Adversarial Attacks against Network Intrusion Detection in IoT Systems

Han Qiu, *Member, IEEE,* Tian Dong, Tianwei Zhang, Jialiang Lu, Gerard Memmi, *Member, IEEE,* and Meikang Qiu, *Senior Member, IEEE*

***Abstract*—Deep Learning (DL) has gained popularity in network intrusion detection, due to its strong capability of recognizing subtle differences between normal and malicious network activities. Although a variety of methods have been designed to leverage DL models for security protection, whether these systems are vulnerable to adversarial examples is unknown.**

In this paper, we design a novel adversarial attack against DL-based Network Intrusion Detection Systems (NIDS) in the IoT environment, with only black-box accesses to the DL model in such NIDS. We introduce two techniques: (1) model extraction is adopted to replicate the black-box model with a small amount of training data; (2) a saliency map is then used to disclose the impact of each packet attribute on the detection results, and the most critical features. This enables us to efficiently generate adversarial examples using conventional methods. With these tehniques, we successfully compromise one state-of-the-art NIDS, Kitsune: the adversary only needs to modify less than 0.005% of bytes in the malicious packets to achieve an average 94.31% attack success rate.

***Index Terms*—Adversarial examples, network intrusion detection, IoT, deep learning.**

I. Introduction

The increased number and severity of cyber-attacks against modern networks highlight the urgent need for effective and efficient protection methodologies. Network Intrusion Detection Systems (NIDS) [1] have become practical to detect malicious network activities and guard critical but vulnerable services and systems. A NIDS is generally deployed as a gatekeeper for Internet devices to monitor network traffic and generate alerts when anomalous or suspicious events are identified. It plays an important role in defeating cyber-attacks against enterprises, individuals and governments [2], [3], for various computer systems (e.g., cloud computing [4], Internet of Things (IoT) [5], edge computing [6], service computing [7], etc.). It is anticipated that billions of US dollars will be invested to enhance the intrusion detection and protect the network environment in 2021 [8].

Early NIDSs employed the *misuse* detection to identify network attacks [9]. They compared the signature (e.g., a sequence of bytes in the payload) of the network traffic with a pre-established threat dataset. Such detection methods are then proved to be ineffective in today's complex environment due to three reasons. First, network packets are normally encrypted for confidentiality protection, which can prevent the detectors from introspecting into the contents of the packets. Second, these methods require the knowledge of the attack signatures. They are incapable of recognizing emerging zero-day threats [10], which become more common in modern systems. Third, it is computationally inefficient to collect the signature of the target traffic and compare it with a large-scale threat dataset, especially for the resource-constrained devices such as IoT [11].

An alternative strategy is *anomaly* detection, which establishes models to characterize the normal behaviors of network traffic, and identifies any deviations from such models as the evidence of network intrusion. This strategy can dominate the misuse detection as it can detect unknown attacks which have distinct behaviors from the normal ones. Particularly, recent advance in Deep Learning (DL) facilitates the development of anomaly detection techniques, as state-of-the-art DL models can distinguish normal and abnormal network activities with higher accuracy, and less restriction [12], [13]. As such, this direction of solutions is attracting more attention, and showing more potential in network security.

We want to explore one question: *do these DL-based NIDSs have fundamental vulnerabilities, that enable an adversary to efficiently invalidate the detection mechanism?* This question is raised from the observation that DL models are well-known to be vulnerable to Adversarial Examples (AE) [14], where imperceptible perturbation injected to the input sample can cause the DL model to make wrong decisions. Although DL models have satisfactory performance in terms of automation, speed, and possibly accuracy, they can make simple mistakes that humans will never do. Over the past years, adversarial attacks have been successfully introduced in various domains (e.g., computer vision [15], natural language processing [16], speech recognition [17], reinforcement learning [18]), and it is extremely hard to comprehensively mitigate these threats [19]. So we aim to study whether such adversarial attacks can threaten the scenario of DL-based intrusion detection as well.

Designing such attacks is not easy, due to the huge differences between the NIDS and other conventional DL tasks. First, a network packet to be classified has many attributes (e.g., header information, size, sent and arrival timestamp, etc.). These attributes may have different impacts on the detection, and it is unknown which attributes should be selected to perturb for the optimal attack results. Second,

H. Qiu and G. Memmi are with Telecom Paris, Palaiseau, France, 91120. Email: {han.qiu, gerard.memmi}@telecom-paris.fr.

T. Zhang is with Nanyang Technological University, Singapore, 639798. Email: tianwei.zhang@ntu.edu.sg.

T. Dong and J. Lu are with SPEIT, Shanghai Jiao Tong University, Shanghai, China, 200240. Email: dongtian9702@gmail.com, jialiang.lu@sjtu.edu.cn.

M. Qiu (corresponding author) is with Texas A&M University Commerce, TX, USA, 75428. Email: meikang.qiu@tamuc.edu

Manuscript received xxx.

NIDSs usually have very complex input preprocessing and feature extraction procedures, which will mix up all the packet attributes to generate the input vectors for classification. So it is challenging to modify the network attributes to produce the desired vectors, which can lead the model misclassification. There are a few works attempting to solve this question. [10] heuristically searches for the adversarial perturbations in the network packets. [20] only considers the perturbations in the feature vector space. [21] adopted complex and time-consuming algorithms like GAN for AE generation. Those solutions are not practical or efficient for a real-time attack, especially in the IoT context.

In this paper, we design a more efficient and effective adversarial attack against DL-based NIDS. By considering a realistic threat model where the DL model employed by the target NIDS is a black-box to the adversary, we have the two main contributions. (1) We propose to use the model extraction technique [22] to replicate the model for AE generation. This technique only requires a small amount of data (10% of the original training set), which achieves very high efficiency. (2) to copy with the diverse attributes and complex feature extraction procedure, we propose to utilize saliency maps [23] to identify the critical features that impact the detection results, and then use conventional AE methods (e.g., FGSM [14]) to generate the desired perturbations.

We implement our method to attack one state-of-the-art NIDS, Kitsune [24]. We demonstrate two attack scenarios. (1) In an IoT botnet attack (Mirai [25]), an adversary can modify the malicious packets to bypass the detector; (2) In a video streaming application, an adversary can alter the normal traffic flows to make the detector classify them as malicious, and generate unexpected false alarms. Our evaluations indicate that the adversary can slightly modify the network packets (e.g., padding, delaying arrival timestamp, dropping certain packets) to significantly decrease the prediction accuracy of the NIDS.

The remainder of this paper is organized as follows. The research background about NIDS is described in Section II. The threat model and attack goal is provided in Section III. We present our proposed attack method in Section IV, followed by two case studies in Sections V and VI as attack evaluation. We conclude in Section VII.

II. Background about NIDS

A. Learning-based NIDS

In recent years, artificial intelligence technology (e.g., Deep Neural Networks) has been utilized in the design of NIDS to improve the detection accuracy and efficiency. These learning-based NIDS can be mainly classified into two categories. The first one is *misuse-based* NIDS, which was adopted in earlier works [26]. As illustrated in Fig. 1 (a), the basic idea of this solution is to predict whether a network packet is malicious or benign based on its contents. A labeled dataset is first constructed with contents from normal packets as well as malicious packets. Then a DNN model is trained over this dataset, which can classify the new network packets from their payloads. This solution significantly improves the performance over traditional misuse detection methods, attributed to the great capability of DNN models. However, it also has some limitations. Inspecting the payloads of all the packets at runtime is computationally intensive. Besides, this solution does not work when the network packets are encrypted.

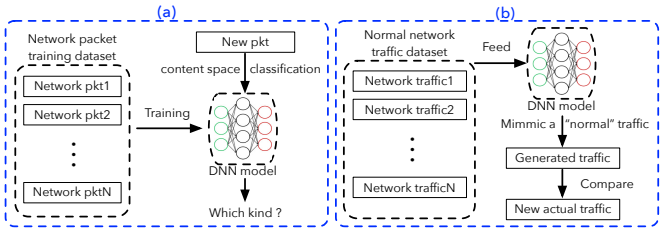

Fig. 1. Development of DNN-based NIDS from (a) misuse detection (e.g. [26]) to (b) anomaly detection (e.g. [24]).

The second category is *anomaly-based* NIDS [24], which can overcome the above issues. As shown in Fig. 1 (b), this solution extracts features from the traffic space, e.g., latency, packet drop rate, etc. A DNN (e.g., Auto-Encoder) is adopted to model the behaviors of normal network traffic. At runtime, the patterns of incoming traffic are analyzed and compared with the normal model. A large deviation indicates a higher possibility of anomaly, and alert will be generated.

The anomaly-based strategy dominates the misuse-based one as the offline model training requires smaller amounts of traffic data, and the online inference is more efficient. This is particularly important in the resource-constrained contexts, e.g., IoT environment. Besides, anomaly-based solution is able to detect zero-day threats, especially the attacks relying on the traffic volumes, e.g., DoS, port scanning, brute force, video injection, and Botnet [21]. So in this paper, we will focus on the attacks against anomaly-based NIDS.

B. Kitsune: an Intrusion Detection System for IoT Networks

As an example, we review the mechanism of a state-of-the-art NIDS: Kitsune [24], which will be the target of our designed adversarial attack in this paper. Kitsune adopts unsupervised learning to identify intrusions in the IoT networks. The system is composed of 5 modules: Packet Capturer, Packet Parser, Feature Extractor (FE), Feature Mapper (FM), and Anomaly Detector (AD), as shown in Fig. 2.

Packet Capturer: this module intercepts and captures the packets from the network. Kitsune adopts multiple external libraries (e.g., NFQueue, tshark) to achieve this goal.

Packet Parser: this module extracts relevant information from packets, e.g., source and the destination IP addresses, MAC address and port, packet size and packet timestamp.

Feature Extractor (FE): this module is used to calculate the temporal statistics of the packets within a stream to achieve high-speed-low-space feature extraction. It introduces "damped incremental statistics" with the time complexity of $O(1)$. Specifically, for a stream of network packets $P_i = \{P_i^1, P_i^2, \dots\}$, the corresponding packet sizes are $S_i = \{x_i^1, x_i^2, \dots\}$. Then the statistic of S_i is defined as a tuple $IS_i = (w, LS, SS, SR_{ij})$, where w is the packet count, LS is the linear sum of the sizes, SS is the squared sum of the sizes, SR_{ij} is the sum of the residual products between

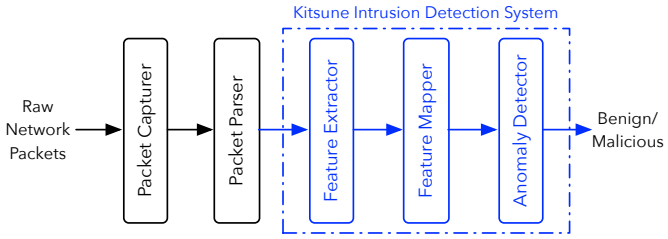

Fig. 2. System overview of Kitsune.

S_i and another stream S_j. This statistic is updated in a "damped incremental" way. Assume the last update occurs at the timestamp T_{last}, and there is a new incoming packet belonging to P_i at the timestamp T_{cur} with the size of x_{cur}. The update rule follows Equation 1:

$$\begin{aligned} w &\longleftarrow \gamma w + 1 \\ LS &\longleftarrow \gamma LS + x_{cur} \\ SS &\longleftarrow \gamma SS + x_{cur}^2 \\ SR_{ij} &\longleftarrow \gamma SR_{ij} + r_i r_j \end{aligned} \tag{1}$$

where $\gamma = 2^{-\lambda(T_{cur}-T_{last})}$ is the decay factor; $r_i = x_{cur} - \mu_i$ is the residual of x_{cur}, and r_j is the most recent residual in stream S_j. With this tuple, we can further calculate the incremental statistics, as shown in Equation 2.

$$\begin{aligned} \text{Weight:} \quad & w = w \\ \text{Mean:} \quad & \mu_{S_i} = LS/w \\ \text{Std:} \quad & \sigma_{S_i} = \sqrt{(|SS/w - (LS/w)^2|)} \\ \text{Magnitude:} \quad & \|S_i, S_j\| = \sqrt{\mu_{S_i}^2 + \mu_{S_j}^2} \\ \text{Radius:} \quad & R_{S_i,S_j} = \sqrt{(\sigma_{S_i}^2)^2 + (\sigma_{S_j}^2)^2} \\ \text{Covariance:} \quad & Cov_{S_i,S_j} = SR_{ij}/(w_i + w_j) \\ \text{Coefficient:} \quad & P_{S_i,S_j} = Cov_{S_i,S_j}/(\sigma_{S_i}\sigma_{S_j}) \end{aligned} \tag{2}$$

The feature set of a network packet P^* is extracted from three streams: (1) all the packets with the same source MAC address and source IP address as P^* (`SrcMAC+SrcIP`); (2) all the packets with the same source IP address and destination IP address as P^* (`SrcIP+DstIP`); (3) all the packets with the same source MAC address and destination MAC address for ARP packets (`SrcMAC+DstMAC`), or source IP address and destination IP address with the same port number for other types of packets (`SrcIPPort+DstIPPort`). TABLE I shows the corresponding statistics collected from those streams. For each λ, there are 20 statistics. We consider $\lambda = 5, 3, 1, 0.1, 0.01$ as in [24] to produce a set of 100 features as the input to the Feature Mapper module, which will map them into clusters for Anomaly Detector.

Feature Mapper (FM): this module clusters the n features into k clusters. Kitsune utilizes the agglomerative hierarchical clustering algorithm based on the correlation distance matrix $D = 1 - C$. Here C is a correlation matrix of n features, computed by the Monte-Carlo method on the received instances.

Anomaly Detector (AD): this module contains two components. The first one is an ensemble layer, where k three-layer Auto-Encoders are integrated to learn the normal behaviors of each cluster in the output of FM. The second component is an output layer, using k normalized RMSE from the Auto-Encoders of the ensemble Layer to produce a global RMSE s. The detection decision is thus made based on s. The traffic stream is flagged as malicious if $s > \beta\phi$, where ϕ is the highest global RMSE obtained during model training, and β is a hyperparameter used to adjust the trade-off between False Negative (FN) errors and False Positive (FP) errors.

In summary, this Kitsune NIDS is one of the state-of-the-art anomaly-based NIDSs. Moreover, the Kitsune system is based on the Auto-Encoder system which does not require the pre-trained models. Therefore, Kitsune can be used in the IoT system in a plug and play fashion due to its high efficiency.

TABLE I
STATISTICS EXTRACTED FROM A NEW ARRIVAL PACKET FOR A GIVEN λ.

Streams	Statistics	# of statistics
`SrcMAC+SrcIP`	w, μ_i, σ_i	3
`SrcIP+DstIP`	$w, \mu_i, \sigma_i, \|S_i, S_j\|$ $R_{S_i,S_j}, Cov_{S_i,S_j}, P_{S_i,S_j}$	7
`SrcIP+DstIP`	w, μ_i, σ_i	3
`SrcMAC+DstMAC` or `SrcIPPort+DstIPPort`	$w, \mu_i, \sigma_i, \|S_i, S_j\|$ $R_{S_i,S_j}, Cov_{S_i,S_j}, P_{S_i,S_j}$	7

III. Attack Goals and Threat Model

A. Attack Goals

DNN models are vulnerable to Adversarial Examples (AEs), where slight changes in the input samples can totally alter the prediction results of the target model. This threat has been intensively studied in computer vision [27], [28], which crafts the malicious samples by adding carefully designed and imperceptible perturbations to the image pixels to mislead a image classification model.

In the context of network systems, we consider a learning-based NIDS (e.g., Kitsune) which detects anomaly via traffic analysis. This is realized by a DL model $f(\cdot)$, which classifies a packet P as normal (label 0) or malicious (label 1) based on the attributes x of this packet: $y = f(x)$. An adversary tries to compromise the DNN model adopted by this NIDS, making it predict wrong results. He achieves this by slightly adjusting the attributes of the packet as $\widetilde{x} = x + \delta$. For instance, he can also delay the packet transmission to change its arrival timestamp. He can pad the packet to increase its size.

With these changes, the adversary can cause two different consequences: (1) he can modify malicious packets to bypass the detection mechanism, while the modified packets can still bring the same damages to the protected system; (2) he can also modify normal packets such that the NIDS will predict them as malicious and block them. This can lead to severe denial-of-service attacks, as the victim system will get a lot of false alarms, and all benign network packets will be denied. For these two cases, the attack process can be formulated as an optimization problem as shown in Equation 3:

$$min\|\delta\|, s.t.\ f(\widetilde{x}) \neq f(x) \tag{3}$$

B. Threat Model

There are two attack scenarios [27] determined by how much the adversary knows about the target DNN system. (1) *White-box scenario*: the adversary knows every detail about the neural network model including the architecture and all the parameters. The AE can then be generated easily by calculating the optimization problem based on the knowledge of the DNN model. (2) *Black-box scenario*: the adversary does not have any knowledge about the victim DNN model. Instead, he can first generate a local shadow model with the same behaviors as the target one, by training a model with the original training set, or a constructed set by querying the black-box model. Then he can generate AEs from this shadow model following the approach in the white-box scenario. Such AEs can be used to attack the victim system with high success rate, due to the transferability feature of DNN models [29]. This black-box scenario is more realistic, but more challenging. We will consider this setting in this paper.

Specifically, in our black-box setting, we assume that the adversary only knows the mechanism deployed in this NIDS system, which is commonly public. Such mechanism includes which kind of DL structure and the mechanism of the FE module. However, he does not know the detailed parameters and hyper-parameters of the DL model. In addition, the parameters for input preprocessing (e.g., the FE and FM modules in Kitsune) are kept secret from the adversary as well.

We assume the integrity of the NIDS is protected so the adversary cannot modify the DNN model or alter the detection results. We also assume that the target model is well-trained with satisfactory accuracy, and does not contain any DNN backdoors [30]. Some NIDSs require the history data of the network traffic flows to train the model (e.g., Kitsune), which is assumed to be correct without poisoned samples. Following the threat model in [24], the adversary can install a malicious network device in the same network, and has the capability of passively monitoring the traffic flows, and actively perturbing the traffic features (e.g., timestamp, packet size) to compromise the NIDS.

IV. Attack Method

This section presents our proposed method to perform adversarial attacks against a learning-based NIDS. The whole attack process consists of two phases. We start with the method overview (Section IV-A), followed by the description of each phase in Sections IV-B and IV-C, respectively.

A. Overview

Fig. 3 illustrates the overview of our black-box adversarial attack against a NIDS. Two phases are involved to complete this attack. The first phase is to extract the DNN model employed by the target NIDS. The adversary needs to reconstruct both the Feature Extractor and Anomaly Detector modules. The second phase is to generate AEs from the extracted shadow model. Our approach builds a saliency map from the packets, which guarantees the adversary can find the optimal features with the least modifications to alter the detection results with high success rates, while preserving the impacts

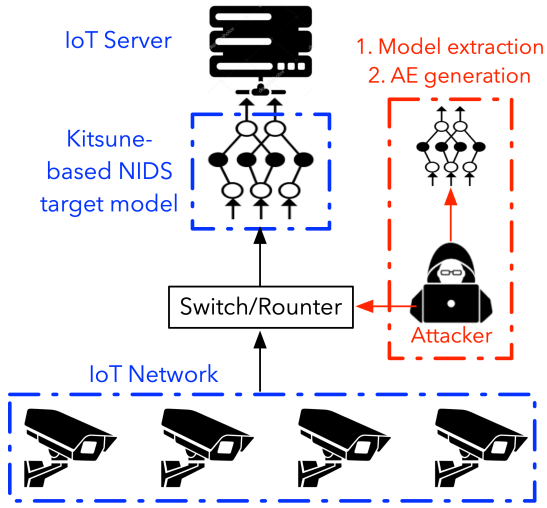

Fig. 3. Overview of the our adversarial attack against a Kitsune-based NIDS.

of the original traffic flow to the target system (the same attack damage, or innocence). Then the adversary can use gradient-based methods to generate AEs over the identified features, and use them to compromise the target victim.

B. Phase 1: Model Extraction

This phase is to reconstruct a shadow model with the same behaviors as the target model. Since the entire NIDS is a black-box to the adversary, he needs to (1) first rebuild the Feature Extractor module used by the system, and then (2) collect a number of traffic packets to train a new model.

We assume the mechanism of the Feature Extractor is public in our threat model (e.g., Section II-B for Kitsune). So the adversary can follow the design to implement this module. Given a stream of packets, this module can produce the same feature vectors as the target system. Since the adversary has the capability of installing his own device (e.g., a switcher or router) within the same network, he can passively monitor the traffic flow to or from the target system, and collect certain amounts of packets without interrupting the victim. He then uses this replicated module to generate the corresponding features for extracting the AD module.

In Kitsune, AD is composed of an encoder and a decoder. The encoder is used to encode an input vector x into an internal representation with a smaller dimension, which will be further decoded by the decoder into an output x' with he same size as x. The adversary knows the network structures of the encoder and decoder, but not the parameters. He can adopt the same structures, and train new models over the extracted features from the replicated FE module. The training goal will be minimizing the Root Mean Square Error (RMSE) between the input feature x and the output of the decoder x'. This shadow model will have the same behaviors as the victim one in the NIDS: it can recover the traffic data for normal network flows, but not for the malicious ones. So a large RMSE from this model will indicate the anomaly of the input packet. It is worth noting that the adversary can just use a much smaller number of packets to extract the model than the original

training set, which maintains enough information about the original model to generate AEs with high transferability. This can significantly reduce the attack cost.

C. Phase 2: AE Generation from Saliency Map

Given the extracted Auto-Enconder model, the next phase is to generate adversarial examples from it and attack the victim system. Achieving this goal is challenging due to the following two reasons. First, different from image classification systems, a NIDS adopts complicated processing operations to extract the features from the attributes of the packets (e.g., packet size, arrival time). How to modify these attribute values to produce the desired features that can fool the model is difficult. Second, an image classification task usually has multiple classes, and it is easy to generate the optimal perturbation that moves the normal data point across the decision boundary. In contrast, a NIDS system only has two labels (benign and malicious), and the malicious points in the feature space are usually far away from the decision boundary. This also increases the difficulty of AE generation. Due to the distinct features between images and network packets, traditional AE generation methods in computer vision cannot be directly applied to this scenario.

To overcome the above challenges, we propose to leverage the saliency map [23] to identify the critical elements in the traffic feature, that determines the detection results. Then we use a gradient-based optimization method to generate AEs by perturbing those critical elements.

We first investigate the impact of each feature from a network packet on the anomaly score produced by AD. The input of AD is a 100-dimensional feature vector $\mathbf{x} = [x_i]_{0 \le i \le 99}$. The anomaly score is denoted as $s = AD(\mathbf{x})$ with the softmax probabilities of benign class ($p_0 = 1 - \texttt{sigmoid}(s - T)$) and malicious class ($p_1 = \texttt{sigmoid}(s - T)$), where T is the threshold. The saliency map of AD on $\mathbf{x}$ is calculated as

$$[\texttt{relu}(\frac{\partial p_c}{\partial x_i})]_{0 \le i \le 99}$$

where $\texttt{relu} : x \mapsto \max(0, x)$ and $c \in \{0, 1\}$ is the corresponding class of the input. Then the critical feature is defined as the one *whose softmax value of the opposite class is most sensitive to the feature perturbations*, i.e., the predicted labels can be flipped with the minimal perturbation.

We randomly sample 10 benign feature vectors (i.e., anomaly score is below the threshold T) as well as 10 malicious feature vectors (i.e., anomaly scoire is above T). Fig. 4 (a) and Fig. 4 (b) show the corresponding saliency maps for the benign and malicious cases, respectively. We observe for malicious vectors, the *49th* and *56th* features have the biggest influence on the value of p_0. Thus, it is possible to reduce the probability of being classified as malicious packets by increasing the values of these features. Note that the *49th* feature represents the statistic "Covariance" between two streams. To increase this feature value, we can vary the size of datagrams by appending redundant bits at the end. The *56th* feature is only dependent on the timestamp of the packet. So we can directly modify its timestamp to mislead the classifier.

To identify the necessary amount of changes on the critical features, we adopt an iterative version of the Fast Gradient Sign Method (FGSM) [14], as shown in Algorithm 1. It iteratively changes the value of the critical feature until the classification result is altered. Then the adversary can use this AE to attack the target NIDS. Due to the high transferability, this AE can mislead the black-box model with high success rates, as we will show in the following two case studies.

ALGORITHM 1: Iterative FGSM

Require:
PE module: f_{FE}
AD modules: f_{AD}
Perturbation step: ϵ
Maximum number of iterations: M
Target threshold T
Critical attribute q that correlates to the critical feature x
Original label l
for $i = 0; i < M; i++$ **do**
  $x = f_{FE}(q)$ // generate the feature vector
  $RMSE = f_{AD}(x)$ // generate the anomaly score
  if $l = 1$ and $RMSE < T$ **then**
    return q
  end if
  if $l = 0$ and $RMSE > T$ **then**
    return q
  end if
  $q \longleftarrow q - \epsilon sign(\frac{\partial (-1)^{l+1} RMSE}{\partial q})$
end for

V. Case Study 1: Evading Intrusion Detection

As the first case, we consider a scenario, where the adversary adds perturbations to malicious network packets to bypass the intrusion detection. We select the Kitsune NIDS as the target, and use the Mirai botnet [25] as the network attack.

A. Experimental Setup and Configuration

The Mirai dataset [25] contains $764,137$ packets from a local network. Among them, $122,660$ packets are normal, while the rest are infected by the "Mirai" malware. Such malware will first use ARP scanning to locate the potential vulnerable devices. Then, there devices will be injected with a malware which can turn network devices into controlled bots and leverage them to launch DDoS attacks against the target IoT devices (e.g., cameras, home routers). These malicious packets can be further classified into two types: packets for ARP scanning of victims, and packets for DDoS attacks.

We re-implement a Kitsune NIDS using Pytorch 1.4.0. We follow the same training process in [24] to generate the system. Specifically, the clustering function in the FM module is trained from the first $5,000$ benign packets in the Mirai dataset, while the Auto-Encoder in the AD module is trained from the next $50,000$ packets with the SGD optimizer and a learning rate of 0.1. The maximal input size of the Auto-Encoder is set as $m = 10$. The entire system is then evaluated with the rest packets in the Mirai dataset. Fig. 5 shows the validation results, where the x-axis denotes the index of each packet, and y-axis is the anomaly score (RMSE). We can observe that malicious packets (ARP scanning, DDoS) have much larger RMSE values than the normal ones before the attack starts.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2020.3048038, IEEE Internet of Things Journal

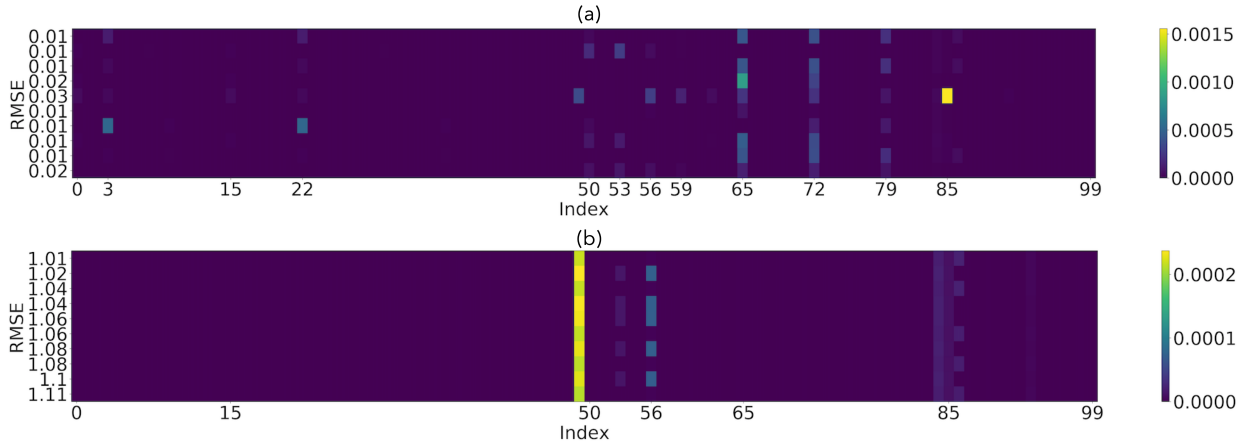

Fig. 4. An example of the saliency maps for 10 benign samples (a) and 10 malicious samples (b) in Kitsune system. In each figure, each row is a sample with the anomaly score (RMSE) shown on the left, and each column is a feature. The colors denote the values of the maps.

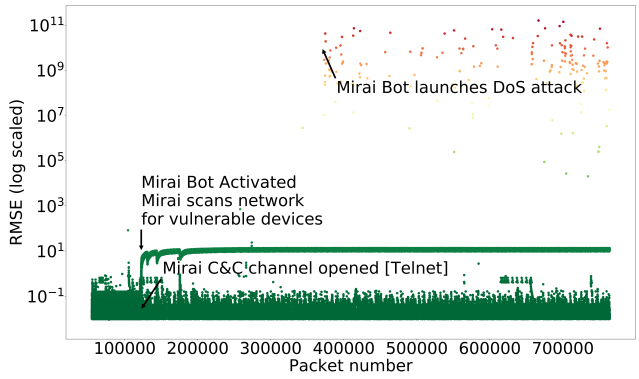

Fig. 5. Anomaly scores from our implemented Kitsune NIDS.

We modify the AD module to predict the packets based on their RMSE values. For one packet with a RMSE of s, the softmax probability for benign and malicious classes are $p_0 = 1 - \texttt{sigmoid}(S-T)$, $p_1 = \texttt{sigmoid}(S-T)$, where T is the threshold for classification. Then the label of this packet will be assigned based on the larger probability.

The threshold T is critical to demtermine the performance of the NIDS. We select different threshold values $T \in [0, 0.01, 0.03, 0.05, 0.1, 0.25, 1, 5, 7, 10, 15, 20]$, and measure the detection performance, as shown in Fig. 6. The x-axis is the threshold value, and y-axis shows the False Positive Rate (FPR: the percentage of normal packets classified as malicious ones), False Negative Rate (FNR: the percentage of malicious packets classified as normal ones), and accuracy (the percentage of packets correctly classified). We can observe that the system can achieve good performance when T is in the range of $[0.1, 10]$: the FPR is close to zero; the FNR is around 10% due to the mislabeling: after the Mirari attack starts, there are still around 10% normal network flows mixed with the attack flows, which are classified as malicious as well. When T is smaller than 0.1, FPR increases significantly as a lot of benign packets will be classified as malicious. When T is larger than 10, Kitsune will miss more malicious packets with an increased FNR, since the RMSE values of most malicious packets are in the order of 10 (Fig. 5). As such, we select $T = 1$ to achieve very satisfactory performance.

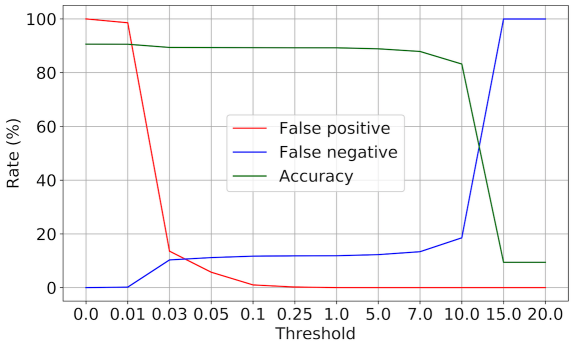

Fig. 6. Detection performance of Mirai under different thresholds.

B. Evaluation Results

The first attack phase is to extract the black-box NIDS, as described in Section IV-B. We find that the adversary can use a much smaller amount of data than the original training set to reproduce the shadow models, whose quality is good enough for further adversarial attacks. In our experiment, we choose a ratio $p_l = 10\%$ of the training data to train the FM model (i.e., 500 benign packets) and the AD model (i.e., 5000 benign packets), randomly sampled from the Mirai set. The adversary sets the threshold of the extracted model as $T = \phi\frac{1}{p_l}$ where ϕ is the maximum loss value during training.

The second phase is to generate AEs from the extracted model. As described in Section V-A, the Mirai attack generates two kinds of malicious packets: ARP request packets to search for victim hosts and DNS query packets for DoS attacks. We modify these packets to evade the detection, respectively.

For the DoS attack packets, we modify their lengths to generate AEs. For a packet with the size of s, we use the method in Section IV-C to calculate the corresponding perturbation Δs. We set $M = 1000$ and $\epsilon = 1$. We iteratively compute the perturbation with FGSM until the RMSE value of the modified packet is lower than the threshold T.

It is more difficult to perturb the ARP packets, since their sizes are not changeable. Instead, we modify their arrival timestamps for adversarial attacks. Changes in the timestamps can disrupt the order of the network packets, and the computation of AEs can slow down the packet sending speed in a real-time attack. So we apply a uniform timestamp change (Δt) to relevant ARP request packets. Specifically, we first compute the timestamp perturbations for the first 100 ARP request packets using Algorithm 1, i.e. $(\Delta t_i)_{1\leq i\leq 100}$. Then, we calculate a timestamp interval $\Delta t = \frac{1}{100}\sum_i \Delta t_i$, as the upper bound of the perturbation under the L_∞ norm. Assume the timestamp of the first ARP packet is $t_{0,scan}$. Then for the i-th relevant packet, we change the timestamp to $t_{0,scan} + (i-1) \times \Delta t$. We abandon the modified ARP packets whose timestamps are larger than the maximum timestamps in the original dataset. After such processing and reordering, there are $219,820$ packets left, among which $6,650$ is relevant to the network scan packets. We generate the perturbation via FGSM, with $M = 1000$ and $\epsilon = 0.1$.

TABLE II
RESULT OF ADVERSARIAL ATTACKS

Malicious Packets	Modified attributes	L_∞	ASR (%)
DoS attack	packet size	7 bits	95.1
ARP scan	timestamp	0.398 s	100

TABLE II demonstrates the Attack Success Rate (ASR) of the two attacks, which is defined as the ratio of the modified malicious packets that are misclassified as normal (i.e., the RMSE is lower than the threshold T). We observe that the adversary can achieve a success rate higher than 95%. This attack effect is obtained under the black-box scenario, as our extracted model can perfectly mimic the behaviors of the target model, and the generated AE has very high transferability.

TABLE III shows the computation time of generating the adversarial packets for two attacks. From the Mirari dataset, for the DoS case, the average time interval between two successive DoS packets is 11.24s. In contrast, the delay caused by the AE generation (0.171s) has little influence on the operations of the target NIDS. For the ARP scanning case, the average computation time of generating AEs is 0.52s, which is also negligible.

TABLE III
COMPUTATION COST OF GENERATING ADVERSARIAL EXAMPLES.

Attack	Mean time	Variance	Max
DoS attack	0.171s	0.058s	0.600s
ARP scan	0.052s	0.001s	0.110s

VI. CASE STUDY 2: INCREASING FALSE ALARMS

In this case, we show that the adversary can modify normal packets to make the NIDS treat them as security threats. This can incur a lot of unexpected false positives, and the protected service will deny all the normal packets.

A. Experimental Setup and Configuration

We adopt the VideoInjection dataset [24] which consists of $2,472,401$ network packets for the camera surveillance scenario. The adversary attempts to inject recorded video clips into the stream by an ARP spoofing attack. We consider the Kitsune system with the same setup in Section V. the first $100,000$ packets are used to train the FE model, and the following $1,000,000$ packets are used to train the AD model. Fig. 7 shows the detection accuracy with different thresholds. We select $T = 0.04$ to achieve the best performance.

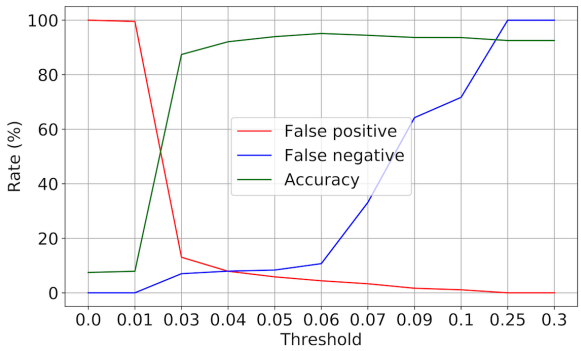

Fig. 7. Detection performance of VideoInjection under different thresholds.

B. Evaluation Results

The goal of this attack is to perturb the normal packets, such that the output of the Auto-Encoder in the AD module will be significantly deviated from normal distributions, and the NIDS will treat them as malicious packets. The adversary can repeat the attack procedure as described in Section IV on normal samples to craft the corresponding adversarial perturbations. First, he randomly samples 10 vectors generated by the FE module from the normal traffic stream, and computes the saliency map over them. Based on this saliency map, he is able to identify the critical features (i.e., the $50th$, $53rd$ and $56th$) that have the largest impact on the final anomaly score. These features are closely correlated to the timestamp attribute of the packets. Then the adversary can change the transmission interval between these packets as the adversarial examples. He can achieve this by randomly dropping UDP packets with a probability of p, Then the average time interval between two successive packets will be changed from Δt to $\frac{1}{1-p}\Delta t$.

Fig. VI-B shows the effects of packet dropping on the detection. We consider different dropping rates: $p \in \{0.1, 0.3, 0.5, 0.7\}$. The x-axis is the packet indexes within the entire video stream, and the y-axis denotes the corresponding RMSE score. The packet dropping occurs during the period of index 1.5 million and 1.8 million. The light blue horizontal line represents the threshold $T = 0.04$, which can achieve the best FPR and FNR under normal circumstances. We observe that during the packet dropping period, the RMSE scores are increased due to our attack. When $p \geq 0.3$, a lot of normal packets have RMSE values higher than the threshold, and will be flagged as malicious. There will be more such false alarms when p is higher: for $p \geq 0.5$, almost all the packets will be classified as "malicious". When the detection threshold is lower (e.g., $T = 0.01$), it will be easier to trigger such attack with a lower packet dropping rate (e.g., $p = 0.1$).

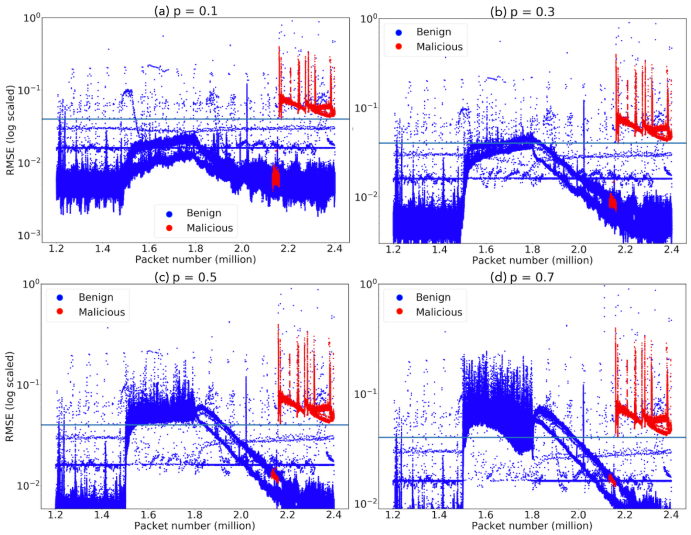

Fig. 8. Anomaly scores for packet dropping rate $p = 0.1, 0.3, 0.5$, and 0.7.

VII. Conclusion

In this paper, we design a new method to generate adversarial network packets, and invalidate modern DL-based NIDSs. Our method leverages the model extraction technique, enabling an efficient attack even when DL models are black-boxes to the adversaries. We also utilize the saliency map to identify the critical features and packet attributes as the target for AE generation. Evaluations show that our solution can successfully attack the state-of-the-art NIDS, Kitsune, by significantly increasing its false positives and false negatives in the scenario of Mirai Botnet and video streaming. In the future, we will focus on the investigation of mitigation solutions to enhance the robustness of DL models in intrusion detection.

References

[1] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *computers & security*, vol. 28, no. 1-2, pp. 18–28, 2009.

[2] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for iot security based on learning techniques," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2671–2701, 2019.

[3] D. E. Whitehead, K. Owens, D. Gammel, and J. Smith, "Ukraine cyber-induced power outage: Analysis and practical mitigation strategies," in *2017 70th Annual Conference for Protective Relay Engineers (CPRE)*. IEEE, 2017, pp. 1–8.

[4] N. Keegan, S.-Y. Ji, A. Chaudhary, C. Concolato, B. Yu, and D. H. Jeong, "A survey of cloud-based network intrusion detection analysis," *Human-centric Computing and Information Sciences*, vol. 6, no. 1, p. 19, 2016.

[5] M. Frustaci, P. Pace, G. Aloi, and G. Fortino, "Evaluating critical security issues of the iot world: Present and future challenges," *IEEE Internet of things journal*, vol. 5, no. 4, pp. 2483–2495, 2017.

[6] M. Chen, W. Li, G. Fortino, Y. Hao, L. Hu, and I. Humar, "A dynamic service migration mechanism in edge cognitive computing," *ACM Transactions on Internet Technology (TOIT)*, vol. 19, no. 2, pp. 1–15, 2019.

[7] R. Casadei, G. Fortino, D. Pianini, W. Russo, C. Savaglio, and M. Viroli, "Modelling and simulation of opportunistic iot services with aggregate computing," *Future Generation Computer Systems*, vol. 91, pp. 252–262, 2019.

[8] R. S. S. Kumar, A. Wicker, and M. Swann, "Practical machine learning for cloud intrusion detection: challenges and the way forward," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 2017, pp. 81–90.

[9] R. Sommer and V. Paxson, "Enhancing byte-level network intrusion detection signatures with context," in *Proceedings of the 10th ACM conference on Computer and communications security*, 2003, pp. 262–271.

[10] M. J. Hashemi, G. Cusack, and E. Keller, "Towards evaluation of NIDSs in adversarial setting," in *Proceedings of the 3rd ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks*, 2019, pp. 14–21.

[11] H. Qiu, Q. Zheng, G. Memmi, J. Lu, M. Qiu, and B. Thuraisingham, "Deep residual learning based enhanced JPEG compression in the internet of things," *IEEE Transactions on Industrial Informatics*, 2020.

[12] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE transactions on emerging topics in computational intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[13] K. Alrawashdeh and C. Purdy, "Toward an online anomaly intrusion detection system based on deep learning," in *2016 15th IEEE international conference on machine learning and applications (ICMLA)*. IEEE, 2016, pp. 195–200.

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

[15] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma, "PCANet: A simple deep learning baseline for image classification?" *IEEE transactions on image processing*, vol. 24, no. 12, pp. 5017–5032, 2015.

[16] W. E. Zhang, Q. Z. Sheng, A. Alhazmi, and C. Li, "Adversarial attacks on deep-learning models in natural language processing: A survey," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 11, no. 3, pp. 1–41, 2020.

[17] N. Carlini and D. Wagner, "Audio adversarial examples: Targeted attacks on speech-to-text," in *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2018, pp. 1–7.

[18] J. Sun, T. Zhang, X. Xie, L. Ma, Y. Zheng, K. Chen, and Y. Liu, "Stealthy and efficient adversarial attacks against deep reinforcement learning," *arXiv preprint arXiv:2005.07099*, 2020.

[19] H. Qiu, Q. Zheng, T. Zhang, M. Qiu, G. Memmi, and J. Lu, "Towards secure and efficient deep learning inference in dependable IoT systems," *IEEE Internet of Things Journal*, 2020.

[20] J. Clements, Y. Yang, A. Sharma, H. Hu, and Y. Lao, "Rallying adversarial techniques against deep learning for network security," *arXiv preprint arXiv:1903.11688*, 2019.

[21] D. Han, Z. Wang, Y. Zhong, W. Chen, J. Yang, S. Lu, X. Shi, and X. Yin, "Practical traffic-space adversarial attacks on learning-based NIDSs," *arXiv preprint arXiv:2005.07519*, 2020.

[22] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 601–618.

[23] X. Huang, C. Shen, X. Boix, and Q. Zhao, "Salicon: Reducing the semantic gap in saliency prediction by adapting deep neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 262–270.

[24] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: an ensemble of autoencoders for online network intrusion detection," *arXiv preprint arXiv:1802.09089*, 2018.

[25] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis *et al.*, "Understanding the mirai botnet," in *26th {USENIX} security symposium ({USENIX} Security 17)*, 2017, pp. 1093–1110.

[26] H. Zheng, Y. Wang, C. Han, F. Le, R. He, and J. Lu, "Learning and applying ontology for machine learning in cyber attack detection," in *17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications*. IEEE, 2018, pp. 1309–1315.

[27] F. Tramer, N. Carlini, W. Brendel, and A. Madry, "On adaptive attacks to adversarial example defenses," *arXiv preprint arXiv:2002.08347*, 2020.

[28] Y. Zeng, H. Qiu, G. Memmi, and M. Qiu, "A data augmentation-based defense method against adversarial attacks in neural networks," in *International Conference on Algorithms and Architectures for Parallel Processing*. Springer, 2020, pp. 274–289.

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

[30] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 707–723.

Han Qiu received the B.E. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2011, the M.S. degree from Telecom-ParisTech (Institute Eurecom), Biot, France, in 2013, and the Ph.D. degree in computer science from the Department of Networks and Computer Science, Telecom-ParisTech, Paris, France, in 2017. He is currently a Researcher with the LINCS Laboratory, Paris, and co-affiliated with Telecom Paris. His research interests include AI security, big data security, and cloud computing.

Tian Dong received the bachelor's degree in French and information engineering from the SPEIT, Shanghai Jiao Tong University, Shanghai, China, in 2019. He is currently working toward the master's degree in a double degree program between Shanghai Jiao Tong University and Ecole Polytechnique, Paris, France. His research interest includes machine learning security and use of machine learning in computer security.

Tianwei Zhang is an assistant professor in School of Computer Science and Engineering, at Nanyang Technological University. His research focuses on computer system security. He is particularly interested in security threats and defenses in machine learning systems, autonomous systems, computer architecture and distributed systems. He received his Bachelor's degree at Peking University in 2011, and the Ph.D degree in at Princeton University in 2017.

Jialiang Lu is an associate professor and the Assistant Dean at ParisTech Shanghai Jiao Tong and a researcher at the Department of Computer Science and Engineering at Shanghai Jiao Tong University, China. He received the M.S. and the M.E. degree with honor from the Department of Telecommunication at INSA Lyon, France, in 2004 and the Ph.D. degree from INSA Lyon, in 2008. His research interests include wireless networks, vehicle networks, and security aspects of machine learning. He has published over 50 publications in international journals and conferences in this area.

Gerard Memmi currently is professor and Head of the Networks and Computer Science Department at Telecom-ParisTech since 2009. He is a member of the executive board of the IRT SystemX since 2012. Before joining Telecom-ParisTech, Gerard Memmi held various executive positions in American start-ups. He succeeded in delivering the industry's best-in-class equivalency checker used to verify electronic design; and focused on improving its architecture and performances. While founding and developing the Applied Research Laboratory for Groupe Bull in the US, he was honored as Principal Investigator for a DARPA grant on Collaborative Software. He has over 80 publications including patents, co-authored a book, gave keynotes presentations in international conferences. He is holding these d'Etat in Computer Science from Universite Pierre et Marie Curie, Paris. Gerard Memmi is constantly involved in the development of key scientific and industrial partnerships. Today, his main scientific topics of interest are about data protection and privacy, energy profiling of software programs, and verification of distributed systems.

Meikang Qiu received the BE and ME degrees from Shanghai Jiao Tong University and received Ph.D. degree of Computer Science from University of Texas at Dallas. He is the Department Head and tenured full professor of Texas A&M University Commerce. He is an ACM Distinguished Member and IEEE Senior Member. He had been selected as Highly Cited Researcher by Web of Science in 2020 and IEEE Distinguished Visitor from 2021-2023. He is the Chair of IEEE Smart Computing Technical Committee. He has published 20+ books, 600+ peer-reviewed journal and conference papers, including 80+ IEEE/ACM Transactions papers. His research interests include Cyber Security, Big Data Analysis, Design Automation, Cloud Computing, Smarting Computing, Intelligent Data, Embedded systems, etc. He is an Associate Editor of 10+ international journals, including IEEE Transactions on Computers, IEEE Transactions on Cloud Computing, IEEE Transactions on Big Data, and IEEE Transactions on SMC.