DeepCloak: Adversarial Crafting As a Defensive Measure to Cloak Processes

Mehmet Sinan İnci
Intel Corporation
mehmet.inci@intel.com

Thomas Eisenbarth
University of Lübeck
thomas.eisenbarth@uni-luebeck.de

Berk Sunar
Worcester Polytechnic Institute
sunar@wpi.edu

ABSTRACT

Over the past decade, side-channels have proven to be significant and practical threats to modern computing systems. Recent attacks have all exploited the underlying shared hardware. While practical, mounting such a complicated attack is still akin to listening on a private conversation in a crowded train station. The attacker has to either perform significant manual labor or use AI systems to automate the process. The recent academic literature points to the latter option. With the abundance of cheap computing power and the improvements made in AI, it is quite advantageous to automate such tasks. By using AI systems however, malicious parties also inherit their weaknesses, most notably the vulnerability to adversarial samples.

In this work, we propose the use of adversarial learning as a defensive tool to obfuscate and mask side-channel information. We demonstrate the viability of this approach by first training CNNs and other machine learning classifiers on leakage trace of different processes. After training a highly accurate model (99+% accuracy), we test it against adversarial learning. We show that through minimal perturbations to input traces, the defender can run as an attachment to the original process and cloak it against a malicious classifier.

Finally, we investigate if an attacker can use adversarial defense methods, adversarial re-training and defensive distillation to protect the model. Our results show that even in the presence of an intelligent adversary that employs such techniques, adversarial learning methods still manage to successfully craft perturbations hence the proposed cloaking methodology succeeds.

1 INTRODUCTION

Deep learning (DL) has proven to be a very powerful tool for a variety of tasks like handwritten digit recognition [26], image classification and labeling [25, 38, 66], speech recognition [64], lip reading [3], verbal reasoning [40], playing competitive video games [20, 53] and even writing novels [45]. As with any booming technology, it is also adopted by malicious actors e.g.,posting to internet boards to shift public opinion by social engineering. One of the latest known examples of this is the millions of AI-generated comments on the FCC net-neutrality boards [32, 59]. These AI-crafted posts were semantically sound and not easy to detect as fake. Another example of malicious use of AI is for spam and phishing attacks. It is now possible to craft custom phishing e-mails using AI with higher 'yield' than human crafted ones [15, 57]. AI system can even participate in hacking competitions where human creativity and intuition was thought to be irreplaceable. In 2016, DARPA sponsored a hacking competition for AI systems where the task was to find and fix vulnerabilities in computer systems within a given time period [33]. According to a survey among cybersecurity experts, the use of AI for cyber attacks will become more common with time [62].

Classical side-channel attacks (SCA) deal with bulk amounts of noisy data that require human interpretation and intuition to process. Such task are perfectly suited for AI systems and it is reasonable that malicious parties are aware of this opportunity. The academic literature already shows the use of AI for processing side-channel leakage. In 2011 Hospodar et al. [28] demonstrated the first use of machine learning, LS-SVM specifically, on a power SCA on AES and showed that the ML approach yields better results than the traditional template attacks. Later, Heuser et al. [27] showed the superiority of multi-class SVM for noisy data in comparison to the template attacks. Martinasek et al. [42, 43] showed that artificial neural networks can recover AES keys from power measurements with a success rate of 96%. In 2015 Beltramelli [4] used LSTM to collect meaningful keystroke data via motions of the smart watch user. In 2016, Maghrebi et al. [41] compared four DL based techniques with template attacks to attack an unprotected AES implementation using power consumption and showed that CNN outperforms template attacks. Finally in 2017, Gulmezoglu et al. [24] showed that machine learning can be used to extract meaningful information from cache side-channel leakage to recover web traffic of users.

A straightforward countermeasure against SCAs is to drown the sensitive computation leakage in noise to cloak it. However, this defense has proven to be ineffective in addition to being computationally expensive with significant performance overhead. In this study, we argue that we can do much better than random noise and craft much smaller noise by using adversarial learning (AL). By using AL, we achieve a stronger cloaking effect using smaller changes to the trace hence minimal overhead to the system. Also, the proposed defense does not require redesigning the software or the hardware stacks. The proposed framework can be deployed as an opt-in service that users can enable or disable at wish, depending on their privacy needs at the time.

In summary, attacking machine learning is easier than defending it [18] and if used strategically in a non-traditional way i.e., as a defensive countermeasure, AL against malicious parties with AI capabilities can be quite advantageous. In this work, we expand this idea and show that AL is indeed a useful defensive tool to cloak private processes from AI capable adversaries.

Our Contribution

In this work, we propose a framework and explore the necessary steps to cloak processes against SCAs as well as the defenses a malicious party can use. More specifically in this paper we;

- show how to profile crypto processes with high accuracy via their side-channel leakage using deep learning and various classical machine learning models. We classify 20 types of processes using readily available, high resolution Hardware Performance Counters (HPC). Further, we investigate the effect of parameter choices like the number of features, samples and data collection intervals on the accuracy of such classifiers.
- present the use of AL methods to craft perturbations and add them to the system hardware trace to cloak the side-channel leakage of private processes. We show that this is a strong defense against an attacker using DL classifiers.
- test and quantify the efficiency of different AL methods and present the accuracy and applicability of each attack.
- show that even when adversarial defense methods adversarial re-training or defensive distillation is employed by the attacker, adversarial perturbations still manage to cloak the process.

2 BACKGROUND

In this section, we provide the necessary background information to better understand the attack, adversarial sample crafting as a countermeasure and the improved attack. More specifically, we go over micro-architectural attacks, hardware performance counters, convolutional neural networks (CNNs), and AL attacks.

2.1 Micro-architectural Attacks

Over the last decade, there has been a surge of micro-architectural attacks. Low-level hardware bottlenecks and performance optimizations have shown to allow processes running on shared hardware to influence and retrieve information about one another. For instance, cache side-channel attacks like Prime&Probe and Flush+Reload exploit the cache and memory access time difference to recover fine-grain secret information and even recover secret crypto keys [5, 21–23, 30, 31, 36, 47, 54, 55, 65, 69]. In these works, the attacker exploits micro-architectural leakages stemming from memory access time variations, e.g.,when the data is retrieved from small but faster caches as opposed to slower DRAM memory.

2.2 Hardware Performance Counters

Hardware Performance Counters (HPCs) are special purpose registers that provide low-level execution metrics directly from the CPU. This low-level information is particularly useful during software development to detect and mitigate performance bottlenecks before deployment. For instance, low number of cache hits and high cache misses indicate an improperly ordered loop. By reordering some operations, a developer can significantly improve the performance.

HPCs are also useful to obtain system health check and/or anomaly detection in real-time. For instance, in [1] Alam et al. leverages *perf_event* API to detect micro-architectural side-channel attacks. In 2009, Lee et al. [67] showed that HPCs can be used on cloud systems to provide real-time side-channel attack detection. In [10, 11], researchers used HPCs to detect cache attacks. Moreover, Allaf et al. [2] used a neural network, decision tree, and kNN to specifically detect Flush+Reload and Prime&Probe attacks on AES. Moreover, researchers have shown that by using the fine-grain information provided by HPCs, it is possible to violate personal privacy as well. In [24], Gulmezoglu et al. showed that HPC traces can be used to reveal the visited websites in a system using variety of ML techniques such as auto-encoder, SVM, kNN and decision trees and works even on privacy conscious Tor browser. More applications of HPCs in cyber security countermeasures can be found in [17].

2.3 Convolutional Neural Networks

Convolutional Neural Networks (CNN) is a supervised feed-forward artificial neural network architecture. One important aspect of CNNs is that they don't saturate easily and can reach high accuracy with more training data. Also, unlike the classical ML methods, CNNs do not require data features to be identified and preprocessed before training. Instead, CNNs discover and learn relevant features in the data without human intervention, making them very suitable for automated tasks. In the past decade, CNNs surpassed humans in many tasks that were considered nearly impossible to automate. This breakthrough is fueled by the rapid increase in GPU powered parallel processing power and the advancements in deep learning. CNNs have been successfully applied to image, malware and many other classification problems. Training a CNN model is done in 3 phases. First, the labeled dataset is split into three parts; training, validation and test. The training data is fed to the CNN with initial hyper-parameters and the classification accuracy is measured using the validation data. Guided by the validation accuracy results, the hyper-parameters are updated to increase the accuracy of the model while maintaining its generality. After the model achieves the desired validation accuracy, it is tested with the test data and the final accuracy of the model is obtained.

2.4 Adversarial Learning

AL is a sub-field of machine learning (ML) that studies the robustness of trained models under adversarial settings. The problem stems from the underlying assumption that the training and the test data comes from the same source are consistent in their features. Studies have shown however that by introducing some small external noise or in this context what is commonly referred to as `adversarial perturbation`, it is possible to craft adversarial samples and manipulate the output of ML models. In other words, by carefully crafting small perturbations, one can push a test sample from the boundaries of one class to another. Due to the mathematical properties of the high-dimensional space that the classifier operates in, this modification can be very small. AL refers to the group of techniques that are used to perturb test samples to classifiers and force misclassification. While there are many different methods of crafting such perturbations, ideally they are desired to be minimal and not easily detectable.

Adversarial attacks on classical ML classifiers (under both white-box and black-box scenarios) have been known for quite some time [6–9, 29, 35, 39, 70]. However, it was Szegedy et al. [60] that first introduced AL attacks on DNNs. The 2013 study showed that very small perturbations that are indistinguishable to human eye can indeed fool CNN image classifiers like ImageNet. The perturbations in the study are calculated using the technique called Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS). This algorithm searches in the variable space to find parameter vectors (perturbation) that can successfully fool the classifier. Later in 2014, Goodfellow et al. [19] improved the attack by using the Fast Gradient Sign Method (FGSM) to efficiently craft minimally different adversarial samples. Unlike the L-BFGS method, the FGSM is computationally conservative and allows much faster perturbation crafting. In 2016, Papernot et al. [50] further improved upon Goodfellow's FGSM by using Jacobian Saliency Map Attack (JSMA) to craft adversarial samples. Unlike the previous attacks, JSMA does not modify randomly selected data points or pixels in an image. Instead, it finds the points of high importance with regards to the classifier decision and then modifies these specific pixels. These points are found by taking the Jacobian matrix of the loss function given a specific input sample, allowing an attacker to craft adversarial samples with fewer modifications.

In 2016 [34, 37, 48], multiple new adversarial attacks were discovered. Moreover, the research showed that these adversarial samples are transferable i.e., perturbations that can fool a model can also work on other models trained on the same task. In [49], Papernot et al. showed that adversarial attacks can also succeed under the black-box attack scenario where an attacker has only access to the classification labels. In this scenario, the attacker has no access to the model parameters such as weights, biases, classification confidence or the loss, therefore, cannot directly compute or use the gradients to craft a perturbation. Instead, the attacker uses the target model as an oracle that labels the inputs and then uses these labeled images to train her own classifier. Authors demonstrated the feasibility of the attack on MetaMind and Deep Neural Network (DNN) classifiers hosted by Amazon and Google. With 84.24%, 96.19% and 88.94% misclassification rates respectively, they were able to fool the targeted classifiers.

In [14], researchers have shown that by iteratively morphing a structured input, it is possible to craft adversarial samples under black-box attack scenario. Authors have implemented the attack against a PDF malware classifier and have reported 100% evasion rate. Moreover, the study acknowledges the fact that black-box attack model has a cost of obtaining labeled data from observations and defines and uses a cost function that takes into account the number of observations. The attack works by adding and/or removing compilable objects to the PDF. Black-box scenario does not assume to obtain confidence scores from the model under attack, only the class output. In summary, the AL is an active research area with plethora of new attacks, defenses and application cases emerging daily [12, 16, 44, 58, 63].

In addition to attack classifiers, adversarial learning have also been used to provide privacy for streaming traffic and facial recognition databases [46, 68]. In contrast to these works, DeepCloak uses adversarial learning to craft additional traffic on micro-architectural level to mask the overall side-channel leakage.

3 METHODOLOGY

Our goal is to show that side-channel classifiers can be successfully stopped using the concept of AL. To validate this assumption, we first train DL-based classifiers using real side-channel data, and show their degradation as the result of AL techniques, even if the DL-based classifier is aware of the AL based cloaking defense. In our experiments, we take the following steps:

(1) Training the process classifier $\mathbf{C}$ using side-channel leakage $\boldsymbol{\Omega}$.
(2) Crafting adversarial samples δ to cloak the user processes and force C to misclassify.
(3) Training a new classifier C' with adversarial defense methods; Defensive Distillation and Adversarial Re-training.
(4) Testing previously crafted adversarial samples δ against the new classifier C'. Also crafting and testing new adversarial samples δ' against the protected classifier C'.

We outline this methodology in Figure 1. In the first stage, Alice the defender runs a privacy sensitive process X. The eavesdropper Eve collects the side-channel leakage Ω and feeds it into her classifier C and discovers what type of process X is. Then in stage 2, Alice cloaks her process by crafting the adversarial sample δ. When faced with this adversarial sample, Eve's classifier C

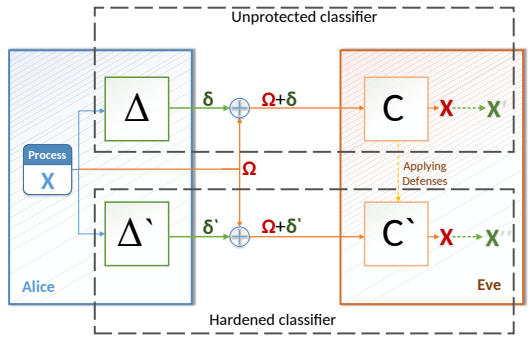

Figure 1: Alice, the defender runs the process X and leaks the Ω. Eve, the attacker obtains the leakage and identifies the process X using the classifier C. Then, Alice crafts the adversarial perturbation δ and forces C to misclassify the trace as X'. Eve then trains C' with adversarial re-training and defensive distillation. Now, Eve can classify δ'+Ω partially correct. However, when Alice crafts δ against C', X is again misclassified.

fails and misclassifies the leakage trace as X'. In the third stage, Eve trains a new classifier C' using *defensive distillation* and *adversarial re-training* to protect it from misclassification cause by the adversarial perturbation δ. In the final stage, Alice first tests previously crafted adversarial samples against Eve's protected classifier C'. Then, Alice updates her adversarial sample crafting target to fool C' rather than the original classifier C.

We apply this methodology to a scenario where a malicious party trains a CNN to classify running processes using the HPC trace as the input. This information is extremely useful to the attacker since it helps to choose a specific attack or pick a vulnerable target among others. Once a a target is found, an attacker can perform micro-architectural or application specific attacks. To circumvent this information leakage and protect processes, the defender attempts to mask the process signature. Ideally, the masking is minimal and does not interfere with the running process.

Specifically, in our methodology, the attacker periodically collects 5 HPC values over 10 msec total with 10 usec intervals, resulting in total of 5000 data points per trace. Later, trace is fed into classical ML and DL classifiers. In this section, we explain our choice of the specific HPCs, the application classifier design and implementation details, the AL attacks applied to these classifiers and finally test the efficiency of adversarial defenses against our cloaking method.

3.1 HPC Profiling

HPCs are special purpose registers that provide detailed information on low-level hardware events in computer systems. These counters periodically count specified event like cache accesses,branches, TLB misses and many others. This information is intended to be used by developers and system administrators to monitor and fine-tune performance of applications. The availability of a specific counter depends on the architecture and model of the CPU. Among many available HPCs, we have selected the following 5 for the classification task;

(1) **Total Instructions:** The total number of retired i.e., executed and completed CPU instructions.
(2) **Branch Instructions:** The number of branch instructions (both taken and not taken).
(3) **Total Cache References:** The total number of L1, L2, and L3 cache hits and misses.
(4) **L1 Instruction Cache Misses:** The occurrence of L1 cache instruction cache misses.
(5) **L1 Data Cache Misses:** The occurrence of L1 cache data cache misses.

We have selected these HPCs to cover a wide variety of hardware events with both coarse and fine-grain information. For instance, the `Total Instructions` does not directly provide any information about the type of the instructions being executed. However, different instructions execute in varying number of cycles even if the data is loaded from the same cache level. This execution time difference translates indirectly into the total instructions executed in the given time period and hints about the instruction being executed.

`Branch Instructions` HPC provides valuable information about the execution flow as well. Whether the branches are taken or not taken, the total number of branches in the executed program remains constant for a given execution path. This constant in the leakage trace helps eliminate noise elements and increases classification accuracy. The `Total Cache References` HPC provides similar information to the Branch Instructions HPC in the sense that it does not leak information about the finer details like the specific cache set or even the cache level. However it carries information regarding the total memory access trace of the program. Regardless of the data being loaded from the CPU cache or the memory, the total number of cache references will remain the same for a given process. The `L1 Instruction Cache Miss` and the `L1 Data Cache Miss` HPCs provide fine-grain information about the *Cold Start* misses on the L1 cache. Since the L1 cache is small, the data in this cache level is constantly replaced with new data, incrementing these counters. Moreover, separate counters for the instruction and the data misses allows the profiler to distinguish between arithmetic and memory intensive operations and increase accuracy. Finally, all five of the HPCs are interval counters meaning that they count specific hardware events within selected time periods.

3.2 Classifier Design and Implementation

In the first part of the study, we design and implement classifiers that can identify processes using the HPC leakage. To show the viability of such classifier, we chose 20 different ciphers from the OpenSSL 1.1.0 library as the classification target. Note that these classes include ciphers with both very similar and extremely different

performance traces e.g.,AES-128,ECDSAB571, ECDSAP521, RC2 and RC2-CBC. Moreover, we also trained models to detect the version of the OpenSSL library for a given cipher. For this task, we used OpenSSL versions 0.9.8, 1.0.0, 1.0.1, 1.0.2 and 1.1.0.

3.2.1 ***Classical ML Classifiers:*** In this study, we refer to non-neural network classification methods as classical ML classifiers. In order to compare and contrast classical ML methods with CNNs, we trained a number of different classifiers using the Matlab Classification Learning Toolbox. The trained classifiers include SVMs, decision trees, kNNs and variety of ensemble methods.

3.2.2 ***Deep Learning Classifier:*** We designed and implemented the CNN classifier using Keras with Tensorflow-GPU back-end. The model has the total of 12 layers including the normalization and the dropout layers. In the input layer, the first convolution layer, there are a total of 5000 neurons to accommodate the 10 msec of leakage data with 5000 HPC data points. Since the network is moderately deep but extremely wide, we used 2 convolution and 2 MaxPool layers to reduce the number dimensions and extract meaningful feature representations from the raw trace.

In addition to convolution and MaxPool layers, we used batch normalization layers to normalize the data from different HPC traces. This is a crucial step since the hardware leakage trace is heavily dependent on the system load and scales with overall performance. Due to this dependency, the average execution time of a process or parts of a process can vary from one execution to another. Moreover, in the system-wide leakage collection scenario, the model would train over this system load when it should be treated as noise. If not handled properly, the noise and shifts in the time domain results in over-fitting the training data with the dominant average execution time, decreasing the classification rate. By using the batch normalization layer, the model learns the features within short time intervals and the relation between different HPC traces. Finally, the output layer has 20 neurons with softmax activation, each representing a classes of process. To train the model, we use *Categorical Cross-entropy* loss function with the *Adam Optimizer*.

3.3 Adversarial Learning Attacks

AL remains an important open research problem in AI. Traditionally, AL is used to fool AI classifiers and test model robustness against malicious inputs. In this study however, we propose to use AL as a defensive tool to mask the side-channel trace of applications and protect against micro-architectural attacks and privacy violations. In the following, we explain the specific adversarial attacks that we have used. We consider the following 10 attacks:

- **Additive Gaussian Noise Attack (AGNA):** AGNA adds Gaussian Noise to the input trace to cause misclassification. The standard deviation of the noise is increased until the misclassification criteria is met. This AL attack method is ideal to be used in the cloaking defense due to the ease of implementation of the all-additive perturbations. A sister-process can actuate such additional changes in the side-channel trace by simply performing operations that increment specific counters like cache accesses or branch instructions.
- **Additive Uniform Noise Attack (AUNA):** AUNA adds uniform noise to the input trace. The standard deviation of the noise is increased until the misclassification criteria is met. Like AGNA, AUNA is easy to implement as a sister-process due to its additive property.
- **Blended Uniform Noise Attack (BUNA):** BUNA blends the input trace with Uniform Noise until the misclassification criteria is met.
- **Contrast Reduction Attack (CRA):** CRA formulates perturbations by reducing the 'contrast' of the input trace until a misclassification occurs. In case of the side-channel leakage trace, the attack smooths parts of the original trace and reduces the distance between the minimum and the maximum data points.
- **Gradient Attack (GA):** GA creates a perturbation with the loss gradient with regards to the input trace. The magnitude of the added gradient is increased until the misclassification criteria is met. The attack only works when the model has a gradient.
- **Gaussian Blur Attack (GBA):** GBA adds Gaussian Blur to the input trace until a misclassification occurs. Gaussian blur smooths the input trace and reduces the amplitude of outliers. Moreover, this method reduces the resolution of the trace and cloaks fine-grain leakage.
- **GSA (Gradient Sign Attack) [19]:** Also called the Fast Gradient Sign Method, the attack has been proposed by Goodfellow et al. in 2014. GSA works by adding the sign of the elements of the gradient of the cost function with regards to the input trace. The gradient sign is then multiplied with a small constant that is increased until a misclassification occurs.
- **L-BFGS-B Attack (LBFGSA) [51]:** LBFGSA attack utilizes the modified Broyden-Fletcher-Goldfarb-Shanno algorithm, an iterative method for solving unconstrained nonlinear optimization problems, to craft perturbations that have minimal distance to the original trace. The attack morphs the input to a specific class. However, in our experiments, we did not target a specific class and chose random classes as the target.
- **Saliency Map Attack (SMA) [52]:** SMA works by calculating the forward derivative of the model to build an adversarial saliency map to detect which input features e.g.,pixels in an image, have a stronger effect on the targeted misclassification. Using this information, an adversary can modify only the features with high impact on the output and produce smaller perturbations.
- **Salt and Pepper Noise Attack (SPNA):** SPNA works by adding Salt and Pepper noise (also called impulse noise) to the input trace until a misclassification occurs. For images, salt and pepper values correspond to white and black pixels respectively. For the side-channel leakage

trace however, these values correspond to the upper and the lower bounds in the trace.

3.4 Adversarial Learning Defenses

In order to see the viability of any defense method that can be used by an attacker against our adversarial perturbations, we have explored two methods: *adversarial re-training* and *defensive distillation* (DD). These defenses are an integral part of this study since an attacker capable of overcoming adversarial perturbation would deem any cloaking mechanism moot.

3.4.1 ***Gradient Masking:*** The term gradient masking defense has been introduced in [49] to represent group of defense methods against adversarial samples. The defense works by hiding the gradient information from the attacker to prevent it from crafting adversarial samples. Papernot et al. [49] however showed that the method fail under the oracle access scenario.An attacker can query the classifier with enough samples to create a cloned classifier. Since the clone and the original classifiers have correlated gradients, the attacker can use the gradient from the clone and craft adversarial samples, bypassing the defense. Due to the known weaknesses and limitations of this defense method, we do not further investigate it in this study.

3.4.2 ***AdversarialRe-training:*** This defense idea was first proposed by Szegedy et al. in 2013 [60]. Later in 2014, Goodfellow et al. [19] improved the practicality of the method by showing how to craft adversarial samples efficiently using the Fast Gradient Sign Method. In this defense, the model is re-trained using adversarial samples. By doing so, the model is 'vaccinated' against adversarial perturbations and can correctly classify them. In other words, the method aims to teach adversarial perturbations to the model so that it can generalize better and not be fooled by small perturbations. While this method works successfully against a specific type of attack, it has been shown to fail against attack methods that the model was not trained for. Nevertheless, we apply this defense method to our classifiers and investigate its applicability to side-channel leakage classifiers.

3.4.3 ***Defensive Distillation:*** The DD has been proposed by Papernot et al. [48] in 2016 to protect DL models against AL attacks. The goal of this technique is to increase the entropy of the prediction vector to protect the model from being easily fooled. The method works by pre-training a model with a custom output layer. Normally, the softmax temperature is set to be as small as possible to train a tightly fitted, highly accurate model. In the custom layer however, the temperature value is set to a higher value to distill the probability outputs. The first model is trained with the training data using hard labels i.e., the correct class label is set to '1' and all other class labels are set to '0'. After the model is trained, the training samples are fed into it and the probability outputs are recorded as *soft labels*. Then these soft labels are used to train the second, distilled model with the same training data. This process smooths the model surface on directions that an adversary would use to craft perturbations. This smoothing process increases the perturbation size required to craft an adversarial samples and invalidates some of the previously crafted adversarial samples. This smoothing can be set to different levels by adjusting the temperature value. Note that however, the DD can reduce the classification accuracy significantly if the temperature value is set too high.

4 EXPERIMENT SETUP AND RESULTS

In this section, we give details of our experiment setup, and the results of different adversarial attacks on the crypto-process classifier, and finally present the results of hardened adversarially re-trained and distilled models.

Experiment Setup: DL models perform large amounts of matrix multiplications that can run efficiently in modern GPU systems. For that reason, we used a workstation with two Nvidia 1080Ti (Pascal architecture) GPUs, 20-core Intel i7-7900X CPU and 64 GB of RAM. On the software side, the classifier model is coded using *Keras v2.1.3* with *Tensorflow-GPU v1.4.1* back-end and other Python3 packages such as *Numpy v1.14.0, Pandas, Sci-kit, H5py etc*.

The HPC data is collected from a server with Intel Xeon E5-2670 v2 CPU running Ubuntu 16 LTS. The CPU has 85 HPCs of which 50 are accessible from user-space. To access the HPCs, we had the choice of using *Perf* and *PAPI* libraries. Due to the lower sampling rate of Perf, we chose to use the PAPI with *QuickHPC* [13] front-end. QuickHPC is a tool developed by Marco Chiappetta to collect high-resolution HPC data using the PAPI back-end. It is over 30000 times faster than *perf-stat* and provides an easy to use interface.

4.1 Classification Results

The classifiers are trained to identify 20 classes representing a diverse set of different ciphers of five different versions of OpenSSL, as detailed in Section 3.2. For training, we split our dataset into three parts as training (%60), validation (%20) and test (%20).

4.1.1 ***CNN Classifier:*** For the CNN classifier, we firstly investigated the effect of the number of HPCs collected and trained our models for 100 epochs with data from a varying number of HPCs. Not surprisingly, even using a single HPC, our CNN classifier achieved 81% validation accuracy by training more epochs. Moreover, we noticed that after the 30th epoch, the model overfitted the training data i.e., the validation accuracy started to drop while the training accuracy kept increasing. When we increased the number of HPCs collected, our models became much more accurate and achieved over 99% validation accuracy as seen in Figure 3. Moreover, when we use the data from all 5 HPCs, our model achieved 99.8%

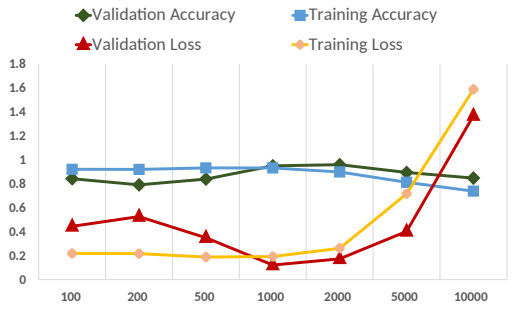

Figure 2: Results for the CNN classifier trained using varying number of features. Models reach highest validation accuracy with 1000 and 2000 features.

validation accuracy in less than 20 epochs. While our validation accuracy saturates even with only 2 HPCs, *Total Instructions* and *Branch Instructions* we have decided to use all 5 of them. We made this decision because in a real-world attack scenario, an attacker might be using any one or more of the HPCs. Since it would not be known which specific hardware event(s) an attacker would monitor, we decided to use all 5, monitoring different low level hardware events to provide a comprehensive cloaking coverage. To find the optimum number of features per HPC, we have trained multiple models with using various number of features. As shown in Figure 2, the validation accuracy saturates at 1000 and 2000 features, validation loss drops after 1000 features. For this reason, we chose to use 1000 features for our experiments.

Further, we investigated how the number of training samples affect the validation accuracy. For that, we have trained 6 models with a varying number of training samples. For the first model, we have used only 100 samples per class (2000 samples in total) and later on trained models with 300, 1000, 3000, 10000 and 30000 samples per class. In the first model, we achieved 99.8% validation accuracy after 40 epochs of training. When we trained models with more data, we have reached similar accuracy levels in much fewer epochs. To make a good trade-off between the dataset size and training time, we have opted to use 1000 samples per class. This model reached 100% accuracy with 20 epochs of training as shown in Figure 4. Finally, our last model achieved 100% accuracy just after 4 epochs when trained with 30000 samples per class.

We also show that in addition to detecting the process type, we can also distinguish between different versions of OpenSSL. For each of the 20 analyzed ciphers, we built classifiers to identify the library version. Figure 5 presents the classification results of two models trained using 1 and 5 HPC traces respectively. As cipher updates between versions can be very small, the added information from sampling several HPCs is essential for high classification rates, as shown in the results.

Table 1: Application classification results for the classical ML classifiers with and without PCA feature reduction.

Classification Method	Without PCA	With PCA (99.5% variance)
Fine Tree	98.7	99.9
Medium Tree	85.4	94.8
Coarse Tree	24.9	25
Linear Discriminant	99.6	99.7
Quadratic Discriminant	N/A	99.4
Linear SVM	99.9	98.2
Quadratic SVM	99.9	96.9
Cubic SVM	99.9	94.3
Fine Gaussian SVM	40	88.2
Medium Gaussian SVM	98.3	92.1
Coarse Gaussian SVM	99.7	13.4
Fine kNN	96.8	11.1
Medium kNN	94.9	7.8
Coarse kNN	85.5	5.2
Cosine kNN	92.5	19.6
Cubic kNN	85.2	7.7
Weighted kNN	95.9	8.3
Boosted Trees	99.2	99.8
Bagged Trees	99.9	94.8
Subspace Discriminant	99.8	99.7
Subspace kNN	84.8	88.1
RUSBoosted Trees	76	92.8
Best	**99.9**	**99.9**

4.1.2 ***ClassicalML Methods:*** In our training of ML classifiers, the first challenge was the fact that the side-channel leakage data is extremely wide. We have chosen to train our models using 1000 data points per HPC with 5 HPCs monitored simultaneously. This parameter selection is done empirically to provide wide cloaking coverage and train highly accurate models as explained in Section 4.1.1. Using 1000 data points with 10 usec intervals per HPC allowed us to obtain high quality data in a short observation window. Nevertheless, 5000 dimensions is unusually high for classifiers, especially considering that we are training multi-class classifiers with 20 possible classes.

In order to find optimal settings for the hardware leakage trace, we tried different parameters with each classifier. For instance in the case of decision trees, we have trained the 'Fine Tree', 'Medium Tree' and 'Coarse Tree' classifiers. The difference between these classifiers is that respectively they allow 5, 20, and 100 splits (leaves in the decision tree) to better distinguish between classes.For the case of Gaussian SVM, fine, medium and coarse refers to the kernel scale set to sqrt(P)/4, sqrt(P) and sqrt(P)*4 respectively. As for the kNN, the parameters refer to the number of neighbors and the different distance metrics.

Results for the classical ML classifiers are given in Table 1. Classic ML algorithms achieve very high success rates for the given classification task. The trained models can in fact classify running processes by using their HPC traces. Note that the Quadratic Discriminant did not converge to a solution without the PCA application hence no score is given in the table.

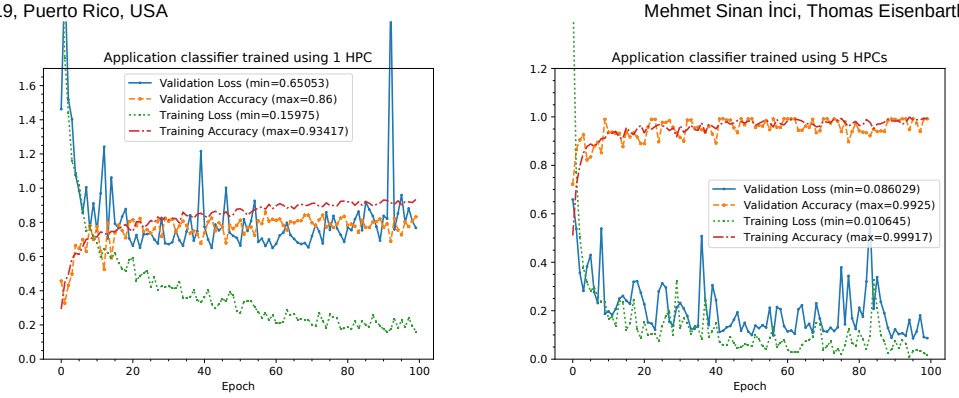

Figure 3: Classification accuracy of models trained using 1 and 5 HPCs. Using data from a single HPC trace is enough to obtain high accuracy top-1 classification rates, albeit taking longer to train.

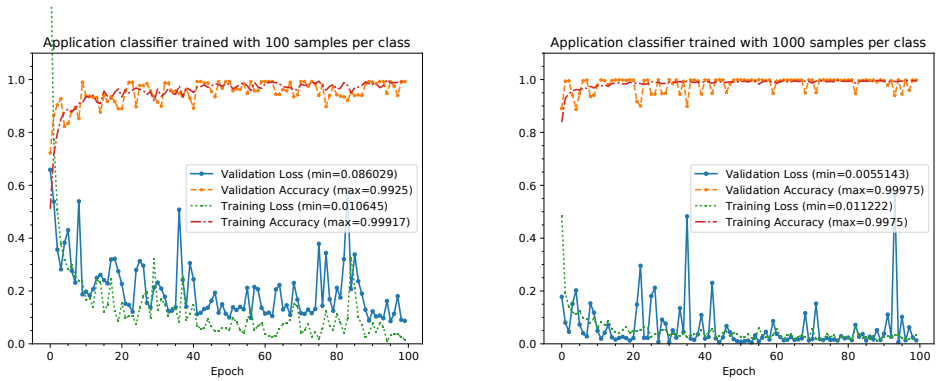

Figure 4: Classification accuracy of models trained using 100 and 1000 samples per class. Both models reach 99% accuracy in 20 and 40 epochs of training respectively.

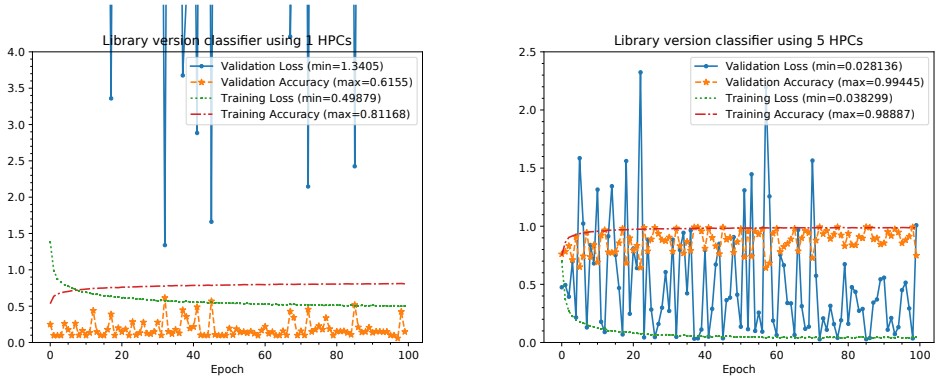

Figure 5: Library version classifier accuracy for models trained using 1 and 5 HPCs. The former model saturate at 61% while the latter reach 99%.

4.2 AL on the Unprotected Model

Next, we crafted adversarial perturbations for the unprotected classifiers by using and adapting the publicly available Foolbox [56] library to our scenario. The library provides numerous adversarial attacks and provides an easy to use API. For a selected attack, Foolbox crafts necessary perturbations on a given sample and classifier model pair to 'fool' the given model. Detailed information about these attacks can be found in Section 3.3.

Table 3 presents the classification accuracy of perturbed samples. As the results show, almost all test samples are misclassified by the classifier model with

**Table 2: Effectiveness of adversarial re-training and DD on 100,000 previously crafted adversarial samples. Th
show what percentage of previously successful adversarial samples are ineffective on the hardened models.**

Adversarial Attack	Adversarial Re-training	DD with T=1	DD with T=2	DD with T=5	DD with T=10	DD with T=20	DD with T=30	DD with T=40	DD with T=50	DD with T=100
AGNA	42	77	60	70	83	83	63	64	62	75
AUNA	43	77	60	70	83	82	63	65	61	75
BUNA	94	92	92	91	94	94	94	96	94	100
CRA	94	95	99	94	94	94	94	99	88	95
GA	97	99	72	83	99	99	96	80	90	90
GBA	84	83	84	88	82	94	93	91	93	88
GSA	99	91	90	91	99	99	99	95	99	98
LBFGSA	51	76	63	63	78	87	65	65	63	71
SMA	26	71	50	52	62	82	49	47	32	48
SPNA	4	84	76	78	94	93	76	79	73	80

**Table 3: Perturbation results against both the unprotected and the hardened (Adversarial Re-training) CNN cla
The new adversarial samples have up to 29% lower misclassification confidence compared to the unprotected m
However, adversarial samples are still misclassified with quite high confidence values in the range of 63-98%**

	Unprotected Classifier			Hardened Classifier (Adv. Re-training)		
Adversarial Attack	**Original Sample Classification Confidence**	**Adversarial MisClassification Confidence**	**Perturbation Size (MAD)**	**Original Sample Classification Confidence**	**Adversarial MisClassification Confidence**	**Perturbation Size (MAD)**
AGNA	99	96	0.00294	92	82	0.00294
AUNA	99	97	0.00292	91	82	0.00332
BUNA	99	99	0.05000	96	93	0.05000
CRA	99	99	0.05254	97	98	0.04999
GA	99	99	0.00250	88	97	0.00398
GBA	99	97	0.00080	93	74	0.00071
GSA	99	99	0.00499	89	97	0.00596
LBFGSA	99	86	0.00025	89	72	0.00031
SMA	99	92	0.00001	88	63	0.00008
SPNA	99	96	0.01528	92	74	0.08268

very high accuracy at over 86%. Another important metric for the adversarial learning is the Mean Absolute Distance (MAD) and the Mean Squared Distance (MSD) of the perturbed traces from the originals. These metrics quantify the size of the changes i.e., perturbations made to the original traces by various adversarial attack methods. The difference between the MAD and the MSD is that, the latter is more sensitive to the larger changes due to the square operation. For instance, if an adversarial perturbation requires a significant change in 1 sample point among the 5000 features, it will have a stronger impact in the final MSD value than average change distributed over few points. MAD however is more dependent on the overall change in the trace, i.e., all 5000 sample points have the same impact on the final distance. Our results show that with most adversarial attacks, perturbation MAD is around or well below 1% and within the ideal range.

4.3 AL on the Hardened Models

Here we present the results of the AL attacks on the hardened classifier models. As explained in Section 3, we wanted to test the robustness of our cloaking mechanism against classifiers hardened with Adversarial Re-training and DD. To recap the scenario, Alice the defender wants to cloak her process by adding perturbations to her execution trace so that eavesdropper Eve cannot correctly classify what Alice is running. Then Eve notices or predicts the use of adversarial perturbations on the data and hardens her classifier model against AL attacks using adversarial re-training and DD.

In order to test the attack scenario on hardened models, we first craft 100,000 adversarial samples per adversarial attack against the unprotected classifier. Then we harden the classifier with the aforementioned defense methods and feed the adversarial samples. Here, we aim to measure the level of protection provided by the adversarial re-training and the DD methods. As presented in Table 2, the application of both the adversarial retraining and the DD invalidates some portion of the previously crafted adversarial samples. For the adversarial re-training, the success rate varies between 99% (GSA) and 4% (SPNA). In other words, 99% of the adversarial samples crafted using GSA against the unprotected model are invalid on the hardened model. As for the DD, we see similar rates of protection ranging from 61% up to 100% for old perturbations. Impressively, 100% of the adversarial samples crafted using the BUNA are now ineffective against the model trained with DD at temperature T=100. In short, by using the adversarial re-training or the DD, Eve can indeed harden her

Table 4: MAD sizes of adversarial perturbations crafted against the DD applied classifier. Application of the ad re-training and DD only marginally increases the perturbation size needed to fool the classifier in most cases.

Adversarial Attack	Unprotected Model	Adversarial Re-trained	DD with T=1	DD with T=2	DD with T=5	DD with T=10	DD with T=20	DD with T=30	DD with T=40	DD with T=50	DD with T=100
AGNA	0.00294	0.00294	0.00035	0.00265	0.00062	0.01059	0.01389	0.00452	0.00766	0.01797	0.00431
AUNA	0.00292	0.00332	0.00033	0.00238	0.00071	0.01114	0.01336	0.00514	0.00948	0.01787	0.00451
BUNA	0.05000	0.05000	0.04989	0.05301	0.08198	0.10293	7.52282	0.05006	0.05002	0.16376	0.07748
CRA	0.05254	0.04999	0.04999	0.10247	0.08548	0.10780	NA	0.04998	1.44158	0.27992	0.07448
GA	0.00250	0.00398	0.00355	0.00283	0.10224	0.00783	0.00364	0.00860	2.75805	0.00849	0.00303
GBA	0.00080	0.00071	0.00058	0.00092	0.00060	0.04992	NA	0.00169	0.00046	0.00062	0.00083
GSA	0.00499	0.00596	0.00504	0.00534	0.13665	0.02482	0.00540	0.01064	0.04121	0.01670	0.00550
LBFGSA	0.00025	0.00031	0.00533	0.00178	0.00024	0.00210	0.07670	0.00088	0.02872	0.00521	0.00920
SMA	0.00001	0.00008	0.00003	0.00012	0.00007	NA	NA	0.00001	0.00001	NA	NA
SPNA	0.01528	0.08268	0.01060	0.00940	0.00360	0.01804	0.02000	0.00260	0.02000	0.02000	0.01980

classifier against AL. However, keep in mind that Alice can observe or predict this behavior and introduce new adversarial samples targeting the hardened models.

4.3.1 ***AdversarialRe-training:*** After training the DL classifier model and crafting adversarial samples, we use these perturbations as training data and re-train the classifier. The motivation here is to teach the classifier model to detect these perturbed samples and correctly classify them. With this re-training stage, we expect to see whether we can 'immunize' the classifier model against given adversarial attacks. However, as the results in Table 3 show, all of the adversarial attacks still succeed albeit requiring marginally larger perturbations. Moreover, while we observe a drop in the misclassification confidence, it is still quite high at over 63% i.e., Eve's classifier can be fooled by adversarial samples.

4.3.2 ***Defensive Distillation:*** We have used the technique proposed in [51] and trained hardened models with DD at various temperatures ranging from 1 to 100. Our results show that, even if the eavesdropper Eve hardens her model with DD, the trained model is still prone to adversarial attacks albeit requiring larger perturbations in some cases. In Table 4, we present the MAD i.e., the perturbation size, of various attack methods on both unprotected and hardened models. Our results show that the application of DD indeed offers a certain level of hardening to the model and increases the perturbation sizes. However this behavior is erratic compared to the adversarial re-training defense i.e., the MAD is significantly higher at some temperatures while much smaller for others. For instance, the MAD for the AUNA perturbations against the unprotected model is 0.00292 in average for 100,000 adversarial samples. The perturbation size for the same attack drops to 0.00033 when the distillation defense is applied with temperature T=1. This in turn practically makes Eve's classifier model easier to fool. Same behavior is observed with the adversarial samples crafted using AGNA and GBA as well. For the cases that the DD actually hardens Eve's model against adversarial samples, the MAD is still minimal. Hence, Alice can still successfully craft adversarial samples with minimal perturbations and fool Eve's model. Finally, NA values in Table 4 represent cases where the model had very low classification accuracy and could not correctly classify original samples.

5 CONCLUSION

Side-channel leakage on shared hardware systems pose a real and present danger to the security and the privacy of users. Even when the software is perfectly isolated, co-resident tenants still share the underlying hardware and are prone to side-channel attacks. Especially considering the wide adoption of AI across many disciplines, it is not surprising that such attacks will become automated and even easier to perform in the future. There is a clear need for users to cloak their execution fingerprints from the underlying shared system.

With this work we took a first step in this direction. Specifically, by making clever defensive use of adversarial crafting we introduced a new cloaking defense against the side-channel leakage classifiers. We first demonstrated the threat side-channel leakage poses by processing leakage profiles to yield highly accurate AI models which may be used by an adversary to violate privacy and security policies of applications. We trained various types of classifiers including the classical ML methods and showed how the parameter selection affects the learning rate and the validation accuracy. While this is a strong threat to shared hardware systems, we showed that it can be mitigated using carefully crafted adversarial samples. Moreover, we investigated defenses that can potentially help an attack to bypass the adversarial samples. Our results show that even in the presence of defensive distillation and adversarial re-training, the defender can craft working adversarial samples and fool the attacker. These perturbations can be implemented as a sister-process that will run side-by-side with the original and easily cause misclassification to the attacker's model without any significant overhead. Using the adversarial crafting-based cloaking mechanism that we have outlined in this work, users can enable such services on-demand for sensitive operations. Efficient design and implementation of such defenses for shared hardware systems like cloud remains an open research problem. Finally, to the best of our knowledge, this work is the first use of adversarial crafting for defensive purposes. We envision the same approach to be useful in other application scenarios.

REFERENCES

[1] Alam, M., Bhattacharya, S., Mukhopadhyay, D., and Bhattacharya, S. Performance counters to rescue: A machine learning based safeguard against micro-architectural side-channel-attacks. Cryptology ePrint Archive, Report 2017/564, 2017. https://eprint.iacr.org/2017/564.

[2] Allaf, Z., Adda, M., and Gegov, A. A comparison study on flush+reload and prime+probe attacks on aes using machine learning approaches. In *Advances in Computational Intelligence Systems* (Cham, 2018), F. Chao, S. Schockaert, and Q. Zhang, Eds., Springer International Publishing, pp. 203–213.

[3] Assael, Y. M., Shillingford, B., Whiteson, S., and de Freitas, N. Lipnet: Sentence-level lipreading. *arXiv preprint arXiv:1611.01599* (2016).

[4] Beltramelli, T., and Risi, S. Deep-spying: Spying using smartwatch and deep learning. *arXiv preprint arXiv:1512.05616* (2015).

[5] Bernstein, D. J. Cache-timing attacks on AES, 2004. URL: `http://cr.yp.to/papers.html#cachetiming`

[6] Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G., and Roli, F. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases* (2013), Springer, pp. 387–402.

[7] Biggio, B., Fumera, G., and Roli, F. Security evaluation of pattern classifiers under attack. *IEEE transactions on knowledge and data engineering 26*, 4 (2014), 984–996.

[8] Biggio, B., Nelson, B., and Laskov, P. Support vector machines under adversarial label noise. In *Asian Conference on Machine Learning* (2011), pp. 97–112.

[9] Biggio, B., Nelson, B., and Laskov, P. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389* (2012).

[10] Briongos, S., Irazoqui, G., Malagón, P., and Eisenbarth, T. Cacheshield: Protecting legacy processes against cache attacks. *arXiv preprint arXiv:1709.01795* (2017).

[11] Briongos, S., Irazoqui, G., Malagón, P., and Eisenbarth, T. Cacheshield: Detecting cache attacks through self-observation. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy* (New York, NY, USA, 2018), CODASPY '18, ACM, pp. 224–235.

[12] Carlini, N., and Wagner, D. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on* (2017), IEEE, pp. 39–57.

[13] Chiappetta, M. Quickhpc. https://github.com/chpmrc/quickhpc, 2015.

[14] Dang, H., Huang, Y., and Chang, E.-C. Evading classifiers by morphing in the dark. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (2017), ACM, pp. 119–133.

[15] Emmanuel, Z. Security experts air concerns over hackers using AI and machine learning for phishing attacks. https://www.computerweekly.com/news/450427653/Security-experts-air-concerns-over-hackers-using-AI-and-machine-learning-for-phishing-atttacks/.

[16] Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Song, D., Kohno, T., Rahmati, A., Prakash, A., and Tramer, F. Note on attacking object detectors with adversarial stickers. *arXiv preprint arXiv:1712.08062* (2017).

[17] Foreman, J. C. A survey of cyber security countermeasures using hardware performance counters. *arXiv preprint arXiv:1807.10868* (2018).

[18] Goodfellow, I., and Papernot, N. Is attacking machine learning easier than defending it? http://www.cleverhans.io/security/privacy/ml/2017/02/15/why-attacking-machine-learning-is-easier-than-defending-it.html.

[19] Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[20] Gordon, R. AI beats pros at Super Smash Bros. http://www.csail.mit.edu/ai_beats_pros_at_super_smash_bros. Accessed: 2017-10-27.

[21] Gruss, D., Maurice, C., Wagner, K., and Mangard, S. Flush+ flush: a fast and stealthy cache attack. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* (2016), Springer, pp. 279–299.

[22] Gruss, D., Spreitzer, R., and Mangard, S. Cache template attacks: automating attacks on inclusive last-level caches. In *Proceedings of the 24th USENIX Conference on Security Symposium* (2015), USENIX Association, pp. 897–912.

[23] Gullasch, D., Bangerter, E., and Krenn, S. Cache games–bringing access-based cache attacks on AES to practice. In *Security and Privacy (SP), 2011 IEEE Symposium on* (2011), IEEE.

[24] Gülmezoglu, B., Zankl, A., Eisenbarth, T., and Sunar, B. PerfWeb: How to Violate Web Privacy with Hardware Performance Events. In *Computer Security - ESORICS 2017 - 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part II* (2017), pp. 80–97.

[25] Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A., Venugopalan, S., Widner, K., Madams, T., Cuadros, J., et al. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *Jama 316*, 22 (2016).

[26] Hadad, Y. 30 amazing applications of deep learning. http://www.yaronhadad.com/deep-learning-most-amazing-applications/, Aug 2017. Accessed: 2017-10-27.

[27] Heuser, A., and Zohner, M. Intelligent machine homicide. *COSADE 7275* (2012), 249–264.

[28] Hospodar, G., Gierlichs, B., De Mulder, E., Verbauwhede, I., and Vandewalle, J. Machine learning in side-channel analysis: a first study. *Journal of Cryptographic Engineering 1*, 4 (2011), 293.

[29] Huang, L., Joseph, A. D., Nelson, B., Rubinstein, B. I., and Tygar, J. Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence* (2011), ACM, pp. 43–58.

[30] Inci, M. S., Gulmezoglu, B., Irazoqui, G., Eisenbarth, T., and Sunar, B. Cache attacks enable bulk key recovery on the cloud. In *International Conference on Cryptographic Hardware and Embedded Systems—CHES* (2016), Springer Berlin Heidelberg, pp. 368–388.

[31] Irazoqui, G., Inci, M. S., Eisenbarth, T., and Sunar, B. Wait a minute! a fast, cross-vm attack on aes. In *International Workshop on Recent Advances in Intrusion Detection* (2014), Springer, pp. 299–319.

[32] Jon Brodkin . 2 million people-and some dead ones-were impersonated in net neutrality comments. https://arstechnica.com/tech-policy/2017/12/dead-people-among-millions-impersonated-in-fake-net-neutrality-comments/.

[33] Keren Elazari . Hackers are on the brink of launching a wave of AI attacks. http://www.wired.co.uk/article/hackers-ai-cyberattack-offensive.

[34] Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236* (2016).

[35] Laskov, P., and Lippmann, R. Machine learning in adversarial environments, 2010.

[36] Lipp, M., Gruss, D., Spreitzer, R., Maurice, C., and Mangard, S. Armageddon: Cache attacks on mobile devices. In *25th USENIX Security Symposium (USENIX Security 16)* (2016), USENIX Association, pp. 235–252.

[37] Liu, Y., Chen, X., Liu, C., and Song, D. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770* (2016).

[38] Liu, Y., Gadepalli, K., Norouzi, M., Dahl, G. E., Kohlberger, T., Boyko, A., Venugopalan, S., Timofeev, A., Nelson, P. Q., Corrado, G. S., et al. Detecting cancer metastases on gigapixel pathology images. *arXiv preprint arXiv:1703.02442* (2017).

[39] Lowd, D., and Meek, C. Adversarial learning. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining* (2005), ACM, pp. 641–647.

[40] MacDonald, F. A Deep Learning Machine Just Beat Humans in an IQ Test. https://www.sciencealert.com/a-deep-learning-machine-just-beat-humans-in-an-iq-test.

[41] Maghrebi, H., Portigliatti, T., and Prouff, E. Breaking cryptographic implementations using deep learning techniques. In *International Conference on Security, Privacy, and Applied Cryptography Engineering* (2016), Springer, pp. 3–26.

[42] Martinasek, Z., Hajny, J., and Malina, L. Optimization of power analysis using neural network. In *International Conference on Smart Card Research and Advanced Applications* (2013), Springer, pp. 94–107.

[43] Martinasek, Z., and Zeman, V. Innovative method of the power analysis. *Radioengineering 22*, 2 (2013), 586–594.

[44] Meng, D., and Chen, H. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (2017), ACM, pp. 135–147.

[45] O'Brien, E. Romance novels, generated by artificial intelligence. https://medium.com/towards-data-science/romance-novels-generated-by-artificial-intelligence-1b31d9c872b2, Aug 2017.

[46] Oh, S. J., Fritz, M., and Schiele, B. Adversarial image perturbation for privacy protection a game theory perspective. In *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), IEEE, pp. 1491–1500.

[47] Osvik, D. A., Shamir, A., and Tromer, E. Cache attacks and countermeasures: the case of AES. In *Cryptographers Track at the RSA Conference* (2006), Springer, pp. 1–20.

[48] Papernot, N., McDaniel, P., and Goodfellow, I. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277* (2016).

[49] Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security* (2017), ACM, pp. 506–519.

[50] Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on* (2016), IEEE, pp. 372–387.

[51] Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (SP), 2016 IEEE Symposium on* (2016), IEEE, pp. 582–597.

[52] Papernot, N., McDaniel, P. D., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. The limitations of deep learning in adversarial settings. *CoRR abs/1511.07528* (2015).

[53] Peng, P., Yuan, Q., Wen, Y., Yang, Y., Tang, Z., Long, H., and Wang, J. Multiagent Bidirectionally-Coordinated Nets for Learning to Play StarCraft Combat Games. *arXiv preprint arXiv:1703.10069* (2017).

[54] Percival, C. Cache missing for fun and profit, 2005.

[55] Pessl, P., Gruss, D., Maurice, C., Schwarz, M., and Mangard, S. Drama: Exploiting dram addressing for cross-cpu attacks. In *USENIX Security Symposium* (2016), pp. 565–581.

[56] Rauber, J., Brendel, W., and Bethge, M. Foolbox v0.8.0: A python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131* (2017).

[57] Simonite, T. This AI Will Craft Tweets That Youll Never Know Are Spam. https://www.technologyreview.com/s/602109/this-ai-will-craft-tweets-that-youll-never-know-are-spam/.

[58] Su, J., Vargas, D. V., and Kouichi, S. One pixel attack for fooling deep neural networks. *arXiv preprint arXiv:1710.08864* (2017).

[59] Susan Decker . FCC Rules Out Delaying Net Neutrality Repeal Over Fake Comments. https://www.bloomberg.com/news/articles/2018-01-05/fcc-rules-out-delaying-net-neutrality-repeal-over-fake-comments.

[60] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).

[61] Tabacof, P., and Valle, E. Exploring the space of adversarial images. *CoRR abs/1510.05328* (2015).

[62] The Cylance Team . Black Hat Attendees See AI as Double-Edged Sword. https://threatmatrix.cylance.com/en_us/home/black-hat-attendees-see-ai-as-double-edged-sword.html.

[63] Tramèr, F., Kurakin, A., Papernot, N., Boneh, D., and McDaniel, P. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204* (2017).

[64] Xiong, W., Droppo, J., Huang, X., Seide, F., Seltzer, M., Stolcke, A., Yu, D., and Zweig, G. Achieving human parity in conversational speech recognition. *arXiv preprint arXiv:1610.05256* (2016).

[65] Yarom, Y., and Falkner, K. FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack. In *(USENIX Security 2014)* (2014).

[66] Yu, K.-H., Zhang, C., Berry, G. J., Altman, R. B., Ré, C., Rubin, D. L., and Snyder, M. Predicting non-small cell lung cancer prognosis by fully automated microscopic pathology image features. *Nature communications 7* (2016).

[67] Zhang, T., Zhang, Y., and Lee, R. B. Cloudradar: A real-time side-channel attack detection system in clouds. In *International Symposium on Research in Attacks, Intrusions, and Defenses* (2016), Springer.

[68] Zhang, X., Hamm, J., Reiter, M. K., and Zhang, Y. Statistical privacy for streaming traffic. In *NDSS* (2019).

[69] Zhang, Y., Juels, A., Reiter, M. K., and Ristenpart, T. Cross-tenant side-channel attacks in PaaS clouds. In *CCS* (2014), pp. 990–1003.

[70] Zhou, Y., Kantarcioglu, M., Thuraisingham, B., and Xi, B. Adversarial support vector machine learning. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining* (2012), ACM, pp. 1059–1067.