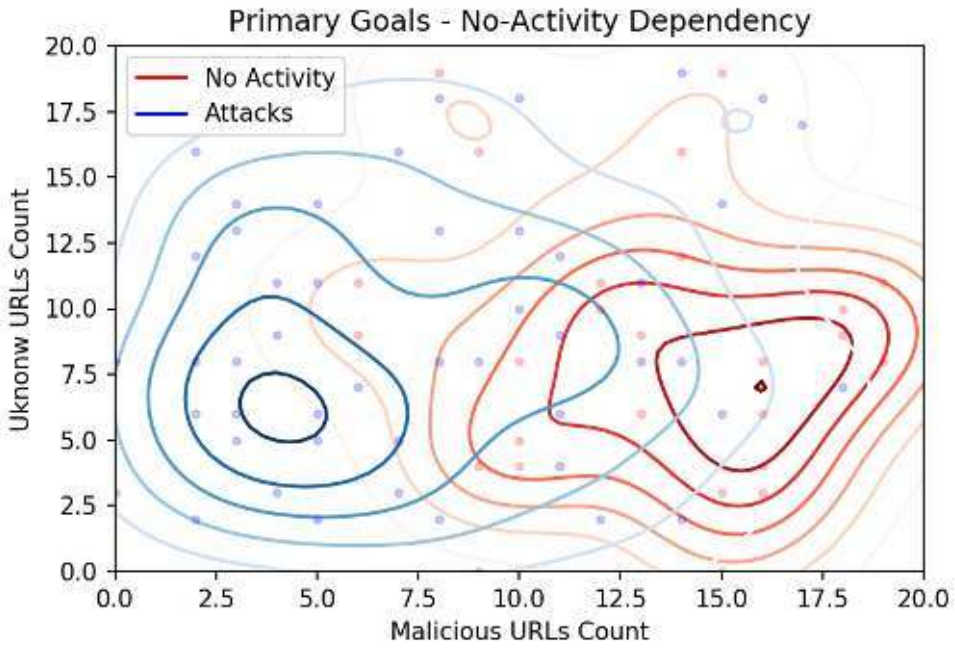
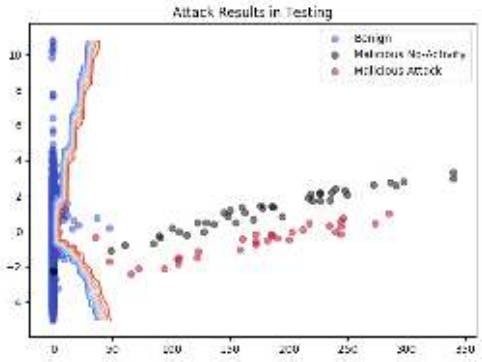
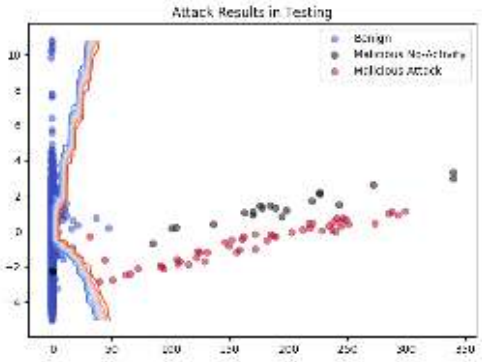


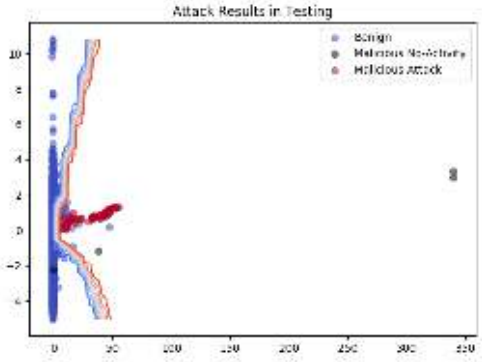
Figure 4: Primary Goals – No-Activity Dependency. We found that there is an emerging pattern in the obfuscation ability against the adversarial detector. Some primary goals tend to be too costly to be obfuscated so the algorithm turns them to No-Activity . This figure shows the pattern: we draw primary goals (primary URL sets) that are modified to No-Activity in red and primary goals that are turned into an activity history in blue. The axes correspond to parameters of the primary goals we generated: the x-axis shows a number of URLs with a malicious reputation score in a primary goal U^{pr}; the y-axis shows the number of unrated URLs. The figure depicts individual primary goals as dots and an estimated density distributions with contours. Clearly, primary goals with more 10 malicious URLs tend to become No-Activity whereas primary goals with fewer than 10 malicious URLs tend to be conversed to an obfuscated activity history and a corresponding attack is carried out. All data are from a test set of primary goals and the results are taken from an attack against the adversarial detector with $FPR = 0.1\%$.



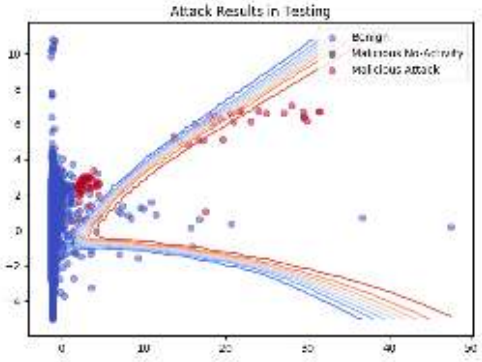
(a) Attack with $T = 400$ and $L_u = 0.05$



(b) Attack with $T = 800$ and $L_u = 0.05$



(c) Attack with $T = 800$ and $L_u = 0.005$



(d) Attack with $T = 1600$ and $L_u = 0.0005$

Figure 5: The figures plot single activity histories as points in a feature space. All images are PCA-transformations of the feature space with identical principal components. Black points are No-Activity , red points are obfuscation activity of individual attacker instances, blue points are benign activity histories. The contours correspond to the class posteriori probability modelled with the detector. All pictures show the results of the adversarial detector. Upper left, the results of the gradient attack with 400 iterations – NAR is 57.33%. Upper right, attack with 800 iterations - NAR is 29.33%. Bottom left, attack with 800 iterations but less expensive per-request cost, $L_u = 0.005$. Bottom right, attack with 1600 iterations but extremely inexpensive per-request cost, $L_u = 0.0005$. Note that attacks are located in different areas as we change attacker's costs. In the extreme case (bottom right), nearly all attacks are located beyond the detector's contours.

	L_u	Iterations	NAR	SAR	OBR	MAL
Train Time Attack	0.05%	400	57%	1%	3%	20
More Iterations	0.05%	800	29%	1%	1%	19
Cheaper Requests	0.005%	800	5%	29%	30%	146
The Cheapest Requests	0.0005%	1,600	0%	94%	94%	2003

Table 5: Attack Cost Analysis

to become an obfuscated activity history.

This means that the optimal detector is more exploitable by attacker instances that have fewer truly malicious URLs in their primary goals whereas an attacker instance with a lot of truly malicious URLs tends to be detected. This implies that, after deploying this detector, an attacker which has full knowledge of the setting deals with a fact that employing more than ten malicious URLs a day leads to too costly obfuscation and it is rational not to attack.

No-Activity Rates The attacks performed in our experiments tend to have large No-Activity rates (usually NAR is between 50% and 80%). A No-Activity occurs if, given a primary goal, the attack algorithm fails to create an obfuscation activity history because carrying out the attack is too costly. However, it seems this happens too often due to imperfections of the attack algorithm: it for example gets stuck in a local optimum or convergence takes too many iterations. We argue relatively high NAR is mainly caused by the number of iterations of the attack algorithm which, despite already being high (400), is sometimes insufficient for finding a less costly activity history. This can be seen when we attack a fitted detector with a gradient attack that runs in 800 iterations. By doubling the number of iterations, NAR drops from 57.33% to 29.33%. Interestingly enough, these attacks, nonetheless, maintain comparable OBR and SCR, i.e. detector's exploitability remains unchanged even though the attacker uses more iterations to craft the attack. Fig. 5a shows attacks of a 400-iterations attacker and Fig. 5b shows a 800-iterations attacker. The figures depict a PCA-transformed feature space with test data of both benign and malicious classes. The red points are final malicious attacks, while the black points are primary goals turned to No-Activity . The contours show a class posteriori probability modelled by the detector, but projected to a hyperplane attained by PCA. Note that the original problem has over 30 dimensions, thus this view skews distances and causes some relations are misleading. However, the main point can be illustrated: when attacking with 800 iterations the attacker converts more primary goals to an activity history than with 400 iterations.

Attack Cost Analysis The attacker's loss has two constants: a cost for being detected L_0 and a cost for sending one request L_u. As presented in Sec 4.3.1, we use $L_0 = 100$ and $L_u = 0.05$. To repeat, per-request cost $L_u = 0.05$ can be interpreted as

follows: an activity history that is labelled as 0% malicious costs exactly L_0 when it contains $2,000$ additional obfuscation URLs. Or to put it differently, an obfuscation activity may contain at most $2,000$ additional requests. However, as we pointed out in the dataset analysis above, the median of the number of requests per activity history is actually $\sim 1,000$ and the distribution contains well-represented activity histories even with $\sim 5,000$ requests. Thus by this choice, we limit the attacker to create activity histories with fewer than 2000 requests which, however, we argue is a reasonable amount for an attacker.

To check whether attacks with lower per-request cost L_u are able to circumvent the detector, we perform attacks with cheaper costs. In addition: during attack generation, we use 400 iterations in the attack's gradient algorithm which in each iteration changes the number of requests by little (usually the change is 2 or 3 requests). Thus, a smaller L_u, which implies higher maximal number of requests, necessarily requires more iteration steps – which is costly and causes significant increase in training time.

From the reasons above, we attack the adversarial detector with the gradient attack with: 400 iterations and $L_u = 0.05$ (train time attack), 800 iterations and $L_u = 0.05$ (more iterations), 800 iterations and $L_u = 0.005$ (cheaper requests) and 1,600 iterations and $L_u = 0.0005$ (the cheapest requests). The attacks' results can be seen in Tab. 5. The attacks are depicted in Fig. 5.

As discussed above, doubling the iterations number from 400 to 800 maintains the adversarial detector's exploitability. However, if we lower the per-request cost to $L_u = 0.005$ which changes the maximum number of requests to 20,000, the detector's exploitability suffers. The successful attack rate (SAR) increases to 30% from 1%. Following the lower cost of requests, the mean attack length increases as well from 20 to 146. OBR rises to 30.99% with NAR at 5.33%. This also shows that attacks that were carried out previously at higher costs are now turned into an activity history with greater chance of a successful obfuscation. However, the values are not critical and are comparable to the values attained on an anomaly detector with a train time attack ($L_u = 0.05$ and 400 iterations.).

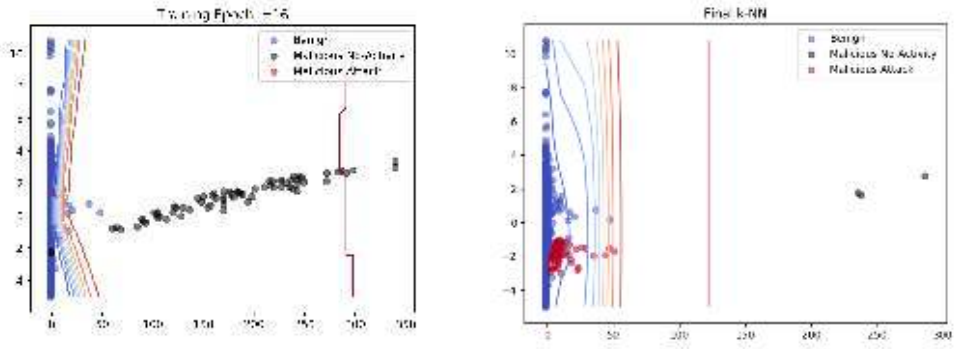
The resulting attacks of this setting can also be seen in Fig. 5c. Note that points representing the attacks moved towards the detector's contours but they remained in the malicious-labeled area. We assuem the shift in attack placements reflects the lower per-request cost as the attacker is able to mix in more obfuscation URLs and move closer to legitimate benign samples with occasional low-score URL appearances.

Finally, an attack with $L_u = 0.0005$ (equivalent to $200,000$ max-requests number) and the number of iterations $1,600$ increases OBR rapidly to 94.67% while entirely erasing NAR (0.0%). Accordingly, MAL increases to $2,000$. The detector is largely exploitable by this attack. Same is seen in Fig. 5d which depicts attacks with the cheapest per-requests cost. Most of the attacks are now moved pass the detector's contours to the area of the feature space with high benign posteriori probability. Note that compared to the previous attacks, the cheapest cost attack instances are placed in

a different area - higher along the y-axis. This corresponds to the fact that obfuscation may occur with more requests creating an activity history with mean attack length around 2,000. These activity histories were not generated during training as the train time attacker was limited to at most 2,000 additional requests. Therefore, we may expect that this is a blind spot of the detector because it was not trained to detect such attacks.

It is important point out that the last attack (the cheapest cost) is off scale compared to the attack used at train time. In addition, we conjecture that a detector becomes robust even to this attack if it is trained against it using the detector's learning algorithm (that is following the same procedure but with much lower attacker costs and more iterations during the attack). This, however, will increase computational requirements and training time.

Anomaly Detector - Adversarial Detector Comparison As argued, an anomaly detector does not incorporate a model of an attacker which means it defends attacks "from all directions". Whereas, an adversarial detector takes advantage from the attacker model and defends only "directions which are susceptible to lure attacks". This is better seen in Fig. 6 which depicts a view of the feature space transformed with PCA. The figure shows contours of detectors' posterior class probability $D_\theta(\mathsf{M}|x)$. In case of the adversarial detector, we see that the detector's contours are shaped to reflect areas in which an attack is likely and entirely omit areas in which attacks are not found. Whereas, in case of the anomaly detector, contours only reflect the benign data distribution. This is the main advantage given to an adversarial detector - it models its posterior probability so that it reflects possible attacks.



(a) Adversarial Detector **(b)** Anomaly Detector

Figure 6: The figures plot single activity histories as points in a feature space. All images are PCA-transformations of the feature space with identical principal components. Black points are No-Activity , red points are obfuscation activity of individual attacker instances, blue points are benign activity histories. The contours correspond to the class posteriori probability modelled with the detector. The pictures compare the shape of detector's posterior class probability that is depicted wtih contours. On left, the adversarial detector shapes its contours to reflect possible attacks, whereas, on right, the anomaly detector omits the attacker's model and thus adjusts its contours to face all possible anomalies. This gives the adversarial detector advantage in that it reflect the attck distribution.

Conclusions

This work examined adversarial machine learning in network security. We focused on a problem of detecting malicious activity while, ideally, not affecting benign users. We started with the assumptions that a detection false positive rate is constrained by a threshold; and that malicious activity cannot be recorded faithfully and, in addition, changes in response to the parameters of a detector. (Sec. 3.1)

To model the setting, we modified the empirical risk minimisation framework to correspond to the Neyman-Pearson task (Sec. 3.1.1). Using the notion of statistical learning, we defined a game in which a detector identifies the best parameters of its stochastic detection classifier and an attacker searches for the optimal obfuscation method of its primary goals (Sec. 3.1.3). Such a model of an attacker builds on the assumption that malicious actors aim to act rationally by minimising their risk (Sec. 3.1.4). In terms of the detector, we argued that it necessarily must be a stochastic detector which means that instead of finding a classifier we solve a task of modelling the posterior class probability and draw the final label from it. (Sec. 3.1.6)

Then, assuming the players play a strong Stackelberg equilibrium, we arrived at a bilevel optimisation problem whose solution is a robust detector that minimises exploitability by an attacker and keeps its false positive rate below a given threshold (Sec. 3.1.5). We then proposed a detector's learning algorithm that approximates the solution of the optimisation problem and outputs a detector which despite being trained solely on legitimate benign data is able to detect unseen malicious activity. The detector's learning algorithm follows a scheme of adjusting parameters according to the attacker's best responses (Sec. 3.4).

In Sec. 3.5, we proposed that the task of detecting malicious activity can be also solved with an anomaly detector. This approach comes with a critical disadvantage – a model of an attacker is omitted entirely and the anomaly detector can be trained only on benign data.

As a running example, we used attacks to a URL reputation system and proposed a formal model of those attacks (Sec. 4.1). We proposed a good query attack (Sec. 4.4) and a gradient attack (Sec. 4.5). The gradient attack, given a primary goal (a set of target URLs), identifies a local-optimum activity that obfuscates this primary goal. The attack is novel in that it is able to create obfuscating activity even in domain that is highly discrete. This is achieved because we conveniently parametrised attacks and used projected gradient descent and a fast gradient sign method to find the local-optimum set of requests (Sec. 4.5.1).

Then we adjusted the proposed detector's learning algorithm to the domain of reputation score requests. We proposed a neural network architecture with five layers that models a posterior class probability (Sec. 4.6.2).

Using real-word data provided by Trend Micro Ltd., we show that an adversarial detector outperforms an anomaly detector in all false positive ratio scenarios (1%, 0.1%

and 0.01%). In terms of meeting the false positive rate constraint, an adversarial detector meets it in all three scenarios, whereas the anomaly detector meets the constraint only on training data. In terms of exploitability (which we measure in successful attack rates), an adversarial detector detects more attacks than an anomaly detector. (Sec. 5.4.1)

Concerning real-time benign data, we found that it contains outliers which our detector confidently labels malicious. On closer manual inspection, those outliers seem to carry out suspicious activity and are, thus, correctly labeled malicious (Sec. 5.4.2).

In future work, a detector's learning algorithm convergence proof may be delivered. Also, our feature map omits sequential nature of activity histories based on sent request- a more complex feature map that takes raw request sequences may be researched.

In conclusion, this work proposed a theoretical background and a practical algorithm to a problem of detecting malicious activity in network security. We like to think of it as a proof-of-concept for real-world adversarial detection problems.

Appendix A - Contents of Attached CD

The attached CD contains implementations of the developed algorithms. All code is written in python and the main library which is used throughout the project is pytorch [34].

The folder sources contains all source codes of the project. It contains several files and subfolders but the main subfolders are: models, threat_model, diagnostics. The folder models contains implementation of all proposed detector models and their training mechanism. The folder threat_model contains implementation of all propose attackers. And finally, the folder diagnostics contains scripts that run the experiments. Given the data are at their expected location (see the script for more details), the training scheme of the adversarial detector is executed by running experiment.py. To run a k-NN anomaly detector, execute experiment_knn.py

Since the data provided by Trend Micro Ltd. contain private information of the company's users, we are not allowed to attach them to this work.

References

[1] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li, "Imagenet large scale visual recognition challenge," *CoRR*, vol. abs/1409.0575, 2014.

[2] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples. corr (2015)," 2015.

[3] A. M. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," *CoRR*, vol. abs/1412.1897, 2014.

[4] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[5] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-gan: Protecting classifiers against adversarial attacks using generative models," *arXiv preprint arXiv:1805.06605*, 2018.

[6] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, (Cambridge, MA, USA), pp. 2672–2680, MIT Press, 2014.

[7] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial examples for malware detection," in *European Symposium on Research in Computer Security*, pp. 62–79, Springer, 2017.

[8] H. S. Anderson, A. Kharkar, B. Filar, D. Evans, and P. Roth, "Learning to evade static PE machine learning malware models via reinforcement learning," *CoRR*, vol. abs/1801.08917, 2018.

[9] D. Lowd and C. Meek, "Good word attacks on statistical spam filters," in *CEAS*, 2005.

[10] V. N. Vapnik, *Statistical Learning Theory*. Wiley-Interscience, 1998.

[11] M. Brückner and T. Scheffer, "Stackelberg games for adversarial prediction problems," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 547–555, ACM, 2011.

[12] K. Amin, S. Singh, and M. P. Wellman, "Gradient methods for stackelberg security games," in *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, UAI'16, (Arlington, Virginia, United States), pp. 2–11, AUAI Press, 2016.

[13] E. Lockhart, M. Lanctot, J. Pérolat, J. Lespiau, D. Morrill, F. Timbers, and K. Tuyls, "Computing approximate equilibria in sequential adversarial games by exploitability descent," *CoRR*, vol. abs/1903.05614, 2019.

[14] J. Neyman and E. Pearson, "On the problem of the most efficient tests of statistical hypotheses," *Philosophical Transactions of the Royal Society, A*, vol. 231, pp. 289–337, 01 1933.

[15] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (G. Gordon, D. Dunson, and M. Dudík, eds.), vol. 15 of *Proceedings of Machine Learning Research*, (Fort Lauderdale, FL, USA), pp. 315–323, PMLR, 11–13 Apr 2011.

[16] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, (USA), pp. 972–981, Curran Associates Inc., 2017.

[17] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *CoRR*, vol. abs/1611.01236, 2016.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, (USA), pp. 1097–1105, Curran Associates Inc., 2012.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017.

[20] J. Z. Kolter and E. Wong, "Provable defenses against adversarial examples via the convex outer adversarial polytope," *arXiv preprint arXiv:1711.00851*, vol. 1, no. 2, p. 3, 2017.

[21] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar, "Adversarial machine learning," in *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, AISec '11, (New York, NY, USA), pp. 43–58, ACM, 2011.

[22] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?," in *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security*, ASIACCS '06, (New York, NY, USA), pp. 16–25, ACM, 2006.

[23] A. Huang, A. Al-Dujaili, E. Hemberg, and U. O'Reilly, "Adversarial deep learning for robust detection of binary encoded malware," *CoRR*, vol. abs/1801.02950, 2018.

[24] B. I. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S.-h. Lau, S. Rao, N. Taft, and J. D. Tygar, "Antidote: understanding and defending against poisoning of anomaly detectors," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, pp. 1–14, ACM, 2009.

[25] I. Evtimov, K. Eykholt, E. Fernandes, T. Kohno, B. Li, A. Prakash, A. Rahmati, and D. Song, "Robust physical-world attacks on machine learning models," *CoRR*, vol. abs/1707.08945, 2017.

[26] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," *arXiv preprint arXiv:1802.00420*, 2018.

[27] N. Papernot, P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against deep learning systems using adversarial examples," *CoRR*, vol. abs/1602.02697, 2016.

[28] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box adversarial attacks with limited queries and information," *arXiv preprint arXiv:1804.08598*, 2018.

[29] J. Kiefer and J. Wolfowitz, "Stochastic estimation of the maximum of a regression function," *Ann. Math. Statist.*, vol. 23, pp. 462–466, 09 1952.

[30] Y. LeCun and C. Cortes, "MNIST handwritten digit database," -, 2010.

[31] V. L. Jaromir Janisch, Tomáš Pevný, "Classification with costly features as a sequential decision-making." (under review), 2019.

[32] W. Suttle, Z. Yang, K. Zhang, Z. Wang, T. Basar, and J. Liu, "A multi-agent off-policy actor-critic algorithm for distributed reinforcement learning," *CoRR*, vol. abs/1903.06372, 2019.

[33] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[34] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: A survey," *J. Mach. Learn. Res.*, vol. 18, pp. 5595–5637, Jan. 2017.

[35] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016.

List of Figures

List of Tables